

REDES NEURAIS

Métodos otimizados para treinamento
das redes MLP e regularização

TÓPICOS

1. ALGORITMOS OTIMIZADOS

1. RMSPROP

2. Adam

2. REGULARIZAÇÃO

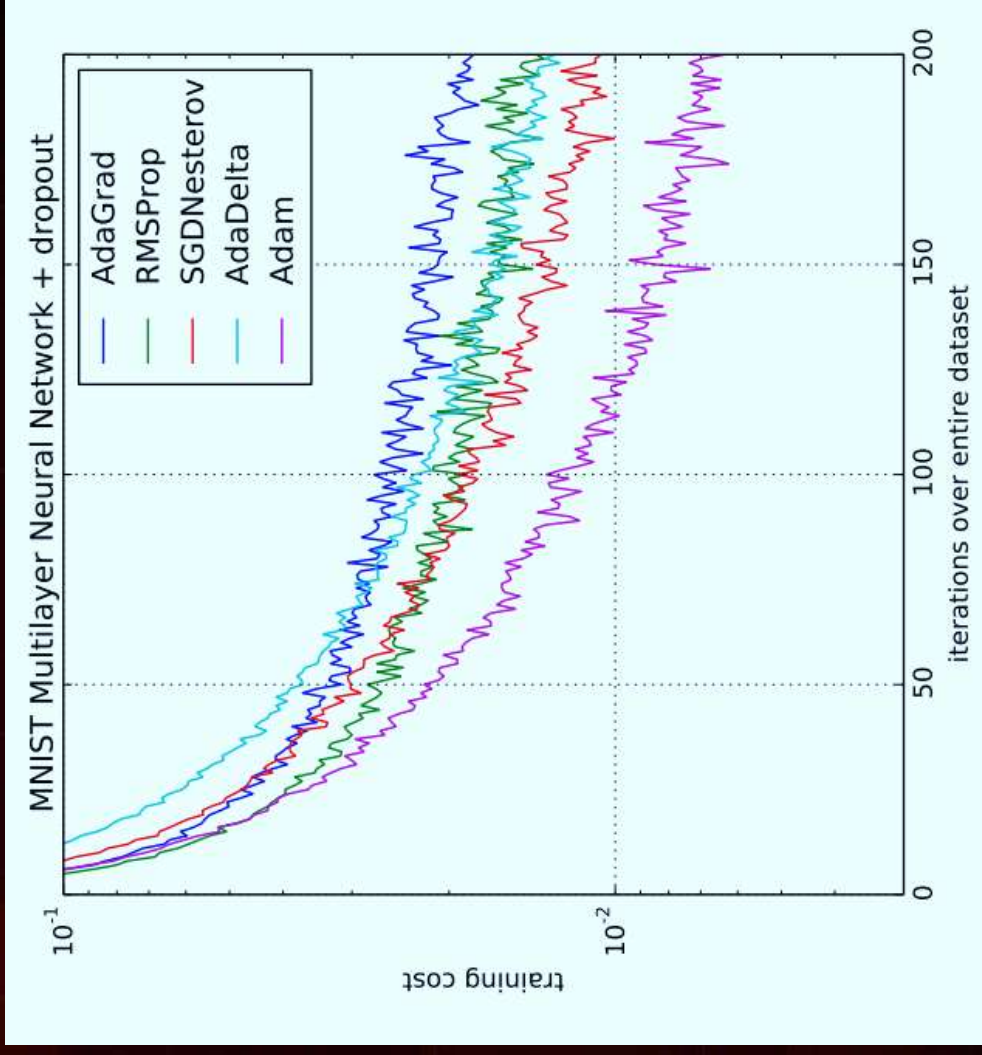
1. L2 e L1

2. Dropout

ALGORITMOS OTIMIZADOS

- Não possuímos informação completa sobre a função de erro
- Dificuldades para encontrar (configurar) uma taxa de aprendizagem adequada
 - Mínimo local em E
 - Plateaus
 - Oscilações
- As adaptações são realizadas em função da derivada parcial do erro quadrático médio

ALGORITMOS OTIMIZADOS

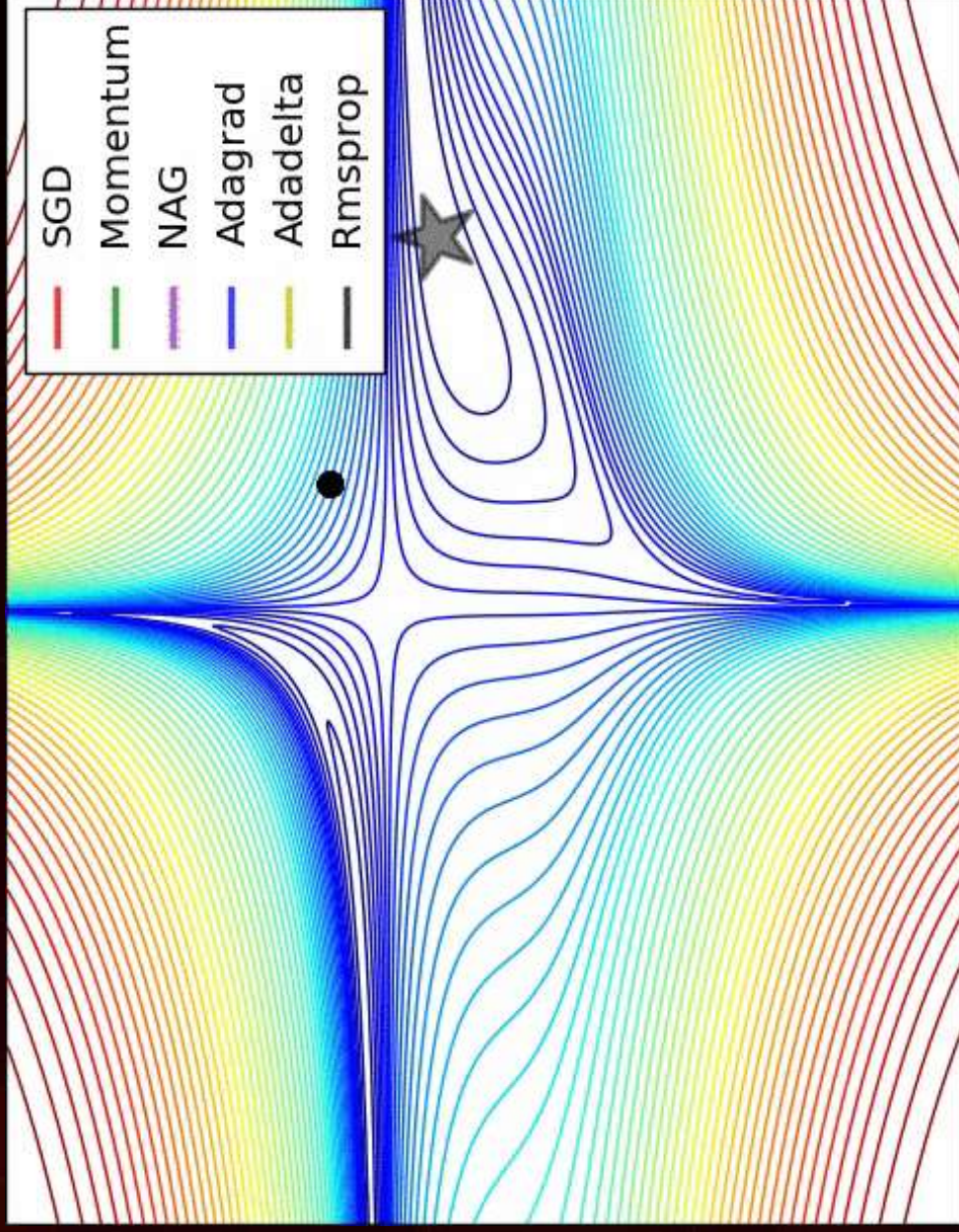


Fonte: <https://arxiv.org/abs/1412.6980>

1. Diversos algoritmos propostos na literatura

1. RProp
2. AdaGrad
3. RMSProp
4. Adam
5. Etc.

ALGORITMOS OTIMIZADOS



Fonte: <https://ml-cheatsheet.readthedocs.io/en/latest/optimizers.html>

RMSProp

- Root Mean Squared Propagation (RMSProp)
- Proposto por Geoffrey Hinton (não-publicado formalmente)
- Calcula a média móvel dos gradientes para cada peso ao longo do treinamento
- Utiliza esse valor para ajustar a atualização
 - Grandes variações reduzem atualização
 - Pequenas variações amplificam a atualização

RMSProp

$$S_w^{rt} = \beta S_w^{rt-1} + (1 - \beta)$$

$$\Delta w = -\frac{\eta}{\sqrt{S_w}} \frac{\partial E}{\partial w}$$

$$\left(\frac{\partial E}{\partial w} \right)^2$$

Parâmetros do algoritmo:
beta = 0,99 (padrão)

Adam

- Adaptive Moment Estimation
- Proposto por Kingma & Ba (2014)
- Consiste numa extensão do RMSProp adicionando o termo de momentum

Referência: <https://arxiv.org/abs/1412.6980>

Adam

$$M_w^t = \beta_1 M_w^{t-1} + (1 - \beta_1) \left(\frac{\partial E}{\partial w} \right)^2$$
$$S_w^t = \beta_2 S_w^{t-1} + (1 - \beta_2) \left(\frac{\partial E}{\partial w} \right)^2$$

PARÂMETROS DO ALGORITMO

- $\beta_1 = 0,9$
(momentum)
- $\beta_2 = 0,999$
(média móvel dos gradientes²)

$$\delta w = -\frac{\eta}{\sqrt{S_w}} M_w$$

REGULARIZAÇÃO

- Tem como objetivo central reduzir a “flexibilidade” do modelo e, como consequência, evitar o overfitting (sobreajuste)
- Regularização L2 / L1
- Dropout

REGULARIZAÇÃO L2

- Estabelece um limite para os pesos evitando a super especialização ou saturação sináptica

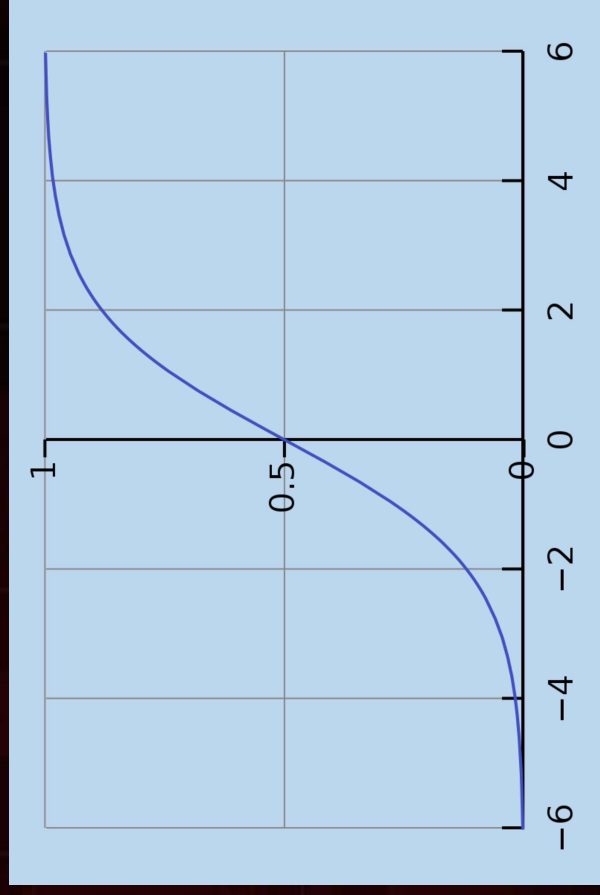
$$E = E_0 + \frac{\lambda}{2n} \sum_w w^2$$

$$E = \frac{1}{2n} \sum_j (d_j - y_j)^2 + \frac{\lambda}{2n} \sum_w w^2$$

REGULARIZAÇÃO L2

O QUE A L2 FAZ?

- Reduz o tamanho dos pesos
- Por que isso ajuda com o overfitting?



Fonte: https://en.wikipedia.org/wiki/Logistic_function

$$v_k = \sum w$$
$$y_k = f(v_k)$$

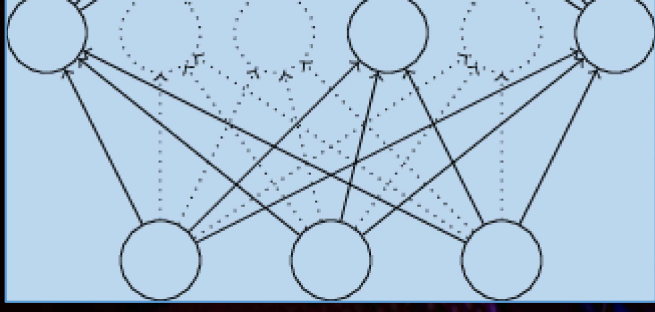
REGULARIZAÇÃO L1

- Semelhante à regularização L2, porém, considera a soma absoluta dos pesos ao invés da soma quadrática
- Como não há penalidade quadrática, alguns pesos podem assumir valores altos sem comprometer o custo
- Bastante utilizada para seleção de atributos

Dropout

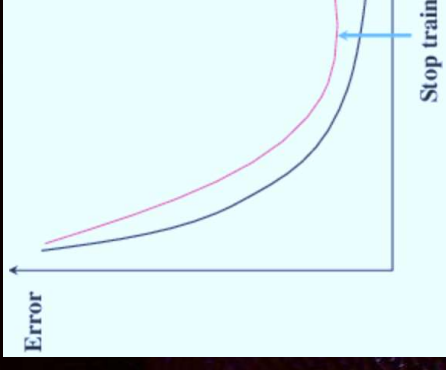
- Desativar algumas unidades durante o treinamento da rede
- Evita que um “único” neurônio decore o padrão
- A “carga” do exemplo é distribuída pelos pesos da rede
- Parâmetro: probabilidade de ativação das unidades
- Efeito prático similar à regularização L2

Fonte: <https://www.deeplearningbook.com.br/capitulo-23-como-funciona-o-dropout/>
Referência: <https://arxiv.org/pdf/1207.0580.pdf>



OUTRAS MELHORIAS

- Obter mais dados representativos
- Aumento artificial de dados (data augmentation)
- Parada prematura do treinamento
- Adequação da função de custo (loss)
 - Diversas funções disponíveis na literatura
 - <https://pytorch.org/docs/stable/nn.html#loss-functions>
 - Funções adequadas para problemas específicos



O QUE VIMOS?

- Conhecemos alguns algoritmos otimizados para treinamento da rede MLP
- Entendemos o conceito de regularização

PRÓXIMA VIDEOAULA

- Realizaremos experimentos práticos com a rede MLP

ATÉ A PRÓXIMA!!