AULA 12
AJUSTE DOS HIPERPARÂMETROS E REGULARIZAÇÃO

1. Objetivos

- Apresentar os problemas de sobre-ajuste ("overfitting") e sub-ajuste ("underfitting") dos dados.
- Apresentar formas de ajustar os hiperparâmetros para obter o melhor desempenho da RNA.
- Apresentar os métodos de regularização para eliminar problemas de sobre-ajuste e sub-ajuste.

2. Sobre-ajuste e sub-ajuste dos dados

- Um problema fundamental em aprendizado de máquina é o compromisso entre **otimização** e **generalização**.
  - **Otimização** refere-se ao processo de ajustar o modelo para obter o melhor desempenho possível com os dados de treinamento.
  - **Generalização** refere-se a quanto que o modelo treinado apresenta um bom desempenho em dados que nunca foram vistos.
- O objetivo do treinamento de uma RNA é obter uma boa generalização, pois isso permite fazer boas previsões com novos dados nunca vistos $\Rightarrow$ contudo, o problema é que não conseguimos controlar a generalização, pois somente somos capazes de ajustar a RNA com os dados de treinamento.
- Existe uma terminologia utilizada em aprendizado de máquina para descrever a capacidade de um modelo de aprender e generalizar o seu aprendizado para novos dados: **sobre-ajuste ("overfitting")** e **sub-ajuste ("underfitting")** dos dados.
- **"Overfitting"**, também conhecido como problema de **alta variância**, representa o quanto que um modelo é capaz de representar os dados utilizados no treinamento.
  - "Overfitting" ocorre quando uma RNA aprende os detalhes e ruídos presentes nos dados de entrada a ponto de impactar negativamente no seu desempenho para novos dados.
  - Uma solução com "overfitting" significa que flutuações presentes nos dados de treinamento, causadas por ruídos e outros problemas, são capturadas e aprendidas pela RNA como se fizessem parte da solução desejada.
  - O problema é que essas flutuações não estão presentes nos novos dados e, portanto, apresentam um impacto negativo na habilidade da RNA generalizar.

- **"Underfitting"**, também conhecido como problema de **alto viés**, se refere à falta de capacidade de um modelo representar tanto os dados de treinamento quanto de generalizar novos dados.

  - Uma RNA que apresenta "underfitting" não é adequada, pois apresenta baixo desempenho tanto para os dados de treinamento como para os dados de teste.

  - O problema de "underfitting" é fácil de ser detectado pelo baixo desempenho da RNA na métrica adotada para avaliar a solução calculada com os dados de treinamento.

  - A solução do problema de "underfitting" é simples, pois em princípio basta melhorar a RNA.

  - O conceito de "underfitting" é importante por representar um contraste ao problema de "overfitting".

- O **ajuste ideal** de um problema de aprendizado de máquina consiste em obter um modelo cuja solução fica entre "underfitting" e "overfitting" $\Rightarrow$ isso é difícil de conseguir na prática.

- Na Figura 1 é apresentado um exemplo de um problema de classificação binária. Na Figura 1a é apresentada uma solução com problema de "underfitting"; na Figura 1b, uma solução com problema de "overfitting"; na Figura 1c uma solução com problemas tanto de "underfitting" como de "overfitting"; e na Figura 1d uma solução com ajuste ideal.

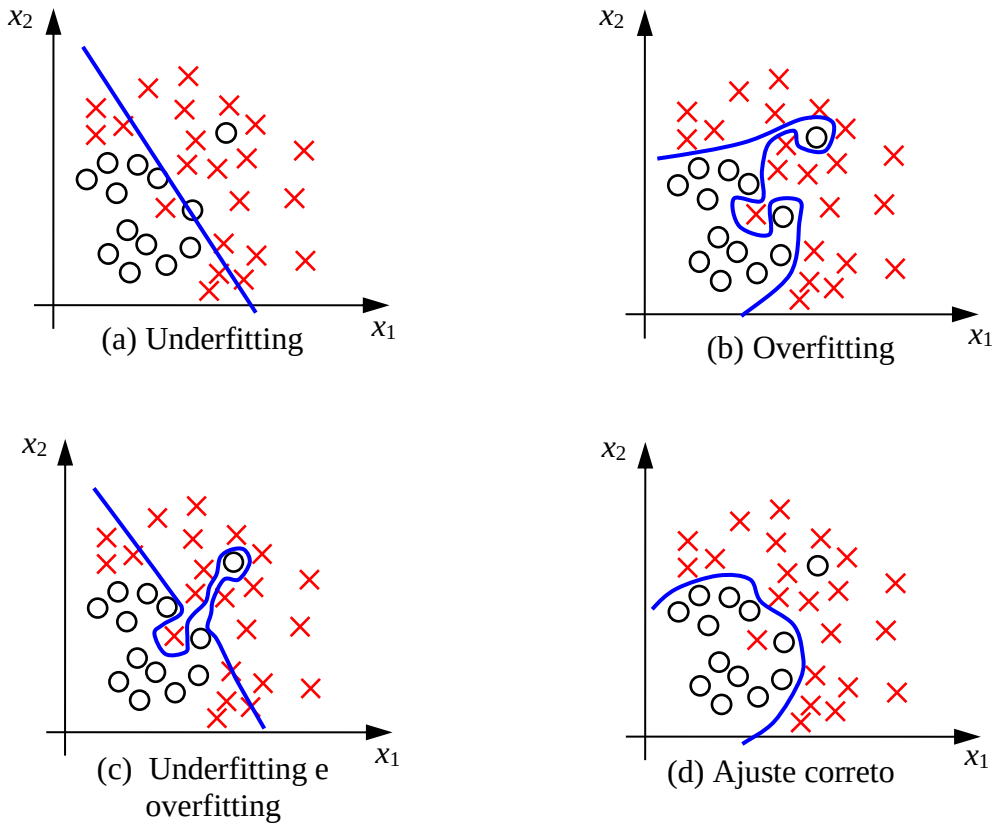

Figura 1. Exemplo de um problema de classificação binária. (a) Solução com problema de "underfitting"; (b) solução com problema de "overfitting"; (c) solução com problemas tanto de "underfitting" como de "overfitting"; e (d) solução com ajuste ideal. (Adaptado de Andrew Ng, deeplearning.ai)

- Na Tabela 1 são apresentados alguns exemplos para valores de métricas calculadas nos conjuntos de treinamento e de validação (ou teste) e o diagnóstico do problema, para um caso de classificação binária. A métrica utilizada é simplesmente a porcentagem dos erros de classificação.

Tabela 1. Exemplos de valores de métrica para um problema de classificação binária e o diagnóstico do problema.

	Porcentagem de erro de classificação			
Dados de treinamento	1%	15%	15%	0,5%
Dados de validação ou de teste	11%	16%	30%	1%
Diagnóstico	"Overfitting"	"Underfitting"	"Overfitting" e "underfitting"	Ajuste adequado

- Obviamente que avaliar uma RNA a partir do valor obtido da sua métrica de desempenho é algo subjetivo e depende do que se deseja da RNA.

- Porém, dada a potencialidade das RNAs, o desempenho que se deseja obter depende principalmente do esforço, tempo e dinheiro que se está disposto a gastar.

- A obtenção de uma RNA que apresenta um ajuste ideal não é fácil de conseguir e algumas regras devem ser seguidas:
  - Em princípio para **eliminar "underfitting" basta aumentar o tamanho da RNA**;
  - Em princípio para **eliminar "overfitting" podemos diminuir o tamanho da RNA, ou obter mais dados de treinamento;**
  - Em princípio deve-se **inicialmente obter uma RNA que realiza "overfitting"** dos dados de treinamento para depois eliminar esse problema;
  - Existem técnicas para obter uma RNA com um ajuste ideal $\Rightarrow$ **processo de regularização da RNA**.

- **Problemas de "overfitting" também podem ser observados pela curva de aprendizado da RNA**, analisando a variação do valor da função de custo, ou da métrica, ao longo do treinamento.

- Na Figura 2 são apresentados os comportamentos da função de custo em função do número de épocas de treinamento para um caso onde não ocorre "overfitting" (Figura 2a) e outro caso onde ocorre "overfitting" (Figura 2b).
  - Nota-se que no caso onde não ocorre "overfitting", a função de custo dos dados de validação tem o mesmo comportamento da função de custo para os dados de treinamento, ou seja, ambos decrescem em função do número de épocas.

- No caso em que ocorre "overfitting" a função de custo dos dados de validação no início diminui e após um determinado momento do treinamento começa a aumentar, indicando que os parâmetros da RNA estão aprendendo os detalhes dos dados de treinamento e não generalizando uma solução.

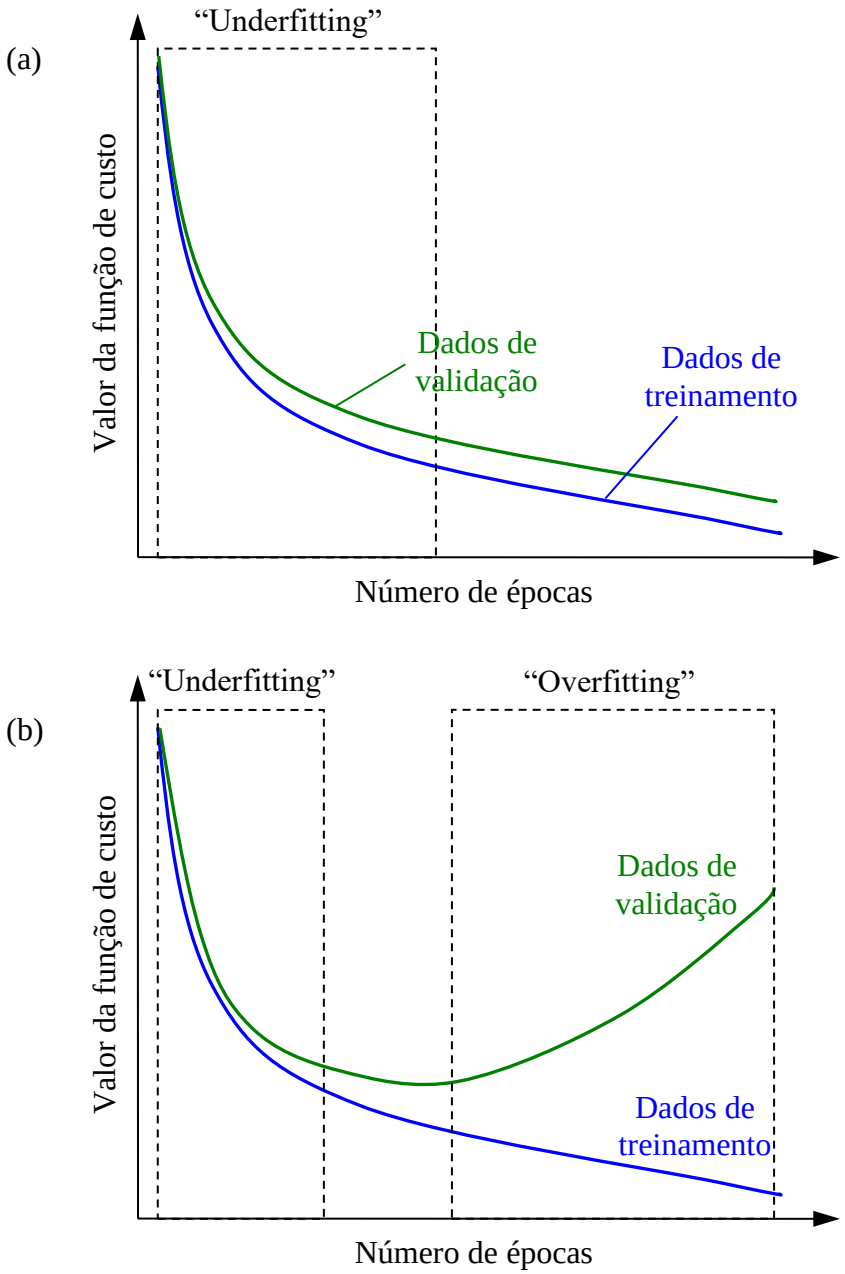

Figura 2. Exemplo da variação da função de custo em função do número de épocas de treinamento. (a) Solução sem problema de "overfitting"; (b) solução com problema de "overfitting".

3. Ajuste dos hiperparâmetros e regularização da RNA

- Para se obter uma RNA com um ajuste ótimo temos que lidar com problemas de "underfitting" e "overfitting" utilizando algum método.

- O fluxograma da Figura 3 apresenta uma "receita" básica para lidar com os problemas de "overfitting" e "underfitting".

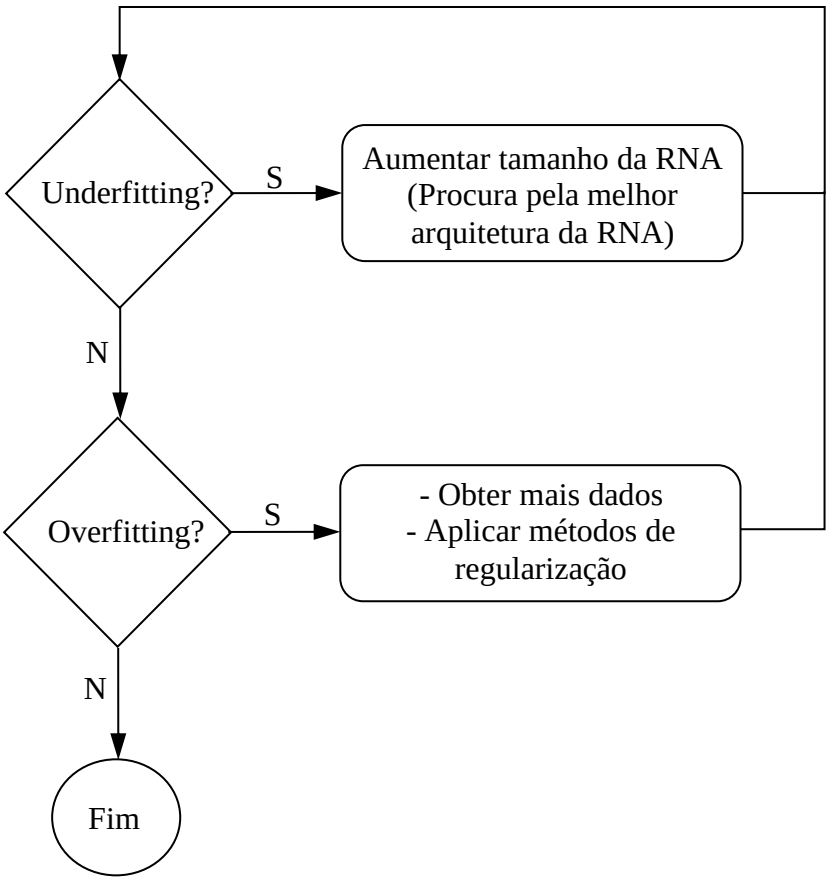

Figura 3. Fluxograma do processo de desenvolvimento de uma RNA para obter um ajuste ótimo.

- No processo de obtenção de uma RNA com um ajuste adequado é necessário ajustar os hiperparâmetros da RNA em função das ocorrências de "underfitting" e "overfitting". O processo segue em linhas gerais as seguintes etapas:

  - Obter uma primeira RNA simples que apresenta desempenho um pouco melhor do que o resultado base para verificar se os dados são adequados;
  - Aumentar o tamanho da RNA (aumentar número de camadas e número de neurônios) de forma a realizar "overfitting" nos dados de treinamento;
  - Ajustar os parâmetros dessa RNA mais complexa e aplicar técnicas para eliminar "overfitting" (regularização), até alcançar a métrica desejada.

  Lembre-se de que um resultado base é qualquer resultado melhor do que um resultado aleatório. Por exemplo, para um problema de classificação binária obter uma taxa de acerto maior do que 50% seria um resultado base, pois se escolhermos aleatoriamente uma das classes temos 50% de chance de acertar.

- A regularização de uma RNA significa eliminar problemas de "overfitting" até obter um ajuste adequado.

- Existem várias formas de regularização de um RNA:
  - Regularização L1;
  - Regularização L2;
  - Regularização por "dropout";
  - Geração artificial de dados ("data augmentation");
  - Parada prematura do treinamento ("early stopping").

- A geração artificial de dados e a parada prematura são artifícios para contornar problemas de "overfitting", que devem utilizados somente quando se tem um pequeno conjunto de dados e com cautela, pois a RNA resultante irá apresentar um desempenho não muito satisfatório.

4. Regularização L1 e L2

- Os métodos de Regularização L1 e L2 consistem em alterar a função de custo adicionando um termo de regularização, que é função dos pesos das ligações da RNA.

Regularização L1

- No caso da Regularização L1 o termo adicionado à função de custo consiste na somatória dos valores absolutos de todos os pesos das ligações da RNA.

- A função de custo com regularização L1 é a seguinte:

$$J_{L1}(\mathbf{W},\mathbf{B}) = \frac{1}{m}\sum_{i=1}^{m} E(\hat{\mathbf{y}}^{(i)},\mathbf{y}^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{N_W}\left|w_j\right| \quad (1)$$

Termo de regularização

onde:

$J_{L1}(\mathbf{W},\mathbf{B})$ = função de custo com regularização L1;

λ = parâmetro de regularização;

N_W = número total de pesos das ligações da RNA em todas as camadas;

w_j = j-ésimo peso de ligação;

$\mathbf{W}$ = vetor com todos os pesos das ligações da RNA;

$\mathbf{B}$ = vetor com todos os vieses da RNA;

m = número de exemplos de treinamento;

$E(\hat{\mathbf{y}}^{(i)},\mathbf{y}^{(i)})$ = função de erro.

Regularização L2

- No caso da Regularização L2 o termo adicionado à função de custo é a somatória de todos os pesos das ligações da RNA ao quadrado.

- A função de custo com regularização L2 é a seguinte:

$$J_{L2}(\mathbf{W},\mathbf{B}) = \frac{1}{m}\sum_{i=1}^{m} E(\hat{\mathbf{y}}^{(i)},\mathbf{y}^{(i)}) + \frac{\lambda}{2m}\sum_{j=1}^{N_W} w_j^2 \qquad (2)$$

Termo de regularização

onde $J_{L2}(\mathbf{W},\mathbf{B})$ é a função de custo com Regularização L2 e os outros parâmetros são os mesmos da equação (1).

- Nota-se que λ presente nas equações (1) e (2) é um hiperparâmetro da RNA que deve ser escolhido.

- Ressalta-se que na escolha do valor de λ deve-se garantir que λ/m seja menor do que 1.

- Nota-se que não se inclui os vieses da RNA no termo de regularização.

- Note que nas funções de custo das equações (1) e (2), quanto menor o valor dos pesos das ligações menor será o valor da função de custo.

Matemática da Regularização L1 e L2

- O treinamento de uma RNA consiste em minimizar a função de custo, assim, ao incluir o termo de regularização na função de custo, o processo de otimização tentará diminuir os pesos das ligações ao máximo para que a função de custo seja a menor possível $\Rightarrow$ esse fato pode ser melhor entendido nas equações que seguem.

- Considerando a Regularização L2, a função de custo com regularização pode ser também escrita como sendo:

$$J_{L2}(\mathbf{W},\mathbf{B}) = J(\mathbf{W},\mathbf{B}) + \frac{\lambda}{2m}\sum_{j=1}^{N} w_j^2 \qquad (3)$$

onde $J(\mathbf{W},\mathbf{B})$ é função de custo sem o termo de Regularização L2, que já vimos anteriormente.

- As derivadas parciais da função de custo com Regularização L2 em relação ao *j*-ésimo peso da ligação da RNA é dada por:

$$\frac{\partial J_{L2}(\mathbf{W},\mathbf{B})}{\partial w_j} = \frac{\partial J(\mathbf{W},\mathbf{B})}{\partial w_j} + \frac{\lambda}{m} w_j \qquad (4)$$

onde $\frac{\partial J(\mathbf{W},\mathbf{B})}{\partial w_j}$ é a derivada parcial da função de custo em relação ao *j*-ésimo peso sem regularização.

- Com o gradiente dado pela equação (3), a atualização do *j*-ésimo peso de ligação será realizada por:

$$w_j = w_j - \alpha\left[\frac{\partial J(\mathbf{W},\mathbf{B})}{\partial w_j} + \frac{\lambda}{m} w_j\right], \text{ para } j = 1,...,N \tag{5}$$

- Rearranjando a equação (5), obtemos:

$$w_j = \left(1 - \frac{\lambda}{m}\right) w_j - \alpha \frac{\partial J(\mathbf{W},\mathbf{B})}{\partial w_j}, \text{ para } j = 1,...,N \tag{6}$$

- Como o termo $(1 - \lambda/m)$ da equação (6) é menor do que um, pode-se concluir que a atualização dos pesos age no sentido de diminuir o seu valor absoluto, independentemente do valor do gradiente da função de custo sem considerar a regularização.

- **Porque as Regularizações L1 e L2 previnem "overfitting"?**
  - As regularizações L1 e L2 previnem "overfitting" porque tendem a diminuir os valores absolutos dos parâmetros da RNA;
  - Parâmetros de menor valor fazem com que os níveis de ativação dos neurônios fiquem também menores;
  - Níveis de ativação menores fazem com que seja necessário um maior número de neurônios contribuindo para calcular as saídas da RNA, ou seja, os neurônios estarão contribuindo de uma forma mais balanceada no cálculo das saídas da RNA;
  - Os neurônios passam a funcionar em conjunto, evitando com que se especializem em calcular valores específicos.

- Além disso, a regularização facilita o treinamento da RNA, pois parâmetros de menor valor fazem com que os estados dos neurônios fiquem mais próximos de zero, diminuindo problemas de saturação e consequentemente problemas de "vanishing" e "exploding gradients".

- No Keras para incluir regularização deve-se antes importar a classe dos regularizadores.

- A regularização no Keras é incluída quando se adiciona as camadas da RNA. O código a seguir mostra um exemplo de como incluir regularização L2 em uma camada densa usando o Keras.

```
# Importa classe dos regularizadores
from tensorflow.keras import regularizers

# Adiciona regularização L2 em uma camada densa
model.add(Dense(64, activation='relu',
          kernel_regularizer=regularizers.l2(0.01)))
```

  - Nesse exemplo o parâmetro λ é igual a 0,01. O valor padrão do parâmetro λ é zero.
  - No Keras a regularização L1 e L2 são incluídas para cada camada e para cada camada pode-se usar um parâmetro λ diferente.

- Regularização L1 é incluída de forma similar à Regularização L2. O código a seguir mostra um exemplo de como incluir regularização L1 em uma camada densa no Keras.

```
# Importa classe dos regularizadores
from tensorflow.keras import regularizers

# Adiciona regularização L1 em uma camada densa
rna.add(Dense(64, activation='relu',
        kernel_regularizer=regularizers.l1(0.02)))
```

  - Nesse exemplo o parâmetro λ é igual a 0,02. O valor padrão do parâmetro λ é zero.

- Ressalta-se que a Regularização L2 é mais eficiente e muito mais utilizada do que a Regularização L1.

5. "Dropout"

- "Dropout" é a forma de regularização mais efetiva e a mais usada para treinar RNAs deep learning.

- Aplicar "dropout" em uma camada da RNA consiste em aleatoriamente zerar as saídas de alguns neurônios da camada durante o treinamento.

- Por exemplo, os níveis de ativação dos neurônios de uma dada camada da RNA são iguais a [0,2; 0,5; 1,3; 0,8; 1,1], para uma determinada entrada. Após aplicar "dropout" nessa camada, os níveis de ativação de alguns neurônios serão iguais a zero, por exemplo, [0; 0,5; 1,3; 0; 1,1].

- A taxa de "dropout" corresponde à fração de neurônios da camada cujos níveis de ativação são zerados durante o treinamento. Essa taxa é usualmente escolhida normalmente com sendo um valor entre 0,2 e 0,5. No exemplo anterior, a taxa de "dropout" foi de 0,4.

- Durante o treinamento em cada mini-batelada de cada época de treinamento, são escolhidos aleatoriamente os neurônios cujas saídas são zeradas, de forma que os neurônios "zerados" durante cada iteração do processo de treinamento variam.

- Na Figura 4 é mostrada uma RNA com "dropout". Os neurônios marcados com o "×" são os neurônios cujas saídas são zeradas naquele instante. A taxa de "dropout" de cada camada está mostrada na figura.

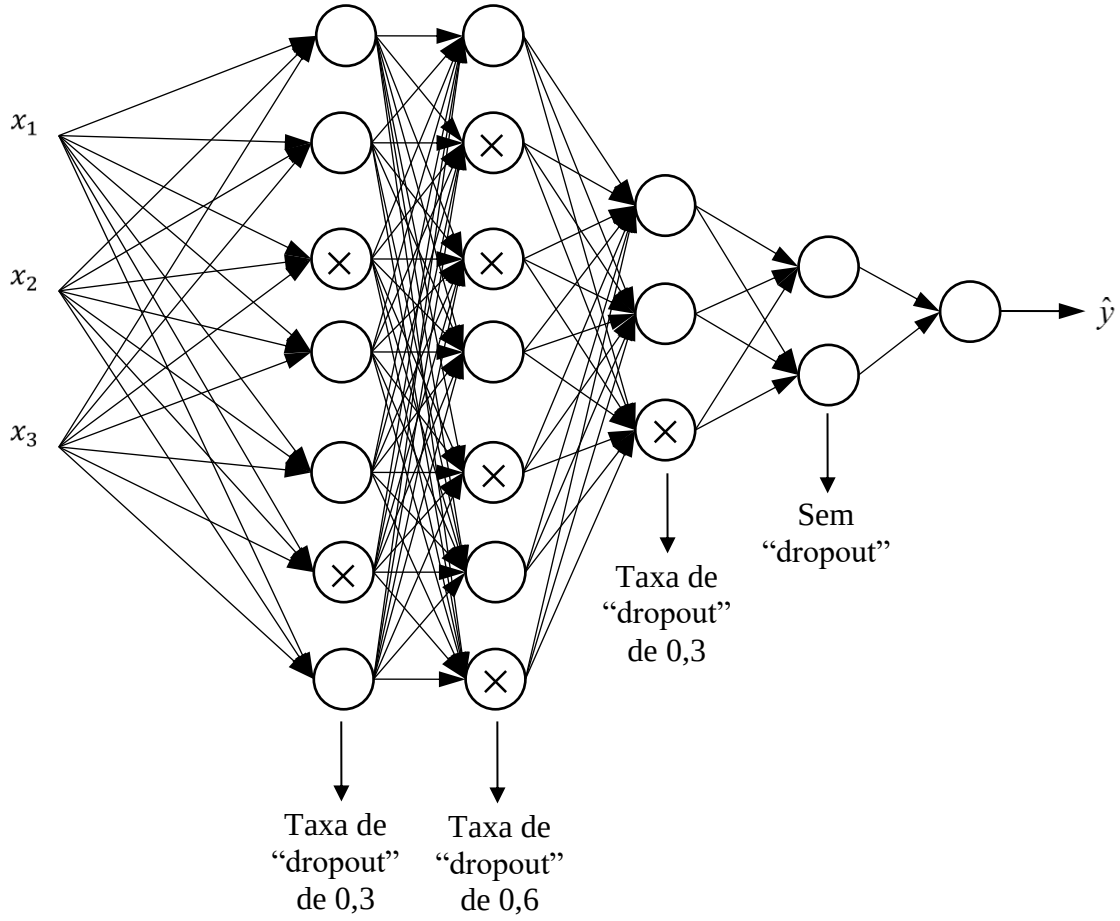

Figura 4. Treinamento de uma RNA com "dropout" (adaptado de Andrew Ng, deeplearning.ai). Neurônios com saídas zeradas estão marcados com "×".

- **Porque "dropout" funciona?**
  - Uma explicação intuitiva da razão do "dropout" funcionar para diminuir "overfitting" é que ao zerar aleatoriamente as saídas de alguns neurônios das camadas durante o treinamento, é como se a RNA ficasse menor e uma RNA menor tem mais tendência de apresentar problemas de "underfitting" e menos de apresentar problemas de "overfitting".
  - Outra explicação seria o fato de que o treinamento com "dropout" é distribuído mais uniformemente entre todos os neurônios da RNA, evitando que grupos de neurônios se especializem em dados de treinamento específicos.

- Vídeo explicativo do processo de "dropout" ⇒ aplicação de "dropout" na camada intermediária.

- No Keras "dropout" é introduzido na forma de uma "camada" de "dropout", que é incluída quando se adiciona as camadas da RNA. O código a seguir mostra um exemplo de como incluir "droput" em uma camada densa.

```
# Importa camadas densas e dropout
from tensorflow.keras.layers import Dense, Dropout

# Adição de uma camada densa com dropout
rna.add(Dense(64, activation='relu'))
rna.add(Dropout(0.2))
```

- "Nesse exemplo, "dropout" com uma taxa de 0,2 é aplicado na camada densa de 64 neurônios que foi adicionada antes da camada de "dropout".

➢ Adicionalmente, o trabalho original do "dropout" recomenda impor uma restrição nos pesos de uma camada tipo densa onde se aplica "dropout", para garantir que a norma dos pesos das ligações da camada não ultrapasse o valor de 3. Isso é realizado definindo o argumento `kernel_constraint` quando se adiciona a camada.

➢ O código a seguir apresenta um exemplo de como incluir a limitação da norma dos pesos de uma camada onde é aplicado "dropout".

```
# Importa camadas densas e dropout, e classe max_norm de
constraints
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.constraints import max_norm

# Adição de uma camada densa com dropout e limitação da norma dos
pesos
rna.add(Dense(64, activation='relu',
        kernel_constraint=max_norm(3)))
rna.add(Dropout(0.2))
```

➢ Outro detalhe que o trabalho original do "dropout" recomenda usar uma taxa de aprendizado de cerca de 10 maior do que se usaria sem "dropout".

➢ **Importante ⇒ por razões óbvias não se usa "dropout" na camada de saída.**

6. Parada prematura

➢ Como visto pelo gráfico da Figura 2b, quando ocorre "overfitting" o valor da função de custo para os dados de validação diminui no início e após um determinado momento do treinamento começa a aumentar, indicando que os parâmetros da RNA estão aprendendo os detalhes dos dados de treinamento e não generalizando para uma solução geral.

➢ A idéia por trás da parada prematura para evitar "overfitting" é parar o treinamento no momento quando o valor da função de custo dos dados de validação começar a aumentar.

➢ Na Figura 5 é apresentado o histórico do treinamento de uma RNA onde ocorre "overfitting". Para evitar "overfitting" o treinamento é interrompido de forma prematura, conforme indicado na Figura 5.

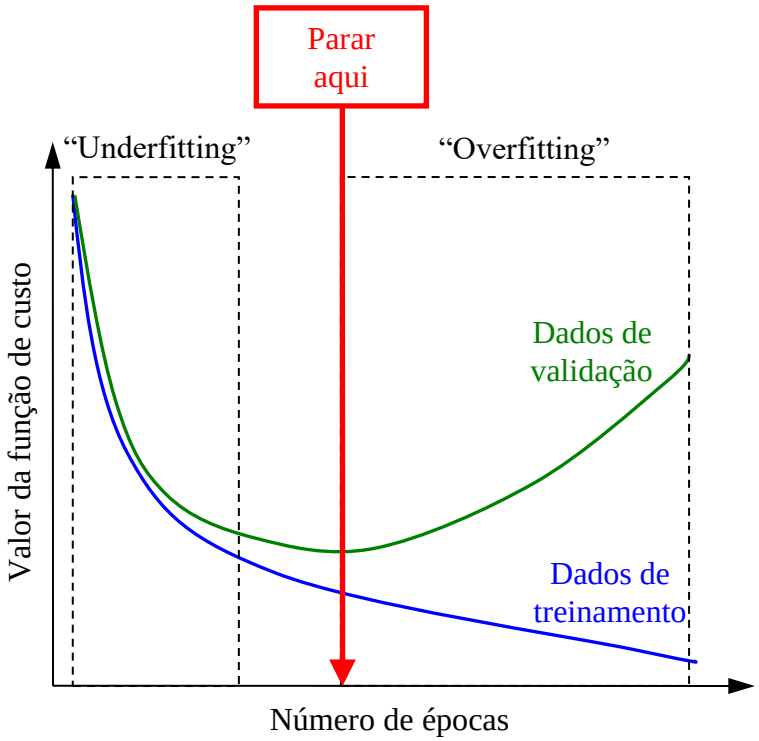

Figura 5. Exemplo de utilização de parada prematura para evitar problema de "overfitting".

- Ressalta-se que parar o treinamento antes da RNA começar a apresentar problemas de "overfitting" não é uma solução muito boa, pois o desempenho da RNA resultante não será muito satisfatório.

- O ideal é utilizar outras formas de regularização, tais como, regularização L1 e L2 e "dropout".

7. "Data augmentation"

- Problema de "overfitting" é causado por um conjunto pequeno de dados de treinamento, o que torna impossível para uma RNA generalizar seu treinamento para novos dados.

- Dado um conjunto infinito de dados a RNA seria exposta a todas as possíveis variações relacionadas ao problema e, assim, nunca teria problema de "overfitting".

- Na prática, problema de "overfitting" está associado à relação entre o número de parâmetros da RNA e o número de exemplos de treinamento $\Rightarrow$ quanto maior o número de camadas e de neurônios de uma RNA, para um dado conjunto de treinamento, maior será o problema de "overfitting".

- Na prática, uma forma muito eficiente de eliminar problemas de "overfitting" em uma RNA é usar um maior número de dados de treinamento.

- Uma RNA que apresenta problemas de "overfitting" para um determinado número de exemplos de treinamento pode apresentar problemas de "underfitting" se aumentarmos o número de exemplos de treinamento.

- Assim, quando é possível e fácil de obter mais exemplos de treinamento, essa é a melhor forma de eliminar problemas de "overfitting".

- Contudo, obter mais exemplos para treinar uma RNA, na maioria dos casos pode ser extremamente difícil e muito trabalhoso.

- Uma alternativa para obter mais exemplos de treinamento é gerar novos exemplos usando os exemplos já existentes.

- A geração de exemplos de forma artificial usando como base os exemplos existentes é chamada de "data augmentation".

- "Data augmentation" é utilizado em problemas que envolvem imagens ⇒ novas imagens são geradas aplicando transformações aleatórias nas imagens existentes. Essas transformações podem ser as seguintes:
  - Translação;
  - Rotação;
  - Distorção;
  - Ampliação;
  - Redução;
  - Redução.

- Apesar de ser uma forma simples e fácil de gerar novos dados, deve-se ter cuidado para usar dados gerados artificialmente, pois esses dados de fato não acrescentam muita informação nova porque possuem as mesmas características básicas dos dados já existentes.

- O Keras fornece uma classe de funções para realizar "data augmentation" ⇒ ImageDataGenerator. Para mais detalhes sobre esse assunto ver: François Chollet, Deep Learning with Python, página138, ou a documentação do Keras.

- Na Figura 6 é apresentado dois exemplos de geração de novas imagens usando imagens já existentes.

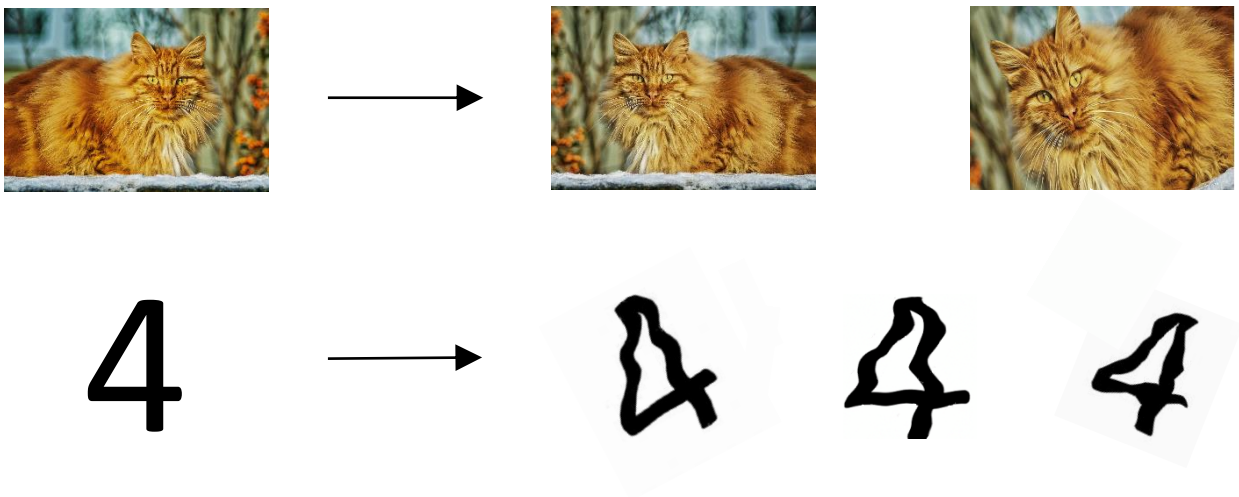

Figura 6. Geração artificial de dados. (a) A imagem do gato original é invertida e rotacionada obtendo-se dois novos exemplos. (b) A imagem original do dígito 4 é distorcida e rotacionada de três formas diferentes (Andrew Ng, deeplearning.ai).