Deep Learning

Overfitting, Underfitting e Regularização

Lucas Silveira Kupssinskü

Machine Learning Theory and Applications Laboratory
PUCRS

agosto de 2026

Visão Geral

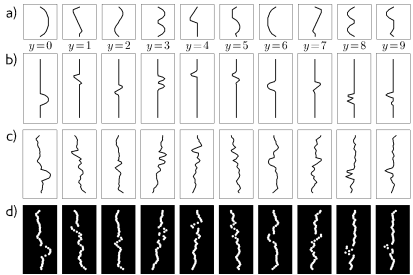
1. **Motivação: overfitting na prática**
2. **Fontes de erro: ruído, viés e variância**
3. **Trade-off viés-variância e double descent**
4. **Regularização explícita e empírica**
5. **Regularização implícita**

Visão Geral

1. **Motivação: overfitting na prática**
2. **Fontes de erro: ruído, viés e variância**
3. **Trade-off viés-variância e double descent**
4. **Regularização explícita e empírica**
5. **Regularização implícita**

Conjunto de dados: MNIST 1D[1]

- Versão simplificada do MNIST, projetada para experimentos rápidos com redes neurais.
- Cada classe possui um *template* unidimensional que sofre transformações estocásticas.
- Pipeline de geração:
  1. *Templates* para as 10 classes.
  2. Distorções aleatórias (translação, escala, espelhamento).
  3. Adição de ruído.
  4. Subamostragem em 40 pontos.
- Repositório: *github.com/greydanus/mnist1d.*



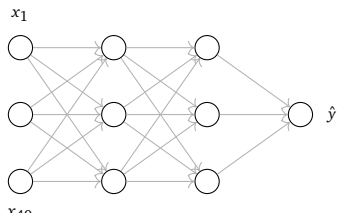
a) *Templates*, b) exemplos de treino, c) ruído, d) subamostragem em 40 pontos.

[1] Sam Greydanus. "Scaling down deep learning". Em: arXiv preprint arXiv:2011.14439 (2020).

MLP usado no experimento

- Entradas: 40 (uma por ponto amostrado).
- Saída: 10 logits, ativação *softmax*.
- Duas camadas ocultas, 100 unidades cada, ativação ReLU.
- Conjunto de treino: 4000 instâncias.
- Otimizador: SGD, *batch size* 100, *learning rate* 0.1.
- Total de iterações: 6000, equivalente a:

$$\text{épocas} = \frac{6000 \cdot 100}{4000} = 150.$$



Esquema (representação reduzida).

Resultados do treinamento

- A taxa de erro no *conjunto de treino* cai a zero.
- A função de custo no *conjunto de teste* primeiro diminui, mas a partir de certo ponto começa a aumentar.
- O modelo continua se ajustando ao ruído do treino e generaliza pior, comportamento clássico de *overfitting*.
- Capacidade alta sem regularização leva a esse padrão.

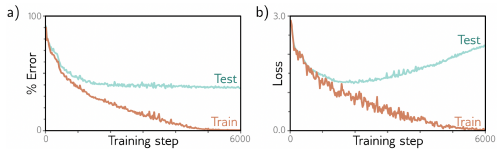


Figure 8.2 MNIST-1D results. a) Percent classification error as a function of the training step. The training set errors decrease to zero, but the test errors do not drop below $\sim 40\%$. This model doesn't generalize well to new test data. b) Loss as a function of the training step. The training loss decreases steadily toward zero. The test loss decreases at first but subsequently increases as the model becomes increasingly confident about its (wrong) predictions.

Por que isso ocorre? Toy MLP de 3 unidades

- MLP com uma camada oculta de 3 unidades ReLU.
- Parâmetros nas linhas pontilhadas estão congelados.
- Variando $\boldsymbol{\phi} = \{\beta, \omega_1, \omega_2, \omega_3\}$ obtemos diferentes funções lineares por partes.
- Em e), f) e g): parametrizações distintas geram saídas distintas, mostrando como diferentes mínimos do treino correspondem a funções muito diferentes.

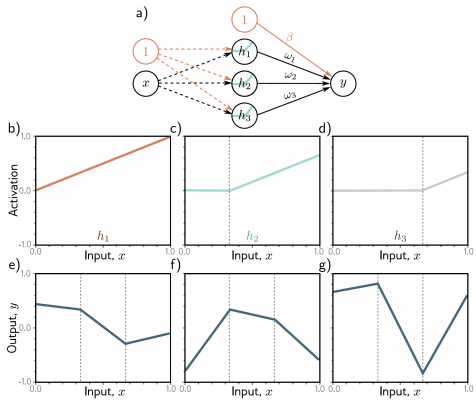


Figure 8.4 Simplified neural network with three hidden units. a) The weights and biases between the input and hidden layer are fixed (dashed arrows). b–d) They are chosen so that the hidden unit activations have slope one, and their joints are equally spaced across the interval, with joints at $x = 0$, $x = 1/3$, and $x = 2/3$, respectively. Modifying the remaining parameters $\boldsymbol{\phi} = \{\beta, \omega_1, \omega_2, \omega_3\}$ can create any piecewise linear function over $x \in [0, 1]$ with joints at $1/3$ and $2/3$. e–g) Three example functions with different values of the parameters $\boldsymbol{\phi}$.

Visão Geral

1. Motivação: overfitting na prática
2. **Fontes de erro: ruído, viés e variância**
3. Trade-off viés-variância e double descent
4. Regularização explícita e empírica
5. Regularização implícita

Três fontes de erro de teste

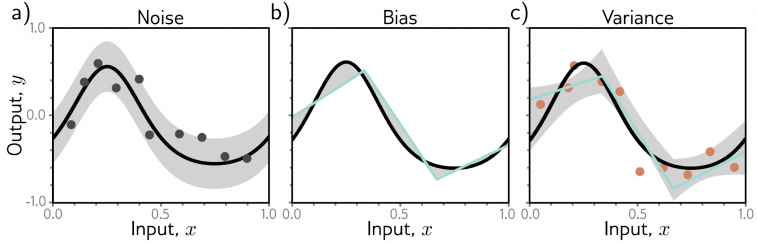
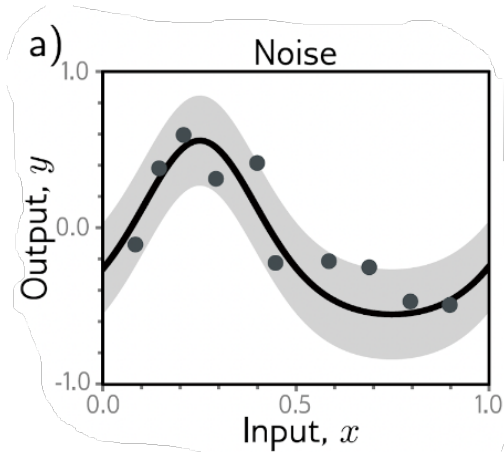


Figure 8.5 Sources of test error. a) Noise. Data generation is noisy, so even if the model exactly replicates the true underlying function (black line), the noise in the test data (gray points) means that some error will remain (gray region represents two standard deviations). b) Bias. Even with the best possible parameters, the three-region model (cyan line) cannot exactly fit the true function (black line). This bias is another source of error (gray regions represent signed error). c) Variance. In practice, we have limited noisy training data (orange points). When we fit the model, we don't recover the best possible function from panel (b) but a slightly different function (cyan line) that reflects idiosyncrasies of the training data. This provides an additional source of error (gray region represents two standard deviations). Figure 8.6 shows how this region was calculated.

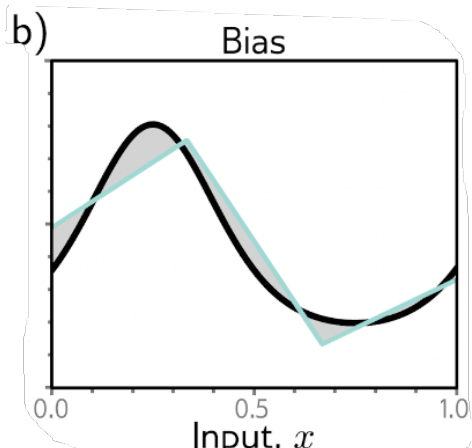
Ruído

- Mesmo recuperando exatamente a função geradora dos dados (em preto), o erro não vai a zero.
- Origem: fatores estocásticos no processo gerador, ou variáveis observadas que não capturam toda a informação relevante.
- Componente irredutível: nenhum modelo elimina.



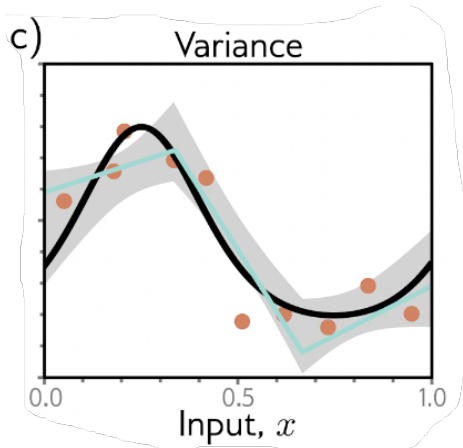
Viés

- Mesmo com os melhores parâmetros possíveis, o MLP em ciano não modela perfeitamente a função em preto.
- Diferença entre a função que o modelo é *capaz* de representar e a função real.
- Todo modelo carrega um *viés indutivo*: sem ele, não há generalização possível.



Variância

- Ao treinar com um número limitado de instâncias, não recuperamos necessariamente o melhor ajuste à função geradora.
- Pequenas mudanças no conjunto de treino produzem modelos diferentes.
- Variância mede o quanto o modelo aprendido oscila com o *dataset* de treino.



Decomposição viés-variância: *setup*

Considere regressão escalar, com:

- Função geradora $\mu(x) = \mathbb{E}[y \mid x]$.
- Observações $y = \mu(x) + \epsilon$, em que $\mathbb{E}[\epsilon] = 0$ e $\mathrm{Var}(\epsilon) = \sigma^2$.
- Predição $f(x; \mathcal{D})$ treinada em um *dataset* $\mathcal{D}$.
- Custo quadrático $\ell = (y - f(x; \mathcal{D}))^2$.

Queremos decompor o erro esperado em x:

$$\mathbb{E}_{\mathcal{D}, y}\left[(y - f(x; \mathcal{D}))^2\right].$$

Estratégia: dois passos de soma e subtração de um termo conveniente. Primeiro $\mu(x)$, isolando o ruído. Depois o modelo médio $\bar{f}(x) = \mathbb{E}_{\mathcal{D}}[f(x; \mathcal{D})]$, separando viés e variância.

Passo 1: isolar o ruído

Fixamos $f(x)$ e tomamos a esperança apenas em y. Somando e subtraindo $\mu(x)$:

$$y - f(x) = \big(y - \mu(x)\big) + \big(\mu(x) - f(x)\big).$$

Elevando ao quadrado:

$$(y - f(x))^2 = (y - \mu(x))^2 \;+\; 2\,(y - \mu(x))(\mu(x) - f(x)) \;+\; (\mu(x) - f(x))^2.$$

Aplicando $\mathbb{E}_y$ e usando que $(\mu(x) - f(x))$ é constante em y, o termo cruzado se anula:

$$\mathbb{E}_y[(y - \mu(x))(\mu(x) - f(x))] = (\mu(x) - f(x))\;\underbrace{\mathbb{E}_y[y - \mu(x)]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_y[(y - f(x))^2] = \underbrace{\mathbb{E}_y[(y - \mu(x))^2]}_{\sigma^2,\ \text{ruído}} + \underbrace{(\mu(x) - f(x))^2}_{\text{erro do modelo}}.$$

Passo 1: isolar o ruído

Fixamos $f(x)$ e tomamos a esperança apenas em y. Somando e subtraindo $\mu(x)$:

$$y - f(x) = \big(y - \mu(x)\big) + \big(\mu(x) - f(x)\big).$$

Elevando ao quadrado:

$$(y - f(x))^2 = (y - \mu(x))^2 \; + \; 2\,(y - \mu(x))(\mu(x) - f(x)) \; + \; (\mu(x) - f(x))^2.$$

Aplicando $\mathbb{E}_y$ e usando que $(\mu(x) - f(x))$ é constante em y, o termo cruzado se anula:

$$\mathbb{E}_y[(y - \mu(x))(\mu(x) - f(x))] = (\mu(x) - f(x)) \underbrace{\mathbb{E}_y[y - \mu(x)]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_y[(y - f(x))^2] = \underbrace{\mathbb{E}_y[(y - \mu(x))^2]}_{\sigma^2,\ \text{ruído}} + \underbrace{(\mu(x) - f(x))^2}_{\text{erro do modelo}}.$$

Passo 1: isolar o ruído

Fixamos $f(x)$ e tomamos a esperança apenas em y. Somando e subtraindo $\mu(x)$:

$$y - f(x) = \big(y - \mu(x)\big) + \big(\mu(x) - f(x)\big).$$

Elevando ao quadrado:

$$(y - f(x))^2 = (y - \mu(x))^2 \; + \; 2\,(y - \mu(x))(\mu(x) - f(x)) \; + \; (\mu(x) - f(x))^2.$$

Aplicando $\mathbb{E}_y$ e usando que $(\mu(x) - f(x))$ é constante em y, o termo cruzado se anula:

$$\mathbb{E}_y[(y - \mu(x))(\mu(x) - f(x))] = (\mu(x) - f(x)) \underbrace{\mathbb{E}_y[y - \mu(x)]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_y[(y - f(x))^2] = \underbrace{\mathbb{E}_y[(y - \mu(x))^2]}_{\sigma^2,\ \text{ruído}} + \underbrace{(\mu(x) - f(x))^2}_{\text{erro do modelo}}.$$

Passo 1: isolar o ruído

Fixamos $f(x)$ e tomamos a esperança apenas em y. Somando e subtraindo $\mu(x)$:

$$y - f(x) = \big(y - \mu(x)\big) + \big(\mu(x) - f(x)\big).$$

Elevando ao quadrado:

$$(y - f(x))^2 = (y - \mu(x))^2 \; + \; 2\,(y - \mu(x))(\mu(x) - f(x)) \; + \; (\mu(x) - f(x))^2.$$

Aplicando $\mathbb{E}_y$ e usando que $(\mu(x) - f(x))$ é constante em y, o termo cruzado se anula:

$$\mathbb{E}_y[(y - \mu(x))(\mu(x) - f(x))] = (\mu(x) - f(x)) \underbrace{\mathbb{E}_y[y - \mu(x)]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_y[(y - f(x))^2] = \underbrace{\mathbb{E}_y[(y - \mu(x))^2]}_{\sigma^2,\ \text{ruído}} + \underbrace{(\mu(x) - f(x))^2}_{\text{erro do modelo}}.$$

Passo 2: separar viés e variância

O termo de erro do modelo ainda depende do *dataset* via $f(x;\mathcal{D})$. Tomamos agora a esperança sobre $\mathcal{D}$, somando e subtraindo $\bar{f}(x)$:

$$\mu(x) - f(x;\mathcal{D}) = \big(\mu(x) - \bar{f}(x)\big) + \big(\bar{f}(x) - f(x;\mathcal{D})\big).$$

Elevando ao quadrado:

$$(\mu - f)^2 = (\mu - \bar{f})^2 + 2\,(\mu - \bar{f})(\bar{f} - f) + (\bar{f} - f)^2.$$

Aplicando $\mathbb{E}_{\mathcal{D}}$, o termo cruzado se anula porque $(\mu - \bar{f})$ é constante em $\mathcal{D}$ e $\mathbb{E}_{\mathcal{D}}[\bar{f} - f] = \bar{f} - \bar{f} = 0$:

$$\mathbb{E}_{\mathcal{D}}\big[(\mu - \bar{f})(\bar{f} - f)\big] = (\mu - \bar{f})\,\underbrace{\mathbb{E}_{\mathcal{D}}[\bar{f} - f]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_{\mathcal{D}}[(\mu(x) - f(x;\mathcal{D}))^2] = \underbrace{(\mu(x) - \bar{f}(x))^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(x;\mathcal{D}) - \bar{f}(x))^2]}_{\text{variância}}.$$

Passo 2: separar viés e variância

O termo de erro do modelo ainda depende do *dataset* via $f(x;\mathcal{D})$. Tomamos agora a esperança sobre $\mathcal{D}$, somando e subtraindo $\bar{f}(x)$:

$$\mu(x) - f(x;\mathcal{D}) = \big(\mu(x) - \bar{f}(x)\big) + \big(\bar{f}(x) - f(x;\mathcal{D})\big).$$

Elevando ao quadrado:

$$(\mu - f)^2 = (\mu - \bar{f})^2 \;+\; 2\,(\mu - \bar{f})(\bar{f} - f) \;+\; (\bar{f} - f)^2.$$

Aplicando $\mathbb{E}_{\mathcal{D}}$, o termo cruzado se anula porque $(\mu - \bar{f})$ é constante em $\mathcal{D}$ e $\mathbb{E}_{\mathcal{D}}[\bar{f} - f] = \bar{f} - \bar{f} = 0$:

$$\mathbb{E}_{\mathcal{D}}\big[(\mu - \bar{f})(\bar{f} - f)\big] = (\mu - \bar{f})\,\underbrace{\mathbb{E}_{\mathcal{D}}[\bar{f} - f]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_{\mathcal{D}}[(\mu(x) - f(x;\mathcal{D}))^2] = \underbrace{(\mu(x) - \bar{f}(x))^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(x;\mathcal{D}) - \bar{f}(x))^2]}_{\text{variância}}.$$

Passo 2: separar viés e variância

O termo de erro do modelo ainda depende do *dataset* via $f(x;\mathcal{D})$. Tomamos agora a esperança sobre $\mathcal{D}$, somando e subtraindo $\bar{f}(x)$:

$$\mu(x) - f(x;\mathcal{D}) = \big(\mu(x) - \bar{f}(x)\big) + \big(\bar{f}(x) - f(x;\mathcal{D})\big).$$

Elevando ao quadrado:

$$(\mu - f)^2 = (\mu - \bar{f})^2 \;+\; 2\,(\mu - \bar{f})(\bar{f} - f) \;+\; (\bar{f} - f)^2.$$

Aplicando $\mathbb{E}_{\mathcal{D}}$, o termo cruzado se anula porque $(\mu - \bar{f})$ é constante em $\mathcal{D}$ e $\mathbb{E}_{\mathcal{D}}[\bar{f} - f] = \bar{f} - \bar{f} = 0$:

$$\mathbb{E}_{\mathcal{D}}\big[(\mu - \bar{f})(\bar{f} - f)\big] = (\mu - \bar{f})\,\underbrace{\mathbb{E}_{\mathcal{D}}[\bar{f} - f]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_{\mathcal{D}}[(\mu(x) - f(x;\mathcal{D}))^2] = \underbrace{(\mu(x) - \bar{f}(x))^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(x;\mathcal{D}) - \bar{f}(x))^2]}_{\text{variância}}.$$

Passo 2: separar viés e variância

O termo de erro do modelo ainda depende do *dataset* via $f(x;\mathcal{D})$. Tomamos agora a esperança sobre $\mathcal{D}$, somando e subtraindo $\bar{f}(x)$:

$$\mu(x) - f(x;\mathcal{D}) = \big(\mu(x) - \bar{f}(x)\big) + \big(\bar{f}(x) - f(x;\mathcal{D})\big).$$

Elevando ao quadrado:

$$(\mu - f)^2 = (\mu - \bar{f})^2 \;+\; 2\,(\mu - \bar{f})(\bar{f} - f) \;+\; (\bar{f} - f)^2.$$

Aplicando $\mathbb{E}_{\mathcal{D}}$, o termo cruzado se anula porque $(\mu - \bar{f})$ é constante em $\mathcal{D}$ e $\mathbb{E}_{\mathcal{D}}[\bar{f} - f] = \bar{f} - \bar{f} = 0$:

$$\mathbb{E}_{\mathcal{D}}\big[(\mu - \bar{f})(\bar{f} - f)\big] = (\mu - \bar{f})\,\underbrace{\mathbb{E}_{\mathcal{D}}[\bar{f} - f]}_{=0} = 0.$$

Restam dois termos:

$$\mathbb{E}_{\mathcal{D}}[(\mu(x) - f(x;\mathcal{D}))^2] = \underbrace{(\mu(x) - \bar{f}(x))^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(x;\mathcal{D}) - \bar{f}(x))^2]}_{\text{variância}}.$$

Resultado: três fontes de erro

Combinando os dois passos, o erro esperado em x se decompõe em três termos:

$$\mathbb{E}_{\mathcal{D},y}\left[(y - f(x;\mathcal{D}))^2\right] = \underbrace{\sigma^2}_{\text{ruído}} + \underbrace{(\mu(x) - \bar{f}(x))^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(x;\mathcal{D}) - \bar{f}(x))^2]}_{\text{variância}}.$$

- **Ruído** σ^2: componente irredutível, depende apenas do processo gerador.
- **Viés**^{2}: distância entre o modelo médio $\bar{f}$ e a função real μ.
- **Variância**: oscilação do modelo aprendido em torno de $\bar{f}$, induzida pela amostra de treino.

Esta decomposição aditiva vale exatamente para o custo quadrático, qualquer que seja o modelo f. Para outras funções de custo (0-1, entropia cruzada) ela não é tão limpa, mas a intuição de viés e variância permanece útil.

Resultado: três fontes de erro

Combinando os dois passos, o erro esperado em x se decompõe em três termos:

$$\mathbb{E}_{\mathcal{D},y}\left[(y - f(x;\mathcal{D}))^2\right] = \underbrace{\sigma^2}_{\text{ruído}} + \underbrace{(\mu(x) - \bar{f}(x))^2}_{\text{viés}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[(f(x;\mathcal{D}) - \bar{f}(x))^2]}_{\text{variância}}.$$

- **Ruído** σ^2: componente irredutível, depende apenas do processo gerador.
- **Viés**^{2}: distância entre o modelo médio $\bar{f}$ e a função real μ.
- **Variância**: oscilação do modelo aprendido em torno de $\bar{f}$, induzida pela amostra de treino.

Esta decomposição aditiva vale exatamente para o custo quadrático, qualquer que seja o modelo f. Para outras funções de custo (0-1, entropia cruzada) ela não é tão limpa, mas a intuição de viés e variância permanece útil.

Visão Geral

1. Motivação: overfitting na prática
2. Fontes de erro: ruído, viés e variância
3. **Trade-off viés-variância e double descent**
4. Regularização explícita e empírica
5. Regularização implícita

Tamanho do *dataset* vs. variância

- Mesmo modelo (capacidade fixa: 3 regiões lineares).
- Com 6 instâncias: ajustes muito variáveis entre *datasets*.
- Com 100 instâncias: ajustes próximos uns dos outros.
- Mais dados, menor variância.

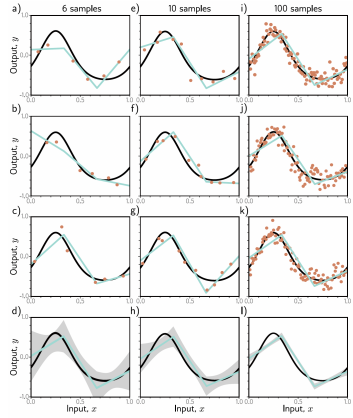
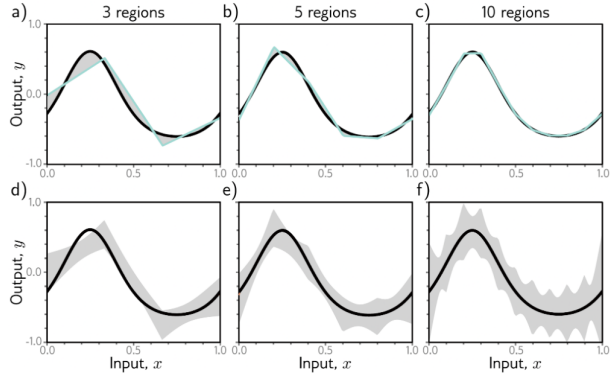


Figure 8.6 Reducing variance by increasing training data. a-c) The three-region model fitted to three different randomly sampled datasets of six points. The fitted model is quite different each time. d) We repeat this experiment many times and plot the mean model predictions (cyan line) and the variance of the model predictions (gray area shows two standard deviations). e–h) We do the same experiment, but this time with datasets of size ten. The variance of the predictions is reduced. i–l) We repeat this experiment with datasets of size 100. Now the fitted model is always similar, and the variance is small.

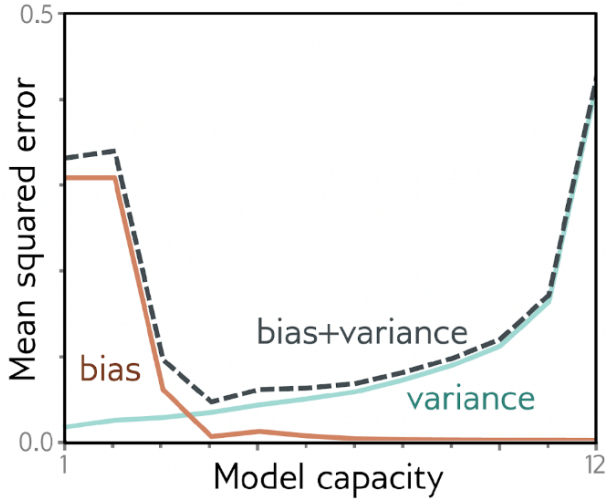
Capacidade do modelo vs. variância

- Mesmo *dataset*, modelos com mais regiões lineares.
- Mais capacidade, ajustes mais sensíveis à amostra de treino.
- Mais capacidade, maior variância.



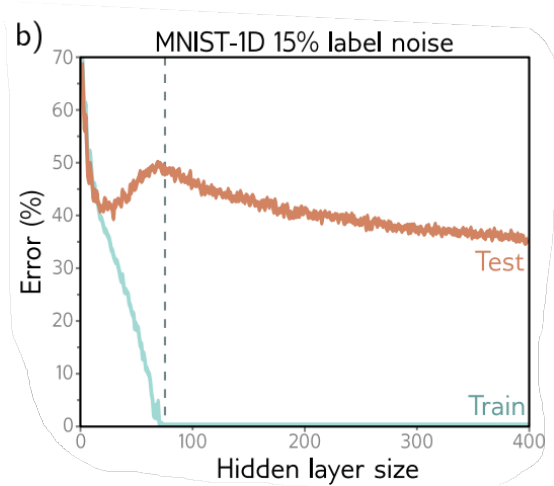
Trade-off viés-variância clássico

- Pouca capacidade: viés alto domina o erro.
- Muita capacidade: variância alta domina o erro.
- Existe um ponto intermediário que minimiza o erro total.
- Esta é a visão *clássica*, antes das redes neurais superparametrizadas.



Double descent em redes neurais[2]

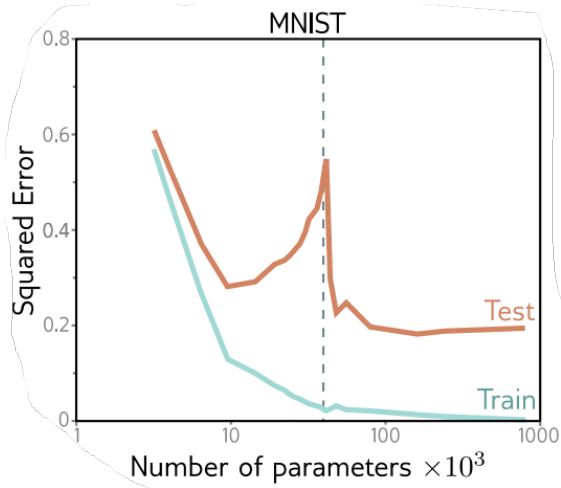
- Em redes neurais o erro de teste pode voltar a cair depois de passar pelo limiar de interpolação.
- Esse limiar ocorre quando o número de parâmetros se aproxima do número de instâncias de treino.
- Padrão observado em vários conjuntos: MNIST-1D.



[2] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". Em: PNAS 116.32 (2019), pp. 15849–15854.

Double descent em redes neurais[2]

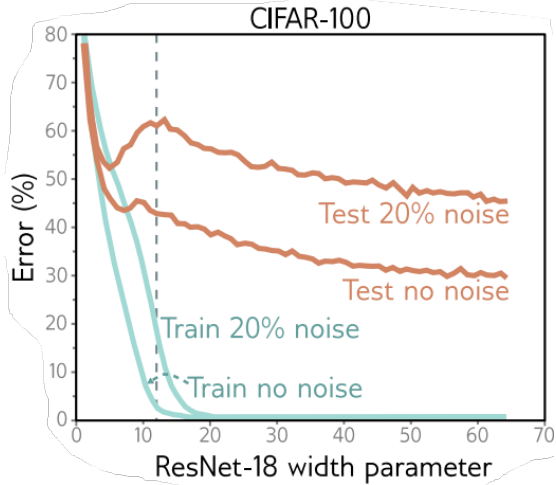
- Em redes neurais o erro de teste pode voltar a cair depois de passar pelo limiar de interpolação.
- Esse limiar ocorre quando o número de parâmetros se aproxima do número de instâncias de treino.
- Padrão observado em vários conjuntos: MNIST-1D, MNIST.



[2] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". Em: PNAS 116.32 (2019), pp. 15849–15854.

Double descent em redes neurais[2]

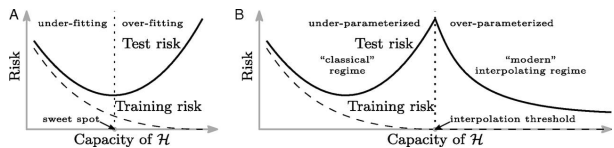
- Em redes neurais o erro de teste pode voltar a cair depois de passar pelo limiar de interpolação.
- Esse limiar ocorre quando o número de parâmetros se aproxima do número de instâncias de treino.
- Padrão observado em vários conjuntos: MNIST-1D, MNIST, CIFAR-100.



[2] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". Em: PNAS 116.32 (2019), pp. 15849–15854.

Por que ocorre? Hipóteses

- Talvez a inicialização favoreça funções mais suaves.
- Talvez o treinamento favoreça funções mais suaves.
- Ainda é um tema aberto, com bons palpites mas sem consenso.
- No regime moderno, a relação capacidade × erro de teste tem dois mínimos: o “sweet spot” clássico e o regime superparametrizado.



(Belkin et al., 2019)

Visão Geral

1. Motivação: overfitting na prática
2. Fontes de erro: ruído, viés e variância
3. Trade-off viés-variância e double descent
4. **Regularização explícita e empírica**
5. Regularização implícita

Tipos de regularização

- **Explícita**: termos extras na função de custo, L1 e L2.
- **Empírica**: técnicas que modificam o procedimento de treinamento, *early stopping*, *dropout*, *data augmentation*, *label smoothing*.
- **Implícita**: efeitos de regularização que surgem do próprio otimizador (descida de gradiente e SGD).
- Todas têm o mesmo objetivo: reduzir variância sem aumentar demais o viés.

Regularização L1 e L2

- Adicionamos um termo de penalidade à função de custo:

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \lambda\, \Omega(\boldsymbol{\theta}), \qquad \Omega_{L2} = \|\boldsymbol{\theta}\|_2^2, \qquad \Omega_{L1} = \|\boldsymbol{\theta}\|_1.$$

- Enviesamos a otimização em favor de parâmetros de baixa magnitude.
- Visão probabilística: equivale a adicionar um *prior* no critério de máxima verossimilhança.
  - L2 corresponde a um *prior* gaussiano, $\boldsymbol{\theta} \sim \mathcal{N}(0, \tau^2 I)$.
  - L1 corresponde a um *prior* laplaciano, que induz esparsidade.
- Como redes neurais atuam em problemas variados, o *prior* precisa ser genérico.

L2 e *weight decay*

Para L2, a atualização SGD se reescreve como:

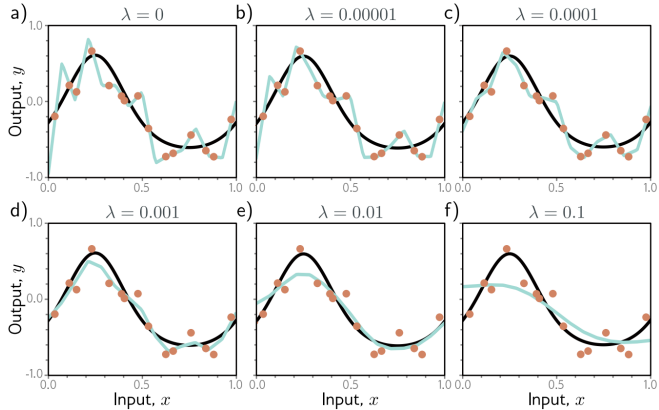
$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta\,\nabla_{\boldsymbol{\theta}}\left[\mathcal{L}(\boldsymbol{\theta}_t) + \lambda\|\boldsymbol{\theta}_t\|_2^2\right] \\ &= \boldsymbol{\theta}_t - \eta\,\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t) - 2\eta\lambda\,\boldsymbol{\theta}_t \\ &= \underbrace{(1-2\eta\lambda)\,\boldsymbol{\theta}_t}_{\text{decaimento}} - \eta\,\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t).\end{aligned}$$

- A cada passo, os pesos encolhem multiplicativamente por $(1 - 2\eta\lambda)$.
- Esse comportamento dá origem ao nome *weight decay*.

No PyTorch, o argumento `weight_decay` corresponde ao coeficiente já em frente de $\boldsymbol{\theta}$ no *update* (i.e., $\lambda_{\text{PT}} = 2\lambda$ desta derivação).

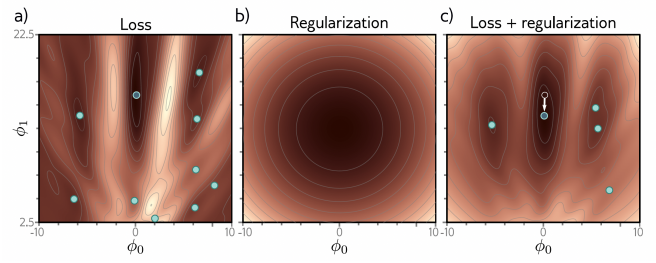
Efeito de aumentar λ

- λ pequeno: o ajuste se aproxima do caso sem regularização, modelo sensível ao ruído dos dados.
- λ grande: o ajuste se torna mais suave, modelo se afasta dos pontos individuais.
- Existe um λ intermediário que generaliza melhor.



L1 vs. L2 na paisagem de *loss*

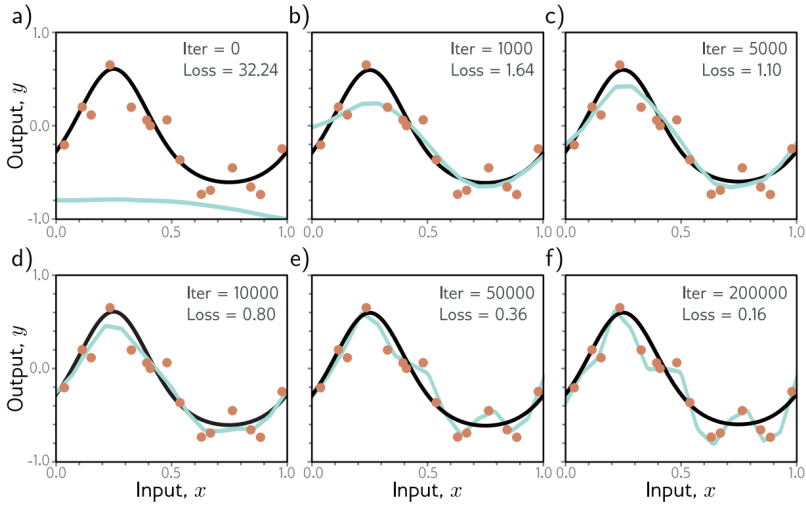
- Painel a): *loss* original com vários mínimos locais.
- Painel b): termo de regularização sozinho, com mínimo único na origem.
- Painel c): soma das duas paisagens, com poucos mínimos próximos da origem.
- L1 favorece componentes exatamente zero (os vértices do losango do L1 ficam sobre os eixos); L2 favorece magnitudes pequenas mas não zero.



Early stopping

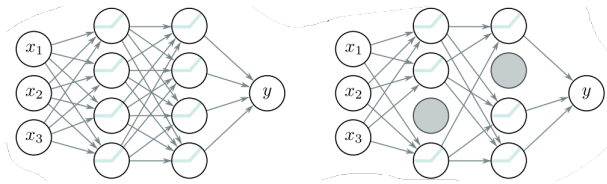
- Monitoramos o desempenho da rede em um conjunto de validação e interrompemos o treinamento antes da *loss* de validação voltar a subir.
- Tem efeito análogo a L2, pois impede que os pesos cresçam indiscriminadamente.
- Hiperparâmetro principal: *patience*, número de iterações sem melhoria que toleramos antes de parar.
- Bônus: também economiza tempo de computação.

Early stopping em ação



Dropout[3]

- Durante o treinamento, unidades de uma camada oculta são desligadas aleatoriamente.
- Equivale a multiplicar as ativações por uma máscara binária $m \sim \text{Bernoulli}(1 - p)$.
- A cada *forward*, uma sub-rede diferente é ativada, efeito similar a treinar um *ensemble* implícito.

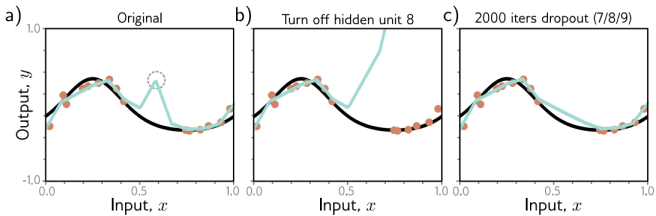


Esquerda: rede densa. Direita: rede com unidades desligadas.

[3] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". Em: JMLR 15.56 (2014), pp. 1929–1958.

Dropout: por que ajuda

- a) Sem *dropout* o modelo já interpola os dados de treino, mas com um pico isolado.
- b) Ao desligar uma unidade, esse pico vira um aumento brusco na *loss*.
- c) Continuando a desligar unidades aleatoriamente, o pico é gradualmente eliminado.



Dropout em inferência

- Em treino, cada unidade é desligada com probabilidade p, então a magnitude esperada das ativações é $(1 - p)$ vezes a magnitude original.
- Em inferência precisamos compensar esse fator. Duas opções:
  - **Rescaling determinístico**: multiplicar as ativações de teste por $(1 - p)$, ou equivalentemente dividi-las em treino por $(1 - p)$ (*inverted dropout*, padrão em PyTorch e Keras).
  - **Monte Carlo Dropout**[4]: rodar o *forward* múltiplas vezes com máscaras diferentes e combinar as saídas, análogo a um *ensemble* explícito; também fornece uma estimativa de incerteza.

[4] Yarin Gal e Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning". Em: ICML. 2016, pp. 1050–1059.

Data augmentation

- Geramos novas instâncias aplicando transformações nos dados originais que preservam o rótulo.
- Em imagens: rotações, *flip*, mudança de cor, equalização, *random crop*, ruído.
- Codificamos invariâncias desejadas no conjunto de treino, sem alterar a arquitetura.
- Efeito: aumenta a diversidade efetiva do *dataset*, reduzindo a variância do modelo.

Original (acima) e variantes geradas (abaixo).

Label smoothing[5]

- *Data augmentation* transforma as entradas; podemos também regularizar transformando os alvos.
- Com alvo *one-hot*, a entropia cruzada só é minimizada quando a probabilidade da classe correta tende a 1, o que exige *logits* cada vez maiores: o modelo é empurrado para o *overconfidence*.
- *Label smoothing* redistribui uma fração α da massa de probabilidade uniformemente entre as C classes:

$$y_{ls}^{(i)} = (1 - \alpha)\, y_{oh}^{(i)} + \frac{\alpha}{C}.$$

- Casos limite: $\alpha = 0$ mantém o alvo *one-hot* original; $\alpha = 1$ produz um alvo uniforme, sem informação de classe.
- Na prática, valores pequenos como $\alpha = 0{,}1$ são os mais comuns.

[5] Christian Szegedy et al. "Rethinking the Inception architecture for computer vision". Em: CVPR. 2016, pp. 2818–2826.

Label smoothing: por que ajuda[6]

- Exemplo com $C = 4$, classe correta 3 e $\alpha = 0{,}1$:

$$\boldsymbol{y}_{oh} = (0,\ 0,\ 1,\ 0) \quad \longrightarrow \quad \boldsymbol{y}_{ls} = (0{,}025,\ 0{,}025,\ 0{,}925,\ 0{,}025).$$

- O alvo suavizado é atingível com *logits* finitos: a diferença entre o *logit* da classe correta e os demais para de crescer, contendo a magnitude dos pesos (efeito análogo ao de L2 na camada de saída).
- Modelos treinados com *label smoothing* tendem a ficar melhor calibrados: a confiança predita se aproxima da acurácia real.
- Custo: as representações da penúltima camada perdem informação sobre a semelhança entre classes, o que prejudica, por exemplo, a destilação de conhecimento para um modelo menor.

[6] Rafael Müller, Simon Kornblith e Geoffrey Hinton. "When does label smoothing help?" Em: NeurIPS. vol. 32. 2019.

Visão Geral

1. Motivação: overfitting na prática
2. Fontes de erro: ruído, viés e variância
3. Trade-off viés-variância e double descent
4. Regularização explícita e empírica
5. **Regularização implícita**

Regularização implícita

- Fato peculiar: a descida de gradiente encontra soluções que generalizam, mesmo em redes muito superparametrizadas.
- Tanto o *batch GD* quanto o SGD não fazem uma trajetória direta para o mínimo local da função de custo: existe uma preferência por algumas soluções em detrimento de outras.
- Essas preferências, que não são impostas explicitamente na função de custo, recebem o nome de regularização implícita.

Implicit Gradient Regularization[7]

- Versão contínua da descida de gradiente: $\dot{\boldsymbol{\theta}}(t) = -\nabla \mathcal{L}(\boldsymbol{\theta}(t))$.
- A versão discreta (passos de tamanho η) aproxima a versão contínua de uma função de custo modificada:

$$\tilde{\mathcal{L}}_{GD}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{\eta}{4} \left\| \nabla \mathcal{L}(\boldsymbol{\theta}) \right\|^2 .$$

- Consequência: a solução é repelida de regiões em que a norma do gradiente é grande (alta declividade da *loss*).
- GD prefere bacias de mínimo *rasas* e largas, que tendem a generalizar melhor.

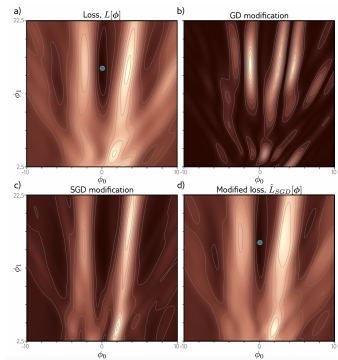
[7] David G. T. Barrett e Benoit Dherin. "Implicit gradient regularization". Em: arXiv preprint arXiv:2009.11162 (2020).

SGD favorece regiões de gradiente estável[8]

Para SGD com *batches* de tamanho B:

$$\tilde{\mathcal{L}}_{SGD}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{\eta}{4}\|\nabla\mathcal{L}\|^2 + \frac{\eta}{4B}\sum_{b=1}^{B}\|\nabla\mathcal{L}_b - \nabla\mathcal{L}\|^2.$$

- O segundo termo extra corresponde à variância do gradiente entre *batches*.
- SGD favorece soluções em que o gradiente é estável ao trocar de *batch*.
- Os mínimos da função de custo original não se movem, mas as trajetórias e o mínimo escolhido podem mudar.



[8] Samuel L. Smith et al. "On the origin of implicit regularization in stochastic gradient descent". Em: arXiv preprint arXiv:2101.12176 (2021).

Leituras Recomendadas

- Capítulo 9 de Simon J. D. Prince. Understanding Deep Learning. MIT Press, 2023, livro disponível em *udlbook.github.io/udlbook*.
- Capítulo 7 de Ian Goodfellow, Yoshua Bengio e Aaron Courville. Deep Learning. MIT Press, 2016. URL: https://www.deeplearningbook.org/, tratamento clássico de regularização.
- Seção 5.5 de Bishop, C. M. *Pattern Recognition and Machine Learning*, Springer, 2006: conexão Bayesiana entre L2/L1 e *priors* gaussiano/laplaciano.
- MNIST 1D: (Greydanus, 2020).
- Double descent: (Belkin et al., 2019).
- Dropout: (Srivastava et al., 2014); Monte Carlo Dropout: (Gal e Ghahramani, 2016).
- Regularização implícita: (Barrett e Dherin, 2020); (Smith et al., 2021).

Referências I

[1] David G. T. Barrett e Benoit Dherin. “Implicit gradient regularization”. Em: arXiv preprint arXiv:2009.11162 (2020).

[2] Mikhail Belkin et al. “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. Em: PNAS 116.32 (2019), pp. 15849–15854.

[3] Yarin Gal e Zoubin Ghahramani. “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning”. Em: ICML. 2016, pp. 1050–1059.

[4] Ian Goodfellow, Yoshua Bengio e Aaron Courville. Deep Learning. MIT Press, 2016. URL: https://www.deeplearningbook.org/.

[5] Sam Greydanus. “Scaling down deep learning”. Em: arXiv preprint arXiv:2011.14439 (2020).

[6] Rafael Müller, Simon Kornblith e Geoffrey Hinton. “When does label smoothing help?” Em: NeurIPS. Vol. 32. 2019.

[7] Simon J. D. Prince. Understanding Deep Learning. MIT Press, 2023.

[8] Samuel L. Smith et al. “On the origin of implicit regularization in stochastic gradient descent”. Em: arXiv preprint arXiv:2101.12176 (2021).

[9] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. Em: JMLR 15.56 (2014), pp. 1929–1958.

[10] Christian Szegedy et al. “Rethinking the Inception architecture for computer vision”. Em: CVPR. 2016, pp. 2818–2826.