



Técnicas de Regularização em Aprendizado Profundo

Rickson Monteiro
Miguel Fernandes
Evandro Rocha

Universidade Federal do Rio de Janeiro
UFRJ/COPPE/PEE

October 14, 2025



Agenda

Introdução

Otimização com Restrições

Data Augmentation

Curvas de Aprendizado

Dropout

Batch Normalization

Combinar Técnicas de Regularização

Conclusão



Introdução



Problema

- ▶ Por quê estudar regularização em *deep learning*?
- ▶ R.: *Overfitting*



Possíveis causas de *Overfitting*

- ▶ Complexidade do modelo;
- ▶ Poucos dados de Treinamento;
- ▶ Ruídos excessivos nos dados;
- ▶ Treinamento por tempo excessivo.



Formalização matemática

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{\mathcal{N}} \ln \left(\frac{4((2\mathcal{N})^{d_{vc}} + 1)}{\delta} \right)}$$

Onde:

- ▶ \mathcal{N} : Número de exemplos de treinamento (tamanho da amostra).
- ▶ \mathcal{H} : Conjunto de hipóteses (espaço de modelos) que o algoritmo está usando (a complexidade da família de modelos).
- ▶ d_{vc} : Dimensão VC (Vapnik-Chervonenkis). Medida da capacidade/complexidade do espaço de hipóteses \mathcal{H} .



Exemplo de Overfitting

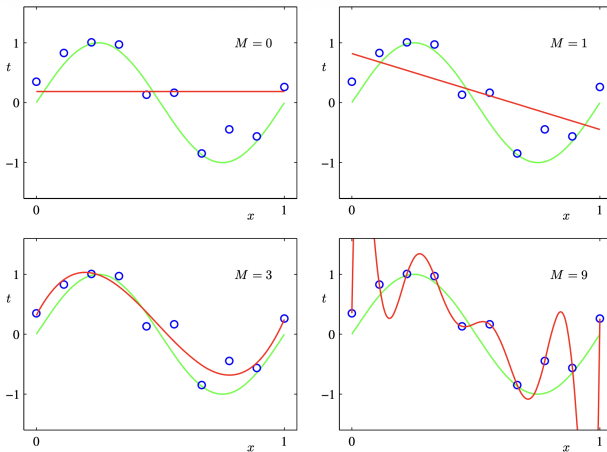
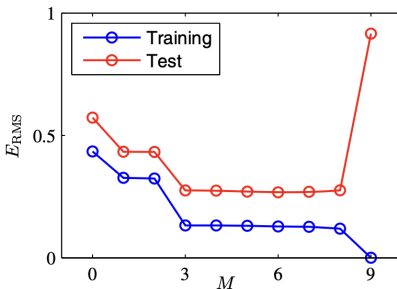


Figure 1.4 Plots of polynomials having various orders M , shown as red curves, fitted to the data set shown in Figure 1.2.



Erro x Grau do polinômio

Figure 1.5 Graphs of the root-mean-square error, defined by (1.3), evaluated on the training set and on an independent test set for various values of M .





Possíveis soluções

1. Aumentar a quantidade de dados de treinamento.



Alternativa 1: aumentar a quantidade de dados no treinamento

"The best way to make a machine learning model generalize better is to train it on more data. Of course, in practice, the amount of data we have is limited." — Goodfellow et al., Deep Learning

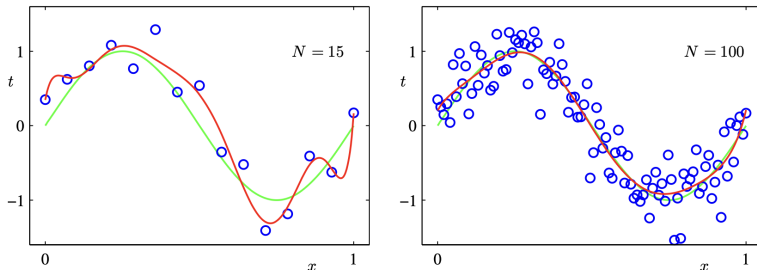


Figure 1.6 Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.



Possíveis soluções

1. Aumentar a quantidade de dados de treinamento.
2. **Regularização.**



Definição

“Any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.”

— Goodfellow et al. Deep Learning



Alternativa 2: aplicar regularização

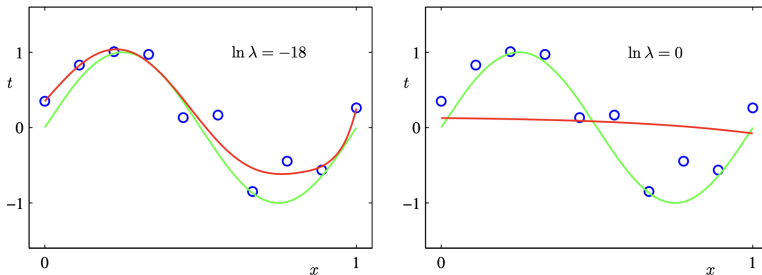


Figure 1.7 Plots of $M = 9$ polynomials fitted to the data set shown in Figure 1.2 using the regularized error function (1.4) for two values of the regularization parameter λ corresponding to $\ln \lambda = -18$ and $\ln \lambda = 0$. The case of no regularizer, i.e., $\lambda = 0$, corresponding to $\ln \lambda = -\infty$, is shown at the bottom right of Figure 1.4.



Table 1.2 Table of the coefficients w^* for $M = 9$ polynomials with various values for the **regularization** parameter λ . Note that $\ln \lambda = -\infty$ corresponds to a model with no **regularization**, i.e., to the graph at the bottom right in Figure 1.4. We see that, as the value of λ increases, the typical magnitude of the coefficients gets smaller.

	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
w_0^*	0.35	0.35	0.13
w_1^*	232.37	4.74	-0.05
w_2^*	-5321.83	-0.77	-0.06
w_3^*	48568.31	-31.97	-0.05
w_4^*	-231639.30	-3.89	-0.03
w_5^*	640042.26	55.28	-0.02
w_6^*	-1061800.52	41.32	-0.01
w_7^*	1042400.18	-45.95	-0.00
w_8^*	-557682.99	-91.53	0.00
w_9^*	125201.43	72.68	0.01

Na prática, o valor de λ pode ser validado com o auxílio do procedimento de validação cruzada.



Otimização com Restrições



Penalização da Norma dos Parâmetros

Regularização L1

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \text{Custo}(y_i, f_{\theta}(x_i)) + \lambda \|\theta\|_1$$

- ▶ Seleção de Variáveis (*Sparsity/Esparsidade*)
- ▶ Redução dos Coeficientes (*Shrinkage/Encolhimento*) a uma taxa constante

Regularização L2

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \text{Custo}(y_i, f_{\theta}(x_i)) + \lambda \|\theta\|_2^2$$

- ▶ Contribui para a estabilidade numérica
- ▶ Redução de Coeficientes a uma taxa proporcional (*Shrinkage/Encolhimento*)

"Regularization terms of the form (above) encourage the model weights to have a smaller magnitude and hence introduce a bias towards functions that vary more slowly with changes in the inputs." — Bishop et al., Deep Learning



Regularização L1 – Gradiente e Shrinkage

Para um modelo de Regressão Linear, onde a função de perda não regularizada é o Erro Quadrático Médio (MSE), a função de custo regularizada L1, $\mathcal{L}(w)$, é definida por:

$$\mathcal{L}(w) = \text{MSE}(w) + \lambda \sum_{j=1}^M |w_j| \quad (1)$$

onde:

$w = (w_1, w_2, \dots, w_M)$ Vetor de coeficientes

$\text{MSE}(w) = \frac{1}{N} \sum_{i=1}^N (y^{(i)} - \hat{y}^{(i)})^2$ Erro quadrático médio;

$\lambda \geq 0$ Intensidade da penalização;

$\sum_{j=1}^M |w_j| = \|w\|_1$ Norma L_1 dos coeficientes.



Regularização L1 – Gradiente e Shrinkage

O objetivo é encontrar o vetor de coeficientes w^* que minimiza a função de custo:

$$w^* = \arg \min_w \left[\text{MSE}(w) + \lambda \sum_{j=1}^M |w_j| \right] \quad (2)$$




Regularização L1 – Gradiente e Shrinkage

Analizando a derivada parcial da função de custo em relação a um coeficiente específico w_j , obtém-se:

$$\frac{\partial \mathcal{L}(w)}{\partial w_j} = \frac{\partial \text{MSE}(w)}{\partial w_j} + \lambda \frac{\partial |w_j|}{\partial w_j} \quad (3)$$

O termo $\frac{\partial \text{MSE}(w)}{\partial w_j}$ representa o gradiente da função de perda não regularizada (g_j).



Regularização L1 – Gradiente e Shrinkage

O termo de penalidade tem a derivada:

$$\frac{\partial |w_j|}{\partial w_j} = \begin{cases} +1, & \text{se } w_j > 0 \\ -1, & \text{se } w_j < 0 \\ \text{indefinido,} & \text{se } w_j = 0 \end{cases} = \text{sgn}(w_j), \quad \text{para } w_j \neq 0 \quad (4)$$

A regra de atualização do coeficiente w_j usando o Gradiente Descendente é (para $w_j \neq 0$):

$$w_j^{\text{novo}} = w_j^{\text{antigo}} - \eta \frac{\partial \mathcal{L}(w)}{\partial w_j} = w_j^{\text{antigo}} - \eta (g_j + \lambda \cdot \text{sgn}(w_j^{\text{antigo}})) \quad (5)$$

$$w_j^{\text{novo}} = w_j^{\text{antigo}} - \eta g_j - \eta \lambda \text{sgn}(w_j^{\text{antigo}}) \quad (6)$$



Regularização L1 – Gradiente e Shrinkage

A solução direta do gradiente é problemática porque $\partial \mathcal{L}(w)/\partial w_j$ não é diferenciável em $w_j = 0$. Isso significa que os pesos podem não ser zerados exatamente na solução.

Uma abordagem para contornar isso é usar o conceito de **subgradiente**, aplicado através da técnica conhecida como *soft-thresholding*, que define cada peso como:

$$w_j^* = \begin{cases} w_j^{OLS} - \frac{\lambda}{Q_j}, & w_j^{OLS} > \frac{\lambda}{Q_j} \\ 0, & |w_j^{OLS}| \leq \frac{\lambda}{Q_j} \\ w_j^{OLS} + \frac{\lambda}{Q_j}, & w_j^{OLS} < -\frac{\lambda}{Q_j} \end{cases}$$

onde w_j^{OLS} é o peso obtido pela regressão linear sem regularização, e $Q_j = \sum_i x_{ij}^2$.

A implementação detalhada da técnica de soft-thresholding não será abordada aqui.



Regularização L2 – Gradiente e Shrinkage

Para a regularização L2 a função de custo é:

$$\mathcal{L}(w) = \text{MSE}(w) + \lambda \sum_{j=1}^M w_j^2 \quad (7)$$

A derivada parcial em relação a w_j é:

$$\frac{\partial \mathcal{L}(w)}{\partial w_j} = \frac{\partial \text{MSE}(w)}{\partial w_j} + \lambda \frac{\partial (w_j^2)}{\partial w_j} = g_j + 2\lambda w_j \quad (8)$$



Regularização L2 – Gradiente e Shrinkage

Logo, a regra de atualização do coeficiente w_j é:

$$w_j^{\text{novo}} = w_j^{\text{antigo}} - \eta \frac{\partial \mathcal{L}(w)}{\partial w_j} \quad (9)$$

$$w_j^{\text{novo}} = w_j^{\text{antigo}} - \eta g_j - 2\eta \lambda w_j^{\text{antigo}} \quad (10)$$



Regularização L2 – Remoção de multicolinearidade

Além da redução proporcional dos coeficientes, a regularização L2 também contribui para a estabilidade numérica do problema de inversão da matriz $X^T X$ no caso da presença de *features* altamente correlacionadas.

$$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = \|y - X\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \quad (11)$$

Derivando em relação a \mathbf{w} e igualando a zero:

$$(X^T X + \lambda I)\mathbf{w} = X^T y \quad (12)$$

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T y \quad (13)$$



L1 x L2

Característica	L1 (Lasso)	L2 (Ridge)
Efeito principal	Seleciona variáveis — coeficientes podem ser exatamente zero	Reduz coeficientes proporcionalmente ao peso
Uso recomendado	Quando se deseja simplificação e seleção automática de variáveis	Quando todas as variáveis são relevantes e deseja-se apenas reduzir variância
Sparsity (esparsidade)	Sim — tende a gerar muitos coeficientes nulos	Não — tende a manter todos os coeficientes diferentes de zero
Robustez à multicolinearidade	Baixa — escolhe uma entre variáveis correlacionadas	Alta — distribui pesos entre variáveis correlacionadas de forma estável



Elastic Net

Fórmula:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \text{Loss}(y_i, f_{\theta}(x_i)) + \lambda_1 \|\theta\|_1 + \lambda_2 \|\theta\|_2^2$$

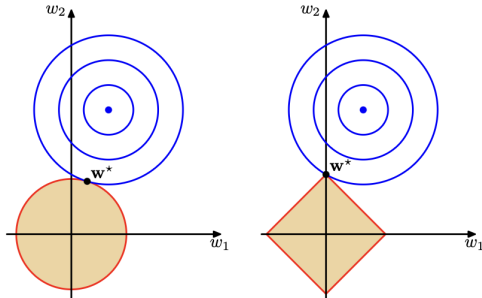
Principais características e problemas que resolve:

- ▶ Combina os benefícios do L1 e L2:
 - ▶ Seleção de variáveis (sparsity) do L1;
 - ▶ Regularização estável e redução de multicolinearidade do L2;



Interpretação Geométrica

Figure 3.4 Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer $q = 2$ on the left and the lasso regularizer $q = 1$ on the right, in which the optimum value for the parameter vector \mathbf{w} is denoted by \mathbf{w}^* . The lasso gives a sparse solution in which $w_1^* = 0$.





Data Augmentation



Aumento de Dados

- ▶ Em determinadas tarefas, a predição deve ser **equivariante**.
Ex: segmentação em objetos com translação;
- ▶ Em outras, a predição deve ser **invariante** a uma ou mais transformações nos dados de entrada: translação, tamanho, rotação, **ruído**, etc.



Aumento de Dados

Formas de tornar o modelo invariante a transformações:

- ▶ Pre-processamento: gerar *features* invariantes às transformações (Ex. Histograma, FFT);
- ▶ Regularização: penaliza alterações na saída do modelo para uma mesma entrada com as transformações aplicadas;
- ▶ Modificar a estrutura da rede;
- ▶ **Aumentar o conjunto de dados de treinamento.**



Invariância

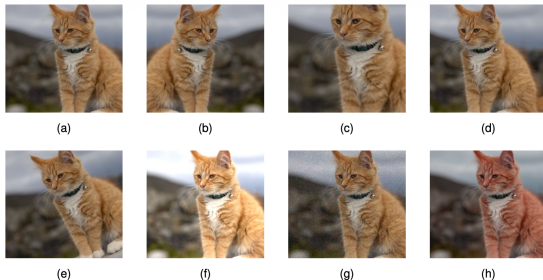


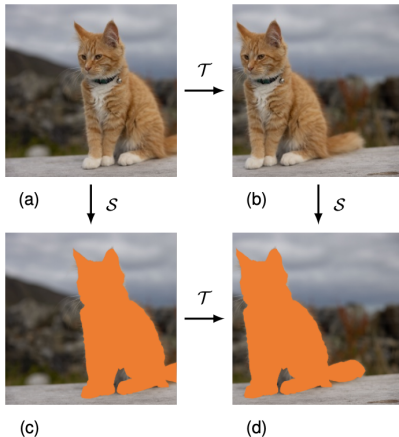
Figure 9.1 Illustration of data set augmentation, showing (a) the original image, (b) horizontal inversion, (c) scaling, (d) translation, (e) rotation, (f) brightness and contrast change, (g) additive noise, and (h) colour shift.

- Reconhecimento de imagem
- OCR: cuidado (b e d, 6 e 9)



Equivariância

Figure 9.2 Illustration of equivariance, corresponding to (9.2). If an image (a) is first translated to give (b) and then segmented to give (d), the result is the same as if the image is first segmented to give (c) and then translated to give (d).





Curvas de Aprendizado



Encerramento Antecipado (*Early stopping*)

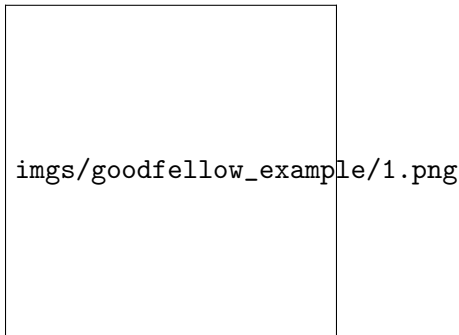


Figure: Erro de log-verossimilhança em função da quantidade de épocas de treinamento. Rede maxout treinada com dataset MNIST. Retirado de Goodfellow et al., *Deep Learning*.

Encaramos o “tempo de treinamento” como um hiperparâmetro.



Encerramento Antecipado (*Early stopping*)

- ▶ Regularização menos intrusiva.
- ▶ “Hiperparâmetro especial” por ser obtido em uma única execução do treinamento.
- ▶ O maior custo computacional seria o cálculo do erro para o conjunto de validação, mas pode ser assíncrono.



Encerramento Antecipado (*Early stopping*)

```
1: Let  $n$  be the number of steps between evaluations.
2: Let  $p$  be the "patience," the number of times to observe worsening validation set error before giving up.
3: Let  $\theta_0$  be the initial parameters.
4:  $\theta \leftarrow \theta_0$ 
5:  $i \leftarrow 0$ 
6:  $j \leftarrow 0$ 
7:  $v \leftarrow \infty$ 
8:  $\theta^* \leftarrow \theta$ 
9:  $i^* \leftarrow i$ 
10: while  $j < p$  do
11:   Update  $\theta$  by running the training algorithm for  $n$  steps.
12:    $i \leftarrow i + n$ 
13:    $v' \leftarrow \text{ValidationSetError}(\theta)$ 
14:   if  $v' < v$  then
15:      $j \leftarrow 0$ 
16:      $\theta^* \leftarrow \theta$ 
17:      $i^* \leftarrow i$ 
18:      $v \leftarrow v'$ 
19:   else
20:      $j \leftarrow j + 1$ 
21:   end if
22: end while
23: Return best parameters  $\theta^*$  and best number of training steps  $i^*$ .
```



Encerramento Antecipado (*Early stopping*)

Após encontrar os melhores parâmetros, há duas estratégias possíveis para o treinamento:

- ▶ Reiniciar os pesos do modelo e treinar do zero até a quantidade “ótima” de passos;
- ▶ Manter os pesos do ponto ótimo e continuar o treinamento com todos os dados (treinamento + validação).
 - ▶ Perde-se a referência da quantidade de passos,
 - ▶ Mas evita o custo de reiniciar o treinamento.



Convenção “clássica”: compromisso viés-variância

Assumimos que os dados são gerados por uma função $f(x)$ tal que

$$y = f(x) + \varepsilon,$$

onde o ruído ε tem média zero e variância σ^2 . Assim,

$$y_i = f(x_i) + \varepsilon_i,$$

onde ε_i é uma amostra de ruído.



Convenção “clássica”: compromisso viés-variância

Queremos encontrar uma função $\hat{f}(x; D)$ que aproxime a função verdadeira $f(x)$ tanto quanto possível

$$D = \{(x_1, y_1), \dots, (x_n, y_n)\}.$$

Ou seja, minimizar o erro quadrático médio entre y e $\hat{f}(x; D)$ para os pontos de treinamento x_1, \dots, x_n e para pontos fora do conjunto de treinamento.

Por conveniência, omite-se o subscrito D , tal que $\hat{f}(x; D) = \hat{f}(x)$.



Convenção “clássica”: compromisso viés-variância

Por conveniência, omite-se o subscrito D , tal que $\hat{f}(x; D) = \hat{f}(x)$.
Define-se o erro quadrático médio (MSE) do modelo como

$$\begin{aligned}\text{MSE} &\triangleq \mathbb{E}\left[(y - \hat{f}(x))^2\right] \\ &= \mathbb{E}\left[(f(x) + \varepsilon - \hat{f}(x))^2\right] \quad \text{dado que } y \triangleq f(x) + \varepsilon \\ &= \mathbb{E}\left[(f(x) - \hat{f}(x))^2\right] + 2\mathbb{E}\left[(f(x) - \hat{f}(x))\varepsilon\right] + \mathbb{E}[\varepsilon^2].\end{aligned}$$

O segundo termo da equação é zero:

$$\begin{aligned}\mathbb{E}\left[(f(x) - \hat{f}(x))\varepsilon\right] &= \mathbb{E}\left[f(x) - \hat{f}(x)\right] \mathbb{E}[\varepsilon] \quad \varepsilon \text{ é independente de } x \\ &= 0 \quad \text{uma vez que } \mathbb{E}[\varepsilon] = 0.\end{aligned}$$



Convenção “clássica”: compromisso viés-variância

O terceiro termo da equação anterior é a variância do ruído:

$$\mathbb{E}[\varepsilon^2] = \sigma^2.$$

Expandindo o termo restante:

$$\begin{aligned}\mathbb{E}\left[(f(x) - \hat{f}(x))^2\right] &= \mathbb{E}\left[(f(x) - \mathbb{E}[\hat{f}(x)] + \mathbb{E}[\hat{f}(x)] - \hat{f}(x))^2\right] \\ &= \mathbb{E}\left[(f(x) - \mathbb{E}[\hat{f}(x)])^2\right] + \mathbb{E}\left[(\mathbb{E}[\hat{f}(x)] - \hat{f}(x))^2\right] \\ &\quad + 2\mathbb{E}\left[(f(x) - \mathbb{E}[\hat{f}(x)])(\mathbb{E}[\hat{f}(x)] - \hat{f}(x))\right].\end{aligned}$$



Convenção “clássica”: compromisso viés-variância

Uma vez que $f(x)$ não é uma variável aleatória, mas uma função determinística de x :

$$\begin{aligned}\mathbb{E}\left[\left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2\right] &= \mathbb{E}\left[f(x)^2\right] - 2\mathbb{E}\left[f(x)\mathbb{E}[\hat{f}(x)]\right] + \mathbb{E}\left[\mathbb{E}[\hat{f}(x)]^2\right] \\ &= f(x)^2 - 2f(x)\mathbb{E}[\hat{f}(x)] + (\mathbb{E}[\hat{f}(x)])^2 \\ &= \left(f(x) - \mathbb{E}[\hat{f}(x)]\right)^2.\end{aligned}$$



Convenção “clássica”: compromisso viés-variância

Pelo mesmo raciocínio, podemos expandir o termo cruzado e demonstrar que é igual a zero:

$$\begin{aligned} & \mathbb{E} \left[(f(x) - \mathbb{E}[\hat{f}(x)]) (\mathbb{E}[\hat{f}(x)] - \hat{f}(x)) \right] \\ &= \mathbb{E} \left[f(x) \mathbb{E}[\hat{f}(x)] - f(x) \hat{f}(x) - (\mathbb{E}[\hat{f}(x)])^2 + \mathbb{E}[\hat{f}(x)] \hat{f}(x) \right] \\ &= f(x) \mathbb{E}[\hat{f}(x)] - f(x) \mathbb{E}[\hat{f}(x)] - (\mathbb{E}[\hat{f}(x)])^2 + (\mathbb{E}[\hat{f}(x)])^2 \\ &= 0. \end{aligned}$$



Convenção “clássica”: compromisso viés-variância

Finalmente:

$$\text{MSE} = (f(x) - \mathbb{E}_{\mathbb{D}}[\hat{f}(x)])^2 + \mathbb{E}_{\mathbb{D}} \left[(\mathbb{E}_{\mathbb{D}}[\hat{f}(x)] - \hat{f}(x))^2 \right] + \sigma^2 \quad (14)$$

$$= \text{Bias}[\hat{f}(x)]^2 + \text{Var}[\hat{f}(x)] + \sigma^2. \quad (15)$$

A função de custo do erro quadrático médio é obtida a partir do valor esperado em relação ao espaço de entradas X :

$$\text{MSE} = \mathbb{E}_x \left\{ \text{Bias}_D[\hat{f}(x; D)]^2 + \text{Var}_D[\hat{f}(x; D)] \right\} + \sigma^2. \quad (16)$$



Convenção “clássica”: compromisso viés-variância

imgs/bishop_example/113.png



Convenção “clássica”: compromisso viés-variância

imgs/bishop_example/114.png

Crença clássica: número de parâmetros deve ser limitado ao tamanho do conjunto de dados



Convenção “moderna”: Double Descent

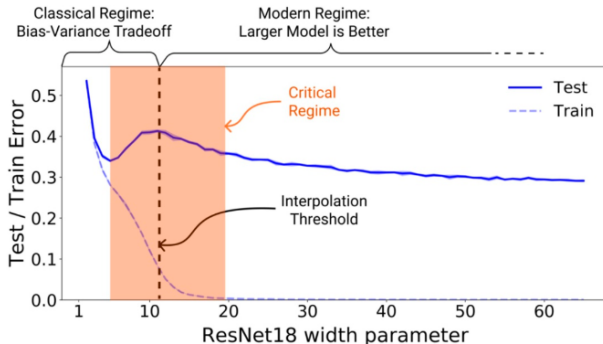


Figure 9.9 Plot of training set and test set errors for a large neural network model called ResNet18 trained on an image classification problem versus the complexity of a model. The horizontal axis represents a hyperparameter governing the number of hidden units and hence the overall number of weights and biases in the network. The vertical dashed line, labelled ‘interpolation threshold’ indicates the level of model complexity at which the model is capable, in principle, of achieving zero error on the training set. [From Nakkiran *et al.* (2019) with permission.]

- **Redes profundas:** bom desempenho mesmo quando a quantidade de parâmetros excede em muito o necessário para ajustar aos dados de treinamento



Convenção “moderna”: **Double Descent**

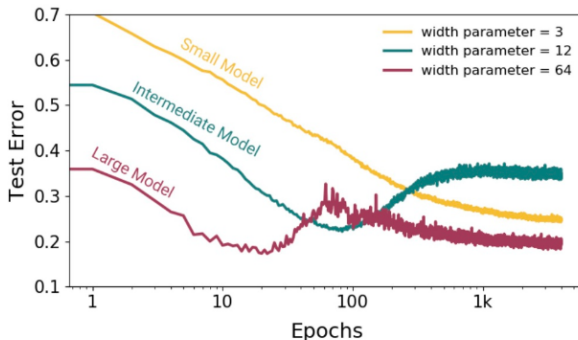


Figure 9.10 Plot of test set error versus number of epochs of gradient descent training for ResNet18 models of various sizes. The effective model complexity increases with the number of training epochs, and the double descent phenomenon is observed for a sufficiently large model. [From Nakkiran *et al.* (2019) with permission.]

Complexidade efetiva do modelo: quantidade máxima de dados de treinamento tal que o modelo atinja erro zero (limiar de interpolação). Dali em diante, a descendência dupla ocorre quando a complexidade do modelo excede esse limiar.



Dropout



Dropout - Motivação

- ▶ **Problema:** Redes neurais profundas sofrem com *overfitting*
- ▶ **Causa:** Co-adaptação excessiva entre neurônios
- ▶ **Solução:** Durante o treinamento, “desligar” aleatoriamente alguns neurônios

Intuição

Força a rede a não depender demais de neurônios específicos, criando representações mais robustas

Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting” (2014)



Dropout - Como Funciona

Treinamento:

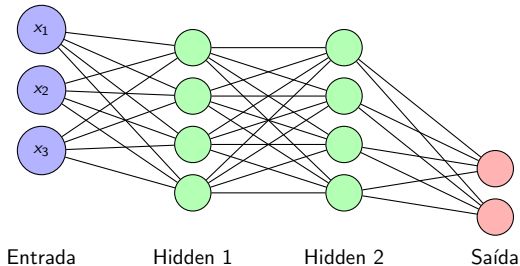
- ▶ Para cada iteração:
 - ▶ Gerar máscara binária aleatória
 - ▶ $m_i \sim \text{Bernoulli}(p)$
 - ▶ Aplicar: $h_i = m_i \cdot a_i$
- ▶ Taxa de dropout p : prob. de manter neurônio

Teste/Inferência:

- ▶ Usar todos os neurônios
- ▶ Escalar saídas por p :
- ▶ $h_i = p \cdot a_i$



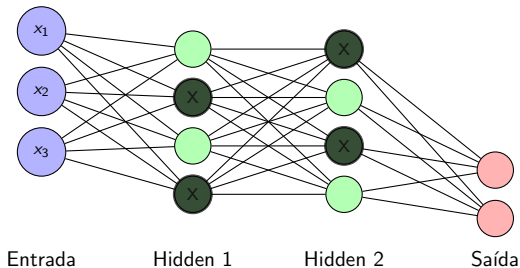
Exemplo Visual: Implementação do Dropout





Exemplo Visual: Implementação do Dropout

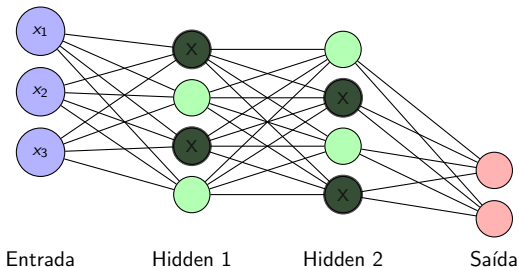
Batch 1: $p = 0.5$





Exemplo Visual: Implementação do Dropout

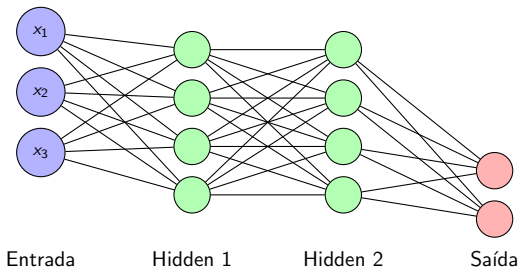
Batch 2: Nova máscara aleatória





Exemplo Visual: Implementação do Dropout

Teste: Todos neurônios ativos (escalados por p)





Dropout - Implementação Matemática

Durante o Treinamento

$$\mathbf{r} \sim \text{Bernoulli}(p)$$

$$\tilde{\mathbf{x}} = \mathbf{r} \odot \mathbf{x}$$

$$\mathbf{y} = W\tilde{\mathbf{x}} + \mathbf{b}$$

Durante o Teste

$$\mathbf{y} = W(p\mathbf{x}) + \mathbf{b}$$

Notação:

- ▶ \mathbf{r} : máscaras binárias
- ▶ \mathbf{x} : entrada (input)
- ▶ $\tilde{\mathbf{x}}$: entrada com dropout
- ▶ \mathbf{y} : saída (output)
- ▶ W : matriz de pesos da camada
- ▶ \mathbf{b} : vetor de bias
- ▶ \odot : produto elemento a elemento
- ▶ p : prob. manter neurônio

Por que $p\mathbf{x}$ no teste?

Compensar diferença de escala: no treinamento só p dos neurônios



Dropout - O que é a Matriz W ?

A matriz W contém os pesos da camada:

Se temos:

- ▶ Entrada \mathbf{x} : 3 elementos
- ▶ Saída \mathbf{y} : 2 elementos

Então W é uma matriz 2×3 :

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}$$

Cada w_{ij} conecta o elemento j da entrada ao elemento i da saída.

Cálculo da saída:

$$\mathbf{y} = W\mathbf{x} + \mathbf{b}$$


Exemplo: Se W é 2×3 :

$$W = \begin{bmatrix} 0.5 & -0.2 & 0.8 \\ 0.3 & 0.7 & -0.1 \end{bmatrix}$$

Com dropout no teste:

$$\mathbf{y} = W(\mathbf{p}\mathbf{x}) + \mathbf{b}$$

Primeiro: $\mathbf{p}\mathbf{x}$, depois:
 $W \times (\mathbf{p}\mathbf{x})$.



Dropout - Explicação da Implementação

Passo a passo da implementação:

1. **Gerar máscaras aleatórias:** Para cada neurônio, sorteia-se um valor binário (0 ou 1) com probabilidade p de ser 1
2. **Aplicar máscaras:** Multiplica-se cada ativação pela máscara correspondente (neurônios com máscara 0 são "desligados")
3. **Propagar adiante:** Usa-se as ativações modificadas para calcular a saída da camada
4. **Durante o teste:** Não há dropout, mas multiplica-se as ativações por p para compensar a diferença de escala

Por que funciona?

- ▶ Força a rede a não depender de neurônios específicos
- ▶ Cria um efeito de ensemble implícito (várias subredes diferentes)
- ▶ Reduz co-adaptação entre neurônios



Dropout - Exemplo Prático

Exemplo numérico:

Suponha uma camada com entrada $\mathbf{x} = [0.2, 0.5, 0.8, 0.3]$ e $p = 0.5$.

Durante o treinamento:

- ▶ Sorteamos $\mathbf{r} \sim \text{Bernoulli}(0.5)$, por exemplo $\mathbf{r} = [1, 0, 1, 0]$
- ▶ Aplicando o dropout: $\tilde{\mathbf{x}} = \mathbf{r} \odot \mathbf{x} = [0.2, 0, 0.8, 0]$
- ▶ O cálculo da saída: $\mathbf{y} = W\tilde{\mathbf{x}} + \mathbf{b}$

Durante o teste:

- ▶ Compensamos a escala:
$$\mathbf{y} = W(p\mathbf{x}) + \mathbf{b} = W([0.1, 0.25, 0.4, 0.15]) + \mathbf{b}$$

Interpretação: No treinamento, os neurônios 2 e 4 foram “desligados”, forçando a rede a usar apenas os neurônios 1 e 3.



Dropout - Vantagens e Limitações

Vantagens:

- ▶ Simples de implementar
- ▶ Muito eficaz contra overfitting
- ▶ Compatível com diversas arquiteturas
- ▶ Baixo custo computacional

Limitações:

- ▶ Pode tornar treinamento mais lento
- ▶ Menos eficaz em redes muito profundas

Dica Prática

- ▶ Começar com $p = 0.5$ para camadas ocultas
- ▶ $p = 0.8$ para camada de entrada
- ▶ Ajustar baseado na performance de validação



Dropout - Variações e Extensões

Inverted Dropout:

- ▶ **Treinamento:** $h_i = \frac{m_i \cdot a_i}{p}$
- ▶ **Teste:** $h_i = a_i$ (sem scaling)
- ▶ **Vantagem:** Não precisa ajustar no teste
- ▶ **Uso:** Implementação mais simples

DropConnect:

- ▶ Remove conexões (pesos) ao invés de neurônios
- ▶ $r_{ij} \sim \text{Bernoulli}(p)$
- ▶ $\tilde{W}_{ij} = r_{ij} \cdot W_{ij}$
- ▶ Mais granular que dropout

Spatial Dropout:

- ▶ Para CNNs: remove feature maps inteiros
- ▶ Evita correlação espacial
- ▶ Melhor para dados com estrutura espacial

Gaussian Dropout:

- ▶ Usa ruído gaussiano ao invés de máscaras binárias
- ▶ $h_i = a_i \cdot \mathcal{N}(1, \sigma^2)$
- ▶ Transição mais suave



Batch Normalization



Batch Normalization - Motivação

- ▶ **Problema:** Internal Covariate Shift
 - ▶ Distribuição das ativações muda durante o treinamento
 - ▶ Cada camada precisa se adaptar às mudanças das camadas anteriores
 - ▶ Torna o treinamento mais lento e instável
- ▶ **Solução:** Normalizar as entradas de cada camada
- ▶ **Benefícios:**
 - ▶ Treinamento mais rápido e estável
 - ▶ Permite learning rates maiores
 - ▶ Reduz dependência da inicialização
 - ▶ Efeito regularizador (reduz overfitting)

Ioffe & Szegedy "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" (2015)



Batch Normalization - Como Funciona

Algoritmo (para um mini-batch)

Para cada feature i no mini-batch $\mathcal{B} = \{x_1, x_2, \dots, x_m\}$:

1. **Calcular média:** $\mu_{\mathcal{B}} = \frac{1}{m} \sum_{k=1}^m x_k$
2. **Calcular variância:** $\sigma_{\mathcal{B}}^2 = \frac{1}{m} \sum_{k=1}^m (x_k - \mu_{\mathcal{B}})^2$
3. **Normalizar:** $\hat{x}_k = \frac{x_k - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$
4. **Escalar e deslocar:** $y_k = \gamma \hat{x}_k + \beta$

Onde:

- ▶ γ, β : parâmetros aprendíveis (scale e shift)
- ▶ ϵ : pequena constante para estabilidade numérica ($\sim 10^{-5}$)

Importância de γ e β :

- ▶ Permitem que a rede desfaça a normalização se necessário
- ▶ $\gamma = \sigma_{\mathcal{B}}$ e $\beta = \mu_{\mathcal{B}}$ recuperam a distribuição original
- ▶ Inicializados como $\gamma = 1$ e $\beta = 0$



Exemplo Visual: Batch Normalization

Entrada (Mini-batch):

x1	x2	x3
2.1	0.8	-1.2
1.8	1.1	-0.9
2.3	0.9	-1.1
1.9	1.0	-1.0

Após Normalização:

\hat{x}_1	\hat{x}_2	\hat{x}_3
0.45	-1.22	-1.22
-0.89	1.22	1.22
1.34	-0.41	-0.41
-0.45	0.41	0.41

Cálculos por feature:

- ▶ $\mu_1 = 2.0, \sigma_1 = 0.22$
- ▶ $\mu_2 = 0.95, \sigma_2 = 0.12$
- ▶ $\mu_3 = -1.05, \sigma_3 = 0.12$

Cada coluna agora tem: $\mu = 0$,
 $\sigma = 1$

Saída Final:

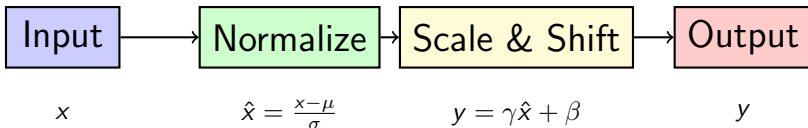
$$y = \gamma \hat{x} + \beta$$

onde γ e β são aprendidos



Batch Normalization - Fluxo do Processo

Sequência de Transformações:



Resumo

- ▶ **Normalizar:** Centra e padroniza os dados ($\mu = 0, \sigma = 1$)
- ▶ **Escalar & Deslocar:** Permite à rede aprender a distribuição ideal
- ▶ **Parâmetros:** γ (escala) e β (deslocamento) são treináveis



Batch Normalization - Posicionamento

Posição Padrão (Ioffe & Szegedy):

- ▶ Antes da função de ativação
- ▶ $y = f(BN(Wx + b))$
- ▶ **Mais comum na prática**

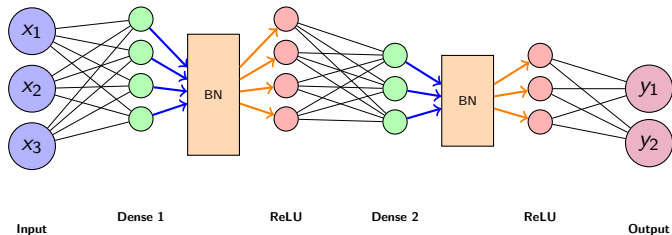
Posição Alternativa:

- ▶ Após a função de ativação
- ▶ $y = BN(f(Wx + b))$
- ▶ Menos utilizada



Batch Normalization - Representação visual

Exemplo: Rede com Batch Normalization



Fluxo: Input \rightarrow Dense \rightarrow BatchNorm \rightarrow ReLU \rightarrow Dense \rightarrow BatchNorm \rightarrow ReLU \rightarrow Output



Batch Normalization - Vantagens e Limitações

Vantagens:

- ▶ Acelera convergência significativamente
- ▶ Permite learning rates maiores
- ▶ Menos sensível à inicialização
- ▶ Evita overfitting (efeito regularizador)
- ▶ Reduz gradient vanishing
- ▶ Amplamente suportado

Limitações:

- ▶ Problemas com batch size pequeno
- ▶ Custo computacional adicional
- ▶ Memória extra para estatísticas



Alternativas ao Batch Normalization

▶ **Layer Normalization:**

- ▶ Normaliza ao longo das features (eixo das características)
- ▶ Independe do tamanho do batch
- ▶ Muito útil para RNNs e Transformers

▶ **Group Normalization:**

- ▶ Divide features em grupos e normaliza cada grupo
- ▶ Intermediário entre Layer e Instance Normalization
- ▶ Eficaz para CNNs com batch pequeno

▶ **Instance Normalization:**

- ▶ Normaliza cada amostra independentemente
- ▶ Útil para transferência de estilo em imagens
- ▶ Preserva contraste entre amostras



Combinação de Técnicas de Regularização



Combinação de Técnicas de Regularização

- ▶ Em problemas reais, uma única técnica de regularização pode não ser suficiente para evitar overfitting.
- ▶ É comum e recomendado combinar diferentes técnicas para potencializar o efeito regularizador.
- ▶ Combinações bem escolhidas podem melhorar a generalização e a robustez do modelo.
- ▶ A escolha das técnicas depende do problema, arquitetura e dados disponíveis.



Por que combinar técnicas de regularização?

- ▶ Cada técnica atua de forma diferente para reduzir o overfitting:
 - ▶ L1/L2 controlam a magnitude dos pesos
 - ▶ Dropout força a rede a não depender de neurônios específicos
 - ▶ Batch Normalization estabiliza o treinamento, acelerando a convergência
 - ▶ Data Augmentation aumenta a diversidade dos dados
- ▶ Combinar técnicas potencializa a regularização e cobre diferentes fontes de overfitting
- ▶ Prática comum em redes profundas modernas
- ▶ Pode melhorar robustez e generalização sem grande aumento de custo computacional



Exemplos de combinações

- ▶ **EfficientNet (Tan & Le, 2019):**
 - ▶ Data Augmentation + Batch Norm + Dropout + L2
 - ▶ AutoAugment para melhor generalização em CNNs
- ▶ **DenseNet (Huang et al., 2017):**
 - ▶ Batch Norm + Dropout + L2 + Data Augmentation
 - ▶ Conexões densas para melhor fluxo de gradientes
- ▶ **Show and Tell (Vinyals et al., 2015):**
 - ▶ Early Stopping + Dropout + L2 + Data Augmentation
 - ▶ Image captioning com CNN + LSTM
- ▶ **AWD-LSTM (Merity et al., 2017):**
 - ▶ Dropout + L2 + DropConnect
 - ▶ Regularização agressiva para modelagem de linguagem
- ▶ **Recomendações:**
 - ▶ Testar diferentes combinações, monitorando validação
 - ▶ Cuidado com excesso: regularização demais pode prejudicar o aprendizado



Conclusão



Resumo das Técnicas de Regularização

- ▶ **Regularização é fundamental** para evitar overfitting em deep learning
- ▶ **Múltiplas abordagens complementares:**
 - ▶ **Penalização:** L1, L2
 - ▶ **Estrutural:** Dropout
 - ▶ **Dados:** Data Augmentation
 - ▶ **Critério de Parada:** Early Stopping
 - ▶ **Normalizações:** Batch Normalization
- ▶ **Cada técnica tem suas características:**
 - ▶ **L1/L2:** Simples, sempre aplicável
 - ▶ **Dropout:** Eficaz para fully-connected layers (camadas totalmente conectadas)
 - ▶ **Batch Normalization:** Normaliza as ativações, acelera o treinamento (permite taxas de aprendizado maiores) e atua como regularizador (dificulta o overfitting)



Conclusões e Recomendações

- ▶ **Escolha da técnica** depende do contexto:
 - ▶ **Tipo de dados:** imagem, texto, tabular
 - ▶ **Arquitetura:** CNN, RNN, Transformer
 - ▶ **Recursos:** tamanho do dataset, batch size, tempo
- ▶ **Podemos combinar técnicas** para melhores resultados:
 - ▶ L2 + Dropout + Batch Norm (combinação clássica)
 - ▶ Data Augmentation + técnicas estruturais
- ▶ **Regularização bem aplicada** = melhor generalização e modelos mais robustos (menos sensível a ruídos e variações)



Referências

- ▶ **Goodfellow, I., Bengio, Y., & Courville, A.** (2016). *Deep Learning*. MIT Press. Disponível em: <https://www.deeplearningbook.org/>
- ▶ **Bishop, C. M.** (2006). *Pattern Recognition and Machine Learning*. Springer.
- ▶ **Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.** (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929-1958.
- ▶ **Ioffe, S., & Szegedy, C.** (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ICML 2015*.
- ▶ **He, K., Zhang, X., Ren, S., & Sun, J.** (2016). Deep Residual Learning for Image Recognition. *CVPR 2016*.



Referências

- ▶ **Tan, M., & Le, Q. V.** (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *ICML 2019*.
- ▶ **Huang, G., et al.** (2017). Densely Connected Convolutional Networks. *CVPR 2017*.
- ▶ **Vinyals, O., et al.** (2015). Show and Tell: A Neural Image Caption Generator. *CVPR 2015*.
- ▶ **Merity, S., et al.** (2017). Regularizing and Optimizing LSTM Language Models. *arXiv:1708.02182*.
- ▶ **Ng, A.** (2016). Machine Learning Yearning. Disponível em: <https://www.mlyearning.org/>



Perguntas

Perguntas?