UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS TRINDADE
BACHARELADO EM MATEMÁTICA

Andrey Chiarelli de Souza Lolo Brigida

Métodos de gradiente estocástico para treinamento de redes neurais

Florianópolis
2023

Andrey Chiarelli de Souza Lolo Brigida

Métodos de gradiente estocástico para treinamento de redes neurais

Trabalho de Conclusão de Curso de Graduação em Bacharelado em Matemática do Campus Trindade da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Matemática.
Orientador: Prof. Douglas S. Gonçalves, Dr.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lolo Brigida, Andrey Chiarelli de Souza
Métodos de gradiente estocástico para treinamento de redes neurais / Andrey Chiarelli de Souza Lolo Brigida ; orientador, Douglas Soares Gonçalves, 2023.
58 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro de Ciências Físicas e Matemáticas, Graduação em Matemática - Bacharelado, Florianópolis, 2023.

Inclui referências.

1. Matemática - Bacharelado. 2. Treinamento de redes neurais. 3. Gradiente estocástico. 4. Problemas de classificação. I. Gonçalves, Douglas Soares. II. Universidade Federal de Santa Catarina. Graduação em Matemática - Bacharelado. III. Título.

Andrey Chiarelli de Souza Lolo Brigida

Métodos de gradiente estocástico para treinamento de redes neurais

Este Trabalho Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Matemática e aprovado em sua forma final pelo Curso de Bacharelado em Matemática.

Florianópolis, 28 de Novembro de 2023.

Documento assinado digitalmente
Felipe Lopes Castro
Data: 20/12/2023 14:37:29-0300
CPF: ***.060.710-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Felipe Lopes Castro, Dr.
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Douglas Soares Goncalves
Data: 15/12/2023 18:16:16-0300
CPF: ***.566.318-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Douglas S. Gonçalves, Dr.
Orientador
UFSC

Prof. Mauro Roisenberg, Dr.
Avaliador
UFSC

Prof. Wagner Barbosa Muniz, Dr.
Avaliador
UFSC

Aos meus avós, à minha família, aos meus amigos. A Valquíria, Tali e Naomi.

AGRADECIMENTOS

Ao meu orientador, Douglas S. Gonçalves. Aos membros da banca, Mauro Roisenberg e Wagner Barbosa Muniz. À Universidade Federal de Santa Catarina.

RESUMO

O termo "*machine learning*", especialmente sob a forma de *redes neurais*, tem ganhado especial notoriedade nas últimas décadas, com uma miríade de aplicações. Neste trabalho, primeiro buscamos descrever matematicamente uma classe de redes neurais, através da composição de certas funções. Cada função componente depende de certos parâmetros que devem ser ajustados para que a discrepância (função perda) entre a previsão do modelo (rede neural) e o resultado de observações seja minimizada. Este processo, chamado de *treinamento*, pode ser formulado como um problema de otimização. Como na prática, o número de observações/amostras é muito grande, métodos determinísticos de otimização não se aplicam, e uma alternativa são os métodos de gradiente estocástico (MGE). Seguimos então para uma análise teórica deste tipo de método, e revisitamos resultados de convergência sob certas hipóteses de suavidade e convexidade da função perda. Mais especificamente, foram estudados resultados de convergência em valor esperado para a função objetivo, no caso de funções fortemente convexas, e da norma do gradiente no caso de funções mais gerais. Feito isso, ilustramos tais resultados teóricos através da aplicação do MGE em alguns problemas teste específicos. Por fim, o MGE foi aplicado no treinamento de duas redes neurais profundas para um problema de classificação multiclasse, de reconhecimento de dígitos manuscritos. As medidas de acurácia das redes após o treinamento foram consideradas satisfatórias e competitivas, quando comparadas com resultados prévios da literatura, e parecem indicar a eficácia do método.

Palavras-chave: Redes neurais. Treinamento de redes neurais. Gradiente estocástico. Problemas de classificação. *Deep learning*.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O termo "rede neural" (RN) denota uma grande classe de modelos estatísticos não-lineares e métodos de aprendizado [4, Capítulo 11]. Em geral, utilizamos uma rede neural para fazer predições sobre o *output* de um certo conjunto de dados/características observados. O procedimento, a grosso modo, dá-se da seguinte maneira: tomamos um conjunto de amostras com seus respectivos *outputs* previamente registrados (chamado *conjunto de treino*). Em seguida, "treinamos" uma dada rede neural sobre esse conjunto – neste ponto, limitemo-nos a dizer que tal treinamento consiste em ajustar o modelo aos dados observados. Feito isso, queremos ser capazes de aplicar a rede neural sobre um novo conjunto de amostras com características semelhantes ao conjunto de treino, mas cujos *outputs* desconhecemos, com a intenção de, como já mencionado, prever tais *outputs* para esse novo conjunto [4, Capítulo 11].

Dentre as muitas áreas em que vemos aplicações para RNs, citamos algumas: reconhecimento de imagens e sons; predição de trânsito e transporte; análise preditiva e *decision-making* inteligente para negócios; auxílio em diagnóstico médico por exames de imagem. Para mais alguns exemplos, recomendamos a leitura de [11].

Como objetivo geral, o presente trabalho tem por fim estudar e expor os aspectos matemáticos envolvidos neste processo. Mais especificamente, descrever matematicamente o modelo representado por uma rede neural, explicitar o problema de otimização envolvido na fase de treinamento, ou ajuste de parâmetros, desta rede, e o estudo teórico (análise de convergência) de métodos de gradiente estocástico (MGE) usados na resolução de tal problema de otimização. Para cumprir tal tarefa, estabelecemos alguns objetivos específicos, estruturados e desenvolvidos no texto da seguinte maneira.

No Capítulo 2, nos propomos a apresentar a caracterização matemática de um tipo de rede neural, a saber, de uma *rede neural profunda*. Tal caracterização é baseada majoritariamente em [4, Capítulo 11]. Além disso, formulamos o treinamento de uma RN como um problema de otimização.

A fim de lidar com tal problema de otimização, no Capítulo 3, nosso foco é apresentar e expor alguns resultados de convergência para métodos de gradiente estocástico. Com base na referência [2], são considerados resultados de convergência em valor esperado para a função objetivo, no caso de funções fortemente convexas, e da norma do gradiente no caso de funções mais gerais.

No Capítulo 4, formulamos experimentos numéricos simples, enolvendo minimização de quadráticas, para ilustrar o comportamento do método e alguns resultados teóricos do capítulo anterior.

Por fim, no Capítulo 5, um MGE foi aplicado no treinamento de duas redes neurais profundas para resolver um problema de classificação sobre reconhecimento de algarismos numéricos escritos à mão; para tal, utilizamos o *dataset* MNIST bem conhecido na literatura [4, Capítulo 11].

Também incluímos um Apêndice A com alguns conceitos de Estatística que julgamos necessários para o entendimento do texto; e outro, Apêndice B, com uma discussão sobre *backpropagation*, importante para o cálculo eficiente de derivadas parciais da função perda. Pressupomos que o leitor tenha familiaridade com alguns conceitos de Álgebra Linear, Cálculo Vetorial e/ou Análise, e Probabilidade.

2 BREVE INTRODUÇÃO A REDES NEURAIS

Como dissemos anteriormente, o termo "rede neural" denota uma grande classe de modelos estatísticos não-lineares e métodos de aprendizado (ver Apêndice A). Neste capítulo apresentaremos a formulação matemática precisa de uma *rede neural profunda*.

2.1 CARACTERIZAÇÃO DO MODELO

Vamos começar descrevendo um tipo específico de rede neural, a *rede neural de uma camada oculta* [4, Capítulo 11]. Trata-se de um modelo de regressão ou classificação (ver apêndice A) em dois estágios representado por um diagrama de redes como o da Figura 1 [4, Capítulo 11].

O vetor $y \in \mathbb{R}^K$ é o vetor de saída (*output*) fornecido pelo modelo. Em um problema de classificação de K classes disjuntas, por exemplo, a k-ésima unidade (ou coordenada) y_k do vetor y representa a probabilidade do vetor $x = (x_1, ..., x_N) \in \mathbb{R}^N$, vetor de *input*, ser da classe k, com $k = 1, ..., K$. Assim, como trata-se de probabilidades, cada $y_k \geq 0$ e $\sum_{k=1}^{K} y_k = 1$.

Já o vetor $a \in \mathbb{R}^M$ é chamado de *camada oculta (hidden layer)*. As entradas de tal vetor resultam de uma transformação sobre o vetor de entrada. Cada coordenada a_m é comumente chamada de *neurônio* [4, Capítulo 11], e é caracterizada pela composição de uma função $h_m : \mathbb{R} \to \mathbb{R}$, chamada aqui de *função de ativação*, com uma função afim do vetor x, da seguinte forma:

$$a_m = h_m({w_m}^T x + b_m),$$

com $w_m \in \mathbb{R}^N$ e $b_m \in \mathbb{R}$, $m = 1, ..., M$, em que M representa o número de neurônios na camada oculta.

Por sua vez, o vetor de output y tem coordenadas da forma:

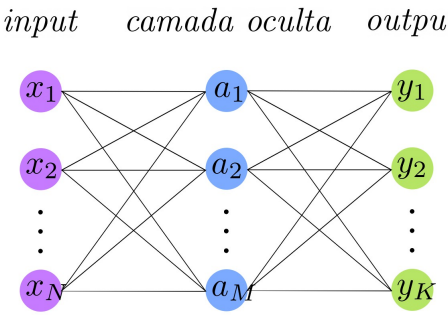

Figura 1 – Rede neural com uma camada oculta

$$y_k = g_k(\bar{w_k}^T a + \bar{b}_k)$$

sendo $g_k : \mathbb{R} \to \mathbb{R}$, com $\bar{w}_k \in \mathbb{R}^M$ e $\bar{b}_k \in \mathbb{R}$, $k = 1, ..., K$, em que K é o número de neurônios na camada de *output*.

Note que temos duas matrizes e dois vetores imediatamente associados ao problema, sendo as matrizes

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & ... & w_{1,N} \\ w_{2,1} & w_{2,2} & ... & w_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M,1} & w_{M,2} & ... & w_{M,N} \end{pmatrix}, \quad \bar{W} = \begin{pmatrix} \bar{w}_{1,1} & \bar{w}_{1,2} & ... & \bar{w}_{1,M} \\ \bar{w}_{2,1} & \bar{w}_{2,2} & ... & \bar{w}_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{w}_{K,1} & \bar{w}_{K,2} & ... & \bar{w}_{K,M} \end{pmatrix}.$$

e os vetores $b = (b_1, \ldots, b_M)^T$ e $\bar{b} = (\bar{b}_1, \ldots, \bar{b}_K)^T$. Chamaremos uma matriz e um vetor desse tipo, no contexto de camadas de uma rede neural, de *matriz de pesos* e vetor de *viés*, respectivamente. Tais pesos estão representados pelas linhas na Figura 1.

Para o caso geral, em um modelo com L camadas ocultas, chamado de *Rede Neural Profunda* (RNP), temos a seguinte forma para as coordenadas dos vetores intermediários:

$$a_j^l = h_j^l((w_j^l)^T a^{l-1} + b_j^l)$$

em que $l = 1, ..., L$, $j = 1, ..., N_l$, e N_l é o número de coordenadas da l-ésima camada. Naturalmente, associa-se um viés $b^l = (b^l_1, ..., b^l_{N_l})$ e uma matriz de pesos à tal camada:

$$W^l = \begin{pmatrix} w^l_{1,1} & w^l_{1,2} & ... & w^l_{1,N_{l-1}} \\ w^l_{2,1} & w^l_{2,2} & ... & w^l_{2,N_{l-1}} \\ \vdots & \vdots & \ddots & \vdots \\ w^l_{N_l,1} & w^l_{N_i,2} & ... & w^l_{N_l,N_{l-1}} \end{pmatrix}.$$

Sendo assim, podemos enxergar uma rede neural como uma composição de funcionais da forma $T^l(x) = W^l x + b^l$, com "funções de ativação" específicas h^l aplicadas coordenada a coordenada sobre o vetor imagem de T^l:

$$H(x) = h^L(T^L(\dots h^2(T^2(h^1(T^1(x)))))) = h^L \circ T^L \circ \cdots \circ h^1 \circ T^1(x),$$

com uma aplicação final $g : \mathbb{R}^K \to \mathbb{R}^K$ sobre um funcional $\bar{T}(H(x)) = \bar{W}H(x) + \bar{b}$. Tal aplicação pode servir, por exemplo, para transformar $\bar{T}(H(x))$ em um vetor de probabilidades, como discutido no início da seção. Em suma, uma rede neural $f(x)$ é dada por:

$$f(x) = g(\bar{T}(H(x)) = g \circ \bar{T} \circ H(x),$$

com $T^l : \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l}$ e $\bar{T} : \mathbb{R}^{N_L} \to \mathbb{R}^K$ e $f : \mathbb{R}^N \to \mathbb{R}^K$.

2.2 O *TREINO*

Uma vez definido o modelo da rede neural, nós passamos para a etapa chamada de *treinamento* [4]. Vamos caracterizá-la ao longo da presente seção.

Dados os seguintes conjuntos:

- $X = \{x^1, ..., x^n\} \subset \mathbb{R}^N$, em que cada x^i é um vetor de entrada de características observadas (*input*) em um determinado conjunto de n amostras;
- $Y = \{y^1, ..., y^n\} \subset \mathbb{R}^K$, em que cada y^i é um vetor de dados de saída observados associado a x^i;
- $f(X) = \{f(x^1), ..., f(x^n)\} \subset \mathbb{R}^K$, em que $f(x^i)$ é a imagem de x^i pelo modelo.

O conjunto de pares (x^i, y^i) é chamado de *conjunto de treino*. A presença desses dados/classificações observados associados aos vetores x^i é o que caracteriza nosso treinamento como um *aprendizado supervisionado* [5, Capítulo 2].

Equipados com esse conjunto, usamos então alguma medida para avaliar o *fitting* do nosso modelo em relação ao conjunto Y. Ou seja, queremos avaliar a discrepância entre o conjunto $f(X)$ gerado pelo modelo e os dados observados. Uma medida comumente usada em problemas de regressão é a *média da soma dos quadrados dos erros* [3, Capítulo 9]:

$$R = \frac{1}{n} \sum_{i=1}^{n} \|f(x^i) - y^i\|_2^2. \tag{1}$$

Note que se alteramos os pesos e os vieses de uma rede neural, nós consequentemente alteramos a imagem de X por esse modelo; em outras palavras, o conjunto imagem $f(X)$ está em função desses parâmetros. A partir de agora agruparemos os pesos w_j^l e vieses b_j^l em um único vetor de parâmetros θ, e reescreveremos a equação acima em função de θ:

$$R(\theta) = \frac{1}{n} \sum_{i=1}^{n} \|f(x^i, \theta) - y^i\|_2^2. \tag{2}$$

O *treinamento* de uma rede neural consiste precisamente em achar θ_* tal que o erro, ou discrepância, entre o modelo e os dados observados seja o menor possível.

Escrevendo de forma geral, queremos resolver o seguinte problema de otimização:

$$\min_{\theta} R(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x^i, \theta), y^i) \tag{3}$$

em que ℓ é uma função não-negativa que mede a discrepância entre $f(x^i, \theta)$ e y^i – aqui nesse contexto chamada de *função de perda* (*loss function*). A função R em (3) é comumente chamada de *risco empírico*[1].

Como dito no início do capítulo, uma vez em posse de uma solução θ para o problema (3), e por conseguinte do modelo (rede neural) resultante, aplicamo-lo a um novo conjunto de amostras a fim de classificá-las e/ou fazer predições sobre esse novo conjunto de dados.

Em resumo, o treinamento consiste em determinar os parâmetros θ com base em um conjunto de dados observados, de modo a minimizar "a diferença" entre os *outputs* observados e os fornecidos pela rede neural.

No próximo capítulo, apresentaremos um método comum na prática de *machine learning* para resolver o problema (3), e exporemos alguns resultados teóricos de convergência sobre tal.

---

[1] Idealmente, gostaríamos de minizar o *risco esperado*

$$R(\theta) = \iint_{\mathbb{R}^N \times \mathbb{R}^K} \ell(f(x, \theta), y)\, dP(x, y)$$

para quaisquer pares $(x, y) \in \mathbb{R}^N \times \mathbb{R}^K$ de *input/output*, em que $P : \mathbb{R}^N \times \mathbb{R}^K \to [0, 1]$ é a distribuição de probabilidade que representa a verdadeira relação entre estes. Como desconhecemos P, e só temos um conjunto de amostras à disposição, acabamos resolvendo (3). Para uma discussão mais detalhada, ver [8, Capítulo 6].

3 MÉTODOS DE GRADIENTE ESTOCÁSTICO

Neste capítulo, passaremos à apresentação e análise de uma escolha comum na prática de *machine learning* (ver A.1 no Apêndice A) para atacar o problema (3), a saber, métodos de gradiente estocástico. Tal método foi primeiramente proposto por Robbins e Monro, no ano de 1951 [2]. A principal referência deste capítulo foi [2], que considera o seguinte problema de otimização:

$$\min_{\theta} \quad F(\theta) := \frac{1}{n} \sum_{i=1}^{n} \rho_i(\theta),$$

com $F, \rho_i : \mathbb{R}^D \longrightarrow \mathbb{R}$. Nesse contexto, chamaremos F de *função objetivo*.

Perceba que se $\rho_i(\theta) = \ell(f(x^i, \theta), y^i)$, para $i = 1, \ldots, n$, então $F(\theta) = R(\theta)$ e o problema acima coincide com (3).

Como, com frequência, o número de amostras n e a dimensão do vetor de parâmetros livres $\theta \in \mathbb{R}^D$ em (3) são significativamente grandes, encontramos dificuldades na implementação de métodos clássicos de otimização para sua resolução, pois a maioria destes demanda a avaliação do "gradiente completo", $\nabla F(\theta)$, o que representa um alto custo computacional por iteração (aplicando um método de Gradiente (ver [1, Capítulo1]) para resolver (3), teríamos um custo proporcional a n). Por exemplo, no caso do problema que será tratado no Capítulo 5 (classificação de algarismos manuscritos), $n = 60.000$ e $D = 164.013$.

3.1 UM PROTÓTIPO DE MÉTODO

Vamos começar apresentando a forma geral do algoritmo MGE [2]. Pressupomos que: (i) temos à disposição um mecanismo para gerar uma realização da variável aleatória λ_k, com espaço amostral $\{1, 2, \ldots, n\}$, e que a sequência $\{\lambda_k\}$ de variáveis aleatórias é con-

juntamente independente; (*ii*) dado o iterado $\theta_k \in \mathbb{R}^D$, temos como calcular o vetor estocástico $g(\theta_k, \lambda_k) \in \mathbb{R}^D$; (*iii*) temos como escolher um escalar $\alpha_k > 0$ como tamanho de passo [2].

Algorithm 1 Protótipo de método de gradiente estocástico (MGE)

Escolha um iterado inicial $\theta_1 \in \mathbb{R}^D$
for k=1,2,... **do**
  Gere uma realização da variável aleatória λ_k
  Compute um vetor estocástico $g(\theta_k, \lambda_k)$
  Escolha um tamanho de passo $\alpha_k > 0$
  Atualize o novo iterado como $\theta_{k+1} = \theta_k - \alpha_k g(\theta_k, \lambda_k)$
end for

Dado o protótipo, vamos discutir como ele se traduz e se aplica para o nosso problema em questão:

1. escolha um vetor inicial $\theta_1 \in \mathbb{R}^D$ de pesos e viéses do modelo. Agora, para $k = 1, 2...$ faça:

2. gere uma escolha aleatória de índice $\lambda_k \in \{1, \ldots, n\}$, em que n é o número total de amostras, e selecione a respectiva amostra x^{λ_k} do conjunto de amostras;

3. compute o vetor $g(\theta_k, x^{\lambda_k})$ como um estimador não viesado do gradiente da função de perda - isto significa dizer que o valor esperado de $g(\theta_k, x^{\lambda_k})$ é igual ao gradiente da função perda (ver [5, Capítulo 3]);

4. escolha um tamanho de passo $\alpha_k > 0$ (também chamado de *learning rate*);

5. atualize o novo iterado como $\theta_{k+1} = \theta_k - \alpha_k g(\theta_k, \lambda_k)$, *à la* gradiente determinístico (ver [1, Capítulo 1]).

Vale notar que é o item 2 que caracteriza nosso algoritmo como um método estocástico (ver apêndice A). Além disso, há a possível abordagem de tomar mais de um índice aleatório por iteração; tal abordagem é conhecida na literatura como *mini-batch* [2].

3.2 ANÁLISE DE CONVERGÊNCIA DO MGE

Para a análise de convergência do MGE, vamos assumir algumas hipóteses sobre a função objetivo. Além disso, introduziremos aqui a notação $\mathbb{E}_{\lambda_k}[.]$ para denotar o valor esperado do argumento em relação a λ_k.

Hipótese 3.1. *A função objetivo F é suave, e o gradiente ∇F é L-Lipschitz, ou seja, existe um escalar $L > 0$ tal que*

$$||\nabla F(\theta) - \nabla F(\bar{\theta})||_2 \leq L||\theta - \bar{\theta}||_2, \quad \forall \theta, \bar{\theta} \in \mathbb{R}^D.$$

De tal hipótese, e como consequência do teorema do valor médio para integrais, ganhamos o seguinte resultado:

$$F(\theta) \leq F(\bar{\theta}) + \nabla F(\bar{\theta})^T(\theta - \bar{\theta}) + \frac{1}{2}L||\theta - \bar{\theta}||_2^2, \quad \forall \theta \in \mathbb{R}^D. \quad (4)$$

Com efeito,

$$\begin{aligned}
F(\theta) &= F(\bar{\theta}) + \int_0^1 \frac{\partial F(\bar{\theta} + t(\theta - \bar{\theta}))}{\partial t} dt \\
&= F(\bar{\theta}) + \int_0^1 \nabla F(\bar{\theta} + t(\theta - \bar{\theta}))^T(\theta - \bar{\theta}) dt \\
&= F(\bar{\theta}) + \nabla F(\bar{\theta})^T(\theta - \bar{\theta}) + \int_0^1 [\nabla F(\bar{\theta} + t(\theta - \bar{\theta})) - \nabla F(\bar{\theta})]^T(\theta - \bar{\theta}) \\
&\leq F(\bar{\theta}) + \nabla F(\bar{\theta})^T(\theta - \bar{\theta}) + \int_0^1 L\|t(\theta - \bar{\theta})\|_2 \|(\theta - \bar{\theta})\|_2 dt,
\end{aligned}$$

de onde segue (4).

Lema 3.2. *Sob a hipótese 3.1, os iterados do MGE satisfazem a seguinte inequação para todo* $k \in \mathbb{N}$*:*

$$\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) \leq -\alpha_k \nabla F(\theta_k)^T \mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)] + \frac{\alpha_k^2}{2} L||\mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)||_2^2], \tag{5}$$

sendo $\mathbb{E}_{\lambda_k}[F(\theta_{k+1})]$ *o valor esperado de* $F(\theta_{k+1})$ *em relação a* λ_k*.*

Demonstração. De (3.1), e de $\theta_{k+1} = \theta_k - \alpha_k g(\theta_k, \lambda_k)$, temos

$$\begin{aligned} F(\theta_{k+1}) - F(\theta_k) &\leq \nabla F(\theta_k)^T(-\alpha_k g(\theta_k, \lambda_k)) + \frac{1}{2} L|| - \alpha_k g(\theta_k, \lambda_k)||_2^2 \\ &= -\alpha_k \nabla F(\theta_k)^T g(\theta_k, \lambda_k) + \frac{\alpha_k^2}{2} L||g(\theta_k, \lambda_k)||_2^2 \end{aligned}$$

Tomando o valor esperado dos dois lados com relação à λ_k, e notando que somente θ_{k+1} e $g(\theta_k, \lambda_k)$ dependem de λ_k, obtemos o resultado desejado. □

Hipótese 3.3. *A função objetivo e o MGE satisfazem as seguintes propriedades:*

1. *a sequência de iterados* $\{\theta_k\}$ *está contida num conjunto aberto no qual* F *é limitada inferiormente por um escalar* F_{inf}*;*
2. *existem escalares* $\mu_G \geq \mu > 0$ *tais que, para todo* $k \in \mathbb{N}$

$$\nabla F(\theta_k)^T \mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)] \geq \mu ||\nabla F(\theta_k)||_2^2 \tag{6}$$

$$||\mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)]||_2 \leq \mu_G ||\nabla F(\theta_k)||_2 \tag{7}$$

3. *existem escalares* $M \geq 0$ *e* $M_V \geq 0$ *tais que, para todo* $k \in \mathbb{N}$

$$\mathbb{V}_{\lambda_k}[g(\theta_k, \lambda_k)] \leq M + M_V ||\nabla F(\theta_k)||_2^2 \tag{8}$$

A primeira condição pede que $F(\theta_k)$ seja limitada inferiormente. A segunda nos diz que as direções de busca adotadas são, em valor esperado, direções de descida - tais condições são bem conhecidas em literatura de otimização [1]. A terceira é uma limitação superior da "variância" definida por [2]:

$$\mathbb{V}_{\lambda_k}[g(\theta_k, \lambda_k)] := \mathbb{E}_{\lambda_k}[||g(\theta_k, \lambda_k)||_2^2] - ||\mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)]||_2^2$$

em função da norma do gradiente de F. Com estas hipóteses é possível deduzir a seguinte propriedade:

$$\begin{aligned} \mathbb{E}_{\lambda_k}[||g(\theta_k, \lambda_k)||_2^2] &\leq M + M_G||\nabla F(\theta_k)||_2^2, \\ \text{com } M_G &:= M_V + \mu_G^2 \geq \mu^2 > 0. \end{aligned} \tag{9}$$

Com efeito, da inequação (8), temos:

$$\mathbb{E}_{\lambda_k}[||g(\theta_k, \lambda_k)||_2^2] - ||\mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)]||_2^2 \leq M + M_V||\nabla F(\theta_k)||_2^2$$

implicando em

$$\begin{aligned} \mathbb{E}_{\lambda_k}[||g(\theta_k, \lambda_k)||_2^2] &\leq M + M_V||\nabla F(\theta_k)||_2^2 + ||\mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)]||_2^2 \\ &\leq M + M_V||\nabla F(\theta_k)||_2^2 + \mu_G^2||\nabla F(\theta_k)||_2^2 \\ &= M + (M_V + \mu_G^2)||\nabla F(\theta_k)||_2^2 \end{aligned}$$

Lema 3.4. *O iterado do MGE satisfaz o seguinte, para $k \in \mathbb{N}$:*

$$\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) \leq -(\mu - \frac{1}{2}\alpha_k L M_G)\alpha_k||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\alpha_k^2 LM \tag{10}$$

Demonstração. Do Lema 3.2 e da inequação (6), temos:

$$\begin{aligned}\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) &\leq -\alpha_k \nabla F(\theta_k)^T \mathbb{E}_{\lambda_k}[g(\theta_k, \lambda_k)] \\ &\quad + \frac{1}{2}\alpha_k^2 L \mathbb{E}_{\lambda_k}[\|g(\theta_k, \lambda_k)\|_2^2] \\ &\leq -\mu\alpha_k ||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\alpha_k^2 L \mathbb{E}_{\lambda_k}[||g(\theta_k, \lambda_k)||_2^2]\end{aligned}$$

Aplicando a inequação (9) sobre $\mathbb{E}_{\lambda_k}[\|g(\theta_k, \lambda_k)\|_2^2]$, e rearranjando os termos, obtemos o resultado desejado. □

Nas seções a seguir, prosseguiremos nossa análise do MGE, primeiro para funções objetivo *fortemente convexas* e depois para casos mais gerais.

3.2.1 Análise para objetivos fortemente convexos

Em alguns cenários importantes de redes neurais, o objetivo a ser minimizado é fortemente convexo - com frequência, via uma função de regularização ou uma reformulação do problema original a fim de torná-lo convexo [2]. Sendo assim, a análise do MGE, quando aplicado para problemas dessa classe, faz-se importante.

Comecemos com a hipótese que dá nome à seção:

Hipótese 3.5 (**Convexidade forte**). *A função objetivo $F : \mathbb{R}^D \to \mathbb{R}$ é fortemente convexa se existe uma constante $c > 0$ tal que*

$$F(\bar{\theta}) \geq F(\theta) + \nabla F(\theta)^T(\bar{\theta} - \theta) + \frac{1}{2}c||\bar{\theta} - \theta||_2^2$$

para todo $(\bar{\theta}, \theta) \in \mathbb{R}^D \times \mathbb{R}^D$.

Uma consequência desta hipótese é que F possui um único minimizador, denotado por $\theta_* \in \mathbb{R}^D$ com $F_* := F(\theta_*)$. De tal, podemos tirar o seguinte:

$$2c(F(\theta) - F_*) \leq ||\nabla F(\theta)||_2^2, \quad \forall \theta \in \mathbb{R}^D \tag{11}$$

Pois dado $\theta \in \mathbb{R}^d$, o modelo quadrático

$$q(\bar{\theta}) := F(\theta) + \nabla F(\theta)^T(\bar{\theta} - \theta) + \frac{1}{2}c||\bar{\theta} - \theta||_2^2$$

tem minimizador único $\bar{\theta_*} := w - \frac{1}{c}\nabla F(\theta)$ com $q(\bar{\theta^*}) = F(\theta) - \frac{1}{2c}\|\nabla F(\theta)\|_2^2$.

Assim, da nossa hipótese, para qualquer $\theta \in \mathbb{R}^D$, fazendo $\bar{\theta} = \theta_*$, temos

$$F_* \geq F(\theta) + \nabla F(\theta)^T(\theta_* - \theta) + \frac{1}{2}c||\theta_* - \theta||_2^2 \geq F(\theta) - \frac{1}{2c}\|\nabla F(\theta)\|_2^2.$$

Teorema 3.6 (**Objetivo fortemente convexo, tamanho de passo fixo**). *Sob as hipóteses 3.1, 3.3 e 3.5 (com $F_{inf} = F_*$), supondo que o método MGE seja executado com tamanho de passo fixo $\alpha_k = \bar{\alpha}$ para todo $k \in \mathbb{N}$, satisfazendo*

$$0 < \bar{\alpha} \leq \frac{\mu}{LM_G} \tag{12}$$

temos que o valor esperado do gap de otimalidade satisfaz a seguinte desigualdade para todo $k \in \mathbb{N}$:

$$\mathbb{E}[F(\theta_k) - F_*] \leq \frac{\bar{\alpha}LM}{2c\mu} + (1 - \bar{\alpha}c\mu)^{k-1}(F(\theta_1) - F_* - \frac{\bar{\alpha}LM}{2c\mu})$$
$$\xrightarrow{k\to\infty} \frac{\bar{\alpha}LM}{2c\mu}$$

Demonstração. Usando o Lema 3.4 e as inequações (11) e (12), temos, para todo $k \in \mathbb{N}$, que

$$\begin{aligned}
\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) &\leq -(\mu - \frac{1}{2}\bar{\alpha}LM_g)\bar{\alpha}||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\bar{\alpha}^2 LM \\
&\leq -(\mu - \frac{1}{2}\frac{\mu}{LM_g}LM_g)\bar{\alpha}||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\bar{\alpha}^2 LM \\
&= -\frac{\mu}{2}\bar{\alpha}||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\bar{\alpha}^2 LM \\
&\leq -\frac{\mu}{2}\bar{\alpha}2c(F(\theta_k) - F_*) + \frac{1}{2}\bar{\alpha}^2 LM \\
&= -\bar{\alpha}c\mu(F(\theta_k) - F_*) + \frac{1}{2}\bar{\alpha}^2 LM
\end{aligned}$$

Subtraindo F_* dos dois lados e manipulando as expressões, temos:

$$\begin{aligned}
\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) - F_* &\leq -\bar{\alpha}c\mu(F(\theta_k) - F_*) + \frac{1}{2}\bar{\alpha}^2 LM - F_* \\
\implies \mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] &\leq -\bar{\alpha}c\mu(F(\theta_k) - F_*) \\
&\quad + F(\theta_k) - F_* + \frac{1}{2}\bar{\alpha}^2 LM \\
\implies \mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] &\leq (1 - \bar{\alpha}c\mu)(F(\theta_k) - F_*) + \frac{1}{2}\bar{\alpha}^2 LM
\end{aligned}$$

Agora, subtraindo $\bar{\alpha}LM/2c\mu$ dos dois lados e prosseguindo, temos:

$$\begin{aligned}
\mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] - \frac{\bar{\alpha}LM}{2c\mu} &\leq (1 - \bar{\alpha}c\mu)(F(\theta_k) - F_*) \\
&\quad + \frac{1}{2}\bar{\alpha}^2 LM - \frac{\bar{\alpha}LM}{2c\mu} \\
&= (1 - \bar{\alpha}c\mu)([F(\theta_k) - F_*] - \frac{\bar{\alpha}LM}{2c\mu})
\end{aligned}$$

Tomando o respectivo valor esperado da iteração em questão, temos

$$\begin{aligned}
&\mathbb{E}_{\lambda_{k-1}}\mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] - \frac{\bar{\alpha}LM}{2c\mu} \\
&\leq (1 - \bar{\alpha}c\mu)(\mathbb{E}_{\lambda_{k-1}}[F(\theta_k) - F_*] - \frac{\bar{\alpha}LM}{2c\mu}) \\
&\leq (1 - \bar{\alpha}c\mu)(1 - \bar{\alpha}c\mu)(F(\theta_{k-1}) - F_* - \frac{\bar{\alpha}LM}{2c\mu}).
\end{aligned}$$

Repetindo o processo, de modo a iterar a desigualdade, obtemos

$$\begin{aligned}
&\mathbb{E}_{\lambda_1}...\mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] - \frac{\bar{\alpha}LM}{2c\mu} \\
&\leq (1 - \bar{\alpha}c\mu)^{k-1}(\mathbb{E}_{\lambda_1}[F(\theta_2) - F_*] - \frac{\bar{\alpha}LM}{2c\mu}).
\end{aligned}$$

E por fim concluímos que

$$\begin{aligned}
&\mathbb{E}_{\lambda_1}...\mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] - \frac{\bar{\alpha}LM}{2c\mu} \\
&\leq (1 - \bar{\alpha}c\mu)^{k}(F(\theta_1) - F_* - \frac{\bar{\alpha}LM}{2c\mu}).
\end{aligned}$$

Note que $\mathbb{E}_{\lambda_1}...\mathbb{E}_{\lambda_k}[F(\theta_l)] = \mathbb{E}_{\lambda_{l-1}}[F(\theta_l)] = \mathbb{E}[F(\theta_l)]$, pois $F(\theta_l)$ só depende de λ_{l-1}. Escreveremos tal encadeamento de valores esperados $\mathbb{E}_{\lambda_1}...\mathbb{E}_{\lambda_k}$ como $\mathbb{E}$, a fim de melhor denotar esse processo de tomar valores esperados em sequência.

Agora, observando que:

$$0 < \bar{\alpha}c\mu \leq \frac{c\mu^2}{LM_g} \leq \frac{c\mu^2}{L\mu^2} = \frac{c}{L} \leq 1$$

concluímos que $(1 - \bar{\alpha}c\mu)^k \xrightarrow{k\to\infty} 0$ e assim obtemos o limite almejado.

□

Se $g(\theta_k, \lambda_k)$ é um estimador não viesado de $\nabla F(\theta_k)$, então $\mu = 1$. E se não existe ruído em $g(\theta_k, \lambda_k)$ então $M_G = 1$ [2]. Se isso acontece, então a condição sobre $\bar{\alpha}$ se reduz a $\bar{\alpha} \in (0, 1/L)$. Note também que o *gap* de otimalidade e o termo de contração $(1 - \bar{\alpha}c\mu)$ dependem do tamanho de passo.

Partimos agora para a análise de uma outra escolha possível em relação ao tamanho de passo, a saber, tomar uma sequência decrescente $\{\alpha_k\}$ de tamanhos de passo satisfazendo as seguintes condições:

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad e \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty \tag{13}$$

Teorema 3.7 (**Objetivo fortemente convexo, tamanho de passo decrescente**). *Sob as hipóteses 3.1, 3.3 e 3.5 (com $F_{inf} = F_*$), suponha que o MGE é iterado com uma sequência de tamanho de passos tal que $\forall k \in \mathbb{N}$,*

$$\alpha_k = \frac{\beta}{\gamma + k}, \text{ para algum } \beta > \frac{1}{c\mu} \text{ e } \gamma > 0 \text{ tal que } \alpha_1 \leq \frac{\mu}{LM_G}. \tag{14}$$

Então o valor esperado do gap de otimalidade é

$$\mathbb{E}[F(\theta_k) - F_*] \leq \frac{\nu}{\gamma + k},$$
$$\text{com } \nu := \max\left\{\frac{\beta^2 LM}{2(\beta c\mu - 1)}, (\gamma + 1)(F(\theta_1) - F_*)\right\}.$$

Demonstração. Assumindo (14), temos que a desigualdade $\alpha_k LM_G \leq \alpha_1 LM_G \leq \mu$ para todo $k \in \mathbb{N}$. Do Lema 3.4 e da hipótese de convexidade forte, temos:

$$\begin{aligned}
\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) &\leq -(\mu - \frac{1}{2}\alpha_k LM_g)\alpha_k ||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\alpha_k^2 LM \\
&\leq -(\mu - \frac{1}{2}\frac{\mu}{LM_g} LM_g)\alpha_k ||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\alpha_k^2 LM \\
&= -\frac{\mu}{2}\alpha_k ||\nabla F(\theta_k)||_2^2 + \frac{1}{2}\alpha_k^2 LM \\
&\leq -\frac{\mu}{2}\alpha_k 2c(F(\theta_k) - F_*) + \frac{1}{2}\alpha_k^2 LM \\
&= -\alpha_k c\mu(F(\theta_k) - F_*) + \frac{1}{2}\alpha_k^2 LM
\end{aligned}$$

Subtraindo F_* dos dois lados e manipulando as expressões, temos:

$$\begin{aligned}
\mathbb{E}_{\lambda_k}[F(\theta_{k+1})] - F(\theta_k) - F_* &\leq -\alpha_k c\mu(F(\theta_k) - F_*) + \frac{1}{2}\alpha_k^2 LM - F_* \\
\implies \mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] &\leq -\alpha_k c\mu(F(\theta_k) - F_*) + F(\theta_k) - F_* \\
&\quad + \frac{1}{2}\alpha_k^2 LM \\
\implies \mathbb{E}_{\lambda_k}[F(\theta_{k+1}) - F_*] &\leq (1 - \alpha_k c\mu)(F(\theta_k) - F_*) + \frac{1}{2}\alpha_k^2 LM
\end{aligned}$$

Tomando o valor esperado $\mathbb{E}$, ficamos com:

$$\mathbb{E}[F(\theta_{k+1}) - F_*] \leq (1 - \alpha_k c\mu)\mathbb{E}[F(\theta_k) - F_*] + \frac{1}{2}\alpha_k^2 LM$$

Seguiremos agora por indução. A definição de ν nos garante que a inequação é satisfeita para $k = 1$. Assuma que vale para algum $k \geq 1$. Então temos, da inequação acima, e denotando $\hat{k} = \gamma + k$, que

$$\begin{aligned}
\mathbb{E}[F(\theta_{k+1}) - F_*] &\leq \left(1 - \frac{\beta}{\hat{k}}\right)\frac{\nu}{\hat{k}} + \frac{\beta^2 LM}{2\hat{k}^2} \\
&= \left(\frac{\hat{k} - \beta c\mu}{\hat{k}^2}\right)\nu + \frac{\beta^2 LM}{2\hat{k}^2} \\
&= \left(\frac{\hat{k} - \beta c\mu}{\hat{k}^2}\right)\nu + \nu - \nu + \frac{\beta^2 LM}{2\hat{k}^2} \\
&= \left(\frac{\hat{k} - 1}{\hat{k}^2}\right)\nu - \left(\frac{\beta c\mu - 1}{\hat{k}^2}\right)\nu + \frac{\beta^2 LM}{2\hat{k}^2}
\end{aligned}$$

Da definição de ν, temos:

$$\left(\frac{\beta c\mu - 1}{\hat{k}^2}\right)\nu + \frac{\beta^2 LM}{2\hat{k}^2} \geq 0$$

Disso, notando que $\hat{k} \geq (\hat{k}+1)(\hat{k}-1)$, concluímos que

$$\left(\frac{\hat{k} - 1}{\hat{k}^2}\right)\nu \leq \frac{(\hat{k}-1)}{(\hat{k}+1)(\hat{k}-1)}\nu = \frac{\nu}{\hat{k}+1}$$

□

3.2.2 Análise para objetivos gerais

Para muitos modelos de redes neurais, o problema de otimização apresentado é não-convexo. Por essa razão, passaremos à análise do MGE quando aplicado a esses casos, a fim de descobrir quais são as garantias teóricas nos são fornecidas - note que, para objetivos não-convexos, alguns desafios se apresentam, como, por exemplo, a possível existência de múltiplos mínimos locais e demais pontos estacionários [2].

Teorema 3.8 (Objetivo não-convexo, tamanho de passo fixo). *Sob as hipóteses 3.1 e 3.3, suponha que o MGE é executado com tamanho de passo fixo $\alpha_k = \bar{\alpha}$ para todo $k \in \mathbb{N}$, satisfazendo*

$$0 < \bar{\alpha} \leq \frac{\mu}{LM_G}.$$

Então, o valor esperado da soma dos quadrados da norma do gradiente de F e sua média, correspondente ao iterado atual do MGE, satisfazem as seguintes inequações para todo $K \in \mathbb{N}$:

$$\begin{aligned} \mathbb{E}\left[\sum_{k=1}^{K} ||\nabla F(\theta_k)||_2^2\right] &\leq \frac{K\bar{\alpha}LM}{\mu} + \frac{2(F(\theta_1) - F_{inf})}{\mu\bar{\alpha}} \\ \implies \mathbb{E}\left[\frac{1}{K}\sum_{k=1}^{K} ||\nabla F(\theta_k)||_2^2\right] &\leq \frac{\bar{\alpha}LM}{\mu} + \frac{2(F(\theta_1) - F_{inf})}{K\mu\bar{\alpha}} \\ &\xrightarrow{K\to\infty} \frac{\bar{\alpha}LM}{\mu} \end{aligned}$$

Demonstração. pelo Lema 3.4 e a condição (3.8) sobre o tamanho de passo, tomando os valores esperados, obtemos:

$$\begin{aligned} \mathbb{E}[F(\theta_{k+1})] - \mathbb{E}[F(\theta_k)] &\leq -(\mu - \frac{1}{2}\bar{\alpha}LM_g)\bar{\alpha}\mathbb{E}[||\nabla F(\theta_k)||_2^2] + \frac{1}{2}\bar{\alpha}^2 LM \\ &= -\frac{1}{2}\mu\bar{\alpha}\mathbb{E}[||\nabla F(\theta_k)||_2^2] + \frac{1}{2}\bar{\alpha}^2 LM \end{aligned}$$

Agora, prosseguimos iterando essa desigualdade para $k \in \{1, ..., K\}$, para obter:

$$\mathbb{E}[F(\theta_{K+1})] - \mathbb{E}[F(\theta_1)] \leq -\frac{1}{2}\mu\bar{\alpha}\sum_{k=1}^{K}\mathbb{E}[||\nabla F(\theta_k)||_2^2] + \frac{1}{2}K\bar{\alpha}^2 LM$$

Notando que $\mathbb{E}[F(\theta_1)] = F(\theta_1)$, pois θ_1 é o ponto inicial, e lembrando da hipótese da existência de um limitante inferior F_{inf} para F, temos:

$$F_{inf} - F(\theta_1) \leq \mathbb{E}[F(\theta_{K+1})] - F(\theta_1) \leq -\frac{1}{2}\mu\bar{\alpha}\sum_{k=1}^{K}\mathbb{E}[||\nabla F(\theta_k)||_2^2] + \frac{1}{2}K\bar{\alpha}^2 LM$$

Agora, da desigualdade entre o primeiro termo à esquerda e o último termo à direita, isolamos o somatório, dividimos por K e tomamos o limite $K \to \infty$ a fim de obter o resultado desejado. □

No capítulo seguinte, iremos desenvolver e expor experimentos numéricos com o MGE a fim de ilustrar alguns dos resultados teóricos estudados neste capítulo.

4 EXPERIMENTOS NUMÉRICOS

Para este capítulo, selecionamos alguns problemas de teste (*toy problems*) sobre os quais aplicaremos alguma versão do MGE. Esperamos que tais experimentos sejam capazes de dar ao leitor uma intuição sobre a natureza estocástica do método em questão, bem como ilustrar alguns resultados teóricos do Capítulo 3.

Todos os *scripts* a partir desse ponto foram implementados na linguagem de programação *Python*; e inteiramente desenvolvidos pelo autor. Além disso, para os problemas dessa seção, foram usados os pacotes *Math*, *NumPy* e *Matplotlib*; todos estes abertamente disponíveis (*open source*) em: `https://github.com/andrey-lolobrigida/tcc-mtm-mge`.

4.1 SOMATÓRIO DE QUADRÁTICAS

Para o primeiro experimento, queremos resolver o seguinte problema de otimização:

$$\min_{x\in\mathbb{R}} f(x) = \frac{1}{n}\sum_{i=1}^{n}(x-a_i)^2, \tag{15}$$

em que f é uma função real de uma variável real, $n = 1000$ e os a_i pertencem a um conjunto de mil pontos igualmente espaçados no intervalo $[-1, 1]$ (ver a função *linspace* do *NumPy*). Ou seja, $a_i \in \{-1, -0.997998, ..., 0.997998, 1\}$. Chamaremos cada parcela do somatório de $f_i(x) = (x - a_i)^2$.

A aplicação do algoritmo 1 se traduz para nosso problema da seguinte maneira: escolha $x_0 \in \mathbb{R}$ ponto inicial.

Agora, para $k = 1, 2 \ldots$ faça:

1. tome um índice i aleatório no conjunto $\{1, 2, ..., 999, 1000\}$;

2. compute a derivada $f_i^{'}(x_k) = 2(x - a_k)$;

3. escolha um tamanho de passo $\alpha_k > 0$;

4. atualize o novo iterado como $x_{k+1} = x_k - \alpha_k f_i^{'}(x_k)$.

Para o item 3, iremos considerar duas versões: uma com tamanho de passo fixo, e outra com tamanho de passo decrescente.

Note que, do fato dos a_i's se distribuírem simetricamente em torno da origem, ou seja, $\sum_{i=1}^{n} a_i = 0$, a função f pode ser expandida e escrita como

$$f(x) = x^2 + a, \tag{16}$$

em que $a = \frac{1}{n}\sum_{i=1}^{n} a_i^2$.

Como f é uma quadrática, ela trivialmente satisfaz nossa condição (3.5) de convexidade. Disto, temos que f possui minimizador global [1, Capítulo 3]. Além disso, como $x^2 \geq 0$ para todo x real, é fácil notar que o ponto $x_* = 0$ é tal minimizador global de (16), cujo valor funcional nesse ponto é dado por $f(x_*) = a$.

Com o objetivo de ilustrar a natureza estocástica do método, não utilizaremos um critério de parada específico (mais será discutido sobre isso no capítulo seguinte). Ademais, conjuntamente com o MGE, foi implementado um método de gradiente determinístico para fins de comparação.

4.1.1 Resultados - Tamanho de Passo Fixo

Aqui, foram empregados tamanhos de passo $\bar{\alpha} \in \{0,1; 0,05; 0,01\}$. Como ponto inicial, tomamos $x_0 = 1$ em todos os experimentos. Nas próximas figuras 2, 3, 4 e 5, plotamos o erro $E_k = f(x_k) - a$ contra o número de iterações. Vale notar o comportamento "errático" em

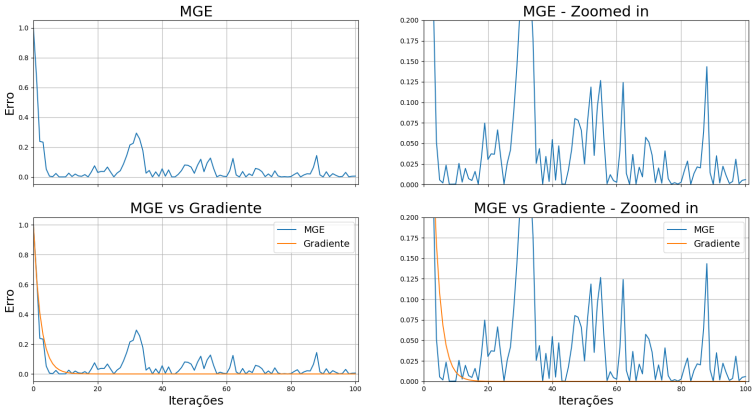

Figura 2 – Tamanho de passo $\bar{\alpha} = 0,1$.

torno do ponto de otimalidade (ponto em que $E_k = 0$), enquanto o método de gradiente determinístico se mantém próximo do ponto.

Nas figuras 6, 7, 8 e 9, plotamos o a média dos erros $E_k = f(x_k) - a$ na k-ésima iteração contra o número de iterações, a fim de ilustrar o Teorema 3.6. Note que, de fato, a média parece convergir para um valor assintótico conforme se dão as iterações; além disso, tais valores assintóticos parecem diminuir em dependência do tamanho de passo, o que acontece de acordo com o previsto por nossos estudos teóricos (ver Teorema 3.6).

4.1.2 Resultados - Tamanho de Passo decrescente

Para variante com tamanho de passo decrescente, começamos com $\alpha_0 = \frac{1}{2}$ e seguimos com $\alpha_k = \frac{1}{2+k}$. Note que a sequência dos $\{\alpha_k\}$ satisfaz as condições requeridas em (13). Os resultados estão expostos nas figuras 10 e 11.

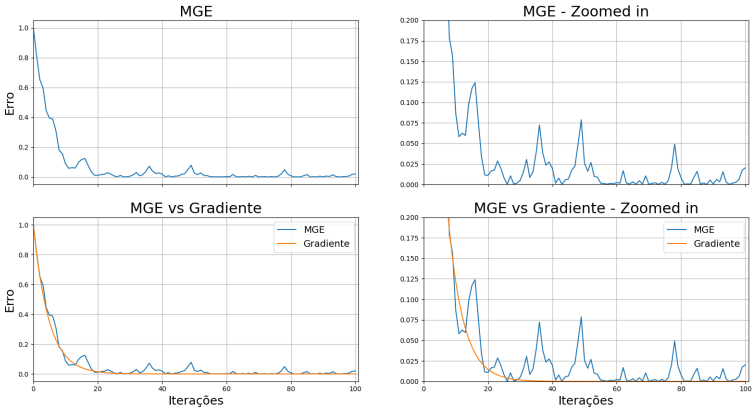

Figura 3 – Tamanho de passo $\bar{\alpha} = 0,05$.

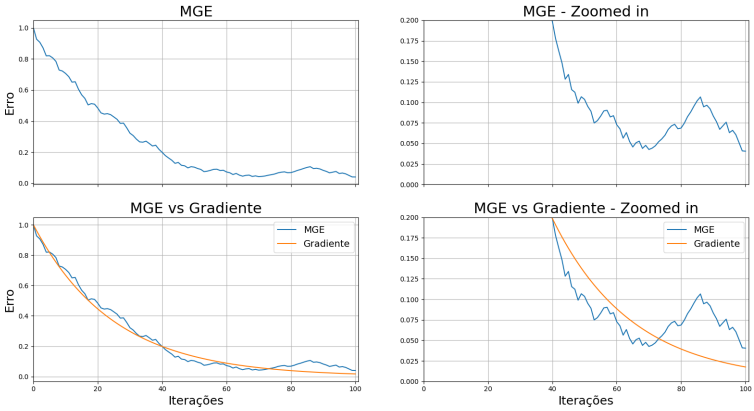

Figura 4 – Tamanho de passo $\bar{\alpha} = 0,01$.

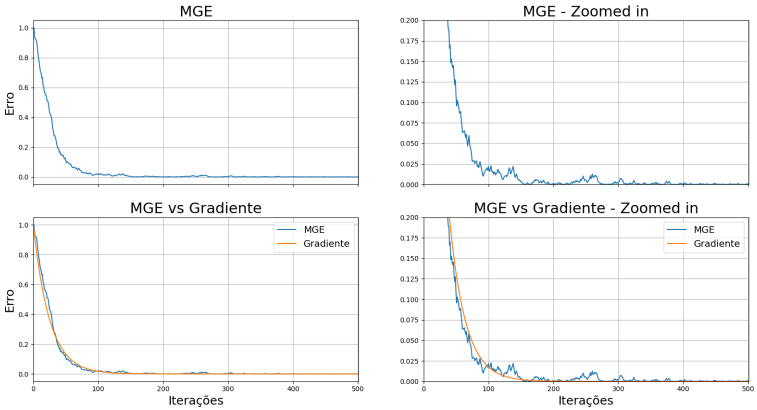

Figura 5 – Tamanho de passo $\bar{\alpha} = 0,01$.

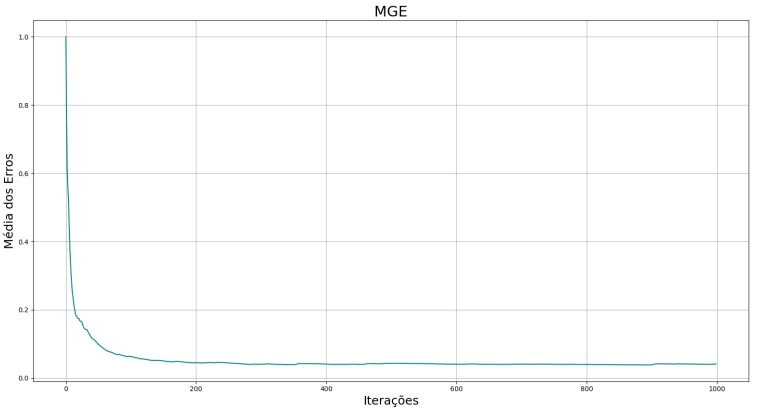

Figura 6 – Tamanho de passo $\bar{\alpha} = 0,1$.

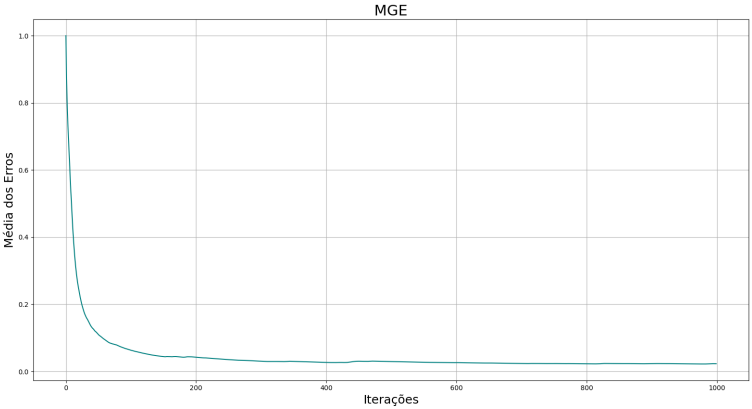

Figura 7 – Tamanho de passo $\bar{\alpha} = 0,05$.

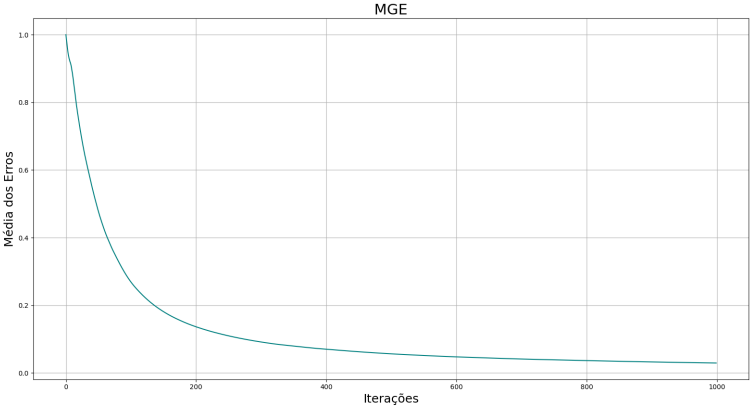

Figura 8 – Tamanho de passo $\bar{\alpha} = 0,01$.

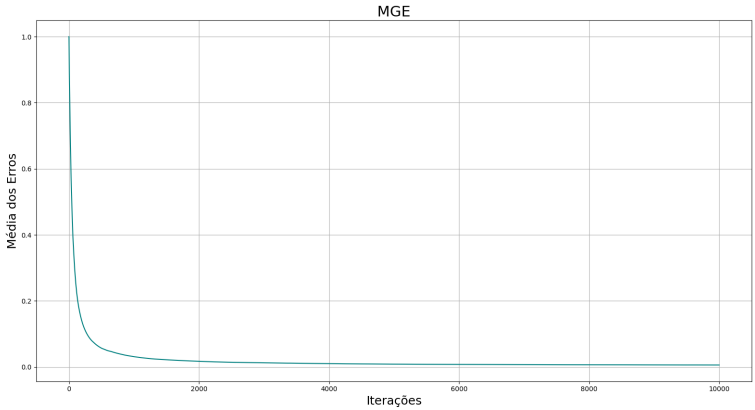

Figura 9 – Tamanho de passo $\bar{\alpha} = 0,01$; 10.000 iterações.

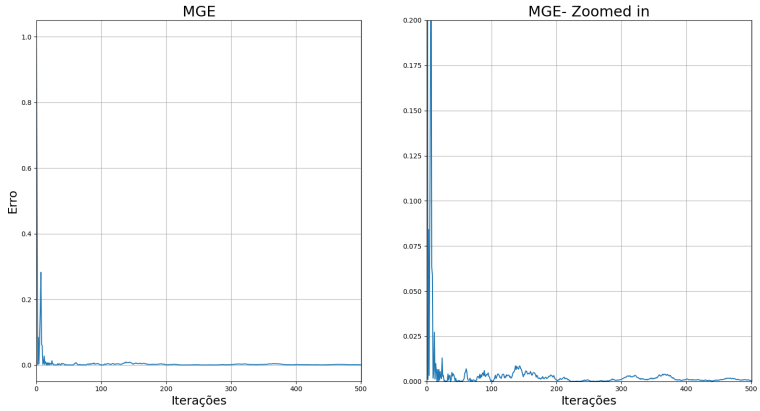

Figura 10 – Tamanho de passo decrescente.

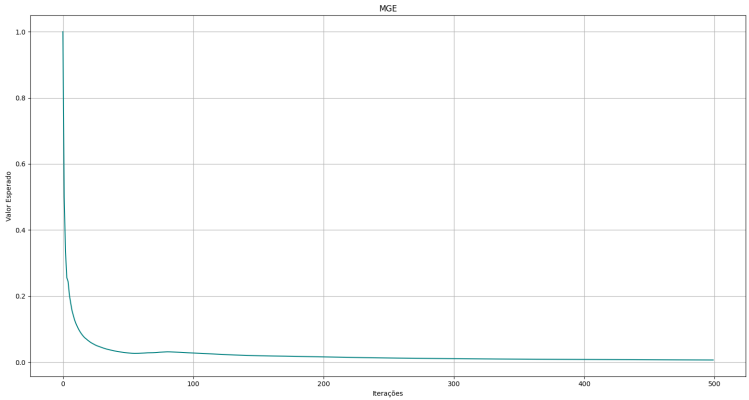

Figura 11 – Tamanho de passo decrescente.

Note que a erraticidade do MGE fica controlada conforme se passam as iterações, e consequentemente o tamanho de passo diminui. Tal fato pode ser visualizado no gráfico da média dos erros. Vale lembrar que o Teorema 3.7 nos diz que o valor esperado fica limitado por um escalar de ordem $\mathcal{O}(\frac{1}{k})$ conforme $k \to \infty$.

4.2 SOMATÓRIO DE QUADRÁTICAS EM DUAS DIMENSÕES

Para melhor ilustrar o comportamento do MGE em torno do ponto de otimalidade de um objetivo fortemente convexo, vamos estender o problema (15) da seguinte maneira:

$$\min_{(x,y)} f(x,y) = \frac{1}{n}\left(\sum_{i=1}^{n}((x-a_i)^2 + (y-b_i)^2)\right), \tag{17}$$

com $f : \mathbb{R}^2 \to \mathbb{R}$, $n = 1000$ e os a_i e b_i pertencentes ao mesmo conjunto descrito no problema anterior.

Procedendo analogamente ao exemplo da Seção 4.1, é fácil ver que o ponto $p_* = (0, 0)$ é minimizador global de (17) com valor funcional dado por $f(p_*) = 2a = 2b$.

4.2.1 Curvas de Nível

Plotamos aqui as curvas de nível da função f juntamente com o caminho percorrido pelo método, ao longo das iterações, a fim de, como já mencionado, observar o comportamento deste em torno do ponto p_* (Figuras 12, 13 e 14). Usamos dois tamanhos de passo fixos, e tamanho de passo decrescente, com a mesma especifição do problema anterior. Para todos os casos, iniciamos o algoritmo com o valor inicial $p_0 = (1, 1)$.

Mais uma vez, nota-se a erraticidade do método em torno do ponto ótimo. Tal fato ilustra muito bem a natureza estocástica do MGE. Além disso, como previsto pelos Teoremas 3.6 e 3.7, os pontos atingidos pelo MGE parecem estar distribuídos dentro de uma área mais ou menos delimitada em torno da origem (pontos em que observa-se o *gap* de otimalidade da maneira descrita pelos mencionados Teoremas).

Para o próximo capítulo, implementamos uma RNP *from scratch* e a treinamos para um problema de classificação de algarismos manuscritos. Resultados e detalhes da implementação serão discutidos adiante.

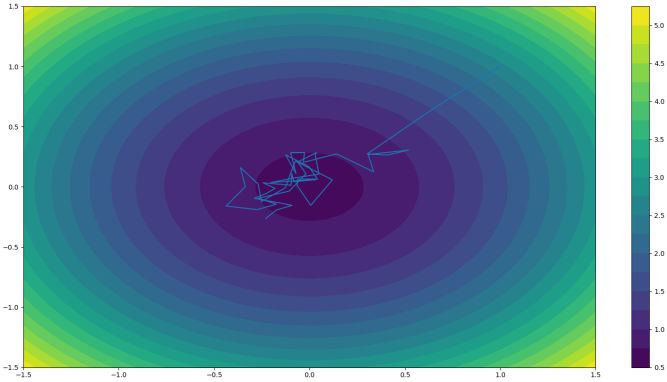

Figura 12 – Tamanho de passo fixo $\bar{\alpha} = 0,1$.

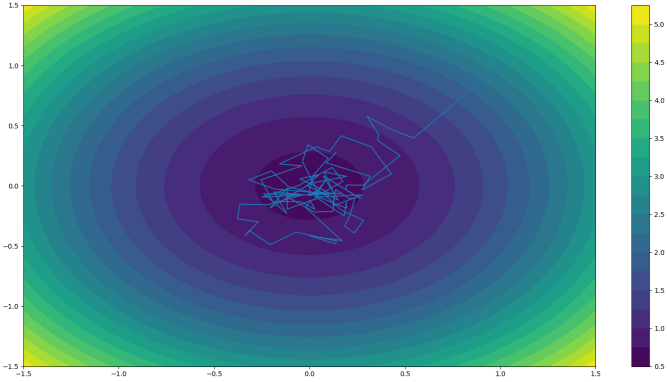

Figura 13 – Tamanho de passo fixo $\bar{\alpha} = 0,01$.

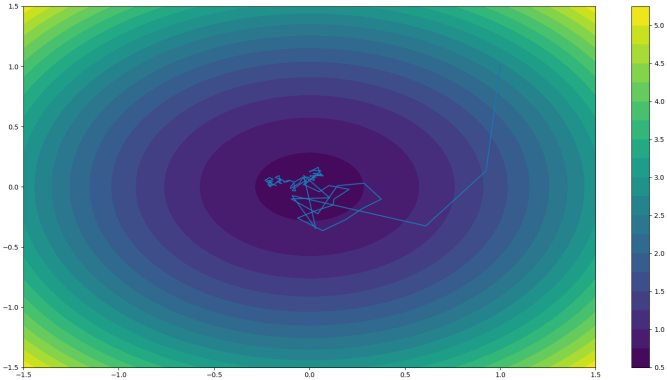

Figura 14 – Tamanho de passo decrescente.

5 IMPLEMENTAÇÃO DE UMA REDE NEURAL PROFUNDA

Neste capítulo, nos propusemos a resolver um problema de classificação de algarismos manuscritos usando uma RNP. Tal problema é um *benchmark* comum na literatura de *machine learning* e redes neurais [4, Capítulo 11].

5.1 CONJUNTO DE TREINO

Como conjunto de treino para nossa RNP, usaremos o *dataset* MNIST, que consiste em um conjunto de 60.000 imagens de algarismos numéricos escritos à mão previamente classificados; cada imagem tem dimensão 28×28 pixels, todas em escalas de cinzas. Neste *dataset*, há também mais um conjunto de 10.000 imagens deste tipo, com o intuito de servir de *conjunto de teste* (mais será dito sobre isso). Mais informações sobre o *dataset* podem ser encontradas em `https://keras.io/api/datasets/mnist/`.

Licença: o conjunto de dados MNIST é disponibilizado sob a licença Creative Commons Attribution-Share Alike 3.0 e é protegido por direitos autorais detidos por Yann LeCun e Corinna Cortes. Para mais informações sobre a licença e o uso do *dataset*, consulte `https://creativecommons.org/licenses/by-sa/3.0/`.

Figura 15 – Exemplo de imagens no MNIST.

5.2 IMPLEMENTAÇÃO

Os códigos da RNP e seu treinamento sobre o MNIST foram escritos em *Python*; e inteiramente desenvolvidos pelo autor. Além disso, foram usados os já mencionados pacotes *NumPy* e *Matplotlib*. Para acessar o *dataset*, utilizamos a API pública *Keras*; também citamos o uso do pacote *Alive Progress* na implementação de barras de progresso. Todos os códigos desenvolvidos para este capítulo estão abertamente disponíveis no repositório: `https://github.com/andrey-lolobrigida/Neural-Network-MNIST-Training/`.

5.2.1 Transformações sobre o conjunto de treino

No nosso caso particular, temos o seguinte:

- número total de amostras $n = 60.000$;
- as amostras x^i são vetores com 784 coordenadas (as entradas da matriz 28×28 que representa cada imagem);
- a classificação $z^i \in \{0, 1 \ldots, 8, 9\}$ é o dígito correspondente à imagem.

Para o treino da RNP, operamos algumas transformações prévias sobre o conjunto de dados. Primeiro, como os valores das coordenadas dos vetores x^i se encontram no intervalo real $[0, 255]$, dividimos tais vetores por 255, a fim de deixá-las com valores entre 0 e 1. Também transformamos a classificação da amostra em um vetor canônico $y^i = e_j \in \mathbb{R}^{10}$, em que $j = z^i + 1$. Ambas as transformações são comuns na prática de *machine learning* [4, Capítulo 11].

5.2.2 Funções de Ativação e Perda

Como *funções de ativação* para as camadas intermediárias, utilizamos a função *sigmóide* $\sigma : \mathbb{R} \to \mathbb{R}$,

$$\sigma(s) = \frac{1}{1 + e^{-s}},$$

aplicada a cada *neurônio* individualmente. Para fins de simplicidade, usamos tal função também para a camada de *output*. Note que a imagem da sigmóide é o intervalo real $(0, 1)$. Tal função é usada em regressão logística a fim de converter funcionais lineares em probabilidades entre 0 e 1 [5, Capítulo 10]. Tangente hiperbólica e a ReLU (*rectified linear unit*) também são comumente usadas como funções de ativação [3, Capítulo 5].

Como *função de perda*, utilizamos a função *squared error*. Isto nos dá, como função objetivo, a chamada *Mean Squared Error (MSE)*:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \|f(x^i, \theta) - y^i\|_2^2, \tag{18}$$

em que, no nosso contexto, n é o número total de amostras, $x^i \in \mathbb{R}^{784}$ é uma amostra, $y^i \in \mathbb{R}^{10}$ a sua classificação correspondente (como descrito na Seção 5.2.1), $\theta \in \mathbb{R}^D$ o vetor de parâmetros livres da RNP, e $f(x^i, \theta)$ o *output* da amostra pelo modelo. Vale notar que, dada esta forma para nossa RNP, a função objetivo de (18) é não-convexa [4, Capítulo 11].

5.2.3 Algoritmo de Treino

A forma que o MGE (algoritmo 1) assume para nosso problema, a grosso modo, dá-se da seguinte maneira: escolha $\theta_0 \in \mathbb{R}^D$ vetor inicial de pesos e vieses da RNP.

Agora, para $k = 1, 2 \ldots, 60.000$ faça:

1. selecione aleatoriamente, e sem reposição, uma amostra (x^i, y^i) no *dataset*;

2. compute o gradiente da função ℓ_i no ponto θ_k, em que $\ell_i = \|f(x^i, \theta_k) - y^i\|_2^2$;

3. atualize o novo iterado como $\theta_{k+1} = \theta_k - \bar{\alpha}\nabla\ell_i(\theta_k)$, sendo $\bar{\alpha}$ tamanho de passo fixo.

Cada procedimento completo deste é comumente chamado de *época (epoch)*. Tipicamente, o treino de uma rede neural é feito em um número pré-estabelecido de épocas [4, Capítulo 11].

Para a escolha de θ_0, geramos valores aleatórios entre -1 e 1 para as matrizes de pesos, e começamos com todas as coordenadas dos vetores de vieses iguais a 0. Tal escolha de valores iniciais é discutida em [4, Capítulo 11].

No item 3, para o cômputo do gradiente $\nabla\ell_i$, implementamos uma técnica chamada de *backpropagation*, técnica esta que permite o cálculo eficiente das derivadas parciais de ℓ_i. Para não interromper o andamento do presente texto, delegamos uma discussão sobre isso ao apêndice B.

Neste ponto, vale notar que, assim como no capítulo anterior, não utilizamos aqui nenhum critério de parada típico na literatura de otimização, como, por exemplo, parar as iterações quando a norma do gradiente no iterado ficar menor que uma certa tolerância $\epsilon > 0$ [1, Capítulo 1]. Isso se deve ao fato de que, na prática de *machine learning*, nem sempre estamos interessados em atingir um mínimo local; atingir tal ponto pode levar a um problema chamado de *overfitting* [4, Capítulo 11].

Em linhas gerais, isso significa dizer que, mesmo que tenhamos minimizado a função de perda sobre o conjunto de treino, o modelo resultante pode ter um desempenho insatisfatório ao fazer predições

sobre dados futuros; como colocado no Capítulo 2, isto é o que queremos cumprir com uma rede neural. Isto nos leva ao próximo ponto.

5.2.4 Métrica de Avaliação do Modelo

Assim como o conjunto de treino, o conjunto de teste também conta com imagens previamente classificadas. Dito isto, para avaliar a perfomance do modelo após o treino, aplicaremo-lo sobre o conjunto de teste e avaliaremos sua *acurácia* sobre tal conjunto, definida aqui como

$$acurácia = \frac{número\ de\ predições\ corretas}{número\ total\ de\ predições}.$$

Para computar a predição do modelo sobre uma amostra $x^i \in \mathbb{R}^{784}$ do conjunto de teste, tomamos o primeiro índice j dentre as coordenadas de $f(x^i) \in \mathbb{R}^{10}$ com maior valor (sendo $f(x^i)$ o *output* da amostra pelo modelo resultante do treino), e comparamos com o respectivo dígito classificado $z^i \in \{0, 1, ..., 8, 9\}$ da amostra. Se $z^i + 1 = j$, então dizemos que o modelo fez a predição correta.

5.3 RESULTADOS

Para nossos experimentos, utilizamos duas redes neurais diferentes, com as seguintes formas:

- Arquitetura I - [784,32,32,10];
- Arquitetura II - [784,196,49,10];

em que o primeiro e último parâmetros são os números de coordenadas do vetor de entrada e de saída, respectivamente, e os dois do meio são os números de neurônios de cada uma das camadas intermediárias (duas camadas para esse caso).

Modelo	*Learning Rate*	Épocas	Acurácia
Arquitetura I	0,1	1	92,23%
Arq. I	0,1	5	95,30%
Arq. I	0,01	5	91,80%
Arq. I	0,1	20	95,45%
Arquitetura II	0,1	1	93,94%
Arq. II	0,1	5	96,60%
Arq. II	0,01	5	93,39%
Arq. II	0,1	20	97,45%

Tabela 1 – Resultados

Com respeito ao tamanho de passo (*learning rate*), usamos a variante fixa, com dois valores $\bar{\alpha} = 0,1$ e $\bar{\alpha} = 0,01$. Reportamos os valores de acurácia para diferentes números de épocas na Tabela 1. Também incluímos uma segunda tabela (Tabela 2) de alguns *benchmarks* registrados na literatura; retirados de [4, Capítulo 11] e [7]. Vale notar que o último modelo da Tabela 2 é similar aos implementados aqui - duas camadas intermediárias totalizando 300 neurônios, e com MSE como função de perda.

No *script* "mnist_training.py" do repositório, o usuário pode alterar a arquitetura de camadas, número de épocas e a *learning rate*; tais valores estão codificados nas variáveis *layers_list*, *epochs_number* e *learning_rate* respectivamente.

Também ilustramos aqui o processo de treinamento com um gráfico de erro contra número de iterações, para a Arquitetura I.

É interessante notar, na Figura 16, o quão rapidamente o erro diminui antes mesmo de completar uma época (60.000 iterações), em comparação com o restante do procedimento.

Por fim, geramos visualizações de acurácia - Arquitetura I com treinamento em 5 épocas e *learning rate* 0,1 - sobre um conjunto de 100 amostras, e sobre o conjunto de teste todo. Nas figuras 17 e 18,

Modelo	Acurácia
Net-1: Single layer network	80,00%
Net-2: Two layer network	87,00%
Net-3: Locally connected	88,50%
Net-4: Constrained network 1	94,00%
Net-5: Constrained network 2	98,40%
2-layer NN, 300 hidden units, mean square error	95,30%

Tabela 2 – *Benchmarks* externos: [4, Capítulo 11] e [7]

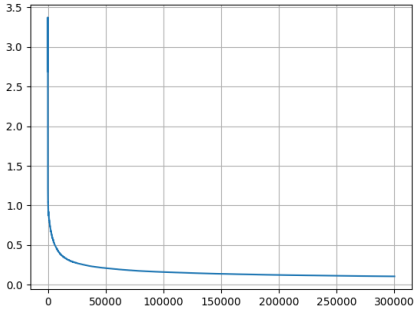

Figura 16 – Tamanho de passo fixo $\bar{\alpha} = 0,1$.

Figura 17 – Conjunto de 100 amostras.

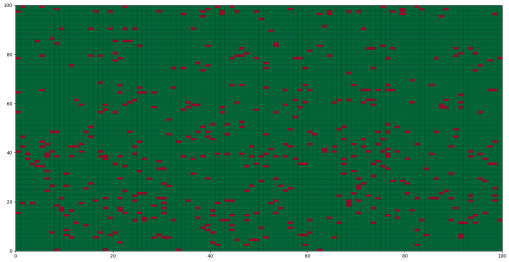

Figura 18 – Todo o conjunto de testes.

os quadrados verdes representam um acerto do modelo, enquanto os vermelhos representam um erro de predição. Dada a comparação com os *benchmarks* externos, consideramos satisfatórios os resultados aqui obtidos.

6 CONCLUSÃO

Neste trabalho fizemos a apresentação e caracterização matemática de uma RN e do problema de treinamento associado. Logo a seguir fizemos a análise de convergência do MGE para funções objetivos fortemente convexos e mais gerais. Desta análise, obtivemos três importantes resultados de convergência: para funções objetivos fortemente convexos, temos garantia de convergência, em valor esperado, do *gap* de otimalidade para tamanhos de passo fixos e decrescente; para funções objetivos gerais, temos convergência, em valor esperado, da média dos quadrados das normas dos gradientes para tamanho de passo fixo.

Em momento seguinte, os resultados da aplicação do MGE em problemas de teste ilustraram, e parecem ter corroborado, os resultados para objetivos convexos. Disto, passamos à implementação de duas RNPs, feita segundo os moldes por nós apresentados, e aplicamos uma versão do MGE no treino destas sobre o *dataset* MNIST – conjunto de imagens de dígitos manuscritos previamente classificados. As medidas de acurácia das RNPs, quando aplicadas sobre um conjunto de testes, foram consideradas satisfatórias quando comparadas a *benchmarks* externos. Tomamos tais resultados como um atestado da eficácia do MGE na realização da tarefa proposta – o treino de redes neurais.

Para possíveis investigações e trabalhos futuros, gostaríamos de continuar estudando variantes de métodos de gradiente estocástico; para citar um exemplo, temos particular interesse no método *Adam*, um método de otimização estocástica recentemente desenvolvido (2014) [6]. Além disso, para o problema específico de classificação de imagens, também gostaríamos de investigar um tipo particular de rede neural desenvolvido nos últimos anos para esse fim: as chamadas

redes convolucionais [7].

REFERÊNCIAS

[1] D. Bertsekas. *Nonlinear programming*. Athena scientific optimization and computation series. Athena Scientific, 2016.

[2] L. Bottou, F. E. Curtis e J. Nocedal. "Optimization Methods for Large-Scale Machine Learning". Em: *SIAM Review* 60.2 (2018), pp. 223–311. DOI: 10.1137/16M1080173.

[3] M. P. Deisenroth, A A. Faisal e C. S. Ong. *Mathematics for machine learning*. Cambridge University Press, 2020.

[4] T. Hastie, R. Tibshirani e J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[5] G. James, D. Witten, T. Hastie e R. Tibshirani. *An introduction to statistical learning: with applications in R*. Springer Publishing Company, Incorporated, 2014.

[6] D.P. Kingma e J. Ba. "Adam: A Method for Stochastic Optimization". Em: *CoRR* abs/1412.6980 (2014).

[7] Y. Lecun, L. Bottou, Y. Bengio e P. Haffner. "Gradient-based learning applied to document recognition". Em: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[8] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[9] M. A. Nielsen. *Neural networks and deep learning*. Determination Press, 2015.

[10] A. Papoulis e S. U. Pillai. *Probability, random variables, and stochastic processes*. McGraw Hill, 2002.

[11] I.H. Sarker. "Machine Learning: Algorithms, Real-World Applications and Research Directions". Em: (2021).

APÊNDICE A – CONCEITOS E DEFINIÇÕES

Vamos aqui registrar alguns conceitos de Estatística necessários para o entendimento do texto.

1. **Aprendizado estatístico** - de modo geral, em um problema de estatísica, temos uma variável *dependente* Y e uma variável *independente* $X = (X_1, X_2, ..., X_p)$, ambas observadas e registradas. Além disso, assumimos que existe a relação

$$Y = f(X) + \epsilon,$$

em que ϵ é um termo aleatório de erro, e f é uma função desconhecida.

Nesse contexto, aprendizado estatístico se refere a um conjunto de técnicas, métodos e ferramentas usadas para estimar f [5, Capítulo 2].

Divide-se tal problema em duas grandes classes, em relação a sua natureza:

- *predição* - quando queremos estimar f a fim de aplicá-la a um novo conjunto de variáveis independentes X e fazer predições sobre sua respectiva saída Y;
- *infererência* - quando estamos interessados em obter *insights* e conhecer relações entre as variáveis X e Y [5, Capítulo 2].

2. **Regressão e Classificação** - em um problema de predição, temos dois casos diferentes em relação à natureza da variável alvo Y a ser predita:

- *regressão* - quando Y é uma variável numérica (*quantitativa*);
- *classificação* - quando Y é uma variável categórica (*qualitativa*) [5, Capítulo 2].

3. **Processo estocástico** - entendemos como processo estocástico, ou *stochastic process*, qualquer conjunto de variáveis aleatórias indexadas por um conjunto numérico. Para o caso em que tal conjunto é $\mathbb{Z}$, um processo estocástico é uma sequência de variáveis aleatórias [10, Capítulo 9].

APÊNDICE B – BACKPROPAGATION

O *backpropagation* é uma maneira de calcular as derivadas parciais da função de perda em relação aos pesos e vieses do modelo.

Sejam W_l e b^l as matrizes de pesos e os vetores de vieses, respectivamente, associadas a cada camada intermediária $l \in \{1, \ldots, L-1\}$. Sejam $\bar{W}$ e $\bar{b}$ as matrizes de pesos e o vetor de vieses associadas à camada de *output*. Então, dado o problema (3) na forma descrita no Capítulo 6 (MSE para a função de perda e sigmóides para as ativações), e seguindo por diferenciação por cadeia, temos a seguinte forma para as derivadas parciais dos pesos e vieses da camada de *output*:

$$\begin{aligned}
\frac{\partial \ell_i}{\partial \bar{w}_{mn}} &= 2(f_m(x^i, \theta_k) - y_m^i)\sigma'(\bar{z}_m)\frac{\partial \bar{z}_m}{\partial \bar{w_{mn}}} \\
&= 2(f_m(x^i, \theta_k) - y_m^i)\sigma'(\bar{z}_m)a_n^L \\
&= s_m a_n^L; \\
\frac{\partial \ell_i}{\partial \bar{b}_m} &= 2(f_m(x^i, \theta_k) - y_m^i)\sigma'(\bar{z}_m) \\
&= s_m;
\end{aligned}$$

em que l_i é a função descrita na seção 5.2.3, $\bar{z_m}$ é a m-ésima coordenada da combinação afim do vetor de ativação $a^L \in \mathbb{R}^{D_L}$ da camada imediatamente anterior; ou seja $\bar{z} = \bar{W}^T a^L + \bar{b}$.

É possível mostrar que, para a camada intermediária imediatamente anterior, as parciais assumem a forma

$$\begin{aligned}
\frac{\partial \ell_i}{\partial w_{mn}^L} &= \delta_m^L a_n^{L-1}, \\
\frac{\partial \ell_i}{\partial b_m^L} &= \delta_m^L;
\end{aligned}$$

em que $\delta^L = \bar{W}^T s \odot \sigma'(z^L)$ com $z^L = W_L^T a^{L-1} + b^L$. Aqui, '$\odot$' denota o produto coordenada a coordenada.

De maneira similar, as derivadas parciais para a l-ésima camada intermediária ficam

$$\frac{\partial \ell_i}{\partial w_{mn}^l} = \delta_m^l a_n^{l-1},$$
$$\frac{\partial \ell_i}{\partial b_m^l} = \delta_m^l;$$

em que $\delta^l = W_{l+1}^T \delta^{l+1} \odot \sigma'(z^l)$.

Tal discussão nos dá uma forma conveninente para o cálculo de todas as parciais, reduzindo os cômputos a multiplicações de matrizes por vetores.

A grosso modo, o procedimento funciona da seguinte maneira: a cada iteração, calculamos todas os vetores de ativação da RNP - um procedimento chamado de *forward pass*. Tal procedimento nos dá todas as quantidades necessárias para calcular os vetores s e δ^l associados a cada camada (note que $\sigma' = 1 - \sigma$). Assim, em posse desses valores, voltamos calculando 'de trás pra frente' tais vetores (por isso o nome *backpropagation*).

Por fim, tendo uma fórmula fechada para todas as derivadas parciais, somos capazes de calcular $\nabla \ell_i$ e atualizar o iterado atual como descrito na seção 5.2.3.

Para uma exposição mais detalhada, encorajamos o leitor a olhar mais em [4, Capítulo 11], e principalmente [9, Capítulo 2].