DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Imputação da ocupação de vias em redes de tráfego urbano por meio de máquina de vetores de suporte por mínimos quadrados

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Augusto José Toso

Florianópolis, Agosto de 2017

Imputação da ocupação de vias em redes de tráfego urbano por meio de máquina de vetores de suporte por mínimos quadrados

Augusto José Toso

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Rodrigo Castelan Carlson

Banca Examinadora:

Prof. Werner Kraus Junior
Orientador na Empresa

Prof. Rodrigo Castelan Carlson
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Prof. Ubirajara Franco Moreno, Avaliador

Gabriel Perotto, Debatedor

Ricardo de Oliveira Gonzalez Aldeyturriaga, Debatedor

Agradecimentos

Ao meu orientador, **Rodrigo Castelan Carlson**, por estar sempre disponível e me auxiliar neste trabalho.

À **BRASCONTROL**, pela bolsa de pesquisa, e por fornecer os dados utilizados.

Ao **Renan Eccel** por me auxiliar na geração das imagens da malha de Santos.

Em especial, ao **Luciano Dionisio Dantas** por realizar as simulações do *AIMSUM*, muitas vezes trabalhosas.

E, finalmente, à minha família, pelo apoio e paciência!

Resumo

Seguindo o exemplo de muitas cidades pelo mundo, Santos, no estado de São Paulo, implementou um "Sistema Inteligente de Trafego" (**SIT**) para controle da sua malha urbana. O SIT instalado se utiliza de detectores do tipo *"Laço indutivo"* para colher os dados de trafego e, com base nesses dados, realizar o controle semafórico. Embora baratos e de fácil instalação, os detectores *"Laço indutivos"* são bastante suscetíveis a defeitos e medições incorretas, que podem inclusive perdurar por vários dias. A perda de dados compromete ou até mesmo impede o controle por parte do SIT. Para contornar esse problema são utilizadas técnicas de imputação de dados. Atualmente a técnica de imputação, utilizada pelo SIT de Santos, é baseada na ponderação dos dados a montante do sensor defeituoso. Este trabalho busca encontrar um novo algorítimo de imputação, mais sofisticado, visando a melhora no desempenho do SIT. Na literatura, o algoritmo de aprendizagem: **"Maquina de Vetores de Suporte por Mínimos Quadrados"** (**MVS-MQ**) surge como o candidato mais promissor para a imputação de dados de trafego. Por isso, ele é o algoritmo selecionado para o estudo de viabilidade realizado neste trabalho. Os resultados obtidos mostram que a técnica MVS-MQ é uma opção mais interessante para imputação de dados de trafego do que a técnica atual.

Palavras-chave: MVS-MQ, Maquina de vetores de suporte, Imputação de dados de trafego, Controle de trafego, Sistema Inteligente de trafego.

Abstract

Following the example of many cities around the world, Santos - SP, implemented a Intelligent Transportation System (**ITS**) to control its urban traffic. The ITS installed have loop detectors to gather data and, based on this data, control the traffic lights. Although inexpensive and easy to install, the loop detectors are highly prone to missing and erroneous data, that can even last for several days. The missing data considerably degrades or even unables the control. To address this problem data imputation techniques are used. The imputation technique currently used in Santos is calculated with the weighted data from the upstream sensors. This paper seeks a new imputation algorithm, more sophisticated, aiming a better performance of the SIT. In the literature, the learning algorithm **Least Squares - Support Vector Machines** (**LS-SVM**) is presented as the most promising candidat for traffic data imputation. Therefore it was the selected algorithm for the research done in this paper. The results shows that LS-SVM is a more suitable option for traffic data imputation than the current one.

Keywords: LS-SVM, Least Square Support Vector Machine, Traffic data imputation, Traffic Control, Intelligent Transportation System.

Lista de ilustrações

Figura 1 – Seção de uma malha viária com a identificação de seus elementos . . . 23
Figura 2 – Exemplo: Medições dos sensores de laço indutivo de Santos 24
Figura 3 – Taxa de ocupação do link L02043 em Santos com diferentes períodos de agregação (Ta) 26
Figura 4 – Representação simplificada de *occt* versus x_m para diferentes posições do detector 26
Figura 5 – Taxa de ocupação temporal da estáção L02145 28
Figura 6 – Imagem do Google Earth destácando as regiões de implementação do SCT 29
Figura 7 – Detalhe da malha de Santos - Região da Avenida da orla, dividida em 3 secções 30
Figura 8 – Detalhe da malha de Santos - Região da Av. Dona Ana Costa 31
Figura 9 – Detalhe da malha de Santos - Região da Av. Nossa Senhora de Fátima 32
Figura 10 – Malha urbana de Chaniá, Grécia 37
Figura 11 – Exemplo de classificação utilizando hiperplanos 44
Figura 12 – Diferença entre os vetores Vx_+ e Vx_- 45
Figura 13 – Exemplo de dados que não são linearmente classificáveis 49
Figura 14 – Projeção dos dados para um espaço de recursos dimensionais superiores. 50
Figura 15 – Diferença de modelos da MVS sem e com sobreajuste 51
Figura 16 – Representação da função ϵ-insensitiva 53
Figura 17 – Contraste entre a esparsidade da RVS e MVS-MQ 56
Figura 18 – Exemplo de trecho do banco de dados processado 58
Figura 19 – Seleção dos sensores vizinhos para estação Sen171 62
Figura 20 – Seleção dos sensores vizinhos para estação Sen177 63
Figura 21 – Seleção dos sensores vizinhos para estação Sen206 63
Figura 22 – Dados das 9 replicações NPC 68
Figura 23 – Imputação do Sen177 - Cenário **a** 69
Figura 24 – Imputação do Sen206 - Cenário **a** 70
Figura 25 – Imputação do Sen177 - Cenário **b** 72
Figura 26 – Imputação do Sen177 - Cenário **b** 73
Figura 27 – Dados reais de Santos agregados a cada 150 segundos 75
Figura 28 – Imputação da estação, *'L01071'* ($Ta = 150$) 77
Figura 29 – Imputação da estação, *'L02082'* ($Ta = 150$) 78
Figura 30 – Imputação da estação, *'D03122'* ($Ta = 150$) 79
Figura 31 – Imputação dos dados de treinamento 82

Lista de tabelas

Tabela 1 – Tabela de correlação de Pearson - estação Sen171 62
Tabela 2 – Tabela dos parâmetros internos γ e σ para *Sen177* e *Sen206*, Cenário **a** 68
Tabela 3 – Tabela comparativa entre métodos - Sen177 - Cenário **a** 71
Tabela 4 – Tabela comparativa entre métodos - Sen206 - Cenário **a** 71
Tabela 5 – Tabela comparativa entre métodos - Media global dos 165 sensores - Cenário **a** 71
Tabela 6 – Tabela dos parâmetros internos γ e σ para *Sen177* e *Sen206*, Cenário **b** 74
Tabela 7 – Tabela comparativa entre métodos - Sen177 - Cenário **a** 74
Tabela 8 – Tabela comparativa entre métodos - Sen206 - Cenário **b** 74
Tabela 9 – Tabela comparativa entre métodos - Media global dos 165 sensores - Cenário **b** 74
Tabela 10 – Tabela de correlação de Pearson - estação *'L01071'* 76
Tabela 11 – Tabela de correlação de Pearson - estação *'L02082'* 76
Tabela 12 – Tabela de correlação de Pearson - estação *'D03122'* 76
Tabela 13 – Tabela dos parâmetros internos γ e σ para as estações. 80
Tabela 14 – Tabela comparativa entre métodos, *'L01071'* ($Ta = 150$) 80
Tabela 15 – Tabela comparativa entre métodos, *'L02082'* ($Ta = 150$) 80
Tabela 16 – Tabela comparativa entre métodos, *'D03122'* ($Ta = 150$) 80
Tabela 17 – Desempenho do SIT utilizando diferentes métodos de imputação 83

Lista de abreviaturas e siglas

Lista de Siglas

MVS: Máquina de Vetores de Suporte

SVM: *Support Vector Machine* (Máquina de Vetores de Suporte)

RVS: Regressão por Vetores de Suporte

MVS-MQ: Máquina de Vetores de Suporte por Mínimos Quadrados

LS-SVM: *Least square SVM* (MVS por Mínimos Quadrados)

SIT: Sistemas Inteligentes de Transporte

SCT: Sistemas de Controle de Tráfego

ITS: *Intelligent Transportation System* (Sistemas Inteligentes de Trafego)

PIB: Produto Interno Bruto

TUC: *Traffic-responsive Urban Control* (Controle Urbano Responsivo ao Tráfego)

TUC_{imp}: Algorítimo de imputação implementado pelo TUC de Santos

ARIMA: Modelo Autorregressivo Integrado de Média Móvel

MI: Múltipla Imputação.

MCMC: *Markov Chain Monte Carlo* (Monte Carlo por Cadeias de Markov)

ME: Maximização de Expectativa

DA: *Data Augmentation*

BPNN: *Back-propagation Neural Network* (Rede Neural de Retro propagação)

ERM: *Empirical Risk Minimization* (Minimização de Riscos Empírica)

SRM: *Structural Risk Minimization* (Minimização de Riscos Estrutural)

PQ: Programação Quadrática

SMO: *Sequential Minimization Otimization* (Otimização Minima Sequencial)

KKT: Condição de Karush-Kuhn-Tucke

RBF: *Radial Basis Function* (Função de Bases Radiais)

CSA: *Chaotic Simulated Annealing* (Arrefecimento Caótico Simulado)

eMax: Erro Máximo

MAE: Media Absoluta do Erro

MEDAE: Mediana Absoluta do Erro

MSE: Media do Quadrado do Erro

MAPE: Media Absoluta Percentual do Erro

Lista de símbolos

Ta	Período de agregação de dados
$occt$	Taxa de ocupação temporal
$T0_k$	Horário de inicio do período k
$d_{i,j}$	Duração da detecção j do sensor i
$t0_{i,j}$	Horário de inicio da detecção j do sensor i
$j0$	Primeira detecção de um período k
jn	Ultima detecção de um período k
da_k	Termo de ajuste da duração total de ocupação do período k
x_m	Numero de veículos no link m
N	Numero de links, com detectores, a montante
y_i	Numero de veículos do link i, proporcional a outro link
Wz	Peso proporcional dos links a montante do link z
$V\vec{x}$	Vetor (multidimensional) de atributos de treinamento
Vy	Valor (label) de saída
Vs	Vetores de Suporte
Vs_+	Vetor de Suporte com label positivo
Vs_-	Vetores de Suporte com label negativo
w	Vetor normal ao hiperplano da Maquina de Vetores de Suporte
b	Valor de bias do hiperplano da Maquina de Vetores de Suporte
ρ	Valor da margem que separa o hiperplano e os Vetores de Suporte
$\mathcal{L}$	Operador de Lagrange
$f(\omega)$	Função de otimização da Maquina de Vetores de Suporte
$g_i(\omega)$	Restrição de desigualdade i da função $f(\omega)$

α_i	Multiplicador de Lagrange relacionado a restrição $g_i(\omega)$
$h_i(\omega)$	Restrições de igualdade i da função $f(\omega)$
β_i	Multiplicador de Lagrange relacionado a restrição $h_i(\omega)$
$\phi(.)$	Função de projeção de $\vec{V}x$ para um espaço de recursos dimensionais superior
$K(.)$	Função Kernel
$K_l(.)$	Kernel Linear
$K_p(.)$	Kernel Polinomial
r	Parâmetro do Kernel Polinomial, referente ao numero de recursos
θ	Parâmetro do Kernel Polinomial, referente a complexidade da transformação
$K_{rbf}(.)$	Kernel Gaussiano de função de Base Radial (*Radial bases function*)
σ	Parâmetro do Kernel Gaussiano, referente a complexidade da transformação
ϵ	Margem de tolerância (insensibilidade) para erros pequenos na função de perda
ξ	Valor da função de perda da RVS
C	Parâmetro de grau de penalização da RVS
η	Multiplicador de Lagrange do tipo de α, relacionado a ξ
γ	Parâmetro de grau de penalização da MVS-MQ
e_i	Erro entre o valor real e o valor estimado
T_t	Quantidade de pares de treinamento
es	Índice de uma estação
n	Numero de atributos de treinamento, equivalente a numero de dimensões de $\vec{V}x$
$C_{Pearson}$	Coeficiente de Correlação de Pearson
$var(.)$	Função que define a variância
$cov(.)$	Função que define a covariância

Sumário

1 INTRODUÇÃO 19
1.1 Visão geral 19
1.2 Problemática 20
1.3 Objetivo 21
1.4 Estrutura do documento 21

2 MALHAS VIÁRIAS URBANAS E DADOS DE TRÁFEGO 23
2.1 Nomes e convenções 23
2.2 Coleta dos dados 24
2.2.1 Estimativa de número de veículos presentes no link 25
2.3 Dados perdidos 27
2.3.1 Identificação de falhas 27
2.3.2 Característica das falhas 27
2.4 Malha de Santos 28
2.4.1 Empresa BRASCONTROL 29
2.4.2 TUC de controle 33
2.4.3 Imputação de dados em Santos 34
2.4.4 Dados disponíveis 35
2.5 Malha de Chaniá, Grécia 35

3 IMPUTAÇÃO DE DADOS DE TRÁFEGO 39
3.1 Visão geral 39
3.1.1 Redes Neurais retro-propagadas 40
3.2 Escolha de MVS-MQ 40

4 MÁQUINA DE VETORES DE SUPORTE 43
4.1 Máquina de Vetores de Suporte Clássica 43
4.1.1 Problema linearmente classificável 44
4.1.2 Casos não-lineares 49
4.2 Regressão por Vetores de Suporte 52
4.3 Máquina de Vetores de Suporte por Mínimos Quadrados 54

5 IMPLEMENTAÇÃO 57
5.1 Softwares Utilizados 57
5.2 Caracterização do Problema 57
5.2.1 Quantidade de dados de treinamento 58

5.2.2 Estações vizinhas . . . 58
5.3 Correlação entre detectores vizinhos . . . 60
5.4 Modelo MVS-MQ . . . 64
5.4.1 Definição dos parâmetros internos da MVS-MQ . . . 64
5.4.2 Treinamento e imputação . . . 64
5.5 Implementação métodos comparativos . . . 65
5.6 Métodos de avaliação de desempenho . . . 66

6 RESULTADOS . . . 67
6.1 Modelo de Chaniá . . . 67
6.1.1 Treinamento e imputação em NPC . . . 68
6.1.2 Treinamento com PCDBN e imputação com NPC, PC e PCQNB . . . 71
6.2 Medições reais da Malha de Santos . . . 74
6.3 Comparativo de desempenho na Malha de Santos . . . 80

7 CONCLUSÕES E PERSPECTIVAS . . . 85
7.1 Trabalhos futuros . . . 85
7.1.1 Treinamento do modelo . . . 85
7.1.2 Solução esparsa . . . 86
7.1.3 Identificação de erros . . . 86
7.1.4 Parâmetros internos da MVS . . . 87
7.2 Conclusão final . . . 87

REFERÊNCIAS . . . 89

ANEXOS 95

ANEXO A – MATLAB: ALGORÍTIMO DE TREINAMENTO E ESTATÍSTICAS DE ERRO . . . 97

ANEXO B – POSTGRESQL: AGREGAÇÃO DOS DADOS . . . 103

ANEXO C – MATLAB: IMPUTAÇÃO PARA SIMULAÇÕES DE DESEMPENHO . . . 107

1 Introdução

1.1 Visão geral

Nas ultimas décadas, com o grande desenvolvimento das cidades, os problemas relacionados à mobilidade urbana têm se agravado. A expansão da malha viária não tem sido capaz de acompanhar o aumento do número de veículos que trafegam em suas vias. Por exemplo, nos Estados Unidos, de 1985 a 2014 a quantidade de quilômetros viajados cresceu 70%, enquanto o de quilômetros pavimentados apenas 8% [1]. Essa disparidade entre capacidade das vias e quantidade de carros circulando é a principal responsável pelo surgimento dos congestionamentos. Segundo o Relatório de Mobilidade Urbana de 2015 [2], a média anual de atrasos por pessoa, nas 471 áreas urbanas analisadas (EUA), foi de 42 horas em 2014, um aumento de 100% comparado com aquela de 1985.

As consequências dos engarrafamentos são graves e não se limitam somente ao tempo perdido pelos motoristas, elas também causam malefícios a toda comunidade como a diminuição da qualidade de vida, o aumentando a poluição sonora e do ar, etc. Somados a isto ainda existem os prejuízos econômicos. Em 2014, congestionamentos causaram aos americanos 6.9 bilhões horas de atraso e um consumo extra de 11.75 bilhões de litros de combustível, totalizando um custo com congestionamentos de $160 bilhões de dólares [2]. Na Europa, congestionamentos custam anualmente cerca de 1% do Produto Interno Bruto (PIB) [3].

A solução do problema de mobilidade urbana não é simples. Limitações orçamentarias, prejuízos socioambientais ou a falta de espaço muitas vezes inviabilizam a expansão da malha viária. Pensando nisso muitas cidades pelo mundo têm adotado **Sistemas Inteligentes de Transporte (SIT)** que visam uma melhor utilização da infraestrutura existente. No caso do tráfego urbano, esses sistemas têm como objetivo melhorar a qualidade das viagens, reduzir os atrasos, e aumentar a eficiência das vias. *"Do ponto de vista de controle, isto pode ser traduzido como o emprego de um sistema de controle adequado e eficiente que responda automaticamente as condições de tráfego existentes..."* [4].

Sistemas de Controle Tráfego (SCT) se baseiam no monitoramento em tempo real das condições de tráfego, coletando dados como velocidade, ocupação, fluxo, etc... e os utilizando para implementação das suas estrategias de controle. Fica claro, portanto, que *"dados de tráfego são essenciais tanto para as pesquisas na área de tráfego quanto para suas aplicações"* [5].

Existem diferentes técnicas e dispositivos capazes de realizar medições de dados de tráfego: sensores fixos de laço indutivo, sensores de infravermelho, radares, sensores

de vídeo, etc... [6] Os avanços recentes na coleta de dados de usuários e processamento de *"big data"* permitem inclusive o monitoramento por "sondagem veicular", que consiste em utilizar os dados fornecido pelos GPS dos usuários. Entretanto, no contexto atual, a solução mais prática e custo eficiente para coleta de dados de tráfego continua sendo o uso de sensores de laço indutivo.

1.2 Problemática

Apesar de os sensores de laço indutivo serem os mais empregados nos sistemas de infraestrutura e servirem como modelo para detecção de tráfego para muitas agencias de transporte [7, 8], pesquisadores na área de engenharia de tráfego têm notado que os dados colhidos com eles estão incompletos, com perdas parciais, ou substancialmente comprometidos por ruído [9]. O programa de monitoramento de mobilidade do Instituto de Transportes do Texas declara que após a filtragem de erros, os conjunto de dados arquivados continham de 16% a 93% de dados perdidos. Al-Deek, Venkata e Chandra [10] relataram uma taxa de perda em sensores de laço indutivo de 15%, referente a coleta na rodovia americana "Interstate I-4". Por fim, pesquisadores do Instituto de Tecnologia da Geórgia publicaram uma taxa de perda de 4% a 14% em coletas naquele estado [11].

Alem das perdas curtas e intermitentes, inerentes dos sensores de laço indutivo, os SCTs ainda sofrem com perdas de longa duração causados por avarias nos sensores. Por causa dos ambientes rigorosos aonde são instalados, os sensores são bastante suscetíveis a defeitos. Ainda, por existirem diferentes empresas operando e realizando obras em uma mesma via, é comum que a infraestrutura do SCT seja negligenciada e acabe danificada. Tais defeitos requerem a manutenção ou troca dos sensores e, até que isto ocorra, os dados daquela via acabam não sendo coletados.

Os problemas com os dados coletados impõem um grande desafio, pois os SCTs precisam realizar o controle do sistema mesmo quando os dados disponíveis são incompletos. *"A falta de medidas reais, devido às falhas dos detectores, podem causar uma séria degradação da performance da estrategia de controle. Para evitar tais situações, uma substituição apropriada dos dados perdidos deve ser realizada para permitir uma degradação mais atenuada do sistema de controle"* [4]. Ou seja, *"o tratamento dos dados perdidos é um importante passo de preparação para um controle efetivo e administração dos SCTs"* [12]. Esse tratamento, mais especificamente a substituição de dados perdidos, é genericamente conhecida como imputação de dados, e é o tema central deste trabalho.

1.3 Objetivo

Santos, no litoral paulista, é o nosso local de interesse. Assim como em outros cidades, Santos possui um Sistema de Controle de Tráfego que realiza seu monitoramento através de sensores de laço indutivo, e por isso sofre com a perda de dados. Atualmente, a técnica de controle utilizada pela BRASCONTROL, empresa responsável pelo SCT, é conhecida como **Controle Urbano Responsivo ao Tráfego (TUC)** e possui um algorítimo de imputação interno que calcula os valores de imputação proporcionais as medições a montante do laço com falha. Apesar de ser de simples implementação, este método não é o mais adequado para a imputação de dados de tráfego.

O objetivo desse trabalho é apresentar um método alternativo de imputação de dados para o SCT de Santos, visando a melhora de sua performance. Dentre as diversas técnicas presentes na literatura a **Máquina de Vetores de Suporte por Mínimos Quadrados** (MVS-MQ), proposta por Suykens et al. [13], é a apontada como mais promissora, e foi escolhida candidata para substituir o atual algoritmo de imputação do SCT de Santos.

1.4 Estrutura do documento

No **Capítulo 2** são apresentados os dados utilizados na pesquisa, e as malhas de tráfego urbano que geraram estes dados, sejam elas reais ou modeladas. Um detalhamento especial é feito para a malha de Santos, explicando também o funcionamento do seu SCT, e a atual técnica de imputação. No **Capítulo 3** o problema da imputação de dados de tráfego é explorada mais a fundo, e algumas das técnicas são brevemente apresentadas. No **Capítulo 4** são descritas detalhadamente a teoria por trás da Maquina de Vetores de Suporte (MVS) e sua adaptação para utilização em problemas de regressão e imputação de dados. No **Capítulo 5** é mostrado detalhes da implementação da **MVS-MQ** para o caso de Santos, explicando também como é feita a seleção dos dados de entrada, e os métodos para avaliação do desempenho. No **Capítulo 6** apresentamos os resultados obtidos. Por fim, o **Capítulo 7** nos traz a conclusão deste trabalho, e sugestões para pesquisas futuras.

2 Malhas viárias urbanas e dados de tráfego

Dos diferentes tipos de dados de tráfego, os mais relevantes são os dados espaciais (que incluem densidade, ocupação espacial), pois se traduzem em número de veículos presentes (ou aguardando em filas) dentro de um mesma via. O conhecimento sobre essa ocupação espacial é um dado fundamental para os SCTs, pois um dos seus objetivos é o escoamento ótimo dos veículos, evitando fenômenos como o *"Queue spillover"*. *Spillovers* ocorrem quando o crescimento das das filas a jusante bloqueiam a chegada dos veículos diretamente a montante, impedindo o avanço dos carros, embora o semáforo esteja aberto [14].

Infelizmente medidas diretas de dados espaciais são caras ou imprecisas. Por isso, no seu lugar, medições locais e temporais mais simples são realizadas, por exemplo utilizando sensores de laço indutivo. No caso de Santos, o parâmetro de entrada do SCT é a taxa ocupação temporal, uma medida local, obtida a partir das medições do laço indutivo. Esse dado é posteriormente utilizado para estimar (indiretamente) a quantidade de veículos presentes no link e então realizar o controle. Portanto, para fins de imputação, o dado de interesse é justamente a **taxa de ocupação temporal**.

Neste **Capítulo** serão dados mais detalhes sobre a coleta dos dados de tráfego, em como eles são obtidos, armazenados, e posteriormente convertidos para serem utilizados nos cálculos de controle do SCT. Em seguida serão apresentadas as malhas viárias que deram origem aos dados utilizados neste trabalho, sejam elas reais, ou modeladas. Por fim, tem-se uma breve descrição sobre o SCT de Santos e seu funcionamento.

2.1 Nomes e convenções

Para auxiliar a compreensão de alguns dos termos utilizados ao longo deste trabalho, são ilustrados na Figura 1.

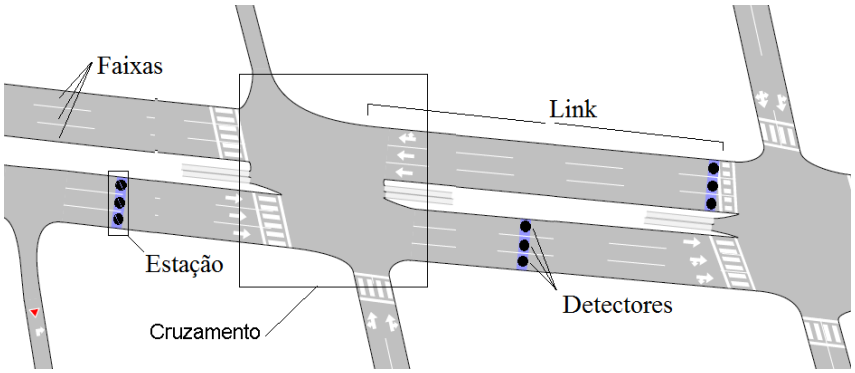


Figura 1 – Seção de uma malha viária com a identificação de seus elementos

Um detector de tráfego é considerado um único sensor de laço indutivo, posicionado em uma faixa da via. Uma estáção é um grupo de detectores posicionados no mesmo local (faixas distintas), e um link corresponde a uma porção da via entre dois cruzamentos semafóricos. Como nas malhas urbanas é comum que seja instalado apenas uma estáção por link, os dois termos serão considerados intercambiáveis neste trabalho.

2.2 Coleta dos dados

Sensores de laço indutivo são capazes de detectar quando um objeto metálico (veículo) se encontra sobre ele. Quando isto ocorre, um pulso elétrico é produzido. Ao processar tais pulsos, e utilizando um limite de tolerância (*threshold*) pré-definido, é possível gerar uma série de eventos discretos contendo o tempo de início e o período em que o objeto permaneceu sobre o sensor. Na Figura 2 é mostrado um exemplo dos dados brutos gerados pelos sensores na malha de Santos. A coluna *seq_detector* indica qual o detector realizou a medição, *vehicle_detection_period* a duração, e *detection_start_time* o horário de início.

seq_detector smallint	vehicle_detection_period real	detection_start_time timestamp with time zone
40	0.542347	2016-09-01 00:00:03.96389-03
38	0.313992	2016-09-01 00:00:07.332168-03
40	0.470987	2016-09-01 00:00:07.417805-03
316	0.399648	2016-09-01 00:00:01.357662-03
309	0.54237	2016-09-01 00:00:01.640256-03
287	0.156979	2016-09-01 00:00:02.06762-03
91	0.385384	2016-09-01 00:00:01.757698-03
92	0.313983	2016-09-01 00:00:01.986086-03
91	0.156993	2016-09-01 00:00:02.542678-03
92	0.313991	2016-09-01 00:00:02.542678-03

Figura 2 – Exemplo: Medições dos sensores de laço indutivo de Santos

A partir da série de eventos discretos é possível obter a taxa de ocupação temporal *occt*, que representa o percentual de tempo em que um sensor detectou veículos, em um dado período Ta de agregação. Este período está usualmente relacionado ao ciclo semafórico, que é o tempo que um semáforo leva para concluir suas 3 fases: verde, amarelo e vermelho.

$$occt_{i,k} = \frac{1}{Ta}\left(\left(\sum_{j0}^{jn} d_{i,j}\right) + da_k\right) \tag{2.1}$$

A Equação 2.1 mostra como é feito o cálculo, em que $occt_{i,k}$ representa a taxa de ocupação temporal k para o sensor i, $T0_k$ o horário de início do período k, $d_{i,j}$ representa

o tempo de duração da detecção j do sensor i e $t0_{i,j}$ o momento de início da detecção j do sensor i.

A detecção $j0$ representa a primeira e jn a ultima da série, dentro do intervalo, tal que:

$$\begin{cases} t0_{i,j0} \geqslant T0_k \\ t0_{i,jn} \leqslant T0_{k+1} \end{cases} \tag{2.2}$$

O termo da_k se refere ao termo de ajuste, necessário devido a ocupações iniciadas anteriormente a $j0$ e que se prolongaram após $T0_k$ e as ocupações que iniciaram em jn e que só terminaram após $T0_{k+1}$. A Equação 2.3 descreve como calcular da_k:

$$\begin{aligned} da_k = min\Big((t0_{ij0-1} + d_{ijn0-1} - T0_k), 0\Big) \\ -min\Big((t0_{ijn} + d_{ijn} - T0_{k+1}), 0\Big) \end{aligned} \tag{2.3}$$

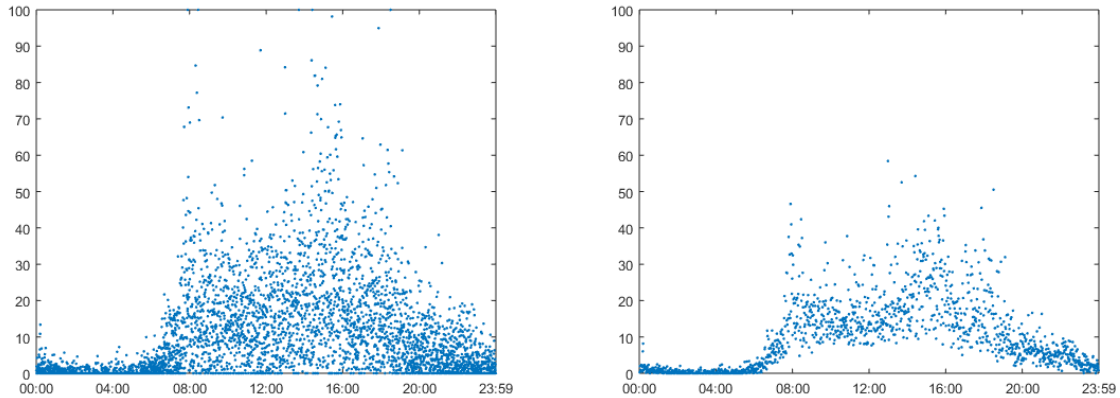
É comum que os dados de taxa de ocupação temporal sejam agregados também por link, e não calculados individualmente para cada detector. Neste caso é feita uma media simples de todos os sensores de uma mesma estáção.

A escolha do período de agregação Ta tem grande influencia na qualidade dos dados de taxa de ocupação temporal. Períodos curtos de amostra tendem a produzir respostas mais ruidosas. Quando este período é estendido, o resultado de ocupação se suaviza revelando assim a tendencia dos dados. A Figura 3 mostra a diferença na qualidade dos dados para diferentes agregações Ta.

2.2.1 Estimativa de número de veículos presentes no link

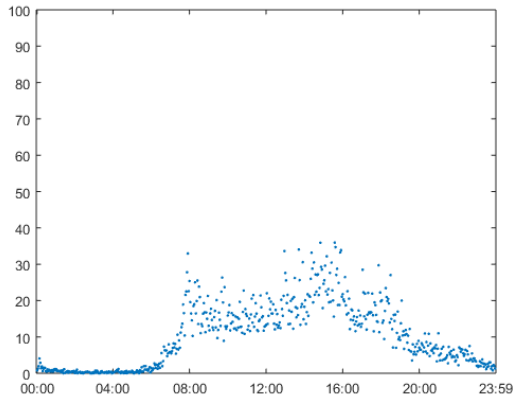
A estimativa do número de veículos, em um link da malha, é feito com base no valor de ocupação temporal. A Figura 4, apresentada por Diakaki [4] mostra uma representação simplificada do mapeamento entre taxa de ocupação e o número de veículos x_m, baseado na posição do sensor, para um link de tamanho normalizado (100 veículos).

A escolha da posição de instalação do detector representa uma solução de compromisso: quanto mais próximos os detectores forem instalados do ponto de parada, menor a quantidade máxima de veículos em fila que ele é capaz de estimar; quanto mais distantes, mais tardia é a detecção e filas muito pequenas podem não serem detectadas.

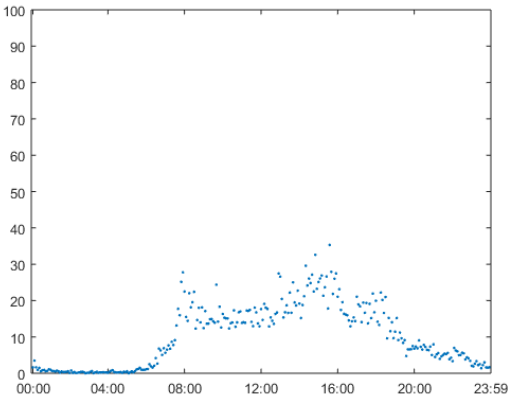


(a) $Ta = 20s$

(b) $Ta = 60s$



(c) $Ta = 150s$



(d) $Ta = 300s$

Figura 3 – Taxa de ocupação do link L02043 em Santos com diferentes períodos de agregação (Ta)

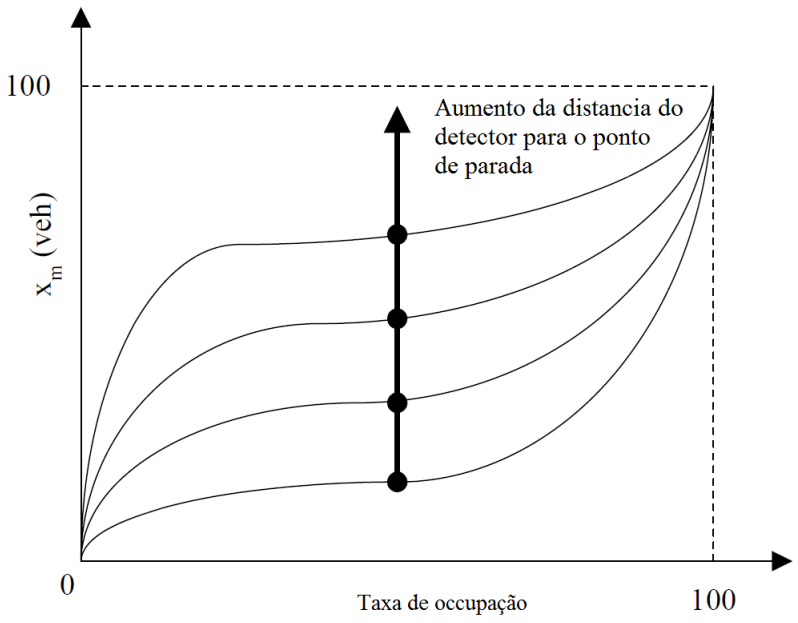


Figura 4 – Representação simplificada de taxa de ocupação versus número de veículos para diferentes posições do detector (Diakaki [4], 1999).

2.3 Dados perdidos

2.3.1 Identificação de falhas

Antes que a imputação possa ser realizada, os dados precisam ser avaliados, identificando dados corrompidos ou inválidos, e os removendo do grupo de dados disponível. Dados com erro são aqueles que não são fisicamente possíveis. Por exemplo, medições com durações muito curtas ou muito longas, que caracterizariam velocidades incoerentes (nulas) ou impraticáveis (muito rápidas). Erros nos dados podem ser causados tanto por falhas no sensor ou transmissão, quanto que por interferências externas (por exemplo, uma caçamba de lixo metálica posicionada próxima do sensor).

A identificação dos erros ocorre na análise dos próprios dados. Os algorítimos de detecção de erros usualmente trabalham com os dados brutos, analisando cada amostra, e realizando o diagnóstico e possível remoção [15]. Para o caso da taxa de ocupação, isso significa que, após o processamento, os "buracos" nos dados acabam sendo mascarados. Por isso, pode não ser possível determinar se um dado nulo, ou próximo de 0, é oriundo da agregação somente de medição reais ou se ele está contaminado por dados omissos.

Existem outros métodos capazes de identificar os erros também após o processamento, porém eles o fazem através de uma análise macroscópica. Por exemplo, Chen et al. [15] propõe um método que analisa todas as amostras relativas a um dia . Por motivos óbvios, métodos desse tipo não são aplicáveis a problemas de tempo real, como são os dos SCTs.

As explicações anteriores servem apenas para exemplificar a complexidade do assunto, uma vez que a detecção de erros não faz parte do escopo deste trabalho. O algorítimo, sugerido nessa pesquisa, realiza as estimativas para todas as estáções e (possivelmente) utilizando todos os dados fornecidos, independente de conterem ou não falhas. Caberá ao SCT decidir entre imputar o valor estimado, ou utilizar o valor medido.

2.3.2 Característica das falhas

A perda de dados de tráfego pode ocorrer em diferentes padrões, tanto temporal quanto espacialmente. Como discutido na introdução, ela ocorre por dois motivos: falhas inerentes do sensor, que causam perdas curtas e distribuídas de forma aleatória; e falhas longas e que podem atingir estáções inteiras, causadas por defeitos ou avarias no sensor e que requerem reparo.

Naturalmente, quando as perdas de dados são menores, ou mais diluídas, existe mais informação disponível para auxiliar na imputação, portanto as estimativas nesse caso produzem melhores resultados. Por exemplo: se um detector, central à estáção, falha por 1 minuto, mas todos os outros da estáção estão disponíveis, uma boa estimativa pode ser

feita. Entretanto, se toda uma estáção é danificada e para de fornecer dados o dia todo, o trabalho de imputação se torna muito mais complexo.

Na literatura, as pesquisas sobre imputação de dados estão tipicamente focadas com os defeitos curtos. Este não é o caso de Santos, aonde a preocupação maior é justamente com as perdas de longa duração. Na Figura 5 é possível ver um exemplo de uma falha, na estáção 'L02145', que causa perda de dados. É possível notar que no dia 22 de setembro há uma falha no sensor que dura varias horas.

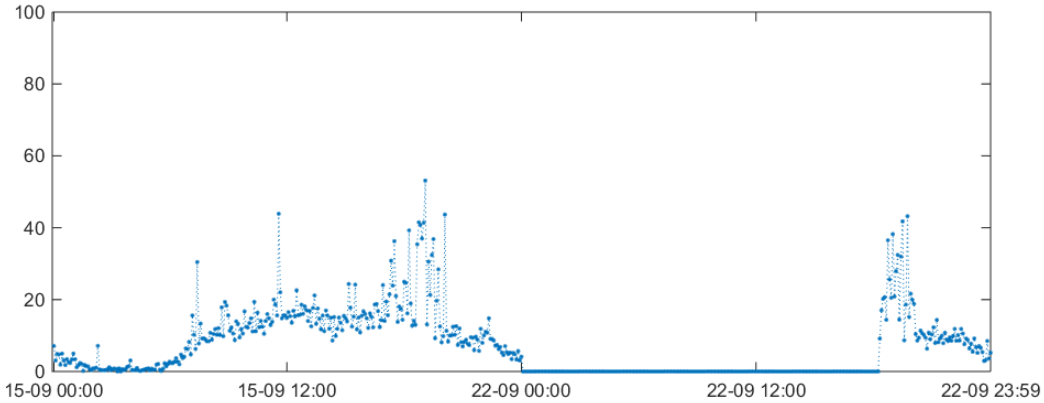


Figura 5 – Taxa de ocupação temporal da estáção L02145

2.4 Malha de Santos

Como mencionado anteriormente, o objetivo principal deste trabalho é o melhoramento da imputação de dados realizada pelo SCT de Santos, e consequentemente melhora no controle da malha. Por isso, a maior parte dos dados utilizados neste trabalho são justamente os coletados em Santos, sendo que eles foram obtidos tanto de medições reais, quanto de resultados simulados utilizando o software AIMSUM.

Santos é um município portuário, localizado no litoral do estádo de São Paulo e a 72 quilômetros da capital. A cidade abriga o maior porto da América Latina, com 13 quilômetros de extensão e por onde passa mais de um quarto de todas as cargas que entram e saem do Brasil . Ao lado das atividades portuárias, os setores do turismo e da pesca completam a lista das atividades econômicas mais relevante do município [16]. Com uma população de aproximadamente 430 mil habitantes (Estimativa IBGE de 2015), e de 1,7 milhões na região metropolitana (IBGE 2008), a cidade sofre com os problemas relacionados aos congestionamentos.

Pensando nisso, ela foi a primeira da região a implementar os semáforos com controle em tempo real. O sistema foi instalado pela empresa BRASCONTROL, e está presente em 3 regiões da cidade. Os equipamentos estão distribuídos em 49 cruzamentos das avenidas da orla (da divisa à Rua Alexandre Martins), Ana Costa (em toda extensão)

e trechos da Nossa Senhora de Fátima/Martins Fontes, entre as ruas Júlia Ferreira de Carvalho e São Sebastião. A Figura 6 mostra uma imagem do Google Earth destácando, em vermelho, as regiões de implementação dos SCT.

Figura 6 – Imagem do Google Earth destácando as regiões de implementação do SCT

As Figuras 7, 8 e 9 mostram detalhadamente cada uma das regiões; respectivamente as avenidas: "da orla", Dona Ana Costa e Nossa Senhora de Fátima.

2.4.1 Empresa BRASCONTROL

Fundada em 1981, a Brascontrol iniciou suas atividades na área de instrumentação industrial, fabricando e comercializando produtos licenciados. Ainda nos anos 80, em parceria com a Petrobras desenvolveu o Primeiro Sistema de Automação de Carregamento

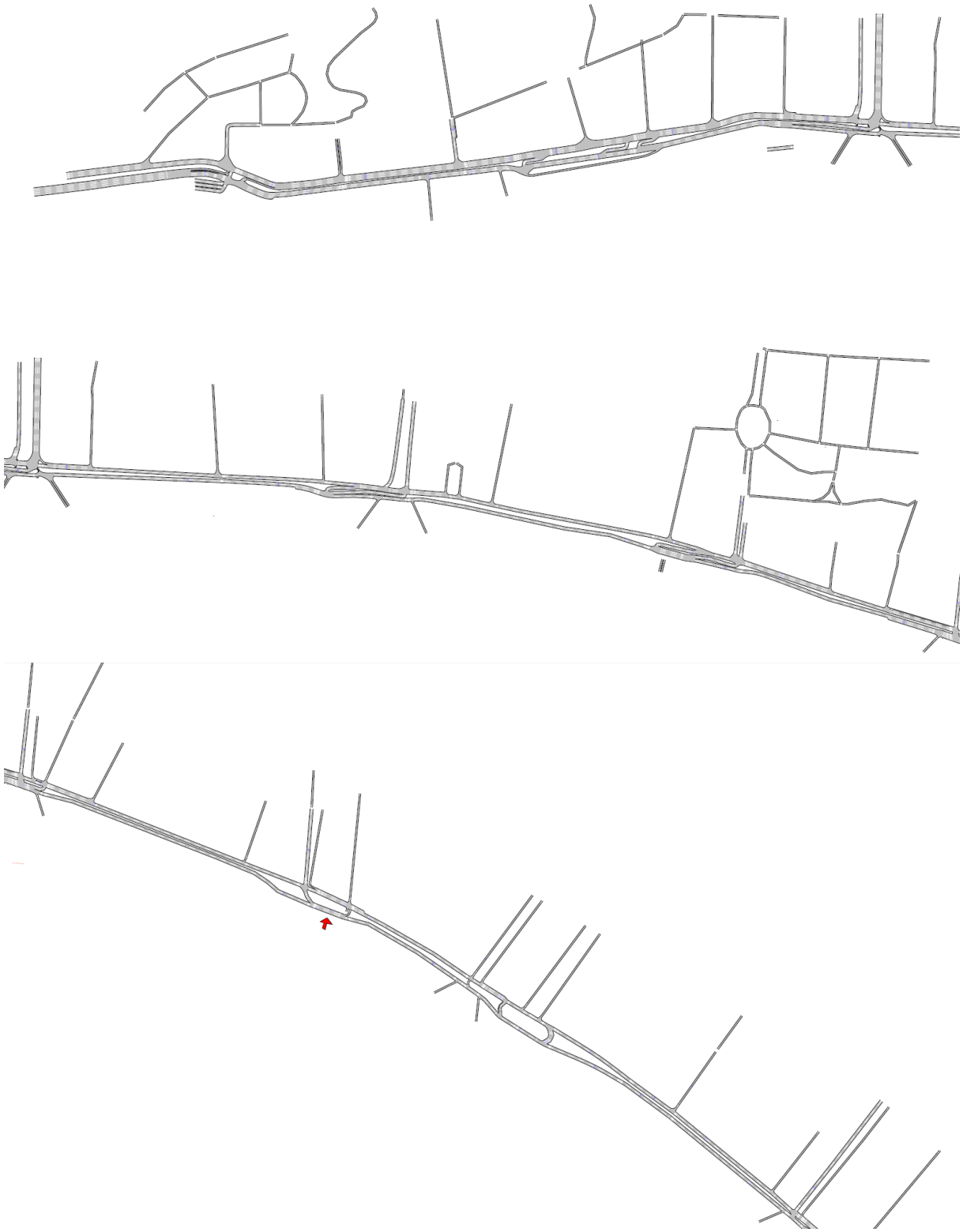


Figura 7 – Detalhe da malha de Santos - Região da Avenida da orla, dividida em 3 secções

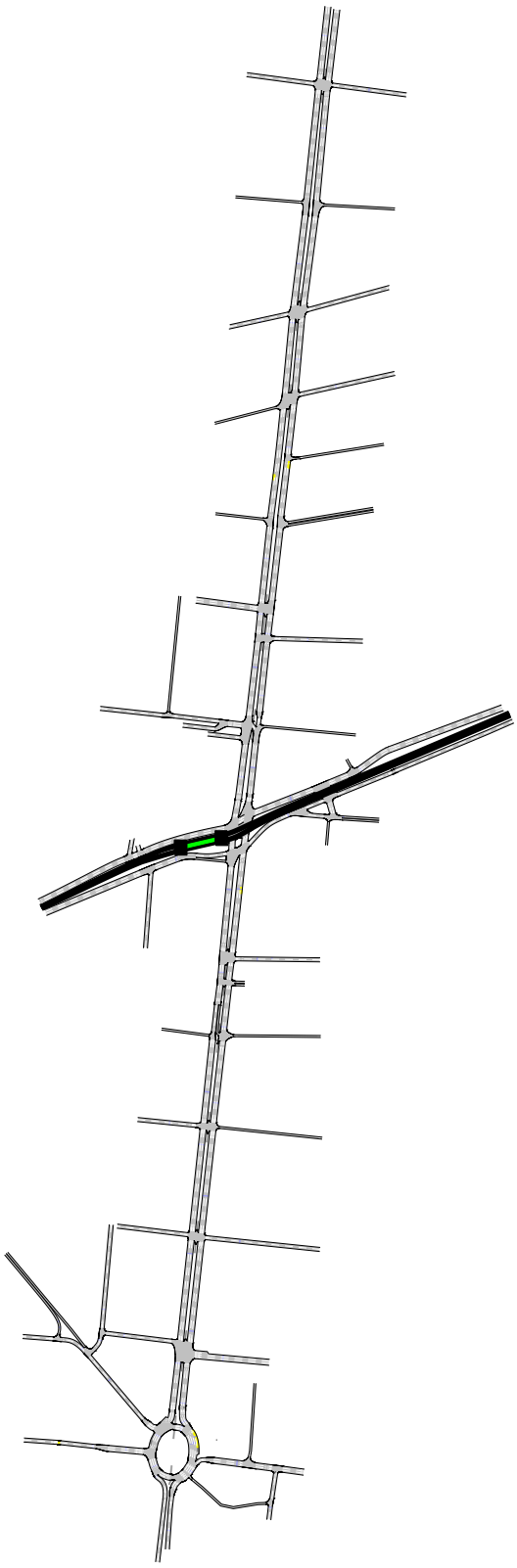


Figura 8 – Detalhe da malha de Santos - Região da Av. Dona Ana Costa

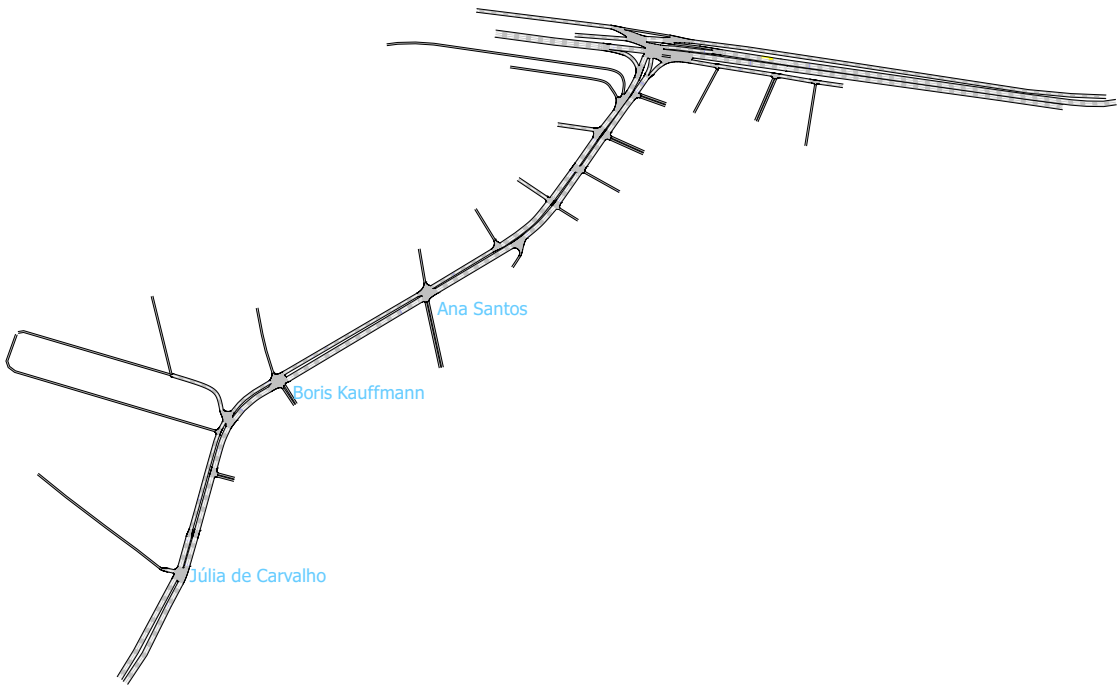


Figura 9 – Detalhe da malha de Santos - Região da Av. Nossa Senhora de Fátima

de Combustível, com tecnologia totalmente nacional, que foi instalado na base da Petrobras de Guarulhos - SP (BAGUAR). Contudo, foi nos anos 90 que a Brascontrol deu o seu grande passo em direção ao futuro. Com a experiência adquirida no desenvolvimento de instrumentação e sistemas, criou a linha de controladores eletrônicos de trânsito urbano e, assim, entrou no segmento de Gerenciamento de Tráfego Urbano. Até hoje este é o principal segmento das atividades da empresa.

Em 2009 a Brascontrol inovou mais uma vez ao lançar a primeira Central de Controle de Trânsito em Tempo Real com tecnologia totalmente nacional. Desta forma, a empresa passou a atuar em um importante segmento para o Brasil: o desenvolvimento de SITs (Sistemas Inteligentes de Transportes).

Atualmente a Brascontrol faz parte de um seleto grupo de empresas que atua no segmento SCT no Brasil. Em 30 anos evoluímos do status de fabricante de equipamentos para a posição de uma das poucas empresas no país capaz de dominar a tecnologia de um conjunto de equipamentos de controle que são interligados na mesma rede de comunicação para realizar gerenciamento e controle de sistemas de transporte como um todo [17].

2.4.2 TUC de controle

O termo *"TUC"* se refere ao Controle Urbano Responsivo ao tráfego, ou em inglês *"Traffic-responsive Urban Control"*, e é a estratégia escolhida pela BRASCONTROL para o SCT de Santos. Esta estratégia de controle é semelhante à implementada em Macaé, no Rio de Janeiro, e foi descrita por Kraus Jr. et al. [18] em seu trabalho. Ela foi desenvolvida para fornecer um controle em tempo real para grandes malhas urbanas, mesmo em SCTuações de saturação de tráfego. Sua principal característica é o controle centralizado, que contrasta com outras estratégias como: SCOOT [19,20], RHODES [21,22] OPAC [21,23] e PRODYN [24].

O TUC utiliza os valores de quantidade de veículos no link, estimadas a partir da taxa de ocupação temporal, para calcular os valores de tempo de ciclo, defasagem entre semáforos e tempo de abertura. Inicialmente os valores de ciclo e defasagem são definidos, e enviados ao modulo responsável pelo cálculo dos tempos de abertura, que por sua vez concilia os valores para obter a estratégia completa de controle para a malha.

Controle de ciclo

O tempo de ciclo é o tempo total que um cruzamento semafórico leva para percorrer todos os seus estágios de aberturas e fechamento. A logica para a definição deste valor é que ciclos maiores reduzem a proporção de tempo perdido, resultante das trocas constantes de estágio, porém podem resultar em desperdício de *"tempo de verde"* e aumento do tempo de espera em vias pouco movimentadas. No caso do TUC em Santos, o cálculo do tempo de ciclo é feito visando limitar o nível máximo de saturação da malha. Um algorítimo com realimentação do estado atual de saturação é utilizado para este cálculo.

Vale ressaltar que para permitir uma coordenação entre cruzamentos, os semáforos de uma mesma rota/região devem possuir um mesmo tempo de ciclo. Portanto, nesse caso, o cálculo é realizado por região.

Controle de defasagem

A defasagem de um semáforo é a diferença entre o momento de início do período de verde, entre dois cruzamentos sucessivos. A principal preocupação, ao se implementar o controle de defasagem, é manter a coordenação entre os cruzamentos de uma rota pré-definida, a qual se deseja dar boa vazão. Essa rota é arbitrariamente selecionada, não precisando respeitar as vias arteriais fisicamente definidas.

O TUC calcula a defasagem ótima baseada nas estimativas de filas presentes em cada link. Considere dois cruzamentos sucessivos j_1 (montante) e j_2 (jusante) e o link z que os conecta. Se o link z não possui nenhum veiculo aguardando em fila, a defasagem ótima entre os cruzamentos é igual ao tempo médio que um veiculo leva de j_1 a j_2. Essa

Situação é conhecida como defasagem positiva, pois a abertura j_1 é anterior a j_2. Por outro lado, uma vez que cresce o número de veículos aguardando no link z, a defasagem ótima diminui. Isso ocorre para permitir a dispersão dos carros aguardando em fila antes que novos cheguem ao cruzamento. Caso haja uma grande quantidade de carros em z, é possível que seja necessário que j_2 anteceda j_1, isto é conhecido como desfasagem negativa.

Controle de tempo de abertura

O objetivo do controle de tempo de abertura é evitar a supersaturação das vias e o bloqueio dos cruzamentos causados pelo fenômeno *"queue spillback"*, aonde os veículos de um cruzamento a montante acabam "bloqueados"pela fila residual do cruzamento a jusante. O controle se baseia em reduzir o tempo de verde de cruzamentos a montante de regiões congestionadas, restringindo assim o fluxo de entrada de veículos nesses links supersaturados.

Antes de serem aplicados, os valores de tempo de abertura são corrigidos para respeitar as restrições de tempos mínimos e máximos de abertura pré-definidos.

2.4.3 Imputação de dados em Santos

Como discutido no capitulo introdutório, quando há perda nos dados de tráfego, um processo de imputação deve ser realizado. No caso do SCT de Santos, a imputação ocorre internamente, imputando os valores (estimados) de número de veículos x. Em contraste, em nossa pesquisa as imputações ocorrem sobre os valores de taxa de ocupação temporal *occt*. Está escolha foi feita para que fosse possível testar o processo de imputação sem alterações no algoritmo TUC, realizando a imputação como um pré-processamento da variável de entrada *occt*.

Ao longo deste trabalho, o processo de imputação interno do TUC será chamado de "Imputação do TUC" ou TUC_{imp}. Este processo é o mesmo descrito por Diakaki [4], aonde o valor calculado é proporcional à média de veículos em links a montante. Abaixo vemos a equação que descreve o cálculo, em que x_z representa o valor a ser imputado para o link z, N o número de links a montante de z (que possuem detectores) e y_i representa o número de veículos no link i expresso proporcional ao tamanho do link z, ou seja: $y_i = x_{z,max} \cdot x_i / x_{i,max}$.

$$x_z = \frac{W_z}{N} \sum_{i=1}^{N} y_i \tag{2.4}$$

Na Equação 2.4, W_z representa o peso proporcional que, historicamente, x_z tem em relação à média histórica das N estimativas. Ele é descrito pela Equação 2.5, aonde

xh_z representa a média histórica de x_z assim como yh_i representa a media histórica de y_i.

$$W_z = \frac{1}{Nxh_z} \sum_{i=1}^{N} yh_i \tag{2.5}$$

Para os casos aonde não existem dados disponíveis de detectores a montante, o valor é estimado como nulo.

2.4.4 Dados disponíveis

Existem dois grupos de dados, oriundos da malha de Santos, e que foram utilizados neste trabalho:

1. **Medições reais** - Os dados foram coletados em 10 quintas-feiras, nos meses de Setembro e Outubro de 2016. O banco de dados possui as medições diretas dos laços indutivos, como apresentado na Figura 2, e informações espaciais como localização, link, estáção, etc...

   O código de PostgreSQL no Anexo B foi utilizado para criar as tabelas contendo todas as taxas de ocupação k desejadas para todos os sensores da malha de Santos. O parâmetro principal do algoritmo é o período de agregação Ta, mas também é possível limitar as buscas por períodos de interesse , por link/estação, ou mesmo por detector.

   Os dados utilizados no trabalho foram agrupados por estação/Link e os períodos de agregação Ta foram de 150, 300 e 900 segundos.

2. **Dados Simulados** - O segundo banco de dados de Santos é oriundo das simulações da malha, realizadas através do AIMSUM. Neste caso, os dados foram repassados já agrupados em estações, e agregados com o Ta correspondente a cada atualização do controle (ciclo dos semáforos da região).

2.5 Malha de Chaniá, Grécia

Apesar da malha de Santos ser a origem de quase todos os dados utilizados nesta pesquisa, o modelo da malha urbana de Chaniá, Grécia, apresentado na Figura 10 também foi utilizado. Os dados foram obtidos através de simulação, utilizando o software AIMSUM. Vale ressaltar que as simulações não fazem parte deste trabalho, sendo que os dados foram obtidos sem nenhuma interação direta com a simulação.

Os dados fornecidos são de taxa de ocupação com período de agregação fixo de 90 segundos. Estão disponíveis 40 replicações de 4 horas cada, sendo que as 2 ultimas são

consideradas de "descarga", ou seja não são consideradas novas "entradas" de veículos na malha.

Os dados foram coletados com a aplicação de 4 estratégias diferentes de controle: *NPC, PC, PCDNB, PCQNB*. As estratégias utilizadas são irrelevantes para a discussão e por isso, os detalhes serão omitidos deste trabalho.

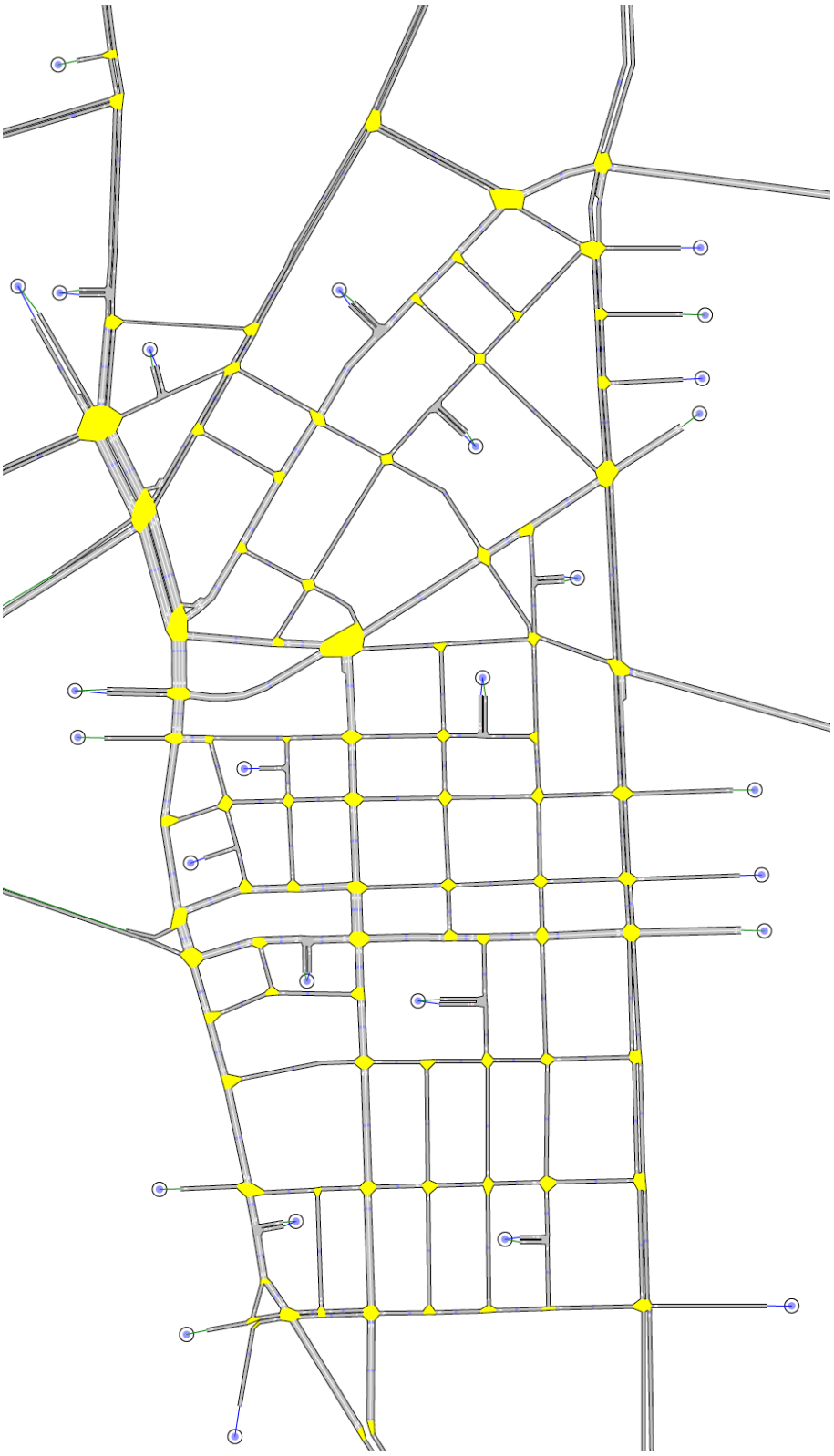


Figura 10 – Malha urbana de Chaniá, Grécia

3 Imputação de dados de tráfego

A imputação de dados se refere a técnica de preenchimento de dados perdidos, ou substituição de dados corrompidos, com estimativas ou palpites. Em se tratando de tráfego, devido a complexidade dos padrões de dados, e a diversidade de cenários e aplicações possíveis, inúmeras pesquisas já foram realizadas, utilizando uma variedade de métodos diferentes para a imputação [12, 25–28].

Neste **capítulo** será apresentado uma visão geral do que a literatura oferece como alternativas para a imputação de dados de tráfego, mostrando uma breve explicação para as técnicas mais relevantes, e por fim explicaremos os motivos que levaram a escolha da técnica **MVS-MQ** para está pesquisa.

3.1 Visão geral

As técnicas mais simples de imputação são as que utilizam as médias dos dados históricos ou de sensores próximos para estimar os valores perdidos. Por exemplo: O modelo histórico preenche os dados perdidos com a média histórica daquele sensor, considerando um mesmo local, horário e dia da semana [29]. É possível também utilizar os dados dos sensores vizinhos realizando a imputação com a média ou a média ponderada dos dados coletados próximos ao local da falha [10,30–32]. Esta é a técnica atualmente implementada no SCT de Santos, que utiliza um valor proporcional à média das medições dos sensores vizinhos.

O filtro de Kalman [33], as séries temporais do modelo auto-regressivo integrado de média móvel - ARIMA [34], e o método de distribuição de vias [35] são alguns exemplos de técnicas mais sofisticadas, baseadas em modelos matemáticos de interpolação, considerando dados tanto espaciais (sensores vizinhos) quanto temporais (dados históricos).

Entretanto, apesar de úteis para outras aplicações, as técnicas citadas anteriormente não são ideias para a imputação de dados de tráfego. *"Predições de curto prazo* [como é a caracteristica dos dados de tráfego] *são mais influenciadas por fatores estocásticos do que predições de longo prazo, com incertezas maiores e leis de comportamento menos claras. Portanto a utilização de modelos matemáticos clássicos para a predição de curto prazo não é capaz de atender satisfatoriamente as demandas de tempo real do controle de tráfego realizado pelos SCTs"* [36].

Por isso, as pesquisas têm migrado para os métodos estatísticos para solucionar o problema da perda de dados de tráfego. Desse grupo, os métodos regressivos e que utilizam múltipla imputação (MI) têm demonstrado ser mais eficientes e com maior precisão do que

outros métodos convencionais de imputação [10, 30, 37, 38]. Diferente de outros modelos de imputação, estes modelos não-paramétricos baseado em dados podem fornecer certas vantagens sobre a concorrência [12]

Um exemplo clássico é a múltipla imputação (MI) de Monte Carlo por cadeias de Markov - MCMC [39–42]. A ideia básica do MCMC é tratar os dados perdidos como parâmetros de interesse, e estima-los colhendo uma serie de amostra para esses parâmetros. Por se tratar de uma técnica de MI, o resultado deste algoritmo é a combinação de múltiplas respostas ao invés de apenas uma.

Outro método bastante eficiente é a maximização de expectativa (ME) [43, 44] combinada com a *Data Agumentation* (DA) [43], aonde a DA nada mais é que um caso especial de MCMC. Neste caso, o algorítimo EM realiza interações e, através da taxa de erros, determina o comportamento de convergência. Em seguida o DA cria cadeias de Markov que convirjam para a combinação entre a distribuição dos dados perdidos e dos parâmetros estimados pelo EM [12].

Alem dos métodos regressivos, outra tipo de método estatístico, que tem ganhado destaque, são os "Algorítimos Aprendizado". **Métodos estatísticos de aprendizado** utilizam os dados históricos para aprender sobre o sistema, e então imputar os dados perdidos de forma iterativa [5, 45–47]. Alem do próprio MVS, que será discutido a fundo no **capítulo 4**, um tipico representante são as Redes Neurais [38, 48, 49] No caso de imputação de dados de tráfego, o modelo mais comum para a Rede Neural é o de retro propagação (Back-propagation Neural Network - BPNN).

3.1.1 Redes Neurais retro-propagadas

O metódo **BPNN** utiliza-se do algoritmo de gradientes decrescentes e regulação ponderada para minimizar sua função objetivo. No entanto, a escolha da função objetivo é um problema para a imputação de dados de tráfego. Ao utilizar o principio de minimização de risco empírico (ERM), a rede neural parece não ser capaz de encontrar uma solução. É possível utilizar a função objetivo como o quadrado da soma da margem entre o valor de entrada e saída, porem está solução da enfase excessiva nos erros, o que costuma causar problemas de sobre-ajuste (*overfitting*) [36]. Portanto a habilidade geral do modelo é limitada, pois as redes neurais enfrentam varias dificuldades como problemas de mínimos locais, e identificação de estrutura de rede, entre outros.

3.2 Escolha de MVS-MQ

Dentre as diversas técnicas presentes na literatura a **Maquina de Vetores de Suporte** (MVS), demonstra grande potencial. Inicialmente proposta por Vapnik. [50] para problemas de reconhecimento de padrões e classificação, ela foi posteriormente adaptada

para a resolução de problemas de regressão não linear. A técnica já foi aplicada com sucesso para diferentes propósitos, incluindo: Classificação, estimação de serie, funções de aproximação, reconhecimento de texto, etc...

O método MVS possui similaridades com as redes neurais "pré-alimentadas"(*feed-foward*). As duas são conhecidas como técnicas "semi-paramétricas", pois são eficientes nos treinamentos como as técnicas paramétricas, mas também tem a habilidade de aprender dependências não-lineares, como é o caso de métodos não-paramétricos.

Entre a propriedades do método se destaca a convergência universal. A técnica utiliza o principio da minimização de risco estrutural (SRM) ao invés da minimização de risco empírica (ERM). Sendo assim, a função objetiva a ser minimizada é convexa e quadrática garantindo que a MVS possua somente uma solução global. Ainda, a MVS se adapta bem a sistemas complexos e é robusta ao lidar com dados corrompidos [51].

Apesar das vantagens aparentes, a técnica de regressão utilizando MVS, conhecida como Regressão por Vetores de Suporte (RVS), inicialmente não obteve grande sucesso na imputação de dados de tráfego, pois exigia uma grande carga computacional. Foi somente quando Suykens et al. [13] propuseram uma reformulação, utilizando como função de maximização os mínimos quadrados, foi que o método começou a ser reconhecido como promissor também para a imputação de dados de tráfego. Essa nova versão é conhecida como Maquina de Vetores de Suporte por Mínimos Quadrados (MVS-MQ), e possui vantagens como rápida convergência, alta precisão e baixa carga computacional [52]. O método tem performance superior em muitas pesquisas sobre tráfego [30, 53] e já foi utilizada em diversas outras áreas, como: Concentração de CO, gás, carga elétrica, renda, precipitações, velocidade do vento, predição de consumo hidroelétrico, etc [54].

Segundo Zhang e Liu [12], a MVS-MQ é mais aplicável e possui mais estabilidade e adaptabilidade do que a técnica de Maximização de Expectativa (ME) . Su, Zhang e Yu [36] apontam que o MVS-MQ, em particular sua versão incremental, tem resultados superiores se comparado a técnica de Rede Neural de retro-propagação. Conclusão semelhante a de Zhao-sheng, Yuan e Qing [55], que afirmam que o MVS é superior às redes neurais em precisão, tempo de convergência, habilidade de generalização e possibilidades de otimização. Por isto a imputação de dados através da Maquina de Vetores de Suporte por Mínimos Quadrados foi escolhida como técnica para este estudo e candidata para substituir o atual algoritmo de imputação do SCT de Santos.

4 Máquina de Vetores de Suporte

Criada por Vapnik. [50], a máquina de vetores de suporte (MVS) é um algoritmo de aprendizagem, inicialmente desenvolvido para problemas de classificação binaria linear. Em 1992, Boser, Guyon e Vapnik [56] adaptaram o método ao incorporar o truque de Kernel [57] que é capaz de criar um hiperplano em um "espaço de recursos dimensionais superior". Esta adaptação possibilitou que o método fosse utilizado também para classificações não lineares. A MVS é baseado em conceitos simples, mas muito poderosos na resolução de problemas não lineares. Os classificadores não-lineares por MVS são utilizados com sucesso, por exemplo, para identificação de escrita manuscrita [58].

Substituindo a função objetivo de maximização de margem pela função de perda ϵ-*insensitiva*, sugerida por Drucker et al. [59], é possível utilizar o algorítimo também para a resolução de problemas de regressão não-linear. Essa técnica, utilizando a função ϵ-*insensitiva*, é conhecida como **Regressão por Vetores de Suporte** (RVS).

Os dois métodos citados, MVS e RVS, necessitam de Programação Quadrática (PQ) e uma solução com um grupo de equações não-lineares para realizar a otimização. Apesar de existirem métodos eficientes para fazer isso, dependendo do tamanho do grupo de treinamento e a quantidade de dimensões do vetor de entrada, esse problema pode se tornar computacional muito custoso.

Pensando nisso, Suykens et al. [13] propôs uma reformulação no método RVS, que altera a função de custo e substitui as restrições de desigualdade por restrições de igualdade. Essa alteração acaba por transformar o problema em uma otimização de um Sistema Linear, ao invés do problema de Programação Quadrática, o que reduz bastante o custo computacional. Esse novo método é chamado de **Máquina de Vetores de Suporte por Mínimos Quadrados**, e pode ser resolvido de forma eficiente pelo método iterativo de Gradientes Conjugados [60], ou pelo método de Otimização Minima Sequencial (SMO) [61, 62].

Neste **capitulo** serão apresentados os três métodos relacionados à **Máquina de Vetores de Suporte**, mostrando seus principais conceitos. As explicações foram baseadas nos trabalhos de Mallinson e Gammerman [63], Ng [64], Wang e Hu [65], Zhang e Liu [12] e Su, Zhang e Yu [36].

4.1 Máquina de Vetores de Suporte Clássica

O método clássico de MVS se refere ao classificador binário não-linear. A **MVS** tem um conceito bem simples: separar as duas classes de dados, através de um hiperplano

que crie o maior "vão" possível entre elas. Isso significa que o modelo treinado nada mais é que um vetor separando essas duas classes.

4.1.1 Problema linearmente classificável

Imagine um grupo de dados $\{Vx_1, Vy_1\}$ ... $\{Vx_n, Vy_n\}$, com o qual se deseja treinar o algoritmo MVS para realização da classificação binaria. O vetor Vx representa a entrada da função de classificação, e Vy representa a saída binaria.

A partir dos dados de treinamento, o algorítimo do MVS ira encontrar um hiperplano adequado. A Figura 11 mostra um exemplo de um problema linearmente classificável. Considerando um grupo de dados de treinamento, a figura destaca 3 candidatos para hiperplano H_1, H_2 e H_3. Nesse exemplo o vetor Vx tem duas dimensões e é represento pela posição dos círculos no plano, e o valor binário Vy é representada pelas cores dos círculos (preto $Vy = -1$ ou branca $Vy = 1$).

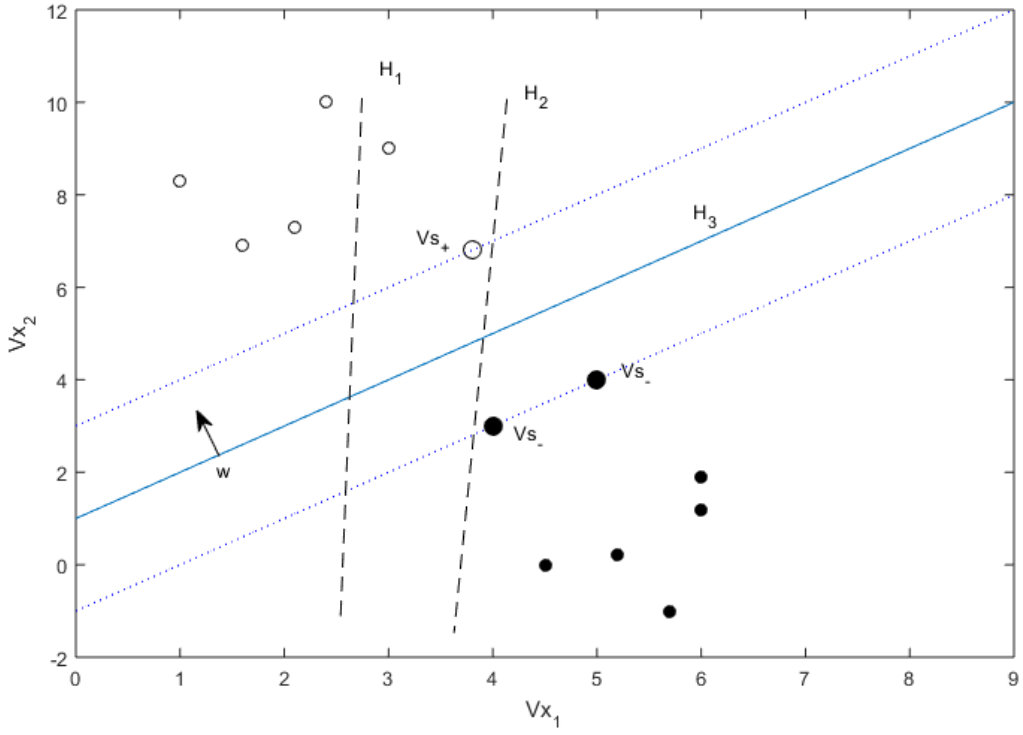


Figura 11 – Exemplo de classificação utilizando hiperplanos

Dos hiperplanos candidatos, presentes na Figura 11, vemos que H_1 não é capaz de separar as duas categorias corretamente, e que H_2, apesar de conseguir realizar a separação, não garante o maior margem entre as classes. É somente H_3 o hiperplano que separa as duas categorias garantindo a maior distancia entre as duas classes de vetores.

Mais especificamente, H_3 se encontra exatamente no centro do "espaço" entre as classes, estando assim equidistante dos vetores mais próximos ao hiperplano: Vs_- (da classe

com Vy negativo) e Vs_+ (da classe com Vy positivo). Esses vetores, os mais próximos da fronteira que separa as classes, são conhecidos como **Vetores de Suporte**, e dão o nome ao método.

Maximização das margens

Mais formalmente, o hiperplano da **MVS** pode ser descrito como um conjunto de vetores $\vec{Vx}$ que satisfaçam Equação 4.1, aonde $\vec{w}$ é o vetor normal ao hiperplano e b é o valor de bias. Sendo que a função de classificação é então descrita pela Equação 4.2.

$$\vec{w} \cdot \vec{Vx} + b = 0 \tag{4.1}$$

$$Vy = sign(\vec{w} \cdot \vec{Vx} + b) \tag{4.2}$$

Uma vez que os **Vetores de Suporte** delimitam o "vão"entre as classes, é possível afirmar que a Equação 4.3 define a largura da margem ρ. A Figura 12 mostra uma representação gráfica dessa equação.

$$\rho = (Vs_+ - Vs_-) \cdot \frac{\vec{w}}{||w||} \tag{4.3}$$

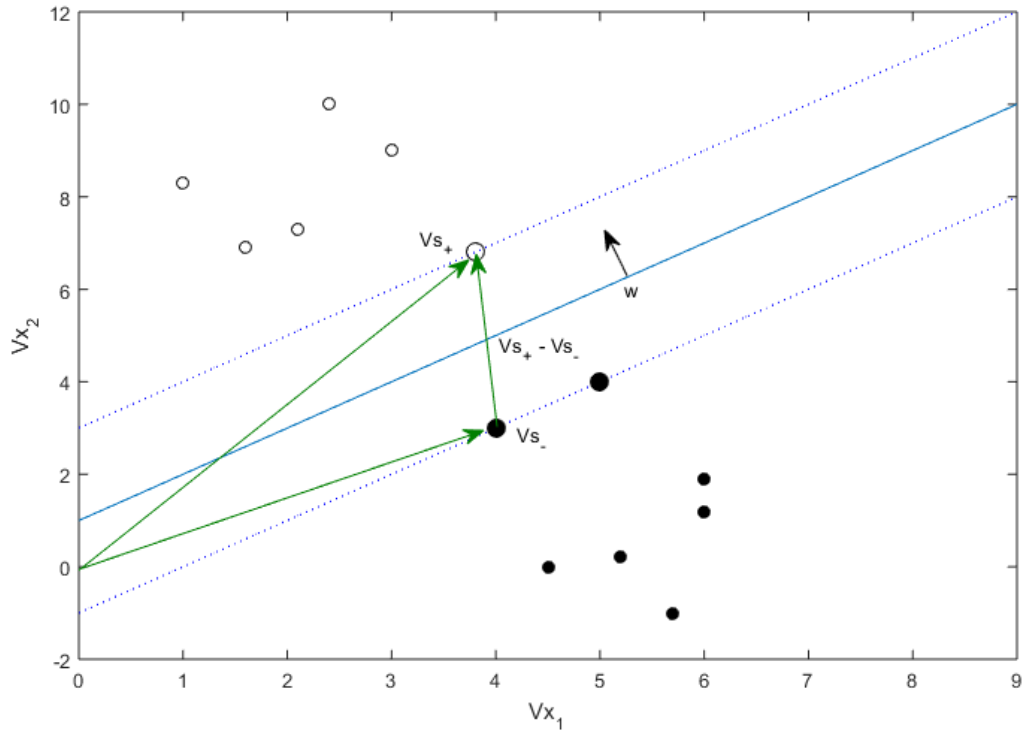


Figura 12 – Diferença entre os vetores Vx_+ e Vx_-

Para encontrar os valores de b e $\vec{w}$, uma restrição é necessária. Os vetores de suporte Vs_+ e Vs_- passam a respeitar as equações 4.4 e 4.5, respectivamente.

$$\vec{w} \cdot \vec{Vs_+} + b = 1 \tag{4.4}$$

$$\vec{w} \cdot \vec{Vs_-} + b = -1 \tag{4.5}$$

Por consequência das condições impostas nas equações 4.4 e 4.5, e sabendo que os Vetores de Suporte são os mais próximos do hiperplano, podemos afirmar que todos os dados de treinamento respeitam a condição:

$$Vy\ (\vec{w} \cdot \vec{Vx} + b) \geqslant 1 \tag{4.6}$$

Ainda, aplicando as equações 4.4 e 4.5 a equação 4.3 temos que:

$$\rho = \frac{2}{||\vec{w}||} \tag{4.7}$$

Portanto, para a maximização de ρ, basta que $||\vec{w}||$ seja minimizado, ou de forma equivalente, que $\frac{1}{2}||w||^2$ seja minimizado, respeitando a condição 4.6. Dessa forma, temos um problema de maximização **quadrático e convexo**, e com **restrições lineares**.

Multiplicadores de Lagrange

Para solução do problema de otimização é necessário a utilização dos multiplicadores de Lagrange, descritos em sua forma genérica pela Equação 4.8.

$$\begin{aligned} & \mathcal{L}(\omega, \alpha, \beta) = f(\omega) + \sum_{i=1}^{k} \alpha_i g_i(\omega) + \sum_{i=1}^{l} \beta_i h_i(\omega) \\ \text{sujeito a} \quad & g_i(\omega) \leqslant 0 \\ & h_i(\omega) = 0 \end{aligned} \tag{4.8}$$

Aonde $f(w)$ é função a ser otimizada, $\alpha_i ... \alpha_n$ e $\beta_i ... \beta_n$ são os multiplicadores de Lagrange, e $g_i(\omega)$ e $h_i(\omega)$ são restrições de desigualdades e igualdades, respectivamente.

O problema de otimização da MVS clássica pode ser descrito pela Equação 4.9.

$$\begin{aligned} \min \quad & f(w) = \frac{1}{2}||w||^2 \\ \text{sujeito a} \quad & g_i(w) = 1 - Vy\ (\vec{w} \cdot \vec{Vx} + b) \leqslant 0 \end{aligned} \tag{4.9}$$

O modelo da MVS não possui condições de igualdade, portanto o termo $h_i(w)$ é nulo, e o multiplicador β desaparece da equação 4.8.

Substituindo os valores da equação 4.9. na equação 4.8, temos:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{n} \alpha_i \ (\ Vy^{(i)} \ (\vec{w} \cdot V\vec{x}^{(i)} + b) - 1) \tag{4.10}$$

Pelo método dos multiplicadores de Lagrange, sabemos que a otimização ocorre quando as derivadas parciais de 4.10 são iguais a 0. Portanto, derivando a função relativo a w e b e igualando a 0 obtemos:

$$\frac{\partial \mathcal{L}}{\partial \vec{w}} : \vec{w} = \sum_{i=1}^{n} \alpha_i \ Vy^{(i)} \ \vec{Vx}^{(i)} \tag{4.11}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{b}} : \sum_{i=1}^{n} \alpha_i \ Vy^{(i)} = 0 \tag{4.12}$$

Condições de Karush-Kuhn-Tucker

Problemas convexos e com restrições lineares, como é o caso da MVS, respeitam as condições de **Karush-Kuhn-Tucker** (KKT) [66]. Sendo assim, essas condições são necessárias e suficientes para que $\vec{w}, b$ e α sejam soluções da otimização [67]. Dentre as condições de KKT, a mais relevante é a condição complementar de dualidade descrita como:

$$\alpha_i \ g_i(w) = 0 \tag{4.13}$$

Sabemos que o valor $g_i(w)$ é nulo somente quando o dado de treinamento $\vec{V_i}$ for um dos **Vetores de Suporte** $\vec{V}s$, ou seja, dados que respeitem a restrição da equação 4.4 ou da equação 4.5. De forma análoga, caso $g_i(w)$ não seja nulo, o dado de treinamento não é um **Vetor de Suporte**.

A equação 4.13 implica que somente os Vetores de Suporte possuam α_i diferente de 0. Consequentemente a a solução da MVS é dita esparsa, pois são somente os Vetores de Suporte, e não todo o conjunto de dados de treinamento, os responsáveis por descrever o hiperplano.

Ao juntarmos a equação 4.10 com as restrições 4.13, 4.11 e 4.12 obtemos o seguinte

problema de otimização duplo de Lagrange:

$$\begin{aligned} \max \alpha \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} Vy^{(i)} Vy^{(j)} \ \alpha_i \alpha_j \ \langle V\vec{x}^{(i)} \cdot V\vec{x}^{(j)} \rangle \\ \text{Sujeito a} \quad & \sum_{i=1}^{n} \alpha_i \ Vy^{(i)} = 0 \\ & \alpha_i \ g_i(w) = 0 \\ & \alpha_i \geqslant 0 \end{aligned} \tag{4.14}$$

Até o momento, nosso problema antes expresso pela equação 4.6, apenas foi alterado para um problema de otimização com restrições mais palpáveis. Para encontrar a solução final ainda será necessário a utilização de métodos numéricos Tais métodos não serão explorados no escopo deste trabalho, porem problemas desse tipo são bem conhecidos e existem métodos eficientes para a sua solução [50, 61].

Uma vez encontrado os α_is, é possível determinar o valor ótimo do bias b, utilizando a equação 4.13, com um i referente a um dos Vetores de Suporte ($\alpha_i \neq 0$). Como mostra as expressões abaixo:

$$b = 1 - \langle \vec{w} \cdot V\vec{s}_+ \rangle \tag{4.15}$$

ou de forma semelhante:

$$b = -1 - \langle \vec{w} \cdot V\vec{s}_- \rangle \tag{4.16}$$

Vale ressaltar que é numericamente mais seguro considerar o valor médio de b para todos os Vetores de suporte. portanto é comum utilizar a equação 4.17 para encontrar o valor de b:

$$b = -\frac{\langle \vec{w} \cdot V\vec{s}_+ \rangle + \langle \vec{w} \cdot V\vec{s}_- \rangle}{2} \tag{4.17}$$

Ainda, utilizando a equação 4.11 é possível determinar o $\vec{w}$ ótimo. Entretanto, o modelo MVS final, responsável pela classificação, é mais comumente expresso em função do b e dos α_is, como mostra a equação a seguir:

$$Vy = sign\Big(\sum_{i=1}^{n} \alpha_i \ Vy^{(i)} \langle V\vec{x}^{\,(i)} \cdot V\vec{x} \rangle + b \Big) \tag{4.18}$$

Portanto, para encontrar a predição do valor de Vy basta calcular os valores do produto interno entre Vx e os dados de treinamento. Ainda, como os α_is são diferentes de 0 somente para os Vetores de Suporte, é necessário calcular o produto interno somente para eles, reduzindo assim a carga computacional do algoritmo final.

4.1.2 Casos não-lineares

A classificação de casos não lineares pode ser alcançada ao se projetar os dados para um **espaço de recursos dimensionais superiores** (higher dimensional feature space) antes do treinamento. Na Figura 13 podemos ver um exemplo de um conjunto de dados que não é linearmente classificável.

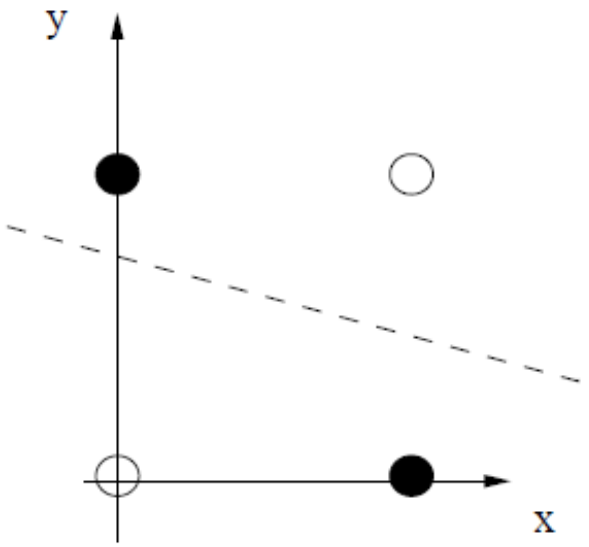


Figura 13 – Exemplo de dados que não são linearmente classificáveis (Mallinson e Gammerman [63], 2003).

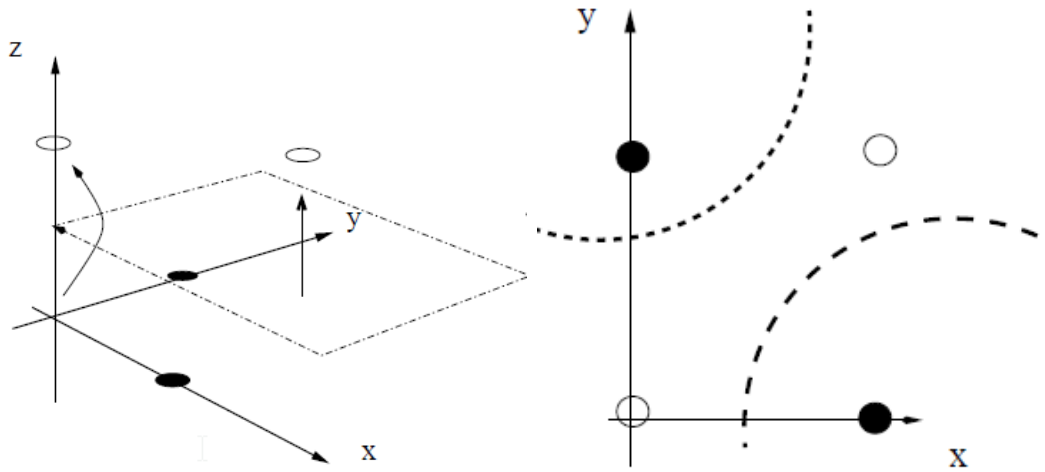
É possível projetar os dados de treinamento bidimensionais (do exemplo anterior), para um espaço de 3 dimensões. Como termologia, chamaremos os valores de entrada das dimensões originais como "**atributos**", e os valores referentes as novas dimensões adicionadas de "**recursos**".

Nesse exemplo, a função de projeção $\phi(.)$ escolhida é: $Vx_3 = 1 + 2Vx_1Vx_2 - Vx_1 - Vx_2$. Desse modo é possível realizar o treinamento da MVS utilizando também os valores dos recursos adicionados.

A Figura 14a mostra os dados projetados no espaço de recursos dimensionais superiores, assim como o hiperplano criado pelo treinamento da MVS. Já a figura 14b mostra esse mesmo hiperplano reprojetado para o espaço bidimensional original.

Truque de Kernel

Para adaptarmos nosso algorítimo ao espaço de recursos dimensionais superiores basta substituir $\vec{Vx}$ por $\phi(\vec{Vx})$ nas equações da MVS. Entretanto, apesar de fácil adaptação, isso obrigaria ao calculo explicito da projeção de todos os Vetores de Treinamento e Vetores de entrada, o que pode ser computacionalmente caro, em especial para projeções que adicionam muitos recursos (dimensões).



(a) Espaço dimensional superior

(b) Espaço dimensional original re-projetado

Figura 14 – Projeção dos dados para um espaço de recursos dimensionais superiores (Mallinson e Gammerman [63], 2003).

Aizerman, Braverman e Rozonoer [57] provaram que uma função **Kernel**, que seja simétrica semi-definida positiva, tem resultado idêntico ao produto interno em um certo espaço de recursos dimensionais.

Como vimos anteriormente, o algorítimo de treinamento e de estimação da MVS dependem apenas dos valores dos produtos internos entre os vetores de entrada: $\langle \vec{Vx}^{(i)} \cdot \vec{Vx}^{(j)} \rangle$. Logo o calculo explicito das projeções não é necessário, basta selecionar a função **Kernel** correspondente a projeção e que respeite o **"Teorema de Mercer"**.

A função **Kernel**, para uma dada projeção $\phi(.)$, é definida como:

$$K(\vec{Vx}^{(i)}, \vec{Vx}^{(j)}) = \langle \phi(\vec{Vx}^{(i)}) \cdot \phi(\vec{Vx}^{(j)}) \rangle \tag{4.19}$$

Portanto, a equação final que descreve o classificador pode ser alterada para:

$$Vy = sign\Big(\sum_{i=1}^{n} \alpha_i \; Vy^{(i)} K(\vec{Vx}^{(i)}, \vec{Vx}) + b \Big) \tag{4.20}$$

Na pratica, para a utilização da MVS, o que é secionado é a função Kernel, sem se preocupar diretamente com a projeção $\phi(.)$. Existem varias funções que satisfazem as condições de Mercer, e correspondem a uma projeção de recursos. Alguns exemplos são:

Kernel Linear:

$$K_l(\vec{Vx}^{(i)}, \vec{Vx}^{(j)}) = \langle \vec{Vx}^{(i)} \cdot \vec{Vx}^{(j)} \rangle \tag{4.21}$$

Kernel Polinomial:

$$K_p(\vec{Vx}^{(i)}, \vec{Vx}^{(j)}) = (\langle \vec{Vx}^{(i)} \cdot \vec{Vx}^{(j)} \rangle + \theta)^r \tag{4.22}$$

Onde r é um parâmetro que define a quantidade de **recursos** que serão adicionados a $\vec{Vx}$. Enquanto que θ influencia na complexidade da transformação, aumentando ou diminuindo a influencia relativa entre os termos de baixa e alta ordem do polionimia

Kernel Gaussiano de Função de Base Radial (RBF):

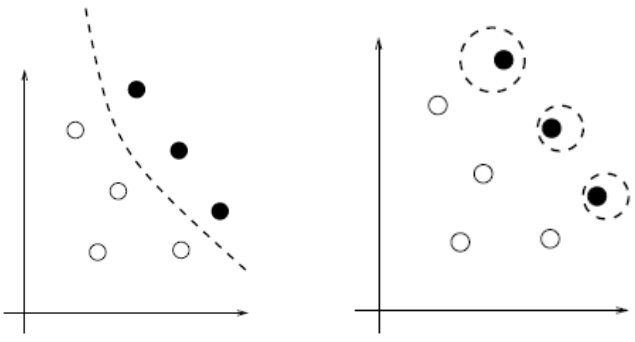
$$K_{rbf}(\vec{Vx}^{(i)}, \vec{Vx}^{(j)}) = exp\left(- \frac{||\vec{Vx}^{(i)} - \vec{Vx}^{(j)}||^2}{2\sigma^2} \right) \tag{4.23}$$

O Kernel Gaussiano pode ser interpretado como uma medida de similaridade entre dois Vetores, uma vez que o valor computado do Kernel cresce a medida que os vetores se tornam mais parecidos. No caso da MVS essa comparação ocorre entre os **Vetores de Suporte** e os Vetores de entrada da imputação.

O Kernel RBF corresponde a uma projeção $\phi(.)$ de dimensões virtualmente infinitas. Por esse motivo, esse Kernel produz um aproximador universal para MVS. Vale ressaltar que, devido as dimensões "infinitas", não é possível calcular explicitamente as projeções $\phi_{rbf}(\vec{Vx}^{(i)})$.

O termo σ que também aparece na equação 4.23 é um parâmetro do usuário, e está relacionado a "largura"da distribuição gaussiana, ou mais formalmente, ao desvio padrão. Em termos práticos, a magnitude do termo σ influencia na complexidade das fronteiras de decisão, sendo que conforme σ tende a 0, o classificador se torna mais preciso, porem pode sofrer com problemas de sobre ajuste

A figura 15 mostra como as fronteiras do modelo MVS podem ser prejudicadas pelo sobre ajuste da função



(a) Modelo normal (b) Modelo com sobreajuste

Figura 15 – Diferença de modelos da MVS sem e com sobre ajuste (Mallinson e Gammerman [63], 2003)

Nesse trabalho utilizaremos o Kernel RBF pela sua boa performance e vantagens na predição de series temporais provadas por pesquisas anteriores [12].

4.2 Regressão por Vetores de Suporte

Posteriormente a introdução do método de classificação, a **MVS** foi estendida para a realização também de regressões não-lineares, resultando assim no nome do novo método: **Regressão por Vetores de Suporte** (RVS).

A definição do problema é semelhante a MVS (classificação), aonde existe um grupo de dados $\{Vx_1, Vy_1\}$ ... $\{Vx_n, Vy_n\}$, com o qual se deseja treinar o algoritmo de RVS. O vetor Vx representa a entrada da função e pode ser multivariável, e Vy representa a saída, porem $Vy \in \mathbb{R}$, diferente do que ocorre na MVS aonde ele é binário.

O objetivo da RVS é encontrar uma função que estime o valor de Vy, dado uma condição de entrada Vx e seguindo a mesma distribuição dos dados de treinamento. A definição da função de estimação é semelhante a 4.2, e é descrita como:

$$Vy = \langle \vec{w} \cdot \vec{Vx} \rangle + b \tag{4.24}$$

A diferença entre os métodos de classificação e regressão está na função de otimização. Enquanto que no método de classificação se deseja maximizar a margem do hiperplano $\vec{w}$, na RVS o objetivo é minimizar alguma estimativa de erro entre os dados de treinamento e o hiperplano $\vec{w}$.

Para o calculo do erro, a RVS utiliza uma função de perda com tolerância para erros pequenos (ϵ). São definidas duas variáveis de saída (ξ_i^* e ξ_i) para está função e que representam o valor do grau da penalização. ξ_i representa a penalização dos pontos acima do tubo ϵ de insensibilidade, e ξ_i^* a penalização dos pontos abaixo. A Figura 16 mostra uma representação do tubo ϵ de insensibilidade e da função de perda.

A função de ϵ-insensitiva, que define os valores ξ_i^* e ξ_i, é descrita como:

$$\begin{aligned} \xi_i &= \begin{cases} 0, & \text{if } Vy - \hat{Vy} \leqslant \epsilon, \\ |Vy - \hat{Vy}| - \epsilon, & \text{else} \end{cases} \\ \xi_i^* &= \begin{cases} 0, & \text{if } Vy - \hat{Vy} \geqslant -\epsilon, \\ |Vy - \hat{Vy}| - \epsilon, & \text{else} \end{cases} \end{aligned} \tag{4.25}$$

Assim como a solução da MVS, a regressão gera uma solução esparsa. Neste caso, os Vetores de Suporte são somente aqueles que se encontram fora da zona de ϵ-insensibilidade.

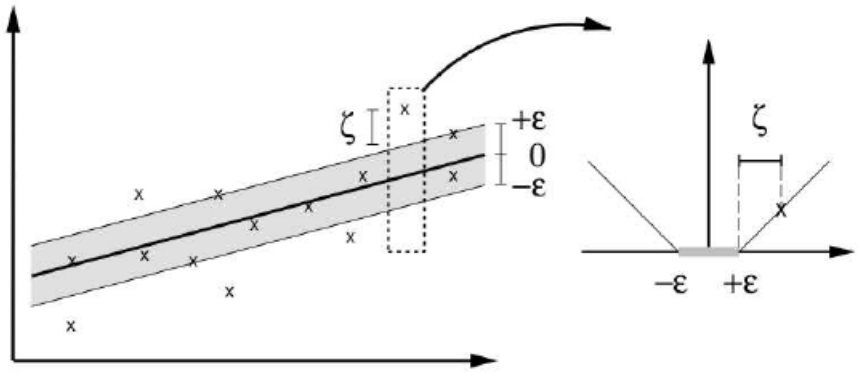


Figura 16 – Representação da função ϵ-insensitiva (Wang e Hu [65], 2005)

Com base na função de perda de ϵ-insensitiva a função de otimização é reformulada para minimizar a equação 4.29.

$$\begin{aligned} \min \quad & f(w) = \frac{1}{2}||w||^2 + C\left(\sum_{i=1}^{n}(\xi_i^* + \xi_i)\right) \\ \text{sujeito a} \quad & Vy_i - \langle \vec{w} \cdot \vec{Vx} \rangle - b \leqslant \epsilon + \xi_i^* \\ & \langle \vec{w} \cdot \vec{Vx} \rangle + b - Vy_i \leqslant \epsilon + \xi_i \\ & \xi_i^*, \xi_i \geqslant 0 \end{aligned} \tag{4.26}$$

C representa um parâmetro do algorítimo RVS, e é uma escolha de compromisso entre simplicidade e precisão do modelo. C pequenos favorecem modelos simples, enquanto que valores maiores de C, passam a penalizar mais fortemente os erros, o que torna o modelo mais complexo, porem mais preciso.

Assim como ocorreu com o método da MVS para classificação, serão utilizados os multiplicadores de Lagrange para simplificar o problema de otimização, assim como uma função Kernel para analises com não-linearidades no espaço de entrada. O problema, considerando estas adições, pode ser descrito como:

$$\begin{aligned} \mathcal{L}(w, b, \xi_i^*, \xi_i, \alpha, \eta) = \; & \frac{1}{2}||w||^2 + C\left(\sum_{i=1}^{n}(\xi_i^* + \xi_i)\right) \\ & - \sum_{i=1}^{k} \alpha_i \Big(\epsilon + \xi_i - Vy^{(i)} + K(\vec{w}, \vec{Vx}^{(i)}) + b\Big) \\ & - \sum_{i=1}^{k} \alpha_i^* \Big(\epsilon + \xi_i^* + Vy^{(i)} - K(\vec{w}, \vec{Vx}^{(i)}) - b\Big) \\ & - \sum_{i=1}^{k} \Big(\eta_i^* \xi_i^* + \eta_i \xi_i\Big) \end{aligned} \tag{4.27}$$

Ao igualar a 0, as derivadas parciais da função dos multiplicadores de Lagrange, relativa as variáveis w, b, ξ_i^* e ξ_i, encontramos a relação entre $\eta_i^*, \eta_i, \alpha_i$ e α_i^* (equação 4.28) e a função final de otimização, descrita pela equação 4.29.

$$\begin{aligned} \eta_i &= C - \alpha_i \\ \eta_i^* &= C - \alpha_i^* \\ \eta_i, \eta_i^*, \alpha_i, \alpha_i^* &\geqslant 0 \end{aligned} \tag{4.28}$$

$$\begin{aligned} \max \quad & -\frac{1}{2}\sum_{i,k=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_k - \alpha_k^*)K(\vec{Vx}^{(i)}, \vec{Vx}^{(k)}) \\ & -\epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} Vy^{(i)}(\alpha_i - \alpha_i^*) \\ \text{sujeito a} \quad & \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C] \end{aligned} \tag{4.29}$$

Ainda, a RVS também respeita a condição complementar de dualidade de KKT 4.13, logo podemos concluir que:

$$\begin{aligned} \alpha_i\Big(\epsilon + \xi_i - Vy^{(i)} + K(\vec{w}, \vec{Vx}^{(i)}) + b\Big) &= 0 \\ \alpha_i^*\Big(\epsilon + \xi_i^* + Vy^{(i)} - K(\vec{w}, \vec{Vx}^{(i)}) - b\Big) &= 0 \end{aligned} \tag{4.30}$$

$$\begin{aligned} \eta_i\xi_i &= \xi_i(C - \alpha_i) = 0 \\ \eta_i^*\xi_i^* &= \xi_i^*(C - \alpha_i^*) = 0 \end{aligned} \tag{4.31}$$

Após apropriada computação numérica, os α_is e α_i^*s são definidos e a função de regressão final pode ser descrita como:

$$Vy = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(\vec{Vx}^{(i)}, \vec{Vx}^{(j)}) + b \tag{4.32}$$

4.3 Máquina de Vetores de Suporte por Mínimos Quadrados

A grande diferença entre a Máquina de Vetores de Suporte por Mínimos Quadrados e a Regressão por Vetores de Suporte é a alteração da função de perda utilizada, e por

consequência, alteração na função objetivo. A nova função objetivo é descrita por:

$$\begin{aligned} \min \quad & f(w) = \frac{1}{2}||w||^2 + \frac{1}{2}\gamma \sum_{i=1}^{n} e_i^2 \\ \text{sujeito a} \quad & Vy_{(i)} = K(\vec{w}, \vec{Vx}^{(i)}) + b + e_i \end{aligned} \tag{4.33}$$

O termo e representa o erro entre o valor real e o estimado, $e = Vy_{(i)} - V\hat{y}_{(i)}$, e γ é um parâmetro do usuário, semelhante a C, responsável por atribuir o peso da penalização dos erros. Podemos ver que o novo problema de otimização agora depende da minimização dos quadrados do erro, tal definição é o que da nome ao método.

Com base na equação 4.34, podemos formular a nova equação dos multiplicadores de Lagrange como:

$$\begin{aligned} \mathcal{L}(w, b, e, \eta) \quad & \frac{1}{2}||w||^2 + \frac{1}{2}\gamma \sum_{i=1}^{n} e_i^2 \\ & - \sum_{i=1}^{n} \beta_i \Big(\vec{w}, \vec{Vx}^{(i)}) + b + e_i - Vy_{(i)} \Big) \end{aligned} \tag{4.34}$$

Podemos notar que as condições de desigualdade da equação 4.29, em RVS, foram substituídas por condições de igualdade, motivo pela alteração do multiplicador de Lagrange α para o multiplicador β. Ao calcular as derivadas da função de Lagrange obtemos os seguintes resultados:

$$\frac{\partial \mathcal{L}}{\partial \vec{w}} : \vec{w} = \sum_{i=1}^{n} \beta_i \phi(\vec{Vx}^{(i)}) \tag{4.35}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{b}} : \sum_{i=1}^{n} \beta_i = 0 \tag{4.36}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{e_i}} : \beta_i = \gamma e_i \tag{4.37}$$

$$\frac{\partial \mathcal{L}}{\partial \vec{\beta_i}} : Vy_{(i)} = K(\vec{w}, \vec{Vx}^{(i)}) + b + e_i \tag{4.38}$$

Pelo resultado obtido em 4.35 4.36 4.37 e 4.38 e após eliminar as variáveis w e e_i, vemos que a função de otimização é na verdade um sistema linear (de ordem igual ao numero de dados de treinamento) descrito por:

$$\begin{bmatrix} 0 & 1_v' \\ 1_v & \mathbf{K}_v + \frac{1}{\gamma} I_v \end{bmatrix} \begin{bmatrix} b \\ \beta_v \end{bmatrix} = \begin{bmatrix} 0 \\ V_y \end{bmatrix} \tag{4.39}$$

Aonde I é uma matriz identidade e 1_v é um vetor de "1s"de dimensões n (quantidade de dados de treinamento), β_v o vetor de βs e $\mathbf{K}_v$ a matriz de Kernel $n \times n$, sendo que cada elemento $\mathbf{K}_{i,j}$ é definido como o valor da função de Kernel dos dados de treinamento i e j, $\mathbf{K}_{i,j} = K(\vec{Vx}^{(i)}, \vec{Vx}^{(j)})$.

Existem diferentes algoritmos capazes de encontrar os β ótimos do sistema linear 4.39. Dentre as opções, Keerthi e Shevade [62] sugerem um adaptação para o algorítimo de Otimização Minima Sequencial (SMO) [61]. Segundo eles, esse novo algorítimo é extremamente fácil de implementar e é eficientemente escalável (quadraticamente) em função do numero de dados de treinamento.

Uma vez encontrados os valores de β, a função de estimação pode ser descrita como:

$$f(\vec{Vx}) = Vy = \sum_{i=1}^{n} \beta_i K(\vec{Vx}^{(i)}, \vec{Vx}^{(j)}) + b \tag{4.40}$$

Vale ressaltar que, apesar da simplificação do processo de otimização, uma grande desvantagem da MVS-MQ, frente a RVS, é que o solução não é mais esparsa. Como vemos pela equação 4.37, agora todos os Multiplicares de Lagrange β possuem valores não nulos. Isso significa que todos os dados de treinamento são considerados como Vetores de Suporte.

A figura 17 ilustra essa diferença.

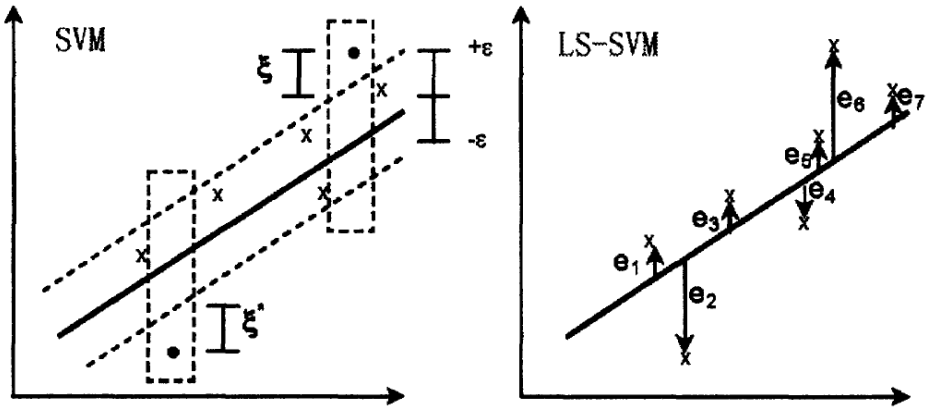


Figura 17 – Contraste entre a esparsidade da RVS e MVS-MQ (Wang e Hu [65], 2005)

5 Implementação

Neste **capítulo** será discutido como o método **MVS-MQ** foi adaptado para utilização na imputação de dados para o SIT de Santos. Serão abordados detalhes mais específicos da implementação como: Caracterização do problema, escolha dos dados de treinamento, algorítimo de treinamento e métodos de avaliação de desempenho.

5.1 Softwares Utilizados

O *MATLAB 2015b* foi o principal software utilizado neste trabalhando, servindo de base para diversas atividades. Dentre as ferramentas (toolbox) utilizadas se destaca a *LS-SVMlab1.5 MATLAB toolbox*, desenvolvido por Pelckmans et al. [68], que foi utilizada para o treinamento dos dados. Foram ainda utilizado 2 softwares para manipulação dos bancos de dados: *PostegreSQL 9.4* e *DB Browser for SQL*. Assim como o já mencionado *AISUM*, para modelagem e simulação das malhas viárias.

5.2 Caracterização do Problema

Como visto no **Capítulo 2**, o problema deste trabalho consiste em imputar dados de tráfego, de sensores falhos, através do método MVS-MQ. Para isso estão a disposição: Os dados históricos (sem perda de dados), e as medições que serão imputadas.

O banco de dados fornecido para o treinamento é semelhante ao apresentado na Figura 2, contendo os dados dos laços indutivos sem nenhum processamento. Portanto é necessário converte-los, selecionando um período de agregação Ta adequado (procedimento discutido no **Capítulo 2**). O processamento transforma os dados em uma matriz, aonde suas colunas representam as estações de sensores, e as linhas representam um certo período do dia (identificado pelo horário de inicio). A Figura 18 mostra um trecho de tal banco de dados.

É a partir deste banco de dados históricos que se irá realizar o treinamento da MVS-MQ. Mais especificamente, será realizado um treinamento para cada uma das estações de sensores. O conjunto de dados de treinamento, para uma estação de sensores es, é composto por pares de dados $\{Vx_i^{(es)}, Vy_i^{(es)}\}$.

$Vy^{(es)}$ é um vetor $1 \times T_t$, contendo T_t resultados históricos da estação es, e $Vx^{(es)}$ é um vetor $n \times Tt$, contendo os resultados históricos de um conjunto de sensores vizinhos, aonde cada dimensão (atributo) n de $Vx^{(es)}$ corresponde a um sensor vizinho.

	L01011	L01012	L01013	L01014	L01015
2016-09-01 02:30:00	0.30606	0.18078	0.73265	0.5376	0.35205
2016-09-01 02:35:00	0.60895	2.4358	0.6708	0.65587	0.2236
2016-09-01 02:40:00	0.41707	0.28545	0.18554	0.98003	0.68032
2016-09-01 02:45:00	0.73265	0	0.57089	0.4805	1.6104
2016-09-01 02:50:00	0.2458	0.18078	1.2369	0.78736	0.77784
2016-09-01 02:55:00	0.66287	0.30924	1.3797	0.86348	0.94197
2016-09-01 03:00:00	0.39011	0.31399	0.65653	0.46861	1.1251
2016-09-01 03:05:00	0.28862	4.001	0.75644	0.58755	0.68032
2016-09-01 03:10:00	0.34888	0.31399	1.1275	0.54711	0.61609
2016-09-01 03:15:00	0.41073	1.9268	0.49953	0.65415	0.11894
2016-09-01 03:20:00	0.60103	0.39011	0.52332	0.29972	0.41866

Figura 18 – Exemplo de trecho do banco de dados processado

Na definição dos dados de treinamento já ficam claros dois parâmetros importantes do algorítimo: T_t e n.

5.2.1 Quantidade de dados de treinamento

A quantidade de pares de treinamento (T_t) representa a *"duração total"* dos dados considerados para o treinamento. Ao utilizarmos um intervalo de tempo maior, teremos mais informações disponíveis, o que, em teoria, "ensina" ao modelo mais "configurações de trafego". Entretanto, vale lembrar que a solução da **MVS-MQ** não é esparsa, o que significa que conforme T_t aumenta, a carga computacional necessária para o treinamento também aumenta, ou mesmo a carga relacionada a imputação (ainda que em menor escala).

Com o aumento de T_t, o tamanho do modelo treinado também aumenta, uma vez que ele precisa armazenar todos os dados de treinamento em conjunto dos valores dos multiplicadores β, um para cada par de dado. Isso pode causar problemas de armazenamento, dependendo do tamanho da memoria disponível.

Pela relação: $T_t = \frac{duração}{Ta}$ vemos que Ta tem uma influencia indireta sobre T_t, sendo que é possível manter fixos T_t e a carga computacional, e ainda assim estender a duração dos dados de treinamento. Neste caso, Ta aumenta, sacrificando predições de curto período para aumentar a "quantidade de informação" disponível para o treinamento.

5.2.2 Estações vizinhas

Por sua vez, a quantidade de "atributos" n de Vx representa a quantidade de estações vizinhas, consideradas na hora de imputar os dados de es. Definimos como estação vizinha, aquelas que possuem dados com um alto grau de correlação, em geral exemplificadas como estações próximas. Como os dados para imputação (pós-treinamento)

são armazenados, informações sobre o estado "passado" da malha também estão disponíveis $(Vy_i^{(es)}, ...Vy_{i-a}^{(es)})$. Dados passados podem ser uteis para a imputação: Imagine uma estação es_0 que se encontra numa posição distante, a montante, do sensor a ser imputado es_3. Nesse caso a influencia do fluxo de carros em um momento i, na estação es_0 só sera relevante para a imputação da estação es_3 após um certo atraso a. Logo, para fins de imputação de $Vy_i^{(es_3)}$ as medições passadas $Vy_{i-a}^{(es_0)}$ são mais relevantes que as medições presentes $Vy_i^{(es_0)}$.

Portanto, para fins de seleção dos vizinhos (atributos) para o treinamento, os dados com atraso também serão considerados. Uma mesma estação pode, inclusive, representar dois atributos de Vx (com atrasos a distintos). Ainda a utilização de dados passados permite, teoricamente, a utilização dos dados passados da própria estação, para estimação dos valores atuais. Entretanto tal implementação não é indicada, pois falhas de longa duração são comuns e nesse caso degradariam a imputação.

Em teoria, a **MVS-MQ** pode ser treinada utilizando os dados de todos os sensores disponíveis (com vários atrasos), entretanto não são todas as estações que realmente possuem dados relevantes entre si. Caso o modelo de uma estação es seja treinado utilizando medições que tem pouca ou nenhuma influencia naquela estação, estaríamos adicionando dados de treinamento que agiriam mais como ruido do que informação útil, degradando a solução.

Por outro lado, é possível que ocorra uma **falha múltipla envolvendo sensores correlacionados**, ou seja: duas estações consideras vizinhas (para fins da MVS) sofrem falhas simultâneas. Neste caso, um certo "erro" será adicionado a entrada da imputação (pós treinamento), que terá influencia maior ou menor dependendo da quantidade n de atributos. Ou seja, um valor maior de n produz um modelo mais robusto.

Existem, é claro, metodologias para se contornar o problema das falhas múltiplas em sensores correlacionados. Possíveis soluções incluem: Utilizar um método secundário, mais simples (e.g. media histórica), para imputar esses dados antes de utilizá-los na MVS, amenizando assim o erro de entrada; A imputação em cascata, realizando a estimação primeiro dos sensores com menos vizinhos falhos, e utilizando os valores imputados deste ciclo para o ciclo seguinte; Ou ainda, o treinamento de múltiplos modelos (em quantidade limitada), e posterior seleção do modelo adequado para cada configuração de falha. Todos esses métodos requerem, porem, a detecção previa das falhas, e como mencionado em 2.3.1, a detecção não faz parte do escopo desse trabalho, por consequência o tratamento de falhas múltiplas também não o faz.

Alem da robustez e precisão, o numero de atributos também tem influencia sobre o tamanho do modelo e da carga computacional, uma vez que torna mais complexo o calculo da **matriz Kernel** (assim como ocorre com T_t). O valor de n deve ser retratar um balanceamento entre o custo computacional (ou velocidade do cálculo das predições) e a

qualidade das predições [69]. Portanto, é fundamental a escolha de um numero n limitado e a seleção adequada dos dados vizinhos. O método utilizado para seleção dos vizinhos será apresentado na seção 5.3.

Dados de entrada

Assim como os dados de treinamento, os dados de entrada do algorítimo (pós-treinamento) são fornecidos como uma matriz (semelhante a Figura 18). Para utilização como entrada, são selecionadas somente as $(a_{max} + 1)$ medições mais recentes da tabela. Aonde a_{max} é um parâmetro do usuário que diz respeito ao atraso máximo considerado na seleção dos atributos para o treinamento.

A partir dessa matriz são gerados os pares de dados $\{Vx_i^{(es)}, Vy_i^{(es)}\}$ que são montados utilizando os mesmo atributos (dados vizinhos) que os utilizados no treinamento. Estes pares de dados serão então utilizados, em seus respectivos modelos MVS-MQ, para estimar o valor de suas medições.

5.3 Correlação entre detectores vizinhos

Como discutido anteriormente, uma escolha adequada dos sensores vizinhos é necessária, tanto para limitar o tamanho dos atributos de treinamento, quanto para evitar a utilização de dados não correlacionados.

Um opção intuitiva é selecionar somente as estações próximas, dando maior prioridade aos detectores a montante (em contraste aos a jusante). Apesar de coerente, esse método requer a inspeção visual da malha, estação por estação, o que, alem de trabalhoso, dificulta a expansão da malha, pois exige uma reanalise (manual) a cada nova estação adicionada. Ainda, *pesquisas recendes descobriram que dados de sensores que não são os "vizinhos mais próximos" também são capazes de fornecer informações uteis* [70]. Portanto, ao utilizarmos somente os sensores próximos, podemos estar ignorando correlações menos obvias entre as vias, mas igualmente importantes.

Pensando nisso, Zhang e Liu [12] sugerem um método automatizado, baseado no **Coeficiente de Correlação de Pearson**, para a seleção dos sensores vizinhos. Este coeficiente mede a relação entre os dados de diferentes estações. A equação 5.1 descreve como o coeficiente é calculado, entre duas estações:

$$C_{Pearson} = \frac{cov(Vy^{(es_i)}, Vy^{(es_j)})}{\sqrt{var(Vy^{(es_i)})var(Vy^{(es_j)})}} \tag{5.1}$$

Aonde "cov"é a função de covariância, e "var"a função de variância. Como anteriormente, $Vy^{(es_i)}$ se refere ao vetor contendo a serie temporal de leituras da estação

es_i.

A equação 5.1 pode ser reescrita de forma mais explicita, como:

$$C_{Pearson} = \frac{\sum_{k=1}^{T_t}(Vy^{(es_i)} - \bar{Vy^{(es_i)}})(Vy^{(es_i)} - \bar{Vy^{(es_i)}})}{\sqrt{(Vy^{(es_i)} - \bar{Vy^{(es_i)}})^2(Vy^{(es_j)} - \bar{Vy^{(es_j)}})^2}}$$

$$\text{aonde:} \quad \bar{Vy^{(es)}} = \frac{1}{T_t}\sum_{k=1}^{T_t} Vy^{(es)} \tag{5.2}$$

$$C_{Pearson} \in [-1, 1]$$

Sendo assim, a interpretação do coeficiente é de que valores altos (positivos ou negativos) indicam um alto grau de correlação, enquanto que valores próximos de 0 indicam baixo grau de correlação. O valor de $C_{Pearson} = 1$ é obtido ao se comparar $Vy^{(es)}$ com ele mesmo. Ao calcular este coeficiente, dois a dois, é possível determinar os vizinhos com maior correlação para cada estação.

O calculo de correlação também é realizado entre os dados de uma estação es_i, e os dados atrasados de uma outra estação es_j. Por motivos óbvios, conforme cresce o numero de ciclos de atraso que se deseja analisar, cresce o custo computacional total do problema. Por isso a quantidade de atrasos explorados é limitada a um valor a_{max}, a ser definido pelo usuário.

A solução, utilizando o Coeficiente de Correlação de Pearson, *"pode ser preferível frente a outros métodos de seleção de subgrupos de variais por causa da sua escalabilidade computacional e estatística"* [71].

No anexo A, entre as linhas 32 e 53 podemos ver o trecho do código MatLab que implementa a classificação por Coeficientes de Pearson.

Utilizando os dados de simulação apresentados em 2.5, o algorítimo de seleção dos sensores vizinhos foi testado ($Ta = 90s$). A tabela 1 mostra as correlações entre a estação Sen171 e algumas das estações vizinhas. Na primeira parte, destacado em azul, os sensores com melhor correlação. Na parte inferior, alguns outros sensores (não selecionados). Destacado em negrito temos a indicação de qual valor, entre os diferentes atrasos, possui a melhor correlação.

A figuras 19 mostra a distribuição espacial dos sensores selecionados. O circulo vermelho corresponde a estação alvo (Sen171), os círculos azuis escuro indicam os 5 sensores com melhor correlação, e os azuis claro indicam os demais sensores entre os 15 com melhor correlação.

Através de inspeção visual, podemos notar que a solução é coerente pois seleciona prioritariamente os sensores próximos ao Sen171. Como esperado, os sensores afastados a

	a_0	a_1	a_2
Sen194	**0.9677**	0.9600	0.9319
Sen3242	**0.9372**	0.9217	0.9008
Sen195	0.9322	**0.9336**	0.9061
Sen188	**0.9328**	0.9089	0.8828
Sen196	0.9226	**0.9321**	0.9145
Sen170	**0.9284**	0.9057	0.8834
Sen211	0.7575	**0.7602**	0.7579
Sen3029	**0.4916**	0.4861	0.4746
Sen3880	**0.5111**	0.5068	0.5013
Sen186	0.6966	**0.6977**	0.6930
Sen245	**0.5500**	0.5447	0.5465

Tabela 1 – Tabela de correlação de Pearson - estação Sen171

montante tem melhor correlação quando se é adicionado um pequeno atraso aos seus dados (Sen195, Sen196). Vemos também que dados a jusante da estação alvo também possuem um bom grau de correlação. Essa informação é coerente, uma vez que o $C_{Pearson}(Vy^{(es_i)}, Vy^{(es_j)})$ e $C_{Pearson}(Vy^{(es_j)}, Vy^{(es_i)})$ são iguais (considerando a analise sem atrasos).

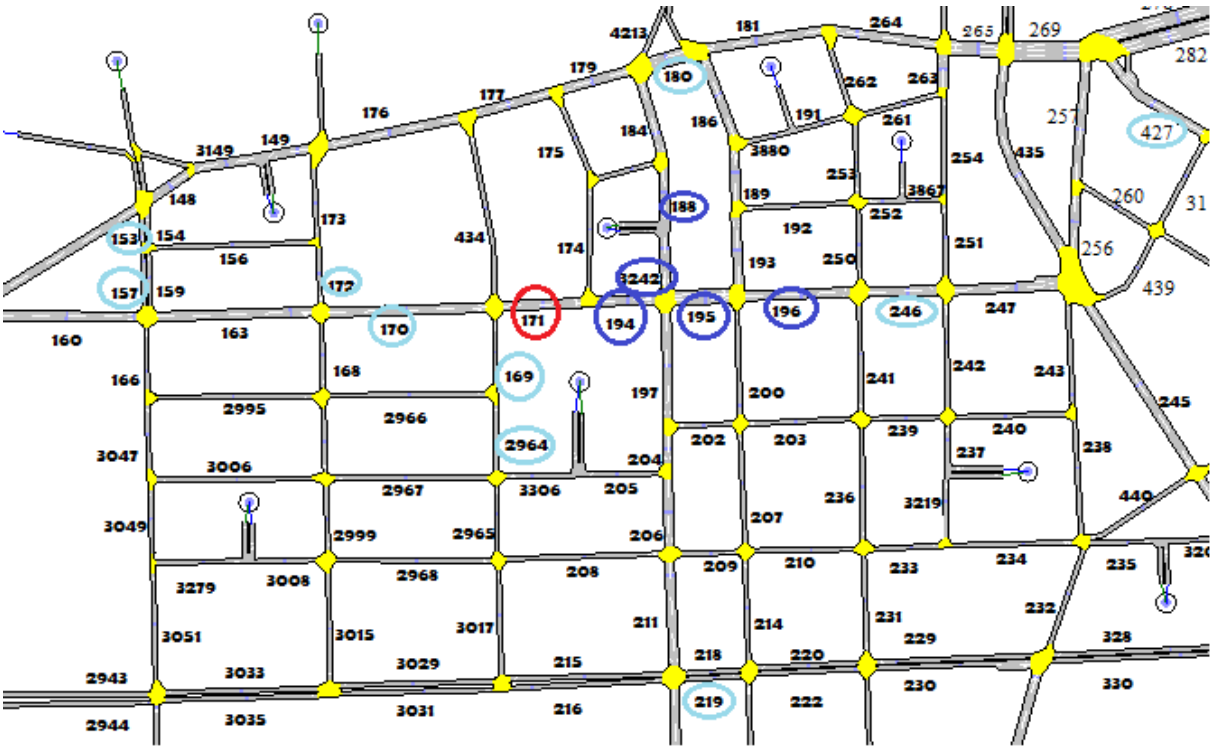


Figura 19 – Seleção dos sensores vizinhos para estação Sen171

As figuras 20 e 21 mostram mais dois outros casos:

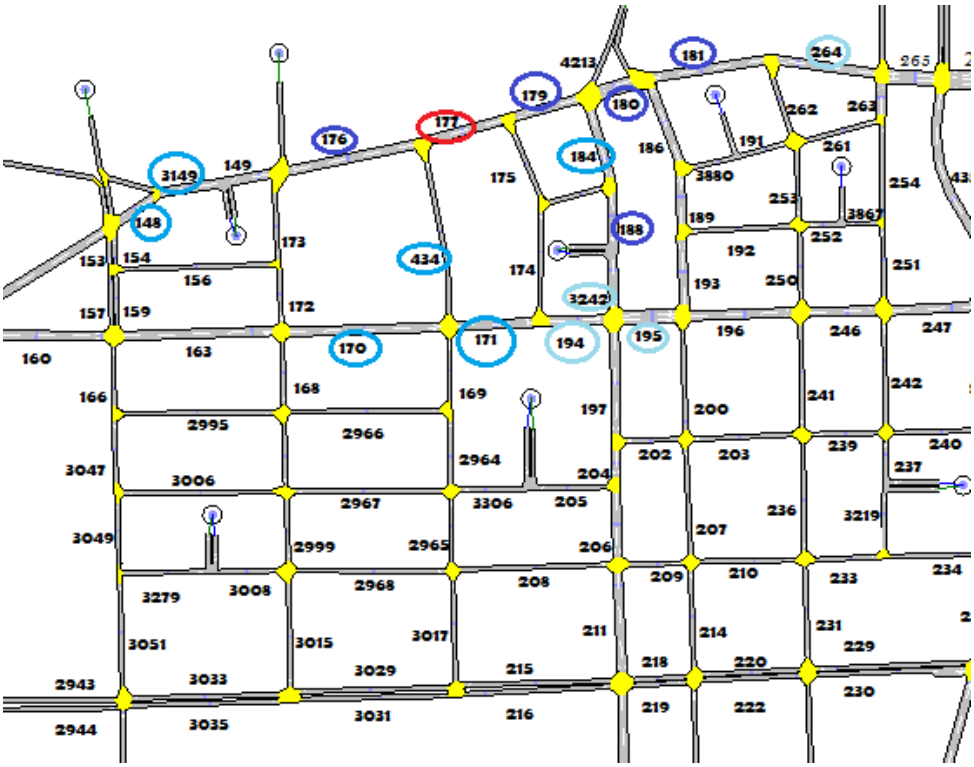


Figura 20 – Seleção dos sensores vizinhos para estação Sen177

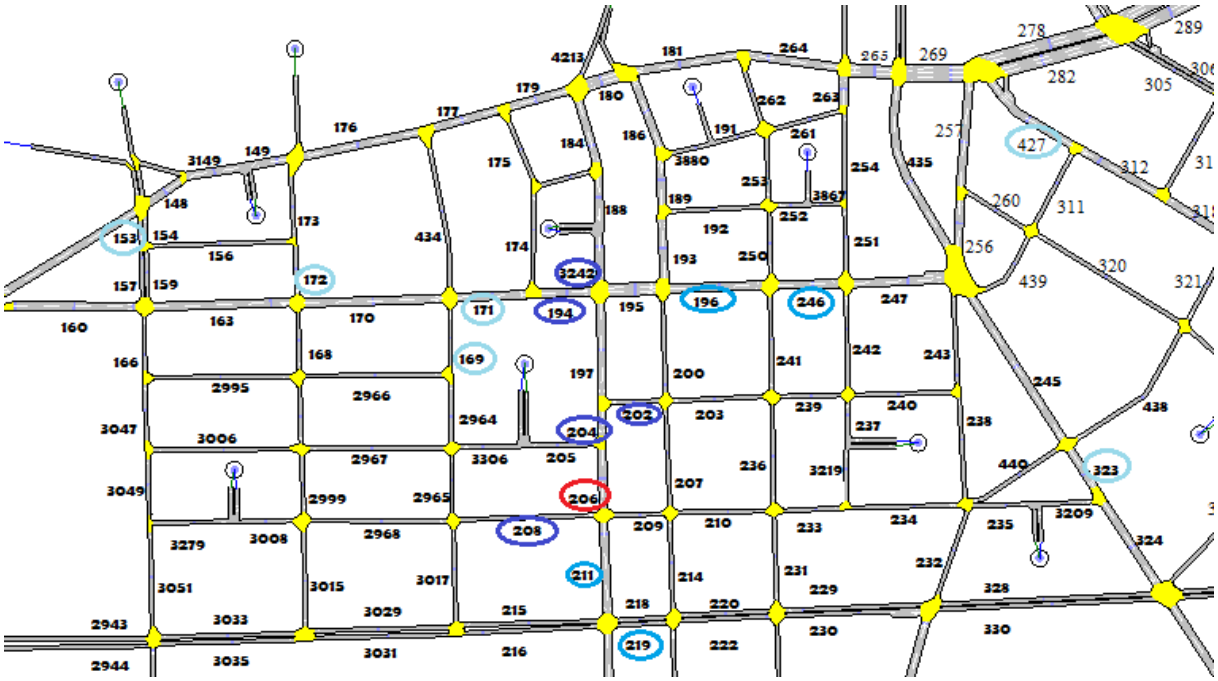


Figura 21 – Seleção dos sensores vizinhos para estação Sen206

5.4 Modelo MVS-MQ

5.4.1 Definição dos parâmetros internos da MVS-MQ

Como já foi apresentado, para a solução do problema da MVS-MQ, é necessário definir alguns parâmetros do usuário: Período de agregação de dados Ta, Numero máximo de atrasos permitido aos dados de treinamento a_{max}, Numero máximo de "atributos"nos dados de treinamento n, Duração do treinamento T_t, Grau de penalização da função de perda γ e Desvio padrão da função de Kernel Gaussiana σ.

Desses, os dois últimos (γ, σ) representam parâmetros de ajuste interno do método MVS-MQ, que influenciam muito na performance do algorítimo [54].

Atualmente, vários algorítimos meta-heurísticos já foram aplicados com sucesso para determinar seus valores ideais, eles incluem: Otimização por enxame de partículas [72], Algorítimo genético [73], Abordagem de evolução caótica diferencial (chaotic differential evolution approach) [74], Algorítimo de colmeia artificial [75] e algorítimo de otimização da mosca da fruta (Fruit Fly Optimization Algorithm) [54].

Fortunadamente, a toolbox *LS-SVMlab* possui uma função (*'tunelssvm'*) que estima os valores ótimos para γ e σ. Mais especificamente, o processo de otimização é realizado em dois passos: Primeiro o algorítimo de Arrefecimento Caótico Simulado (CSA) [62] é utilizado para determinar os valores iniciais de γ e σ. Posteriormente, modelos gerados com esses parâmetros são avaliados utilizando funções de validação cruzada. Através de um método iterativo, utilizando simplex (ou gridsearch), o ajuste fino dos parâmetros é realizado.

O método CSA é uma boa escolha pois consegue reduzir a sensibilidade do algorítimo aos parâmetros iniciais, enquanto guia o processo para um processo quase-ótimo. Isso proporciona uma melhora na eficiência da otimização do método. Por exemplo, já foi comprovado que ele é mais eficiente que a Otimização do Máximo Declive (Multi-start Gradient Descent Optimization) [76].

Apesar de eficiente, a otimização dos dois parâmetros ainda representa o maior gargalo no processo de treinamento. Por isso, em caso de retreinamento, é indicado utilizar os valores de treinamentos anteriores (de uma mesma estação) como pontos de partida do algorítimo.

5.4.2 Treinamento e imputação

Definidos todos os parâmetros de usuário e criado os pares de vetores para o treinamento $\{Vx_i, Vy_i\}$, o modelo já pode ser treinado. Nesse caso mais uma vez recorreremos a uma função (*'trainlssvm'*) da toolbox *LS-SVMlab* para realizá-lo Assim como descrito nesse trabalho, a toolbox formula o problema da MVS utilizando a minimização dos quadrados

do erro (Mínimos Quadrados), e soluciona a otimização do Sistema Linear através do SMO modificado [62].

Como esperado, o modelo final nada mais é do que os vetores dos pares de entrada $\{Vx_i, Vy_i\}$, os parâmetros internos (γ, σ), um valor de bias b, e os valores dos Multiplicadores de Lagrange β, um para cada par de dados de treinamento.

São somente esses dados o SIT precisa armazenar (para cada estação). Aplicando eles e os dados medidos a equação 4.40 podemos obter o valor a ser imputado. É possível facilmente montar a função embarcada para realizar esse procedimento. Dentro do ambiente do Matlab, e por praticidade, será utilizado função *simlssvm*.

O trecho do anexo A, linha 111 mostra como foi executado o treinamento descrito anteriormente.

5.5 Implementação métodos comparativos

Como forma de avaliar a eficacia do método proposto, foram também desenvolvida duas técnicas de imputação:

- **Média histórica**. Aonde o valor a ser imputado é diretamente obtido do valor da media histórica correspondente a aquele mesmo período. O calculo da média histórica é, portanto, realizado por sensor e por período do dia. No anexo A, linha 70, é possível visualizar a implementação do calculo da média e do vetor de saída da imputação.

- **Imputação do SIT de Santos**. Nesse caso são calculadas as proporções entre os dados históricos de uma estação com os dados históricos das estações vizinhas. Então os valores imputados serão as medias dos valores proporcionais das medições dos vizinhos.

  Na imputação real, realizada pelo SIT de Santos, os sensores vizinhos são obtidos a partir de um banco de dados interno, contendo essa informação. Entretanto, na implementação aqui proposta, por praticidade, eles serão selecionados da mesma forma que os atributos da MVS, através do Coeficiente de Pearson. Nesse caso, serão utilizados somente medições sem atraso, pois é dessa forma que o SIT faz seus cálculos. Ainda, o numero de estações vizinhas sera limitada a 5, pois o SIT considera um numero limitado de vizinhos em seus cálculos.

  A implementação da imputação do SIT, para fins de comparação, pode ser vista no anexo A, linha 96.

5.6 Métodos de avaliação de desempenho

Existem diferentes métodos de avaliação de desempenho na literatura. Em nosso trabalho serão utilizadas varias medidas estatísticas para mensurar a qualidade da imputação:

- **Erro máximo absoluto (eMax)**:

$$sup_i(|e|) \tag{5.3}$$

- **Media absoluta do erro (MAE)**:

$$\frac{\sum_{i=1}^{T_n} |e_i|}{T_n} \tag{5.4}$$

- **Mediana absoluta do erro (MEDAE)**:

$$median_{i=1}^{T_n} \ (|e_i|) \tag{5.5}$$

- **Media do quadrado do erro (MSE)**:

$$\frac{\sum_{i=1}^{T_n} e_i^2}{T_n} \tag{5.6}$$

- **Media absoluta percentual do erro (MAPE)**:

$$\frac{1}{T_n} \sum_{i=1}^{T_n} \frac{e_i}{Vy^{(i)}} \tag{5.7}$$

Nas equações, e_i representa o erro da amostra i, T_n o total de imputações, $Vy^{(i)}$ a medida referencia da medição i, $sup(e)$ o maior valor dentre os valores de e, e $median$ a função que retorna a mediana.

Alem das medidas diretas de erro, o ultimo teste ainda realizou uma analisa global do trafego na malha de Santos. Comparando o desempenho do SIT ao utilizar o método atual de imputação, com o desempenho utilizando o novo método sugerido (MVS). Os parâmetros de analise incluem dados como: Tempo de atraso médio, densidade, fluxo, velocidades medias, numero de veículos em fila, tempo de viagem, etc...

6 Resultados

Neste **capítulo** serão apresentados os resultados obtidos ao se aplicar a Máquina de Vetores de Suporte por Mínimos Quadrados para a imputação de dados.

Com base nos bancos de dados disponíveis, foram realizados três cenários de testes:

- **Modelo de Chaniá**: Imputação no banco de dados das simulações (AIMSUM) do modelo Chaniá apresentado em 2.5
- **Dados reais de Santos**: Imputação no banco de dados contendo as medições reais, não processadas, do SIT de Santos.
- **Simulação de desempenho**: Imputação no banco de dados simulado, do modelo (AIMSUM) da malha de Santos, com posterior teste de desempenho do SIT.

Para os testes citados, o atraso máximo tolerado para a entrada foi fixado em 1 ciclos ($a_{max} = 1$), e o numero de atributos de treinamento foi 15 ($n = 15$). Esses valores foram definidos por tentativa e erro, sendo que estes apresentaram bom desempenho nos testes.

6.1 Modelo de Chaniá

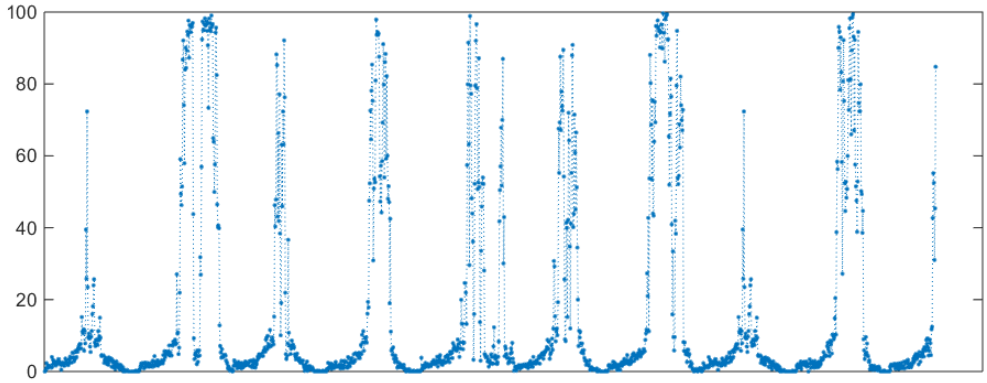
A ideia básica aqui é utilizar uma parte dos dados para realizar o treinamento, e uma segunda parte como valores de entrada e referencia da imputação. As simulações de Chaniá foram realizadas com 4 configurações, por isso foram formulados dois cenários de teste:

a) Treinamento e imputação com os dados de uma mesma configuração (NPC).

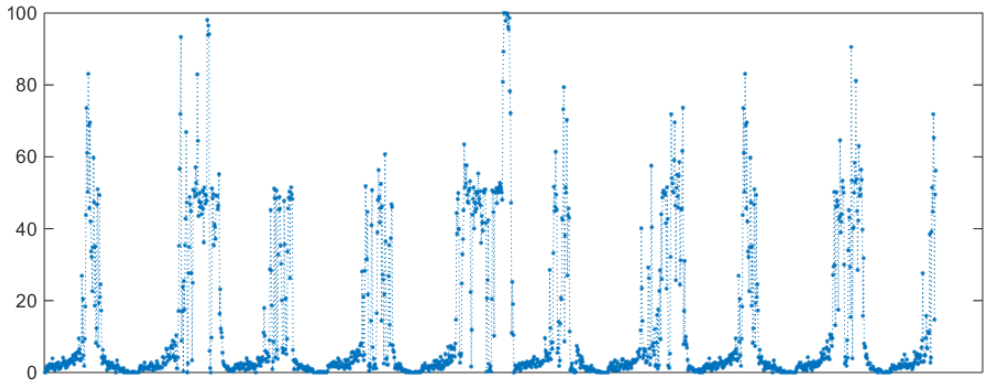
b) Treinamento com um tipo de configuração (PCDBN) e imputação com os dados das outras (PC,NPC,PCQN).

Os ensaios de imputação foram realizados em todos os 165 sensores presentes no modelo. Abaixo serão apresentados com mais detalhes dois deles: *Sen177* e *Sen206*, assim como um resumo dos resultados.

Os sensores *Sen177* e *Sen206* foram escolhido por seu posicionamento espacial relevante, como vimos nas figuras 19 e 21, e por possuírem medições bem comportadas e (relativamente) pouco ruidosas, como vemos na figura 22.



(a) *Sen177*



(b) *Sen206*

Figura 22 – Dados das 9 replicações NPC

6.1.1 Treinamento e imputação em NPC

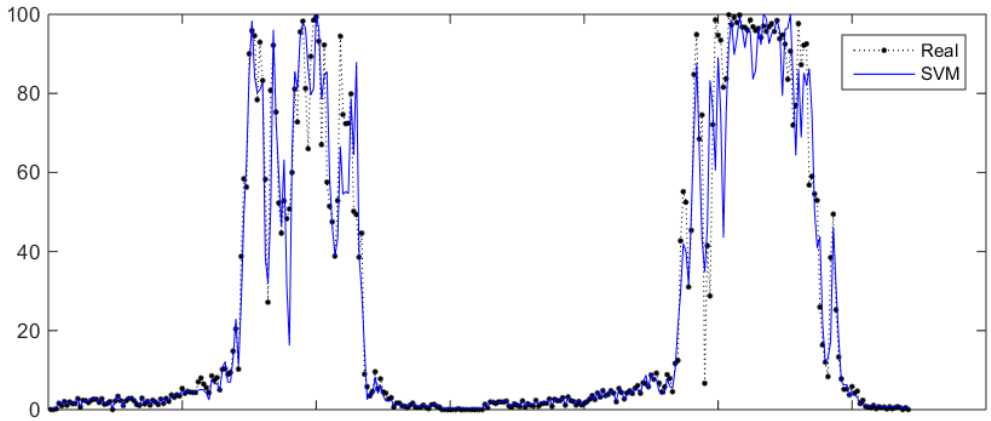
Os dados de NPC foram os utilizados para os ensaios do cenário **a**. Neste cenário, temos o treinamento e a imputação realizados com dados simulados com uma mesma configuração de trafego. Mais especificamente, foram utilizadas 7 replicações para o treinamento, e 2 replicações para gerar os dados dos "sensores vizinhos".

Os valores dos parâmetros internos, definidos pela algorítimo CSA, foram:

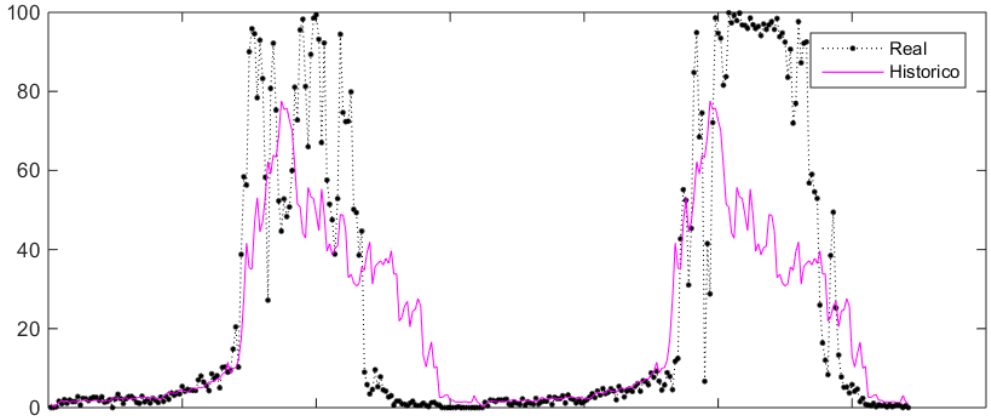
	Sen177	*Sen206*
γ	41.2619	3.1920
σ	8.5479	25.1758

Tabela 2 – Tabela dos parâmetros internos γ e σ para *Sen177* e *Sen206*, Cenário **a**

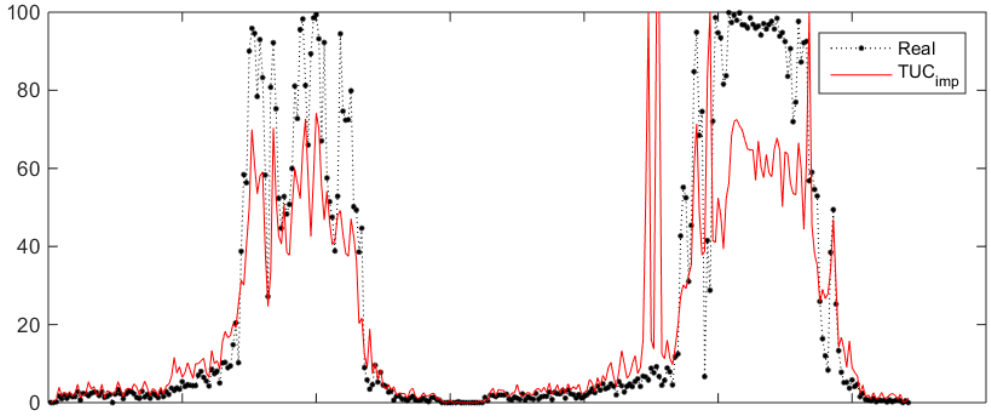
Os resultados obtidos mostram um desempenho superior da técnica MVS-LS frente as outras, como podemos ver abaixo. As figuras 23 e 24 e as tabelas 3 4 mostram os comparativos entre os métodos, para os sensores *Sen177* e *Sen206*, respectivamente.



(a) Destaque: MVS-MQ

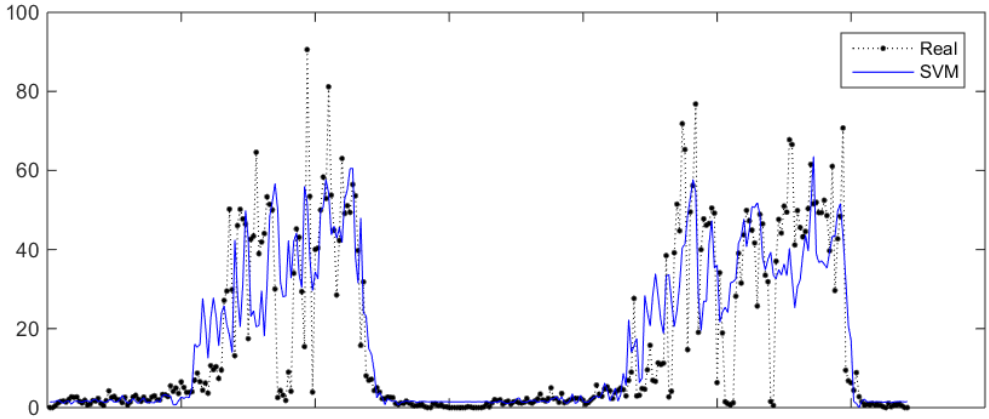


(b) Destaque: média histórica

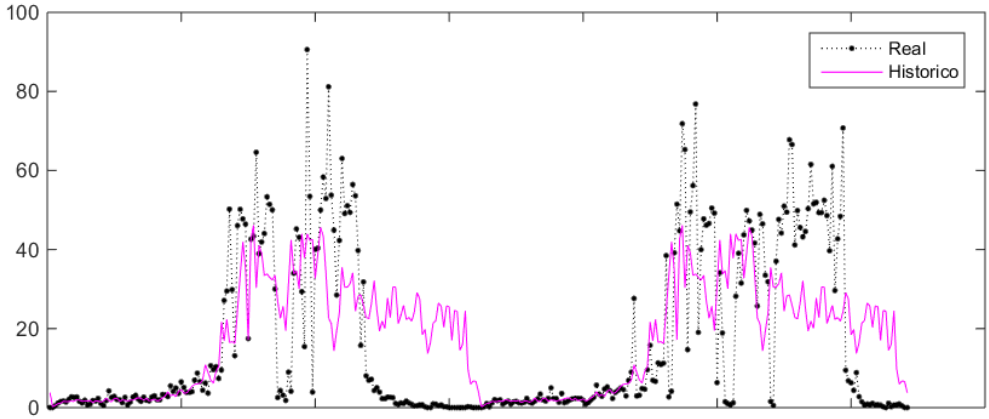


(c) Destaque: algorítimo do TUC_{imp}

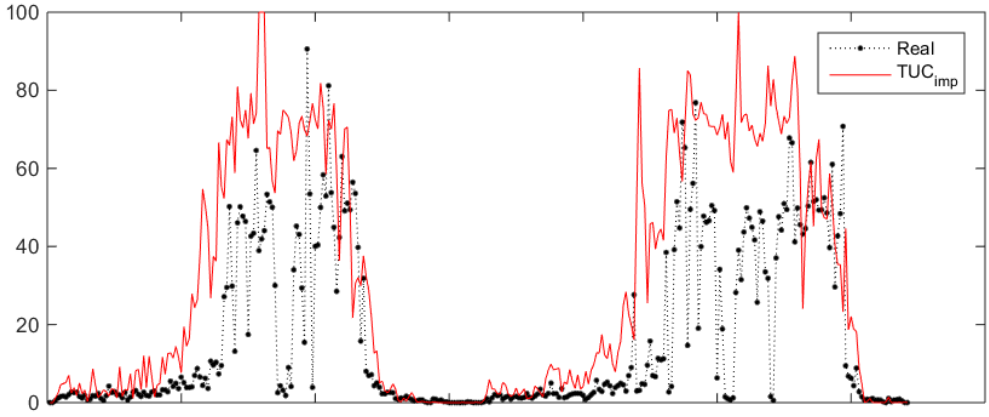
Figura 23 – Imputação do Sen177 - Cenário **a**



(a) Destaque: MVS-MQ



(b) Destaque: média histórica



(c) Destaque: algorítimo do TUC_{imp}

Figura 24 – Imputação do Sen206 - Cenário **a**

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	54.5170	3.7993	0.4012	71.1819	0.2156
Hist	65.5697	14.0676	2.1950	527.2481	2.5339
TUC_{imp}	94.2120	10.7890	2.5441	369.7502	0.9594

Tabela 3 – Tabela comparativa entre métodos - Sen177 - Cenário **a**

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	47.5960	7.6391	1.8640	159.1522	1.6208
Hist	58.5181	11.8219	7.3379	284.0211	5.7792
TUC_{imp}	82.5795	16.3334	7.2541	657.6124	3.8137

Tabela 4 – Tabela comparativa entre métodos - Sen206 - Cenário **a**

Para ilustrar o resultado global obtido, a tabela 5 mostra as médias, entre os 165 sensores disponíveis, dos indicadores estatísticos do erro.

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	64.5294	7.7051	1.9494	271.8071	1.7173
Hist	81.7370	16.2443	8.1876	695.3759	4.8314
TUC_{imp}	84.2803	14.6939	4.8672	700.7734	4.9270

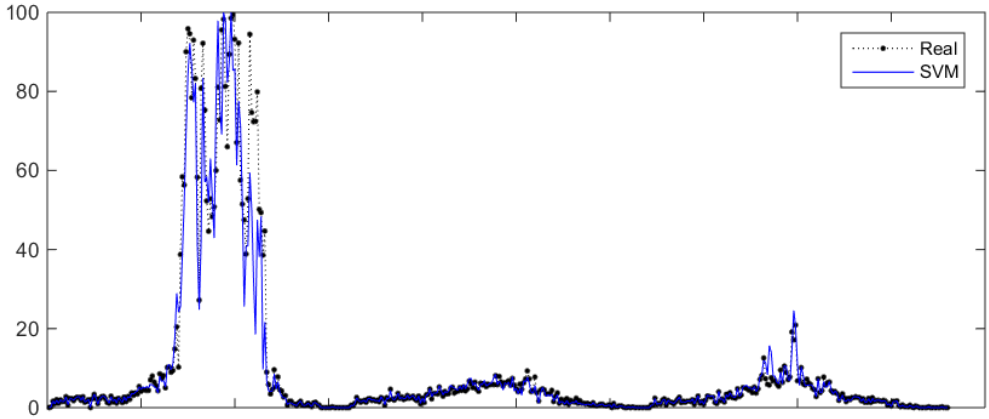
Tabela 5 – Tabela comparativa entre métodos - Media global dos 165 sensores - Cenário **a**

6.1.2 Treinamento com PCDBN e imputação com NPC, PC e PCQNB

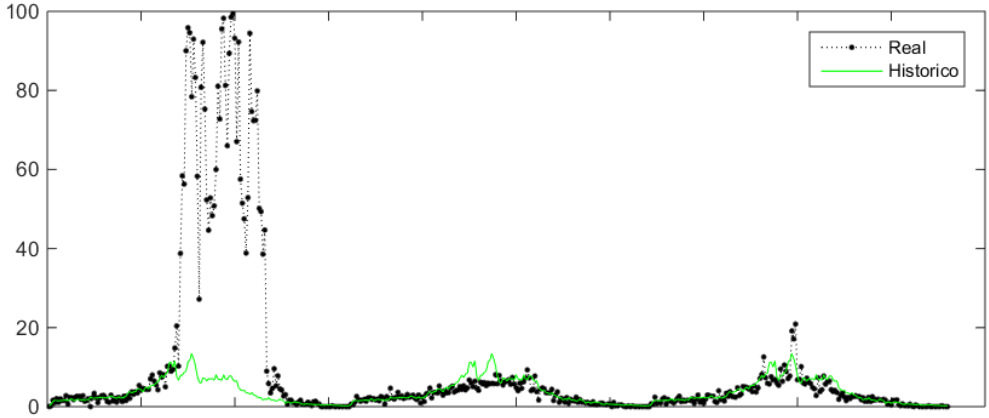
Os ensaios do cenário **b** foram realizados utilizando dados de uma configuração (PCDBN) para o treinamento, e 3 outros grupo de dados, NPC, PC e PCQNB, para utilizar como dados de referencia para a imputação. O objetivo aqui é testar a imputação quando os dados de treinamento não são semelhantes aos dados que precisam ser imputados.

Mais uma vez vemos que a MVS-LS tem resultado superior frente aos outros métodos. Isso demonstra a capacidade da MVS-LS em se adaptar a novas configurações de transito, mesmo que elas não tenham sido aprendidas.

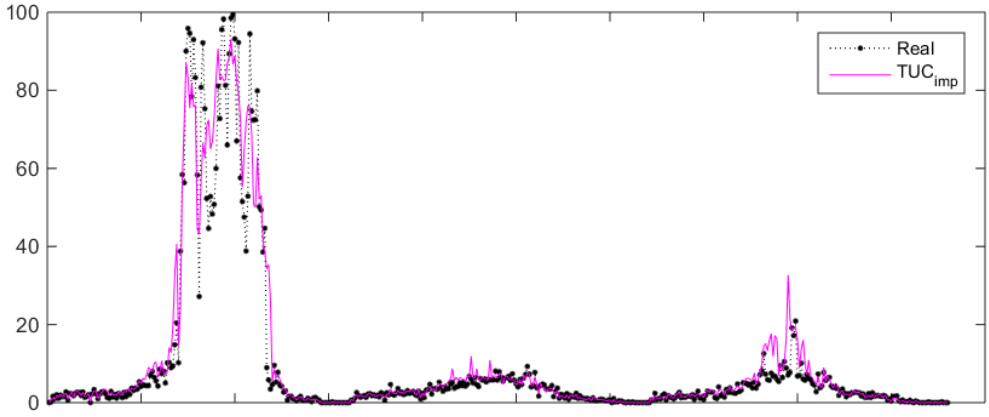
A tabela 6 com os parâmetros internos utilizados, e abaixo, as figuras 25 e 26, e as tabelas 7 e 8 mostram os comparativos entre os métodos.



(a) Destaque: MVS-MQ

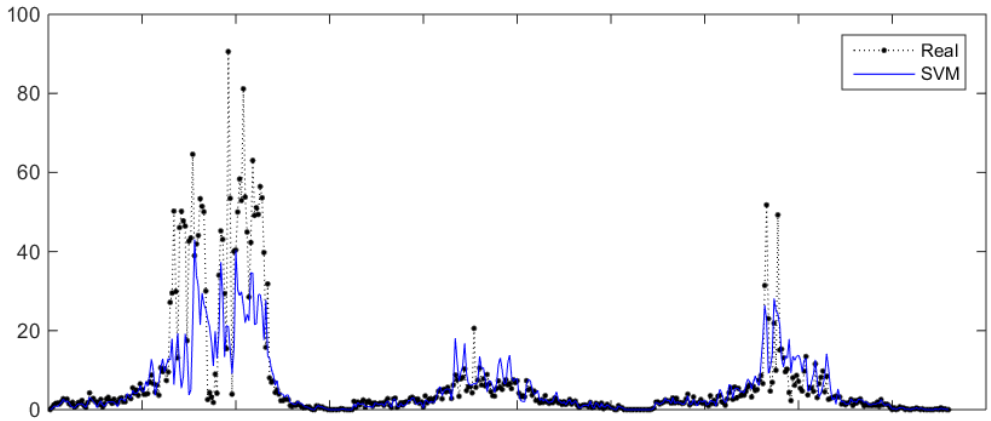


(b) Destaque: média histórica

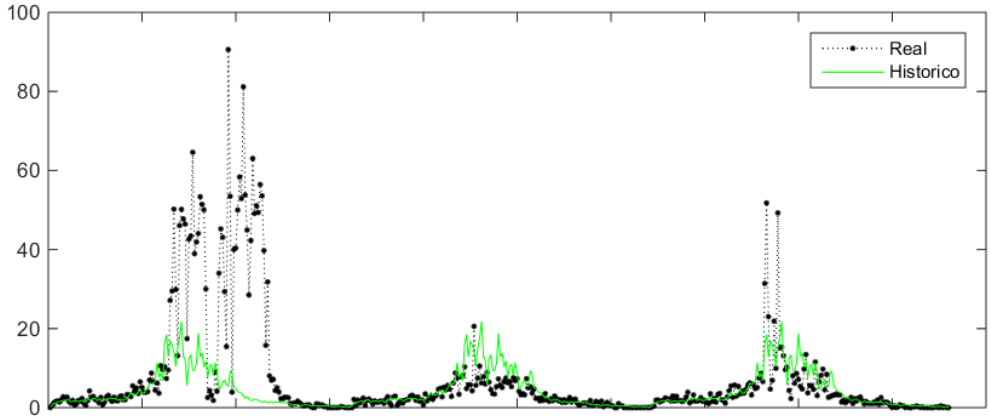


(c) Destaque: algorítimo do TUC_{imp}

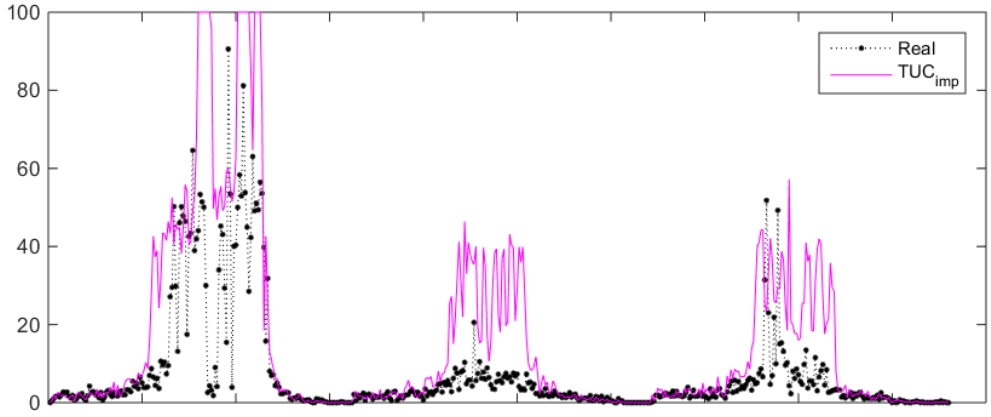
Figura 25 – Imputação do Sen177 - Cenário **b**



(a) Destaque: MVS-MQ



(b) Destaque: média histórica



(c) Destaque: algorítimo do TUC_{imp}

Figura 26 – Imputação do Sen177 - Cenário **b**

	Sen177	*Sen206*
γ	1223	6.5839
σ	3831	149.7008

Tabela 6 – Tabela dos parâmetros internos γ e σ para *Sen177* e *Sen206*, Cenário **b**

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	53.921	1.7558	0.13139	34.709	0.11504
Hist	92.558	7.0149	0.63017	418.52	0.46717
TUC_{imp}	33.142	2.3387	0.37211	33.933	0.31156

Tabela 7 – Tabela comparativa entre métodos - Sen177 - Cenário **a**

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	69.667	3.5191	0.44759	82.552	0.41352
Hist	85.247	5.085	0.6977	166.74	0.56134
TUC_{imp}	97.453	9.1077	1.2468	322.02	1.7782

Tabela 8 – Tabela comparativa entre métodos - Sen206 - Cenário **b**

Como anteriormente, a tabela 9 apresenta as médias globais das estatísticas dos erros.

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	54.0134	3.4108	0.8876	112.1798	0.9028
Hist	67.6636	5.0755	0.8644	188.6768	1.2874
TUC_{imp}	69.7839	8.5602	2.0709	337.9992	3.3583

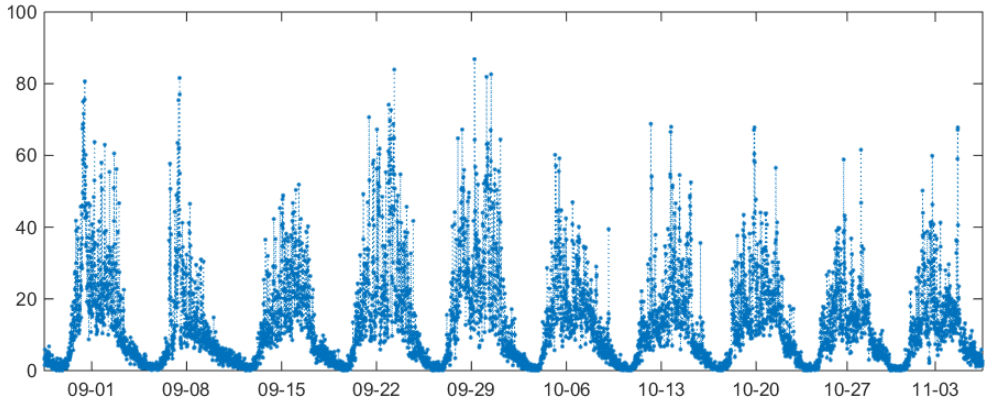
Tabela 9 – Tabela comparativa entre métodos - Media global dos 165 sensores - Cenário **b**

6.2 Medições reais da Malha de Santos

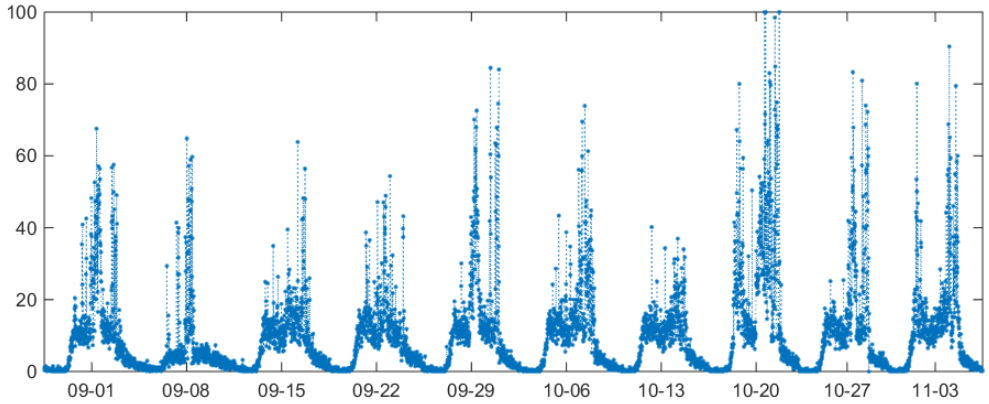
Para o segundo teste foram utilizadas as medições reais da malha de Santos. Como o banco de dados possui apenas dados sem processamento, foi necessária a definição de um valor de agregação Ta para obtenção dos valores de taxa de ocupação temporal da via.

Os resultados que serão apresentados a seguir correspondem a ensaios com período de agregação de 150 segundos. Este é o tempo de ciclo médio do SIT em Santos. Ainda pela figura 3 vimos que é próximo desse valor que os dados começam a definir um padrão mais coerente.

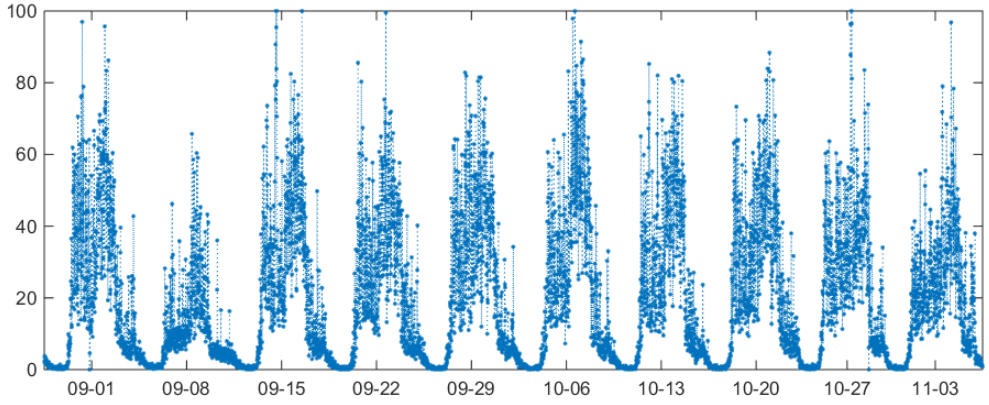
As estações selecionadas para os ensaios foram as estações *'L01071'*, *'L02082'* e *'D03122'*. A figura 27 mostra como são os dados disponíveis para estas estações.



(a) *L01071*



(b) *L02082*



(c) *D03122*

Figura 27 – Dados reais de Santos agregados a cada 150 segundos

Analisando as series temporais podemos notar uma redução na taxa de ocupação no dia 08 de Setembro. Atribui-se essa redução a proximidade do feriado (7 de Setembro). Essa propriedade foi explorada no ensaio para analise do comportamento dos estimadores em caso de mudança no comportamento da malha. Mais especificamente, os dados de

treinamento utilizados foram das quintas feiras de 22 de Setembro a 3 de Novembro, e os dados de referencia foram os dos dias 01, 08 e 15 de Setembro.

Apesar de as estações escolhidas não aparentarem estar degradadas, é possível que elas contenham perda de dados curtas. Ainda, muitos dos dados das outras estações possuem perda de dados longa, como vimos na figura 5, isso significa que os dados de treinamento utilizados não são necessariamente os melhores para a realização do treinamento. Todavia essa configuração é desejada, pois pode ocorrer também na implementação final do SIT.

As tabelas 10, 11 e 12 mostram os coeficientes de correlação das estações vizinhas selecionadas para o treinamento, respectivamente para *'L01071'*, *'L02082'* e *'D03122'*. A tabela 13 mostra os parâmetros internos, definidos pelo algorítimo CSA.

	a_0	a_1
L01081	0.7082	0.6023
L01061	0.6459	0.6311
L01051	0.5954	0.6112
L02043	0.5970	0.5984
L02022	0.5847	0.5853

Tabela 10 – Tabela de correlação de Pearson - estação *'L01071'*

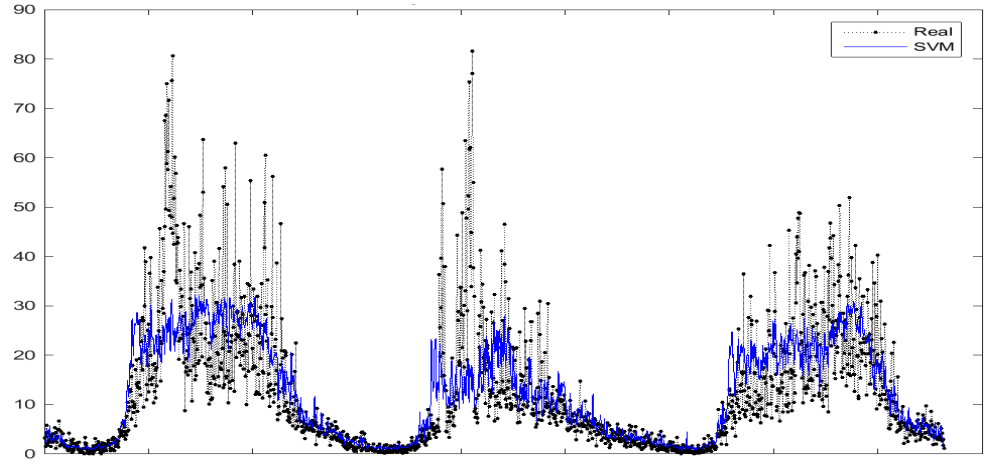
	a_0	a_1
L02093	0.6411	0.6365
L02072	0.6162	0.5879
L01061	0.6156	0.6128
L02043	0.6155	0.6140
L02031	0.6077	0.6029

Tabela 11 – Tabela de correlação de Pearson - estação *'L02082'*

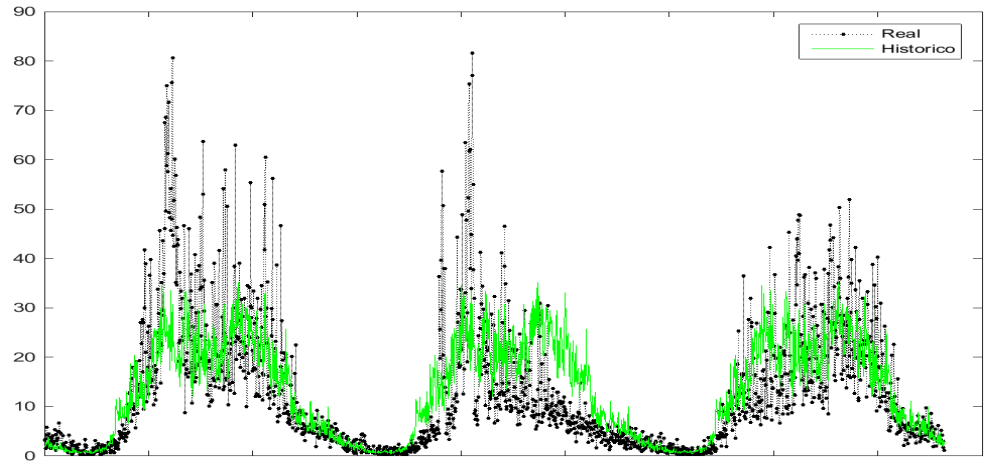
	a_0	a_1
D03135	0.7575	0.7780
D03142	0.6966	0.7353
L02022	0.7010	0.7099
L02043	0.7061	0.7065
D03152	0.7031	0.6995

Tabela 12 – Tabela de correlação de Pearson - estação *'D03122'*

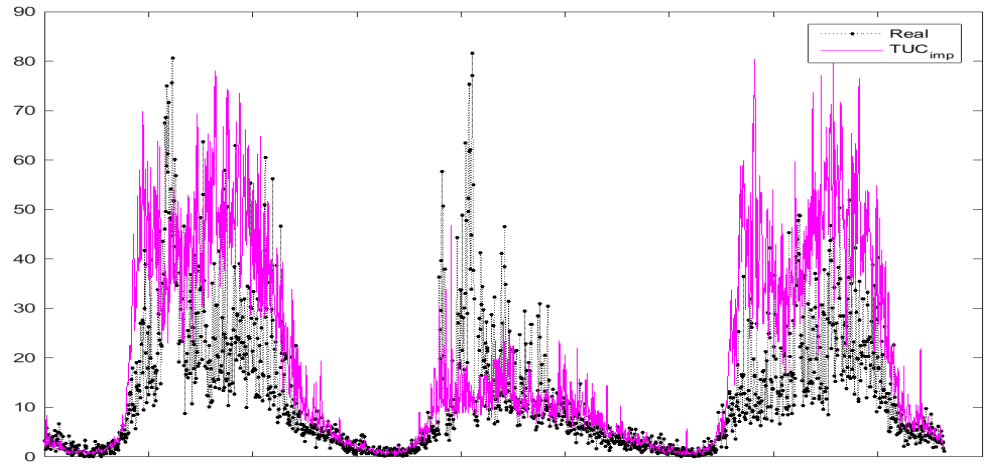
Nas figuras 28, 29 e 30, e tabelas 14, 15 e 16 abaixo, podemos analisar alguns dos resultados:



(a) Destaque: MVS-MQ

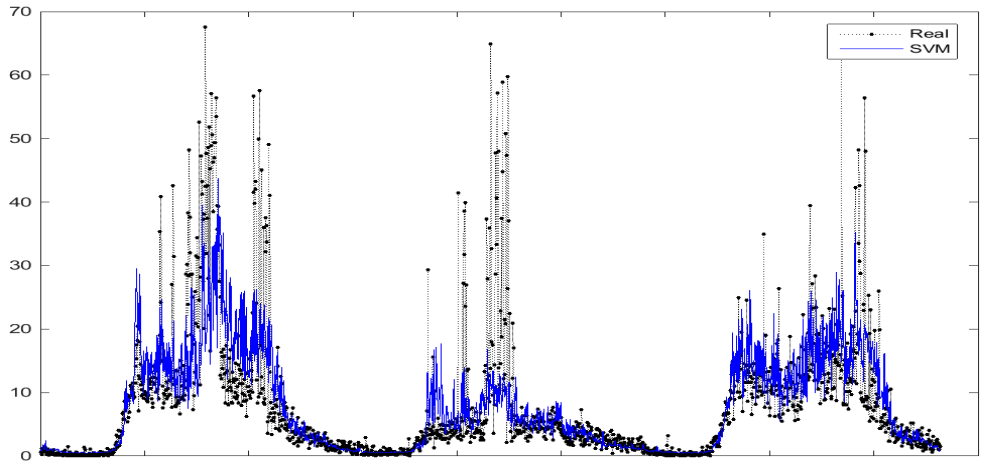


(b) Destaque: média histórica

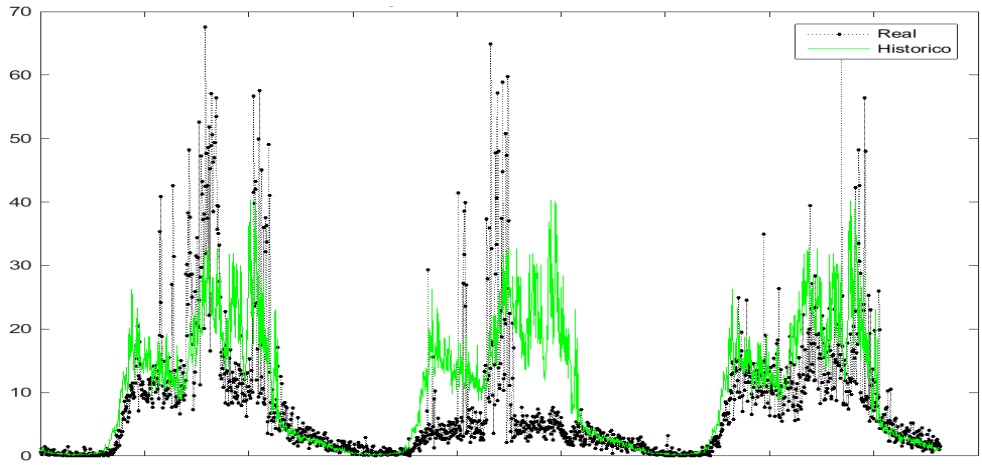


(c) Destaque: algorítimo do TUC_{imp}

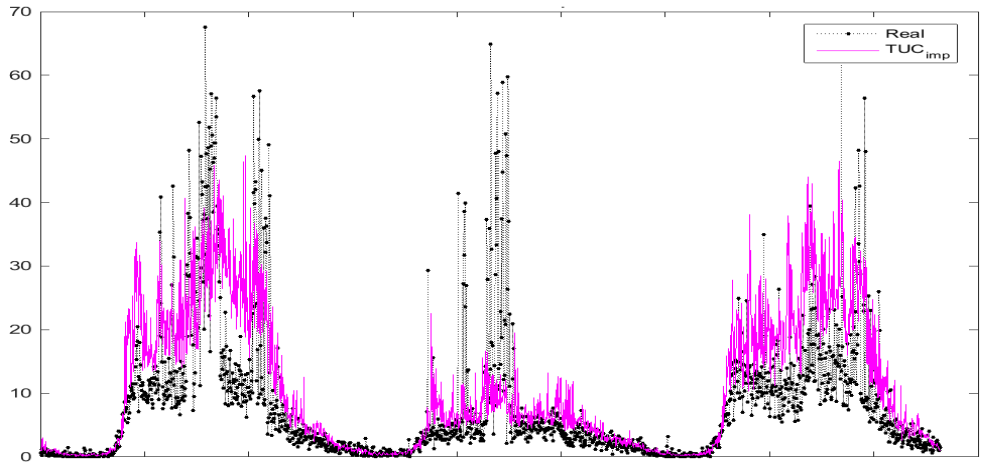
Figura 28 – Imputação da estação, *'L01071'* ($Ta = 150$)



(a) Destaque: MVS-MQ

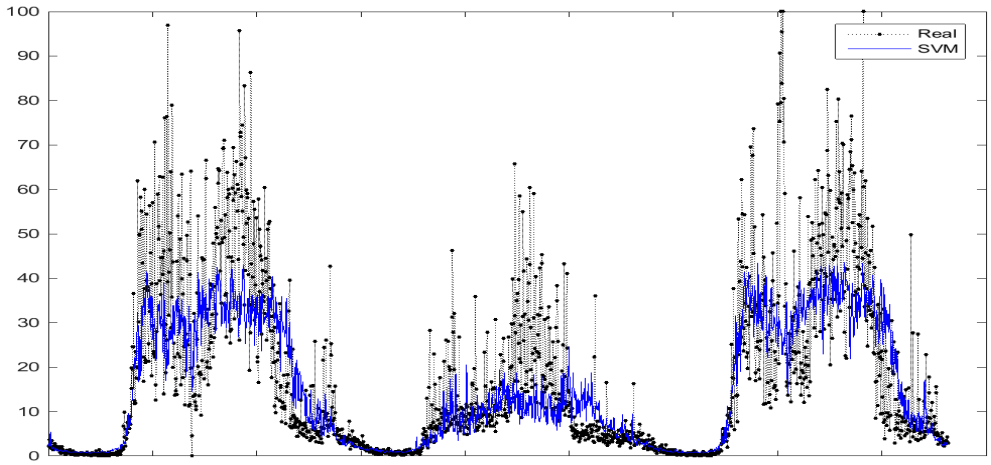


(b) Destaque: média histórica

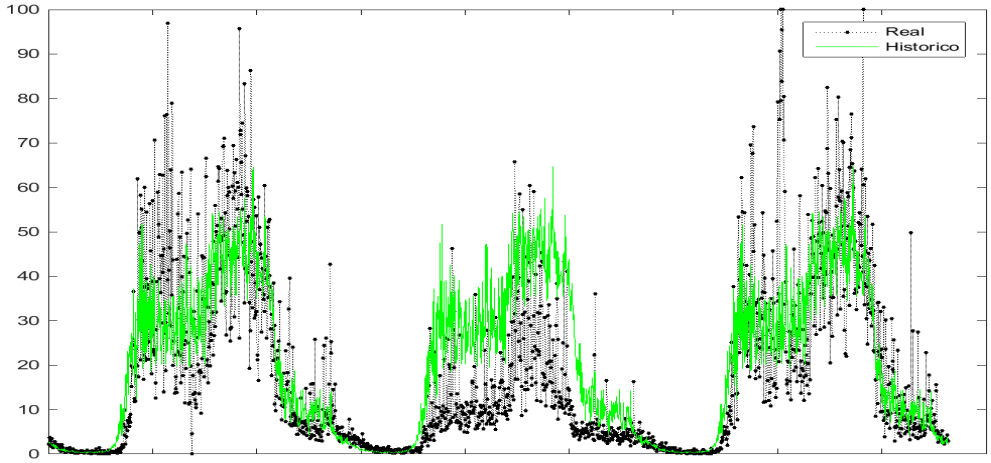


(c) Destaque: algorítimo do TUC_{imp}

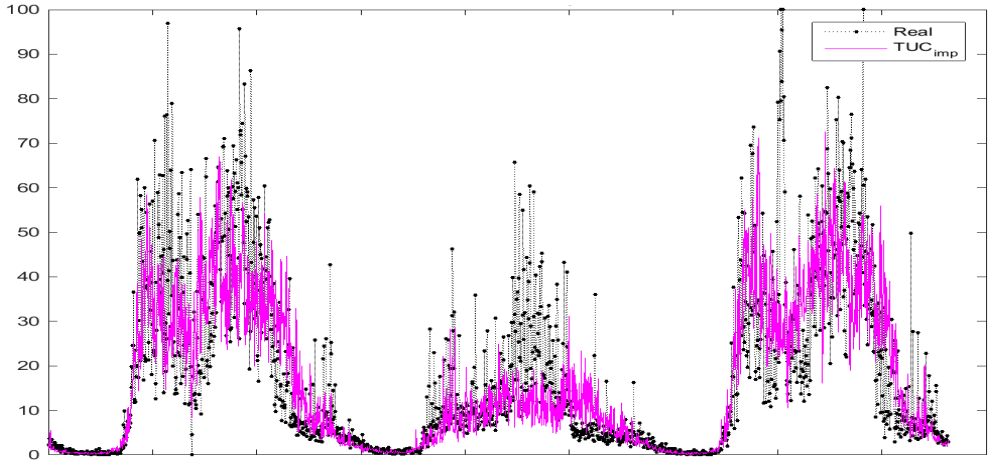
Figura 29 – Imputação da estação, *'L02082'* ($Ta = 150$)



(a) Destaque: MVS-MQ



(b) Destaque: média histórica



(c) Destaque: algorítimo do TUC_{imp}

Figura 30 – Imputação da estação, *'D03122'* ($Ta = 150$)

	L01014	*L02082*	*'D03122'*
γ	0.3146	0.6897	1.4352
σ	34.1237	5.8408	56.6772

Tabela 13 – Tabela dos parâmetros internos γ e σ para as estações.

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	69.114	5.1896	2.1082	81.665	0.53822
Hist	59.017	6.33	3.5153	98.687	0.74235
TUC_{imp}	72.222	10.339	3.2891	279.19	0.93713

Tabela 14 – Tabela comparativa entre métodos, *'L01071'* ($Ta = 150$)

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	54.19	3.65	1.15	55.20	0.69
Hist	49.08	5.58	2.10	84.55	1.14
TUC_{imp}	56.68	5.33	1.93	81.17	0.87

Tabela 15 – Tabela comparativa entre métodos, *'L02082'* ($Ta = 150$)

	eMax	MAE	MEDAE	MSE	MAPE
MVS-MQ	76.581	7.6819	3.4977	163.09	0.70287
Hist	75.122	9.0363	5.1112	190.69	0.98966
TUC_{imp}	77.155	8.2416	3.6018	183.15	0.64518

Tabela 16 – Tabela comparativa entre métodos, *'D03122'* ($Ta = 150$)

Podemos ver pelos resultados apresentados que, apesar de em alguns setores a melhora não ser muito grande, na maioria dos testes a MVS-MQ tem desempenho superior aos outros métodos de imputação.

6.3 Comparativo de desempenho na Malha de Santos

Alem dos testes medindo o erro direto da imputação, foram realizadas simulações para comparar o desempenho do SIT utilizando o algorítimo TUC_{imp}, frente ao SIT utilizando a MVS-MQ.

Primeiro foi realizado uma simulação sem perda de dados, para gerar tanto os dados de treinamento, quanto para estimar os valores de referencia dos parâmetros de comparação (Atrasos, filas, etc...).

Com o modelo treinado, foram feitas simulações no AIMSUM com dois cenários:

No primeiro, nenhum pré-processamento dos dados foi feito antes de utiliza-los como entrada do TUC, na pratica, isso significa que a imputação ocorreu com o algorítimo interno (TUC_{imp}), sempre que ele detectou valores nulos. No segundo, os dados foram pré-processados utilizando o modelo treinado pela **MVS-LS**, na pratica isso significa que todos os valores utilizados foram ”imputados”, contendo perdas ou não.

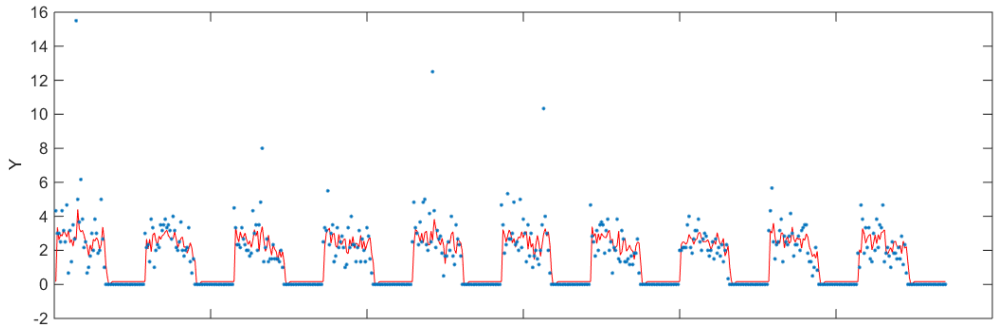
Os dois cenários continham os mesmos 12 detectores com falhas, sendo que 6 deles implicavam na perda total da leitura de 2 links.

Os dados de treinamento já foram fornecidos agregados a cada atualização do controle, ou seja, a cada ciclo. O valor padrão do ciclo em Santos é de 150 segundos, mas esse valor varia de acordo com o controle. Tal variação não caracteriza um problema, desde que os dados mantenham uma coerência entre si, ou seja, os dados de um sensor e seus sensores vizinhos tenham sempre o mesmo Ta, para um mesmo instante. Como o SIT define a mesma duração de ciclo para toda a região, não há problemas quanto ao período Ta variável ao longo do tempo.

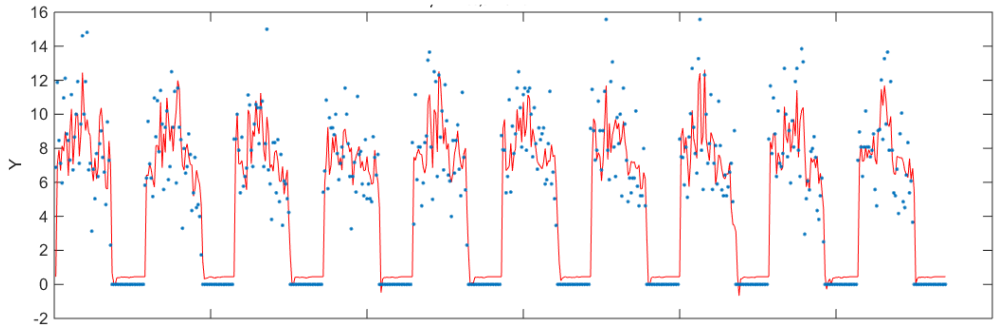
Outra mudança, em relação aos outros ensaios, foi o atraso máximo permitido, para facilitar o acoplamento da MVS-MQ à simulação, somente os dados do presente foram considerados $a_{max} = 0$. Assim os dados de entrada para a imputação (pós-treinamento) precisam apenas conter os dados coletados naquele ciclo.

Um cuidado também foi necessário para os casos de falha múltipla de sensores, mais especificamente o tratamento para o caso de um sensor de entrada da imputação (atributo) também apresentasse falha. Neste caso, por conveniência, os valores utilizados foram fixados em 0. Antes o valor era considerado *"NaN"*.

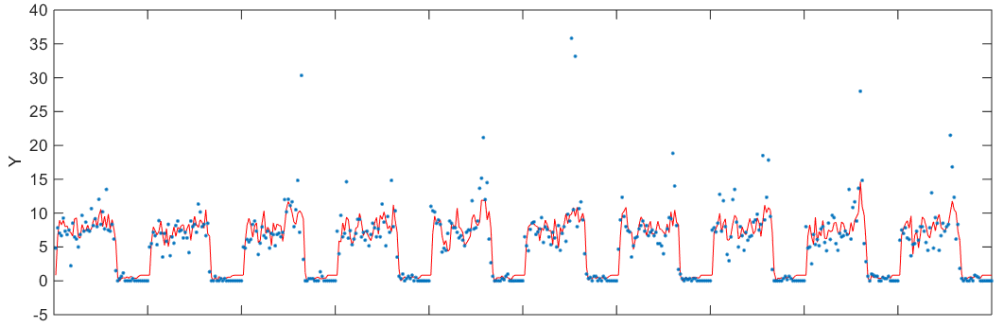
As figuras 31a, 31b e 31c ilustram como os modelos treinados imputariam os dados de entrada. Note que nesse caso os dados de entrada são justamente os dados de treinamento.



(a) Região 01 - Detector L020111



(b) Região 02 - Detector L0203111



(c) Região 03 - Detector L020462

Figura 31 – Imputação dos dados de treinamento

A função de imputação para as simulações está descrita no anexo C. Ela tem como entrada um vetor com todos os modelos treinados (contendo o nome os modelos vizinhos), o vetor com todas as medições, e o nome dos sensores seguindo a mesma ordem das medições. A saída são todos os valores de imputação. Após isso um API para Python faz a comunicação e repassa esses valores para o SIT/AIMSUM.

Os resultados de desempenho médios, entre 10 replicações de cada cenário, podem ser vistos na tabela 17 abaixo:

	Referencia		TUC_{imp}		MVS-MQ		
	Valor	$\sigma_{padrão}$	**Valor**	$\sigma_{padrão}$	**Valor**	$\sigma_{padrão}$	Unidade
Atraso médio	**161.19**	6.96	**181.65**	9.88	**164.28**	6.43	s/km
Densidade	**10.04**	0.46	**11.62**	0.75	**10.31**	0.49	veh/km
Fluxo	**6733.34**	64.94	**6733.34**	64.94	**6733.34**	64.94	veh/h
Velocidade harmônica	**15.75**	0	**14.46**	0	**15.54**	0	km/h
Fila virtual máxima	**81.9**	21.7	**83.6**	22.61	**138.1**	23.94	veh
Fila média	**177.08**	11.88	**223.31**	21.63	**185.37**	12.74	veh
Fila virtual média	**10.88**	4.03	**12.01**	3.35	**32.49**	8.19	veh
"Missed Turns"	**26.3**	5.38	**66.1**	7.56	**23.5**	4.84	#
Numero de trocas de pista	**1205.26**	11.41	**1229.51**	16.01	**1191.76**	16.71	#
Numero de paradas	**0.11**	0	**0.12**	0	**0.11**	0	#/veh.km
Velocidade	**20**	0.39	**18.82**	0.41	**19.88**	0.38	km/h
Tempo parado	**134.71**	6.46	**153.34**	9.27	**137.73**	5.75	sec/km
Total de trocas de pista	**39600.9**	375.05	**40397.7**	526.16	**39157.2**	548.93	#
Total de paradas	**51368**	1450	**57252**	2018	**51722**	1773	#
Tempo médio de viagem	**228.57**	6.95	**249.03**	9.87	**231.66**	6.41	sec/km
Tempo total de viagem	**751.48**	34.95	**863.92**	55.2	**770.36**	36.87	h
Distancia total viajado	**12498**	150.15	**12529**	153.62	**12498**	150.45	km

Tabela 17 – Desempenho do SIT utilizando diferentes métodos de imputação

7 Conclusões e Perspectivas

Nos últimos anos a **Maquina de Vetores de Suporte**, em especial sua versão por **Mínimos Quadrados**, tem ganhado cada vez mais espaço nas pesquisas acadêmicas Embora popular, ainda existem poucos trabalhos focados especificamente na perda de dados de trafego de malhas urbanas. Ainda assim este trabalho foi capaz de apresentar a teoria necessária para a implementação pratica do método por um SCT.

Como vimos pelos resultados apresentados, este novo método possui benefícios frente ao antigo (TUC_{imp}), a **MVS-MQ** consegue realizar a estimação dos dados de trafego de forma satisfatória, mesmo quando a perda de dados é uma perda longa. Em geral, os indicadores estatísticos do método proposto são superiores aos dois comparativos. Esses resultados foram obtidos mesmo quando os dados utilizados para o treinamento eram inadequados, seja por conterem muito ruido, seja por estarem incompletos.

Ainda, ao analisarmos o desempenho da malha simulada, vimos que existe uma melhora significativa no desempenho do SCT de Santos, ao incluirmos este método de imputação como pré-processamento. Portanto, fica claro que a MVS-MQ é uma alternativa valida para ser implementada na malha da baixada santista.

Com a implementação de um sistema de imputação adequado, a malha de Santos ganhara eficiência e por consequência a vida das pessoas e a economia da cidade também sofreram impactos positivos. Ainda, a empresa BRASCONTROL ao dominar uma tecnologia deste tipo ira enriquecer o seu portfólio de produtos umas vez que contara com um SCT mais eficiente.

Algumas ressalvas são necessárias, porem. Os algorítimos utilizados aqui implementam apenas uma parte, ainda que central, do método. Para que ele seja implementado efetivamente e embarcado no SCT, algumas adaptações precisam ser feitas. Já outras mudanças trariam benefícios. Essas adaptações e possíveis mudanças ficarão como sugestões para trabalhos futuros:

7.1 Trabalhos futuros

7.1.1 Treinamento do modelo

Como sabemos, com o método atual de treinamento, não possibilita que ele seja realizado de forma simultânea com o SCT (a cada ciclo de controle). O treinamento completo demanda uma grande carga computacional, e não é capaz de cumprir os requisitos de tempo real do SCT. Por isso é necessário se definir quando e como esse treinamento

ocorrerá.

Um trabalho definindo qual é a melhor escolha seria benéfico ao método. As perguntas pertinentes nesse caso seriam:

- O modelo será periodicamente atualizado?
- Com que frequência o modelo deve ser atualizado?
- Os novos dados irão substituir os velhos, ou serão adicionados ao banco de dados do treinamento?
- Os novos dados serão todos os medidos, ou haverá uma pré-seleção (e como)?
- Haverão múltiplos modelos menores, ou apenas um grande modelo?

A definição das questões levantadas aqui é fundamental. Para a implementação inicial uma escolha intuitiva é suficiente, mas claramente um estudo mais detalhado potencialmente traria melhoras resultados. Por exemplo, na literatura existem estudos sobre a implementação de uma **Maquina de Vetores de Suporte incremental**, neste caso a RVS, que realiza um re-treinamento a cada ciclo [36].

7.1.2 Solução esparsa

A MVS-MQ não possui uma solução esparsa, portanto a adição ilimitada de dados ira obviamente causar problemas de armazenamento e processamento. É possível, no entanto, forçar a condição esparsa ao algorítimo utilizando alguns métodos de remoção de dados.

Wang e Hu [65] sugerem um método iterativo, aonde os modelos são treinados, posteriormente os **vetores de suporte** com os menores βs são removidos, e então o método é re-treinado com esse numero reduzido de vetores de treinamento. O método repete a operação (sem estimar novos σ e γ) até que um certo critério de parada seja alcançado.

Não fica claro qual o aumento no custo computacional do treinamento, porem a eliminação dos vetores de suporte menos relevantes, podem tornar o modelo mais leve e rápido.

7.1.3 Identificação de erros

Durante o nosso trabalho, o tema da **identificação de perda de dados** não foi explorado. Inclusive, por muitas vezes os dados corrompidos foram até utilizados como dados de treinamento, ou dados de entrada. Existem vários benefícios em se identificar

corretamente os dados corrompidos. Essa informação permite um treinamento mais adequado, sem a adição de dados degradados ao modelo. Permite também que a imputação ocorra somente nos dados que necessitam, em contraste a realizar a estimação para todos os dados (como ocorreu em nosso ultimo ensaio).

Ainda, a identificação possibilita um tratamento para o caso de falhas múltipla de sensores correlacionados, ainda que a MVS seja robusta a elas. Por isso, um método de MVS-MQ que também inclua a identificação de erros é desejável.

7.1.4 Parâmetros internos da MVS

Como explicamos no trabalho, os parâmetros internos do método σ e γ tem grande influencia na resposta, e sua estimação representa o maior custo computacional do treinamento. Nesta pesquisa foi utilizado o algorítimo do Modelo de arrefecimento, mas existem outras metodologias que podem ser estudadas visando a melhora de desempenho.

Ainda, poderiam ser utilizadas técnicas para a redução dos dados de treinamento, nem que somente para a estimação dos parâmetros. A redução nos dados de treinamento melhora bastante o tempo necessário para a estimação de σ e γ.

7.2 Conclusão final

Em suma, a MVS-MQ pode ser utilizada como técnica de imputação de dados de trafego, mas as pesquisas acima citadas podem trazer melhorarias importantes para a sua eficiência e precisão.

Referências

1 BUREAU OF TRANSPORTATION STATISTICS. *National Transportation Statistics*. 2017. <https://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/national_transportation_statistics/index.html>. Citado na página 19.

2 SCHRANK, D. et al. Urban mobility scorecard. Agosto 2015. Citado na página 19.

3 EUROPEAN COMMISSION. *Roadmap to a single european transport area towards a competitive and resource efficient transport system*. Brussels, 2011. Citado na página 19.

4 DIAKAKI, C. *Integrated Control of Traffic Flow in Corridor Networks*. Tese (Doutorado) — Technical University of Crete, Janeiro 1999. Citado 5 vezes nas páginas 19, 20, 25, 26 e 34.

5 WANG, J.; ZOU, N.; CHANG, G. Travel time prediction: empirical analysis of missing data issues for advanced traveler information system applications. *Transp. Res. Rec.: J. Transp. Res. Board*, p. 81,91, 2008. Citado 2 vezes nas páginas 19 e 40.

6 KLEIN, L. A.; MILLS, D. R. G. M. K. Traffic detector handbook: Third edition—volume i. In: . [S.l.]: FHWA: Federal Highway Administration, 2006. Citado na página 20.

7 KELL, J. H.; FULLERTON, I. J. Manual of traffic signal design (2nd edition). 1991. Citado na página 20.

8 WANG, Y.; NIHAN, N. L. Can single-loop detectors do the work of dual-loop detectors? *Journal of Transportation Engineering*, n. 129, 2003. Citado na página 20.

9 ZHANG, Y.; LIU, Y. Data imputation using least squares support vector machines in urban arterial streets. November 2008. Citado na página 20.

10 AL-DEEK, H.; VENKATA, C.; CHANDRA, S. R. New algorithms for filtering and imputation of real time and archived dual-loop detector data in the i-4 data warehouse. *Proc., 83rd Transportation Research Board (TRB) Annual Meeting*, Washington D.C., 2004. Citado 3 vezes nas páginas 20, 39 e 40.

11 NI, D. et al. Multiple imputation scheme for overcoming the missing values and variability issues in its data. *Journal of transportation engineering*, December 2005. Citado na página 20.

12 ZHANG, Y.; LIU, Y. Missing traffic flow data prediction using least squares support vector machines in urban arterial streets. November 2008. Citado 7 vezes nas páginas 20, 39, 40, 41, 43, 52 e 60.

13 SUYKENS, J. et al. *Least Squares Support Vector Machines*. Singapore: [s.n.], 2002. ISBN 981-238-151-1. Citado 3 vezes nas páginas 21, 41 e 43.

14 GEROLIMINIS, N.; SKABARDONIS, A. Identification and analysis of queue spillovers in city street networks. *IEEE Transactions on Intelligent Transportation Systems*, v. 12, n. 4, p. 1107–1115, Dec 2011. ISSN 1524-9050. Citado na página 23.

15 CHEN, C. et al. Detecting errors and imputing missing data for single-loop surveillance systems. *Transportation Research Record*, v. 1855, n. 03-3507, p. 160–167. Citado na página 27.

16 Prefeitura de Santos. *Prefeitura de Santos Dados gerais sobre a cidade*. Disponível em: <http://www.santos.sp.gov.br/conheca-santos/dados-gerais>. Citado na página 28.

17 BRASCONTROL. *BRASCONTROL História da Empresa*. Disponível em: <http://www.brascontrol.com.br/br-empresa/historia/>. Citado na página 32.

18 Kraus Jr., W. et al. Cost effective real-time traffic signal control using the tuc strategy. *IEEE INTELLIGENT TRANSPORTATION SYSTEMS MAGAZINE*, Febuary 2010. Citado na página 33.

19 HUNT, P. B. et al. The scoot on-line traffic signal optimisation technique. *Traffic Eng.Contr.*, v. 23, p. 190—-199, 1982. Citado na página 33.

20 ROBERTSON, D. I.; BRETHERTON, R. D. Optimizing networks of traffic signals in real time – the scoot method. *IEEE Trans. Veh. Technol.*, v. 40, p. 11—-15, 1991. Citado na página 33.

21 HEAD, K. L.; MIRCHANDI, P. B.; SHELBY, S. The rhodes prototype: a description and some results. *Proc. 77th Transportation Research Board Annual Meeting*, Tucson, p. 1—-12, 1998. Citado na página 33.

22 MIRCHANDANI, P.; WANG, F.-Y. Rhodes to intelligent transportation systems. *IEEE Intell. Syst.*, v. 20, p. 10—-15, 2005. Citado na página 33.

23 GARTNER, N. H. Opac: a demand-responsive strategy for traffic signal control. *Transport. Res. Rec.*, v. 906, p. 75—-81, 1983. Citado na página 33.

24 FARGES, J. L.; KHOUDOUR, L.; LESORT, J. B. Prodyn: on site evaluation. *Proc. 3rd IEE Conf. Road Traffic Control*, London, U.K. Citado na página 33.

25 FERNANDEZ-MOCTEZUMA, R. J. et al. Toward management and imputation of unavailable data in online advanced traveler information systems. In: *2007 IEEE Intelligent Transportation Systems Conference*. [S.l.: s.n.], 2007. p. 24–29. ISSN 2153-0009. Citado na página 39.

26 TANG, J. et al. A hybrid approach to integrate fuzzy c-means based imputation method with genetic algorithm for missing traffic volume data estimation. *Transp. Res. Part C: Emerg. Technol.*, v. 51, p. 29—40, 2015. Citado na página 39.

27 TAN, H. et al. A tensor-based method for missing traffic data completion. *Transp. Res. Part C: Emerg. Technol.*, v. 28, p. 15—-27, 2013. Citado na página 39.

28 CASTRILLON, F. et al. Comparison of modeling approaches for imputation of video detection data in intelligent transportation systems. *Transportation Research Record: Journal of the Transportation Research Board*, v. 2308, p. 138–147, 2012. Disponível em: <https://doi.org/10.3141/2308-15>. Citado na página 39.

29 ALLISON, P. D. *Missing data (Vol. 136)*. [S.l.: s.n.], 2001. Citado na página 39.

30 CHEN, C. et al. Detecting errors and imputing missing data for single-loop surveillance systems. *Transportation Research Record: Journal of the Transportation Research Board*, v. 1855, p. 160–167, 2003. Citado 3 vezes nas páginas 39, 40 e 41.

31 Van Lint, J. W. C.; HOOGENDOORN, S. P.; van Zuylen, H. J. Accurate freeway travel time prediction with state-space neural networks under missing data. *Transp. Res. Part C: Emerg. Technol.*, v. 13, p. 347—369, 2005. Citado na página 39.

32 KIM, H.; LOVELL, D. J. Traffic information imputation using a linear model in vehicular ad hoc networks. In: *2006 IEEE Intelligent Transportation Systems Conference*. [S.l.: s.n.], 2006. p. 1406–1411. ISSN 2153-0009. Citado na página 39.

33 DAILEY, D. J. Improved error detection for inductive loop sensors. Washington State Dept. of Transportation, Olympia, Wash, 1993. Citado na página 39.

34 NIHAN, N. L. Aid to determining freeway metering rates and detecting loop errors. *Journal of Transportation Engineering*, v. 123, n. 6, 1997. Citado na página 39.

35 SMITH, B.; CONKLIN, J. Use of local lane distribution patterns to estimate missing data values from traffic monitoring systems. *Transportation Research Record: Journal of the Transportation Research Board*, v. 1811, p. 50–56, 2002. Citado na página 39.

36 SU, H.; ZHANG, L.; YU, S. Short-term traffic flow prediction based on incremental support vector regression. *Third International Conference on Natural Computation*, 2007. Citado 5 vezes nas páginas 39, 40, 41, 43 e 86.

37 SMITH, B.; BABICEANU, S. Investigation of extraction, transformation, and loading techniques for traffic data warehouses. *Transportation Research Record: Journal of the Transportation Research Board*, v. 1879, p. 9–16, 2004. Citado na página 40.

38 ZHONG, M.; SHARMA, S.; LINGRAS, P. Genetically designed models for accurate imputation of missing traffic counts. *Transportation Research Record: Journal of the Transportation Research Board*, v. 1879, p. 71–79, 2004. Citado na página 40.

39 NI, D. et al. A multiple imputation scheme for overcoming the missing values and variability issues in its data. *ASCE Journal of Transportation Engineering*, v. 131, p. 931–938, 12/2005 2005. ISSN 0733-947X. Citado na página 40.

40 NI, D.; LEONARD, J. D. Markov chain monte carlo multiple imputation for incomplete its data using bayesian networks. *Transportation Research Record*, v. 1935, p. 57–67, 2005. Citado na página 40.

41 STEIMETZ, S. S.; BROWNSTONE, D. Estimating commuters' "value of time" with noisy data: a multiple imputation approach. *Transportation Research Part B: Methodological*, v. 39, n. 10, p. 865–889, December 2005. Citado na página 40.

42 FARHAN, J.; FWA, T. Airport pavement missing data management and imputation with stochastic multiple imputation model. *Transportation Research Record: Journal of the Transportation Research Board*, v. 2336, p. 43–54, 2013. Citado na página 40.

43 SCHAFER, J. *Analysis of Incomplete Multivariate Data*. London: Chapman and Hall, 1997. Citado na página 40.

44 DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum-likelihood estimation from incomplete data via the em algorithm. *J. Royal Statist. Soc., Series B.*, v. 39, p. 1–38, 1977. Citado na página 40.

45 SMITH, B. L.; SCHERER, W. T.; CONKLIN, J. H. Exploring imputation techniques for missing data in transportation management systems. *Transportation Research Record 1836*, n. 03-2894. Citado na página 40.

46 QU, L. et al. PPCA-based missing data imputation for traffic flow volume: A systematical approach. *IEEE Transactions on Intelligent Transportation Systems*, v. 10, n. 3, p. 512–522, Sept 2009. ISSN 1524-9050. Citado na página 40.

47 LI, L.; LI, Y.; LI, Z. Efficient missing data imputing for traffic flow by considering temporal and spatial dependence. *Transportation research part C: emerging technologies*, v. 34, p. 108–120, 2013. Citado na página 40.

48 MING, Z.; LINGRAS, P.; SHARMA, S. Estimation of missing traffic counts using factor, genetic, neural, and regression techniques. *Transp. Res. Part C: Emerg. Technol.*, v. 12, p. 139—-166, 2004. Citado na página 40.

49 LV, Y. et al. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, v. 16, n. 2, p. 865–873, April 2015. ISSN 1524-9050. Citado na página 40.

50 VAPNIK., V. The nature of statistical learning theory. *New York: Springer Verlag*, 1995. Citado 3 vezes nas páginas 40, 43 e 48.

51 WU, C.-H.; HO, J.-M.; LEE, D. T. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems*, v. 5, n. 4, p. 276–281, Dec 2004. ISSN 1524-9050. Citado na página 41.

52 SUYKENS, J. A. K.; VANDEWALLE, J.; MOOR, B. D. Optimal control by least squares support vector machines. *Neural Netw.*, v. 14, 1998. Citado na página 41.

53 SANSOM, D. C.; DOWNS, T.; SAHA, T. K. Evaluation of support vector machine based forecasting tool in electricity price forecasting for australian national electricity market participants. *Journal of Electrical and Electronics Engineering, Australia*, v. 22, n. 3, p. 227–234, Jan 2003. ISSN 0725-2986. Citado na página 41.

54 CONGA, Y.; WANGA, J.; LIA, X. Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm. School of Communication Engineering, Jilin University, Changchun, Jilin, China, 2016. Citado 2 vezes nas páginas 41 e 64.

55 ZHAO-SHENG, Y.; YUAN, W.; QING, G. Short-term traffic flow prediction method based on svm. College of transportation, Jilin University, Changhun, 13002, China, 2005. Citado na página 41.

56 BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. New York, NY, USA: ACM, 1992. (COLT '92), p. 144–152. ISBN 0-89791-497-X. Disponível em: <http://doi.acm.org/10.1145/130385.130401>. Citado na página 43.

57 AIZERMAN, M. A.; BRAVERMAN, E. A.; ROZONOER, L. Theoretical foundations of the potential function method in pattern recognition learning. In: *Automation and Remote Control,*. [S.l.: s.n.], 1964. (Automation and Remote Control„ 25), p. 821–837. Citado 2 vezes nas páginas 43 e 50.

58 JOACHIMS, T. Text categorization with support vector machines: Learning with many relevant features. Springer Berlin Heidelberg, Berlin, Heidelberg, p. 137–142, 1998. Citado na página 43.

59 DRUCKER, H. et al. Support vector regression machines. In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 9*. [S.l.]: MIT Press, 1997. p. 155–161. Citado na página 43.

60 SUYKENS, J. A. K.; VANDEWALLE, J.; MOOR, B. D. Weighted least squares support vector machines-robustness and sparse approximation. *Neurocomputing*, v. 48, 2002. Citado na página 43.

61 LóPEZ, J.; SUYKENS, J. A. First and second order smo algorithms for ls-svm classifiers. *Neural Process. Lett.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 33, n. 1, p. 31–44, fev. 2011. ISSN 1370-4621. Disponível em: <http://dx.doi.org/10.1007/s11063-010-9162-9>. Citado 3 vezes nas páginas 43, 48 e 56.

62 KEERTHI, S. S.; SHEVADE, S. K. SMO algorithm for least-squares svm formulations. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 15, n. 2, p. 487–507, fev. 2003. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/089976603762553013>. Citado 4 vezes nas páginas 43, 56, 64 e 65.

63 MALLINSON, H.; GAMMERMAN, A. Imputation using support vector machines. 2003. Citado 4 vezes nas páginas 43, 49, 50 e 51.

64 NG, A. *Lecture notes in CS 229 Machine Learning - Lecture notes 3: Support Vector Machines*. Stanford University, 2016. Disponível em: <http://cs229.stanford.edu/materials.html>. Citado na página 43.

65 WANG, H.; HU, D. Comparison of svm and ls-svm for regression. In: *2005 International Conference on Neural Networks and Brain*. [S.l.: s.n.], 2005. v. 1, p. 279–283. Citado 4 vezes nas páginas 43, 53, 56 e 86.

66 KUHN, H. W.; TUCKER, A. W. Nonlinear programming. In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*. Berkeley, Calif.: University of California Press, 1951. p. 481–492. Citado na página 47.

67 FLETCHER, R. *Practical methods of optimization, Volume 1*. John Wiley & Sons, Ltd, 1987. ISBN 9781118723203. Disponível em: <http://dx.doi.org/10.1002/9781118723203.ch1>. Citado na página 47.

68 PELCKMANS, K. et al. *LS-SVMlab: a MATLAB/C toolbox for Least Squares Support Vector Machines*. 2002. Citado na página 57.

69 HINES, J. W.; AL. et. *Technical Review of On-Line Monitoring Techiniques for Perfomance Assessment Vol: 2 Theoretical issues*. U.S. Nuclear Regulatory Commission Office of Nuclear Regulatory Research: [s.n.], 2008. Citado na página 60.

70 NI, D. et al. A multiple imputation scheme for overcoming the missing values and variability issues in its data. *ASCE Journal of Transportation Engineering*, v. 131, p. 931–938, 12/2005 2005. Citado na página 60.

71 SUN, S.; ZHANG, C. The selective random subspace predictor for traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems*, v. 8, n. 2, p. 367–373, June 2007. ISSN 1524-9050. Citado na página 61.

72 LIAO, R. et al. Particle swarm optimization-least squares support vector regression based forecasting model on dissolved gases in oil-filled power transformers. *Electr. Power Syst. Res.*, v. 81, p. 2074–2080, 2011. Citado na página 64.

73 WU, Q. Hybrid model based on wavelet support vector machine and modified genetic algorithm penalizing gaussian noises for power load forecasts. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 38, n. 1, p. 379–385, jan. 2011. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2010.06.075>. Citado na página 64.

74 dos Santos, G. S. et al. Least squares support vector machines with tuning based on chaotic differential evolution approach applied to the identification of a thermal process. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 39, n. 5, p. 4805–4812, abr. 2012. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2011.09.137>. Citado na página 64.

75 SULAIMANA, M. et al. An application of artificial bee colony algorithm with least squares supports vector machine for real and reactive power tracing in deregulated power system. *International Journal Of Electrical Power & Energy Systems*, v. 37, n. 1, p. 67–77, 2012. Citado na página 64.

76 SUYKENS, J. A. K.; VANDEWALLE, J.; MOOR, B. D. Intelligence and cooperative search by coupled local minimizers. *International Journal of Bifurcation and Chaos*, v. 11, n. 8, 2001. Citado na página 64.

Anexos

ANEXO A – MATLAB: Algorítimo de Treinamento e Estatísticas de Erro

```
function SVMmodel = FindSVMModel(TrainingData, SimulationData, StationSelected, nAtributes, nDelay, nTucAtributes, SaveName)
%% Find StationSelected idx from Name(String)
    if ischar(StationSelected)
        StationSelected = find(ismember(TrainingData.Properties.VariableNames, StationSelected));
        if isempty(StationSelected)
            error('StationSelected is not a sensor from table');
        end
    end

%% Clean Training data - Eliminating all rows from Ytraining = 0
    if ~any(TrainingData{:,StationSelected})
        display('All data empty, training stoped')
        SVMmodel = [];
        return
    end
        TrainingData = TrainingData(find(TrainingData{:,StationSelected}),:);

%% Find timestamps
    Ttraining = regexp(TrainingData.Properties.RowNames,' ', 'split');%Split date from time
    %Eliminate empty cells (generated because of tabulation)
        Ttraining = [Ttraining{:}];
        Ttraining = reshape(Ttraining(~cellfun(@isempty,Ttraining)),2,[]) ';
    TraDays = length(unique(Ttraining(:,1)));%Get total of traning days
    UniqueTimes = unique(Ttraining(:,2),'stable');%Get mesured times
    nSamplesDay = length(UniqueTimes);%Get max number of mesures(sameday)

   Tsimulation = regexp(SimulationData.Properties.RowNames,' ', 'split');%Split date from time
   %Eliminate empty cells (generated because of tabulation)
        Tsimulation = [Tsimulation{:}];
```

```
30          Tsimulation = reshape(Tsimulation(~cellfun(@isempty,Tsimulation)),2,[])';
31
32  %% Pearson correlation coefficient
33      CorrelationTable = [];
34      for delay = 0:nDelay
35          CorrelationTable = [CorrelationTable ;  ...
36              abs(corr(TrainingData{1+delay:end,StationSelected},...
37              TrainingData{1:end-delay,:}))];
38      end
39      CorrelationTable(isnan(CorrelationTable)) = 0;  %Eliminte Nan values
40      CorrelationTable(:,StationSelected) = [];          %Eliminate colum of the
        StationSelected.
41      [~,idxAtrib] = sort(reshape(CorrelationTable,[],1),'descend');  %Get the ordered
        idx
42      [delay,idxAtrib] = ind2sub(size(CorrelationTable),idxAtrib); %Gets colum (
        atribute) and row (delay)
43
44      CorrelationTable = array2table(CorrelationTable);%Transforms to table
45      %Add name of stations
46          NamesStations = TrainingData.Properties.VariableNames;%Get names
47          NamesStations(StationSelected) = [];%Exclude StationSelected
48          CorrelationTable.Properties.VariableNames = NamesStations; %Addnames
49      NeighborsTuc = unique(idxAtrib,'stable')';
50      NeighborsTuc = NeighborsTuc(1:nTucAtributes);
51
52      idxAtrib(nAtributes+1:end) = [];     %Select only nAtributes atributes
53      SelectedTable = CorrelationTable(:,unique(idxAtrib,'stable'))  %Creates table
        with only Selected Stations
54
55  %% Create Vectors: Training Vector (X,Y), Test Vector (X,Y);
56      Xtraining = [];
57      Xtest = [];
58      for i = 1:nAtributes
59          Xtraining = [Xtraining ...
60              circshift(TrainingData{:,idxAtrib(i)},delay(i)-1)];
61          Xtest = [Xtest ...
62              circshift(SimulationData{:,idxAtrib(i)},delay(i)-1)];
63      end
64      XTucTraining = TrainingData{:,NeighborsTuc};
```

```
65      XTucSimu = SimulationData{:,NeighborsTuc};
66
67      Ytraining = TrainingData{:,StationSelected};
68      Yreal = SimulationData{:,StationSelected};
69
70  %% Historical Avarage Estimator
71      Yhist = [];
72      for timeStamp = 1:length(UniqueTimes)
73          idxTime = ismember(Ttraining(:,2),UniqueTimes(timeStamp));
74          Yhist = [Yhist; mean(Ytraining(idxTime))];
75      end
76      %Yhist = smooth(Yhist);  %Historic smooth for a full day
77      %Plot Historical Avarage
78          DayHistFig = figure;
79                  PlotHist = plot(Yhist,'DisplayName','DayHist');
80          hold on
81                  plot(smooth(Yhist),'DisplayName','Smooth DayHist');
82                          PlotHist.Parent.XLim(end) = length(UniqueTimes)-1;
83  %                       PlotHist.Parent.XTick(end+1) = length(UniqueTimes)-1;
84                          PlotHist.Parent.XTickLabel = UniqueTimes(PlotHist.Parent.
    XTick+1);
85                          PlotHist.Parent.XTickLabelRotation = 15;
86          legend('show');
87          title (['HistAvarage - ' TrainingData.Properties.VariableNames{
    StationSelected}]);
88      %Recreate Y base on the timestamps
89          YsimHist = Yreal; %Just pre-alocation space
90          for timeStamp = 1:length(UniqueTimes)
91              %Define output as historic value from the same timestamp
92              idxTime = ismember(Tsimulation(:,2),UniqueTimes(timeStamp));
93              YsimHist(idxTime) = Yhist(timeStamp);
94          end
95
96  %% Tuc imputation
97      Xhist = [];
98      for timeStamp = 1:length(UniqueTimes)
99          idxTime = ismember(Ttraining(:,2),UniqueTimes(timeStamp));
100         Xhist = [Xhist; mean(XTucTraining(idxTime,:),1)];
101     end
```

```
102     CoeTucInputation = repmat(Yhist,1,nTucAtributes)./Xhist;
103     CoeTucInputation(isinf(CoeTucInputation)) = NaN;      %Eliminte inf values
104     CoeTucInputation = nanmean(CoeTucInputation);   %Get the coeficientes
105     %Calculate the estimation from TUC
106         YsimTuc = Yreal; %Just pre-alocation space, since they will have the same
        size
107         for idx = 1:length(YsimTuc)
108             YsimTuc(idx) = mean(XTucSimu(idx,:).*CoeTucInputation);
109         end
110         YsimTuc = max(min(YsimTuc,100),0); %Limit to [0,100]
111 %% SVM simulation
112     % Reduce by 3 the data for the Tunning
113     SVMmodel = initlssvm(Xtraining(1:3:end,:),Ytraining(1:3:end,:),'function
        estimation' ,[],[], 'RBF_kernel');
114     % Anneling algorithm, with simplex for gamma sigma tunning
115     SVMmodel = tunelssvm(SVMmodel,'simplex','rcrossvalidatelssvm',{TraDays,'mse'
        });
116     SVMmodel = initlssvm(Xtraining,Ytraining,'function estimation',SVMmodel.gam,
        SVMmodel.kernel_pars,'RBF_kernel');
117     % SMO solution
118     SVMmodel = trainlssvm(SVMmodel);
119     SVMmodel.Selected = [SelectedTable];
120     TrainedFig = figure;
121     plotlssvm(SVMmodel);
122         title ([ 'Training - ' TrainingData.Properties.VariableNames{StationSelected
        }]);
123     % Simulate SVM
124     Ysim = max(min(simlssvm(SVMmodel,Xtest),100),0);
125
126 %% Plot results
127     ImputationFig(1) = figure;
128     ImputationFig(1).Position = [200 200 1000 400];
129 %     ImputationFig(1).PaperPosition = [0 0 800 200];
130     linereal  = plot(Yreal,'k','LineStyle',':','Marker','.','MarkerSize',8,'
        DisplayName','Real');
131     hold on;
132     linesvm = plot(Ysim,'-b','DisplayName','SVM');
133     title ([ 'Imputation SVM ' TrainingData.Properties.VariableNames{StationSelected
        }]);
```

```
134     legend('show');
135
136     ImputationFig(2) = figure;
137     ImputationFig(2).Position = [200 200 1000 400];
138 %     ImputationFig(2).PaperPosition = [0 0 800 200];
139      linereal  = plot(Yreal,'k','LineStyle',':','Marker','.','MarkerSize',8,'
        DisplayName','Real');
140     hold on;
141     lineH = plot(YsimHist,'-g','DisplayName','Historico');
142      title (['Imputation Historic ' TrainingData.Properties.VariableNames{
        StationSelected}]);
143     legend('show');
144
145     ImputationFig(3) = figure;
146     ImputationFig(3).Position = [200 200 1000 400];
147 %     ImputationFig(3).PaperPosition = [0 0 800 200];
148      linereal  = plot(Yreal,'k','LineStyle',':','Marker','.','MarkerSize',8,'
        DisplayName','Real');
149     hold on;
150     lineTuc = plot(YsimTuc,'-m','DisplayName','TUC_{imp}');
151      title (['Imputation TUC_{imp}' TrainingData.Properties.VariableNames{
        StationSelected}]);
152     legend('show');
153
154
155 %% Error Calcularions
156     Results = [Ysim YsimHist YsimTuc];
157     eEval = ones(length(Results(1,:)),5);
158     for  i = 1:length(Results(1,:))
159         e = Yreal - Results(:,i);
160         %Mean Absolute percentual Error
161             mape = abs(e)./Yreal;
162             mape = mean(mape(isfinite(mape)));
163         eEval(i,:)  = [linf(e)  mae(e) medae(e) mse(e)  mape];
164     end
165     eEval = array2table(eEval);
166     eEval.Properties.VariableNames = {'Max', 'MAE','MEDAE','MSE','MAPE'};
167     eEval.Properties.RowNames = {'MVS-QM', 'Hist','TUC'}
168     SVMmodel.mError = eEval;
```

```
169
170 %% Save info
171     %FolderDefinition
172     str = strcat('C:\Users\ATOSO\Desktop\Toso\', SaveName, '\',...
173     char(TrainingData(1,StationSelected).Properties.VariableNames),'_',mat2str(
        nSamplesDay), '\');
174     mkdir(str(1:end-1));
175     %Save Tables
176     writetable(CorrelationTable,[str 'CorrTable.csv'], 'Delimiter',';','
        WriteRowNames',true);
177     writetable(SelectedTable,[str 'CorrTableSelected.csv'], 'Delimiter',';','
        WriteRowNames',true);
178     writetable(eEval,[str 'error.csv'], 'Delimiter',';','WriteRowNames',true);
179     %Save Figs
180     savefig(DayHistFig,[str 'DayHistFig']);
181     savefig(TrainedFig,[str 'TrainedFig']);
182     savefig(ImputationFig,[str 'ImputationFig']);
183         print(ImputationFig(1),[str 'svm'],'-dpng');
184         print(ImputationFig(2),[str 'hist'],'-dpng');
185         print(ImputationFig(3),[str 'tuc'],'-dpng');
186     %Save suport data
187     save([str 'ModelData'],'SVMmodel','StationSelected', 'nAtributes', 'nDelay', '
        nTucAtributes');
188 end
```

ANEXO B – PostgreSQL: Agregação dos Dados

```
1  DROP TABLE IF EXISTS geral;
2
3  CREATE TEMP TABLE geral AS
4  with const AS (select 300 as periodo) --Create constants: Period of agregation
5  SELECT
6  wt_links.cod_link as Link, --Refers to link/station of detectors
7  substring (cod_detector from 1 for 6) as PrefixoGrupo, --Prefix is equivalent to Link
8  avg(number_of_lanes) as nVias, --Number of detectors in a station/link
9  to_char((TIMESTAMP WITH TIME ZONE 'epoch' + INTERVAL '1 second' * floor(
       date_part('epoch', MIN(detection_start_time)) / periodo) * periodo), 'YYYY-
       MM-DD HH24:MI:SS') as Hora_inicio, --Gets Starting time of the pre-definied
       interval
10 (COALESCE(SUM(vehicle_detection_period)*100/periodo, 0)) AS OcupacaoTotal,
       --Adds up all the ocupation from the same station/link
11  COALESCE(count(vehicle_detection_period),0) AS Flow --Estimates the flow
12 FROM wt_op_detector, wt_detectors, wt_link_detector_sets, wt_links --Diferent
       tables in the database
13 cross join const --Adds Constants
14 --Query for specific times
15 WHERE (
16 detection_start_time Between '2016-09-01 00:00:00'::timestamp AND '2016-09-01
       23:59:59'::timestamp
17 OR detection_start_time Between '2016-09-08 00:00:00'::timestamp AND '
       2016-09-08 23:59:59'::timestamp
18 OR detection_start_time Between '2016-09-15 00:00:00'::timestamp AND '
       2016-09-15 23:59:59'::timestamp
19 OR detection_start_time Between '2016-09-22 00:00:00'::timestamp AND '
       2016-09-22 23:59:59'::timestamp
20 OR detection_start_time Between '2016-09-29 00:00:00'::timestamp AND '
       2016-09-29 23:59:59'::timestamp
21 OR detection_start_time Between '2016-10-06 00:00:00'::timestamp AND '
       2016-10-06 23:59:59'::timestamp
22 OR detection_start_time Between '2016-10-13 00:00:00'::timestamp AND '
```

```
    2016-10-13 23:59:59'::timestamp
OR detection_start_time Between '2016-10-20 00:00:00'::timestamp AND '
    2016-10-20 23:59:59'::timestamp
OR detection_start_time Between '2016-10-27 00:00:00'::timestamp AND '
    2016-10-27 23:59:59'::timestamp
OR detection_start_time Between '2016-11-03 00:00:00'::timestamp AND '
    2016-11-03 23:59:59'::timestamp)
--Relate the columns from the several tables
AND wt_op_detector.seq_detector = wt_detectors.seq_detector
AND wt_detectors.seq_detector = wt_link_detector_sets.seq_detector
AND wt_link_detector_sets.seq_link = wt_links.seq_link
-------------------------------------
--      AND wt_links.cod_link = 'L020403'       -- Select specific link
GROUP BY Link, PrefixoGrupo, floor(date_part('epoch', detection_start_time) /
    periodo), periodo -- Group by prefix/link and time interval | constant (periodo) is
     the same for all.
ORDER BY PrefixoGrupo, Hora_inicio; -- Order by link, and time

UPDATE geral --Get the avarage from station and limit to 100%
SET OcupacaoTotal= least(OcupacaoTotal/nVias, 100);

--Save the partial table
Copy (Select * From geral) To '.\data\tempGeral.csv' With CSV DELIMITER ',';

Copy (SELECT *
FROM   crosstab( --Pivote for the predefined links.
'SELECT Hora_inicio, PrefixoGroupo, OcupacaoTotal
FROM   geral
ORDER  BY 1',
$$SELECT unnest('{D02061,D02111,D02121,D02122,D02142,D02171,D02172,D02173,
    D03011,D03012,D03021,D03022,D03023,D03031,D03032,D03033,D03041,D03042,
    D03043,D03051,D03052,D03053,D03061,D03062,D03071,D03072,D03073,D03074,
    D03081,D03082,D03091,D03092,D03093,D03101,D03102,D03103,D03104,D03111,
    D03112,D03113,D03121,D03122,D03123,D03124,D03131,D03132,D03133,D03134,
    D03135,D03136,D03137,D03141,D03142,D03143,D03151,D03152,D03161,D03162,
    D03163,D03171,D03172,D03173,D03174,D03175,D03181,D03182,D03191,D03192,
    D03193,D03201,D03202,D03203,D03211,D03212,D03221,D03222,L01011,L01012,
    L01013,L01014,L01015,L01021,L01022,L01023,L01031,L01032,L01041,L01042,
    L01043,L01044,L01051,L01052,L01053,L01061,L01062,L01063,L01071,L01072,
```

```
    L01081,L01082,L01083,L01091,L01092,L01093,L01094,L02011,L02012,L02013,
    L02021,L02022,L02023,L02031,L02032,L02033,L02041,L02042,L02043,L02044,
    L02045,L02046,L02051,L02052,L02061,L02062,L02063,L02064,L02071,L02072,
    L02073,L02081,L02082,L02091,L02092,L02093,L02101,L02102,L02103,L02111,
    L02112,L02113,L02121,L02122,L02124,L02125,L02131,L02132,L02133,L02134,
    L02135,L02136,L02141,L02142,L02143,L02144,L02145,L02151,L02161,L02162
47  }' :: text []) $$)
48  AS ct ("Hora_inicio" text,  "D02061" text,"D02111" text,"D02121" text,"D02122" text,"
    D02142" text,"D02171" text,"D02172" text,"D02173" text,"D03011" text,"D03012"
    text,"D03021" text,"D03022" text,"D03023" text,"D03031" text,"D03032" text,"
    D03033" text,"D03041" text,"D03042" text,"D03043" text,"D03051" text,"D03052"
    text,"D03053" text,"D03061" text,"D03062" text,"D03071" text,"D03072" text,"
    D03073" text,"D03074" text,"D03081" text,"D03082" text,"D03091" text,"D03092"
    text,"D03093" text,"D03101" text,"D03102" text,"D03103" text,"D03104" text,"
    D03111" text,"D03112" text,"D03113" text,"D03121" text,"D03122" text,"D03123"
    text,"D03124" text,"D03131" text,"D03132" text,"D03133" text,"D03134" text,"
    D03135" text,"D03136" text,"D03137" text,"D03141" text,"D03142" text,"D03143"
    text,"D03151" text,"D03152" text,"D03161" text,"D03162" text,"D03163" text,"
    D03171" text,"D03172" text,"D03173" text,"D03174" text,"D03175" text,"D03181"
    text,"D03182" text,"D03191" text,"D03192" text,"D03193" text,"D03201" text,"
    D03202" text,"D03203" text,"D03211" text,"D03212" text,"D03221" text,"D03222"
    text,"L01011" text,"L01012" text,"L01013" text,"L01014" text,"L01015" text,"L01021"
     text,"L01022" text,"L01023" text,"L01031" text,"L01032" text,"L01041" text,"L01042
    " text,"L01043" text,"L01044" text,"L01051" text,"L01052" text,"L01053" text,"
    L01061" text,"L01062" text,"L01063" text,"L01071" text,"L01072" text,"L01081" text,
    "L01082" text,"L01083" text,"L01091" text,"L01092" text,"L01093" text,"L01094" text
    ,"L02011" text,"L02012" text,"L02013" text,"L02021" text,"L02022" text,"L02023"
    text,"L02031" text,"L02032" text,"L02033" text,"L02041" text,"L02042" text,"L02043"
     text,"L02044" text,"L02045" text,"L02046" text,"L02051" text,"L02052" text,"L02061
    " text,"L02062" text,"L02063" text,"L02064" text,"L02071" text,"L02072" text,"
    L02073" text,"L02081" text,"L02082" text,"L02091" text,"L02092" text,"L02093"
    text,"L02101" text,"L02102" text,"L02103" text,"L02111" text,"L02112" text,"
    L02113" text,"L02121" text,"L02122" text,"L02124" text,"L02125" text,"L02131"
    text,"L02132" text,"L02133" text,"L02134" text,"L02135" text,"L02136" text,"L02141
    " text,"L02142" text,"L02143" text,"L02144" text,"L02145" text,"L02151" text,"
    L02161" text,"L02162" text)
49  --Save final table
50  ) To '.\data\Pivoted.csv' With CSV DELIMITER ',';
```

ANEXO C – MATLAB: Imputação para Simulações de Desempenho

```
function y = SVM_func (SvmTreinado,dados, NomesVariaveis )
%%    Clean empty data

%Make data into a table with the detectors names.
Variaveis = array2table(dados);
Variaveis.Properties.VariableNames = cellstr(NomesVariaveis);

%Itarates for all models in SvmTreinado
y = [];
for entradas = 1:width(Variaveis)
    %Find the respective model
    modelo = SvmTreinado.(Variaveis.Properties.VariableNames{entradas});
    Xteste = [];
    %Find the neighbors
    for rows = 1:length(modelo.Selected.Properties.RowNames)
        Xteste = [Xteste, Variaveis.(modelo.Selected.Properties.RowNames{rows})];
    end
    %Get the values
    y = [y, max(min(simlssvm(modelo, Xteste),100),0)];
end

%Output as a table
 y = array2table(y);
 y.Properties.VariableNames = Variaveis.Properties.VariableNames;
```