Nome: Davi Paulino Conceição BV: 3045455 - BCC T6

Matéria: Visão Computacional - Relatório Pesquisa.

Sumário

Algoritmo: Backpropagation e Gradiente Descendente em Redes Neurais Profundas

1. Introdução ao Processo de Aprendizado

No domínio da Inteligência Artificial, o aprendizado de máquina supervisionado é formalmente definido como um problema de otimização. O desafio reside em encontrar uma função matemática capaz de mapear entradas (X) para saídas (Y) com o menor erro possível.

As Redes Neurais Artificiais (RNAs) emergem como a solução primordial para este desafio, inspirando-se na neurobiologia para criar arquiteturas capazes de aprender padrões altamente não-lineares.

A sofisticação das RNAs permite aplicações críticas na sociedade moderna, desde o reconhecimento facial em sistemas de segurança até diagnósticos médicos complexos e previsões de mercado financeiro. Este relatório técnico objetiva detalhar os mecanismos fundamentais que sustentam essa capacidade de aprendizado: o algoritmo de **Backpropagation** e o método de **Gradiente Descendente**, demonstrando como o ajuste sistemático de parâmetros permite que a rede converja para soluções ótimas.

2. Arquitetura e Fluxo de Dados (Forward Pass)

Uma rede neural *feedforward* organiza-se em uma estrutura estratificada composta por:

- **Camada de Entrada:** Recebe os vetores de características (ex: pixels de uma imagem de 28x28 do dataset MNIST resultando em 784 neurônios).
- **Camadas Ocultas:** Estágios intermediários onde ocorre a extração de abstrações. Quanto maior a profundidade, maior a complexidade dos padrões capturados.
- **Camada de Saída:** Produz a inferência final (ex: probabilidades via Softmax para classificação).

A unidade fundamental de processamento é o neurônio artificial, que executa uma **soma ponderada** dos sinais de entrada, adicionando um termo de viés (*bias*): z =\sum wj xj + b. Para introduzir a não-linearidade necessária para resolver problemas complexos, aplica-se uma **função de ativação** ao resultado z:

- **Sigmóide:**\sigma(z) = 1 / (1 + e^{-z}). Mapeia a saída entre 0 e 1. Útil para modelos probabilísticos, mas suscetível ao desvanecimento do gradiente.
- **Tanh (Tangente Hiperbólica):** Mapeia valores entre -1 e 1, centrando os dados na média zero.
- **ReLU (Rectified Linear Unit):** f(z) =\max(0, z). É o padrão em redes profundas por sua

eficiência computacional e por mitigar a saturação em valores positivos.

- **Softmax:** Utilizada na camada de saída para converter vetores em distribuições de probabilidade que somam 1.

3. O Ponto de Partida: Inicialização de Pesos e Vieses

A inicialização de parâmetros não é um passo meramente protocolar, mas um determinante crítico da convergência. Historicamente, utilizou-se a inicialização Gaussiana padrão (média 0, desvio padrão 1). Contudo, em uma rede com, por exemplo, 501 variáveis de entrada (500 pesos + 1 bias), a soma ponderada z resulta em uma distribuição Gaussiana com desvio padrão \approx 22.4. Esse valor elevado empurra as ativações para os extremos das funções sigmóide ou tanh, onde a derivada é quase nula, causando a **saturação dos neurônios** e paralisando o aprendizado.

Técnicas modernas como a inicialização **Xavier/Glória** (para sigmoide/tanh) e **Inicialização He** (específica para ReLU) visam "esmagar os Gaussianos", calibrando a variância dos pesos para manter o sinal constante entre as camadas.

Comparativo de Velocidade de Convergência (Dataset MNIST)

Técnica de Inicialização	Acurácia na 1ª Época	Comportamento do Sinal
Gaussiana Simples	< 87%	Saturação rápida; convergência lenta.

He / Xavier	~ 93%	Variância constante; convergência acelerada.

Nota: Pesquisas da USP demonstram que a inicialização He é vital para a robustez de redes convolucionais aplicadas à visão computacional.

4. Quantificação do Erro: Funções de Custo (Loss Functions)

A função de custo (C) quantifica a discrepância entre a previsão da rede (a) e o rótulo real (y). Minimizar C é o objetivo último do treinamento.

- **Erro Quadrático Médio (MSE):** Aplicado em regressão.
  - *Fórmula:* C(w,b) = \frac{1}{2n} \sum x \|y(x) - a\|^2
- **Entropia Cruzada (Cross-Entropy):** Padrão para classificação. Penaliza severamente erros onde a rede está "confiante e errada".

Aplicações práticas brasileiras desenvolvidas na **UNICAMP** e **USP** utilizam estas funções para otimizar modelos de reconhecimento de plantas nativas e sistemas de previsão de demanda energética nacional, onde a escolha da função de perda impacta diretamente a resiliência do modelo a *outliers*.

5. O Algoritmo de Gradiente Descendente

O **Gradiente Descendente** é o motor de otimização que busca o mínimo da função de custo. A intuição é a de uma bola rolando para o fundo de um vale: o algoritmo calcula o **gradiente** (vetor de derivadas parciais) para identificar a direção de maior subida e move os parâmetros na direção oposta.

O passo de atualização é controlado pela **Taxa de Aprendizado (\eta)**:

- \está muito alta: Risco de ultrapassar o mínimo e divergir.
- \está muito baixa: Processo excessivamente lento e propenso a ficar preso em mínimos

locais sub-ótimos.

Variantes Principais:

1. **Batch Gradient Descent:** Gradiente calculado sobre todo o dataset. Estável, mas proibitivo para grandes volumes de dados.
2. **Stochastic Gradient Descent (SGD):** Atualização por amostra individual. Ruidoso, mas rápido e eficiente para escapar de mínimos locais.
3. **Mini-batch Gradient Descent:** O padrão da indústria. Divide os dados em lotes (ex: 32, 128 amostras), equilibrando estabilidade e velocidade.

Fórmula de atualização: **w = w - \eta \cdot \nabla C**

6. Backpropagation: O Mecanismo de Retropropagação

Popularizado por Rumelhart, Hinton e Williams (1986), o Backpropagation resolveu o problema de como treinar redes multicamadas. Ele utiliza a **Regra da Cadeia** para decompor o impacto de cada peso no erro total. Para um peso w conectando um neurônio j á um neurônio k, a contribuição para o erro é dada pela decomposição das derivadas parciais:

O Backpropagation opera em quatro etapas fundamentais que definem o ciclo de vida de uma interação de treinamento. A primeira é o cálculo da saída da rede através de sucessivas multiplicações de matrizes. A segunda é a definição do erro na camada de saída, que serve como o sinal primordial para a retropropagação. A terceira etapa envolve a propagação desse erro para trás, calculando o erro local para cada neurônio em cada camada oculta. Finalmente, os gradientes são calculados para cada peso e viés individual.

Um desafio matemático significativo no treinamento de redes muito profundas é o fenômeno dos **Gradientes Explosivos ou Desvanecentes**. Devido à natureza multiplicativa da regra da cadeia, se os gradientes foram consistentemente menores que 1, eles tendem a zero conforme retrocedem para as primeiras camadas, impedindo que os neurônios iniciais aprendam. Inversamente, gradientes maiores que 1 podem crescer exponencialmente, causando instabilidade numérica e "Nas" nos pesos.

6.1 Regularização e Generalização

Para garantir que o gradiente descendente não apenas minimize o erro nos dados de treino, mas também generalize para dados não vistos, técnicas de regularização são indispensáveis. A regularização **L2 (Weight Decay)** adiciona uma penalidade proporcional ao quadrado da magnitude dos pesos à função de custo, forçando a rede a preferir pesos menores e distribuídos. Já o **Dropout** introduz ruído durante o treinamento ao desativar aleatoriamente uma fração dos neurônios, forçando a rede a não depender de características específicas e redundantes.

Roteiro do Processo de Retropropagação:

1. **Forward Pass:** Propagação dos dados para gerar a previsão atual.
2. **Cálculo do Erro:** Avaliação do custo C via verdade absoluta.
3. **Backward Pass:** Propagação do erro da saída para a entrada, calculando os gradientes camada por camada.
4. **Atualização:** Ajuste de pesos e vieses via otimizador.

Pesquisadores da **UFABC** desenvolveram otimizações para este fluxo em GPUs nacionais, permitindo que o Backpropagation seja executado de forma eficiente em hardware de médio porte.

7. Otimizadores e Técnicas de Refinamento

Para superar as limitações do SGD, como a oscilação em "ravinas" da função de custo, utilizam-se técnicas avançadas:

- **Momentum:** Acumula um histórico de gradientes passados para manter o "senso de direção", acelerando a convergência.
- **Adam (Adaptive Moment Estimation):** Combina Momentum e RMSProp. Adapta a

taxa de aprendizado individualmente para cada parâmetro.

Estudos Comparativos (UFMG / IA-Labs): Pesquisas da UFMG indicam que o **Adam** é superior para tarefas de **Processamento de Linguagem Natural (NLP) em português**, enquanto o **SGD com Momentum** é frequentemente preferido para **imagens médicas** devido à sua capacidade de generalização superior em domínios visuais específicos.

8. Implementação Prática e Frameworks

A hegemonia do Python no ecossistema de IA é consolidada por frameworks como Tensor Flow/Keras e PyTorch. Abaixo, um exemplo de implementação Keras:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# Arquitetura para reconhecimento de dígitos
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),
    # Dropout(0.2) atua como um ensemble implícito, prevenindo
    # overfitting ao desativar neurônios aleatoriamente
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
]

# Otimizador Adam com parâmetros padrão da literatura
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

9. Conclusão

O aprendizado em Redes Neurais Profundas é um processo sistemático de redução de incerteza. Através da inicialização criteriosa (evitando o desvio padrão crítico de 22.4), do cálculo

rigoroso de derivadas via Backpropagation (regra da cadeia) e da navegação eficiente pelo espaço de parâmetros via Gradiente Descendente, os modelos de IA tornam-se capazes de generalizar conhecimentos complexos. A pesquisa acadêmica brasileira, capitaneada por instituições como UFMG e USP, continua a refinar estes processos, adaptando algoritmos globais aos desafios e dados específicos do cenário nacional.

Referências e Fontes

- BENGIO, Y. *Practical Recommendations for Gradient-Based Training of Deep Architectures*. [S.l.: s.n.], 2012.
- DEEP LEARNING BOOK, Capítulos 12 (Gradiente) e 25 (Inicialização). [S.l.: s.n.].
- IBM TECHNOLOGY. *Backpropagation*. Disponível em: <https://www.ibm.com /br-pt/think/topics/backpropagation>. Acesso em: 20 maio 2026.
- IBM TECHNOLOGY. *Gradient Descent*. Disponível em: <https://www.ibm.com/br-pt/think/topics/gradient-descent>. Acesso em: 20 maio 2026.
- IA-LABS. *Relatórios de Treinamento em Redes Neurais (Pesquisas UFMG, USP, UNICAMP, UFABC)*. Disponível em: <https://ia-labs.com.br/wp-content/uploads/2025/06/Treinamento-de-Redes-Neurais-Estrutura-Essencial-para-Aprender.pdf>. Acesso em: 20 maio 2026.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning representations by back-propagating errors*. [S.l.: s.n.], 1986.