Universidade Federal de Ciências da Saúde de Porto Alegre

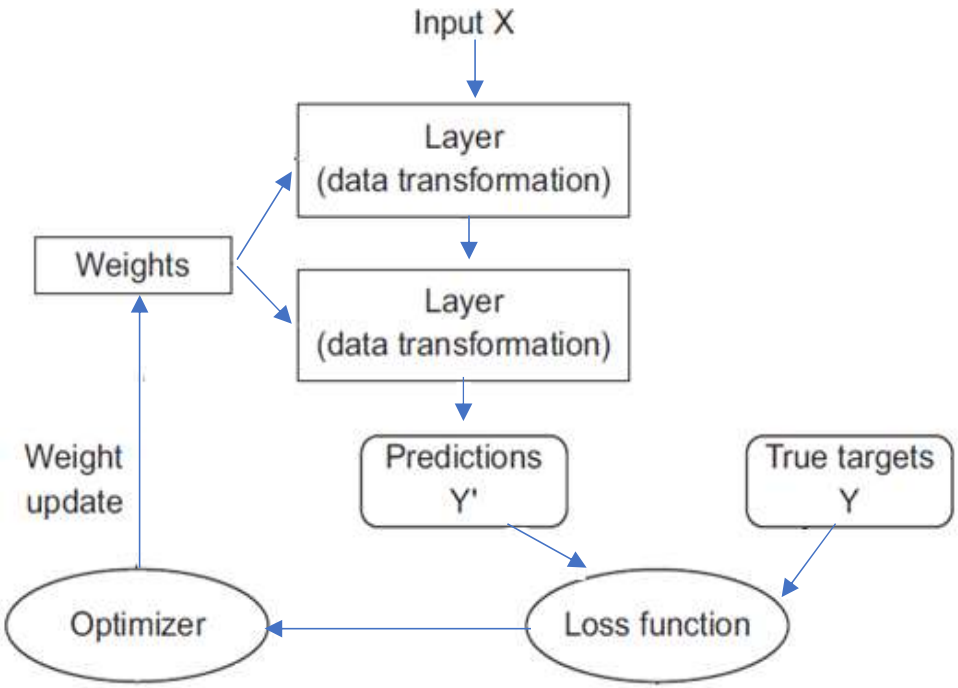
Aprendizado da rede (continuação)

-- Otimização de parâmetros em redes neurais --

Prof. Viviane Botelho
vivianerb@ufcspa.edu.br

Aula anterior: Objetivo de compreender os hiperparâmetros.

- Tipos de *Loss*
- Gradiente Descendente
- *Learning rate*
- Algoritmo de *backpropagation*
- Inicialização de hiperparâmetros
- Número de épocas
- Batch Size

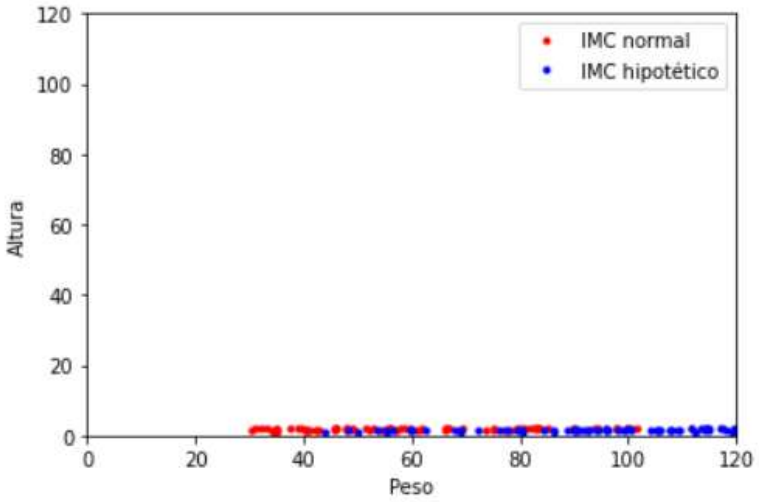


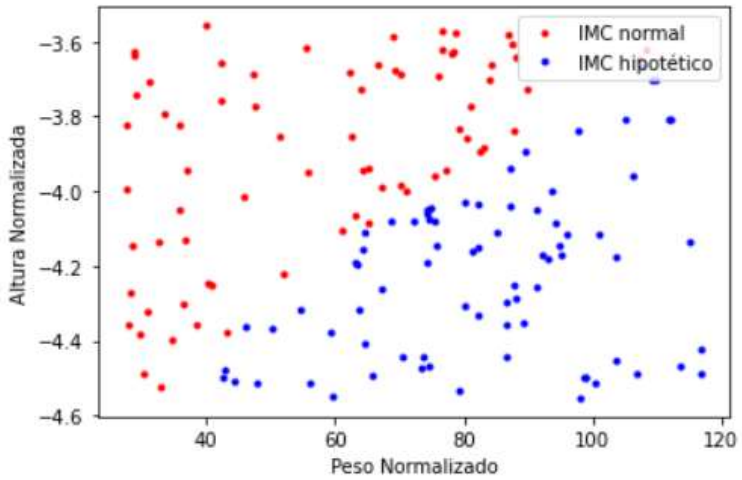
Normalização dos dados: ***Batch normalization***

Deixar os dados de entrada na mesma ordem de grandeza

Normalização das entradas

Exemplo hipotético: Entradas são peso e altura e saída é a classe IMC normal e IMC elevado

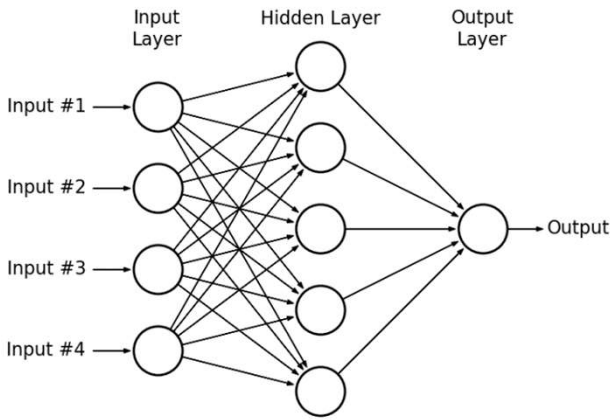




$$X_{i_{norm}} = \frac{X_i - \overline{X_i}}{\sigma_{X_i}}$$

Normalizando os dados apenas antes de sua entrada na rede:

- Pode ser que a normalização de cada *batch* não seja representativa do dataset inteiro.
- Entradas das camadas ocultas não estão normalizadas.



Solução é utilizar *Batch normalization* !

Normalização dos dados: ***Batch normalization***

Batch Normalization é um tipo de camada introduzida entre as camadas densa que pode normalizar de forma adaptativa os dados.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

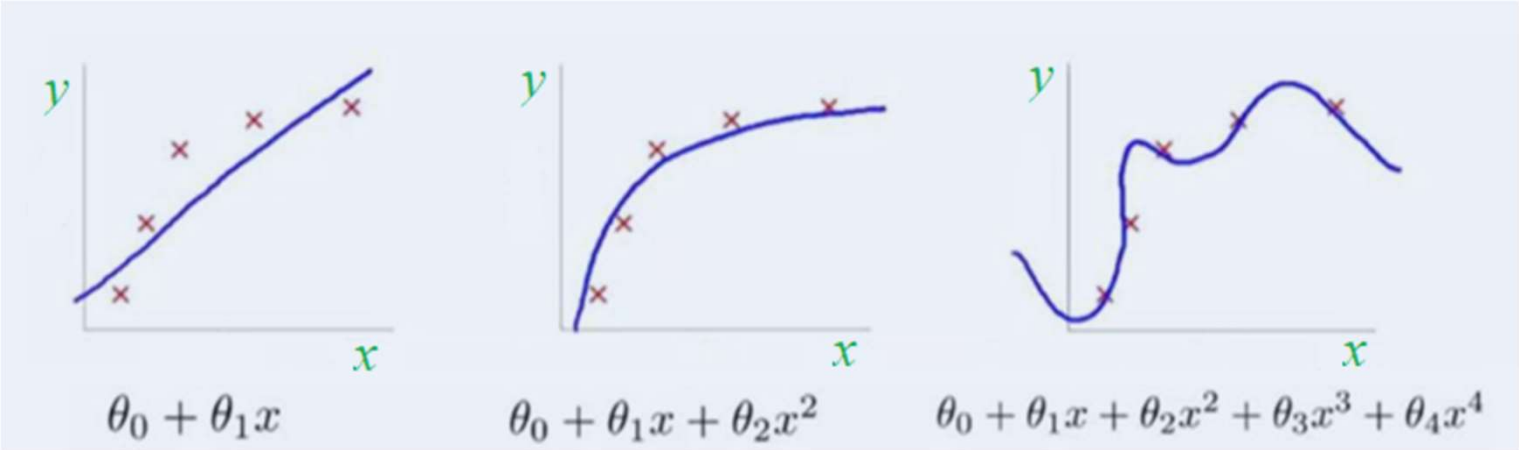
Batch normalization melhora a propagação do gradiente durante a otimização:

- Acelera o treinamento.
- Permite *learning rates* maiores.
- Melhora o processo de treinamento, como um todo.

Regularização: Técnicas para reduzir *overfitting*

Ideia:

- Quanto mais simples o modelo (com menor número de parâmetros), menor é a chance de *overfitting*.



- Forma sistemática "para zerar" os parâmetros desnecessários.

1) Inclusão de penalização na *Loss* (por exemplo, se aplicado a MSE)

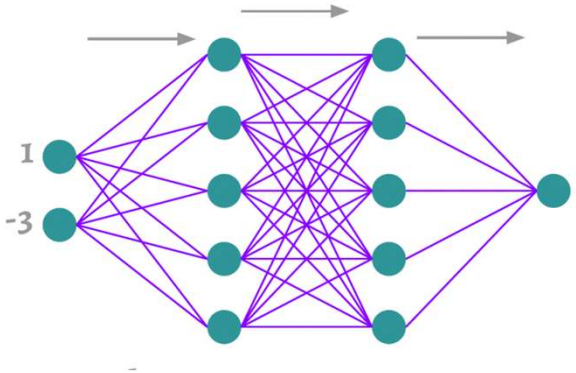
$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} + \begin{cases} \text{L1: } \lambda \sum_{j=0}^{M} |W_j| \\ \text{L2: } \lambda \sum_{j=0}^{M} W_j^2 \end{cases}$$

$\lambda > 0$: parâmetro de regularização!

Regularização: Técnicas para reduzir *overfitting*

2) Camada de *Dropout*

Consiste em zerar **aleatoriamente** (e temporariamente) alguns neurônios ocultos em cada iteração.



Em cada etapa do treinamento, uma rede diferente é gerada. Assim, conceitualmente, o procedimento é semelhante ao uso de um conjunto de muitas redes diferentes. No momento do teste, toda a rede é usada (todas as unidades), mas com pesos reduzidos.

Por que funciona: O *dropout* evita que as unidades se "coadaptem", isto é, as unidades aprendem a não depender muito umas das outras.

Efeitos:

- Força a rede a aprender quais são os neurônios mais robustos.
- O *dropout* aumenta o número de iterações necessárias para convergir.
- O tempo de cada iteração é menor pois a rede é mais simples.

Universidade Federal de Ciências da Saúde de Porto Alegre

Ajuste de hiperparâmetros

Prof. Viviane Botelho
vivianerb@ufcspa.edu.br

Seleção de hiperparâmetros

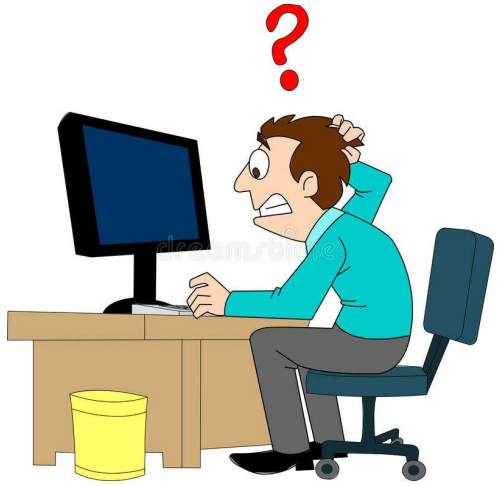
Não há uma metodologia formal para se definir os melhores valores para os hiperparâmetros.

- Aula de hoje: Apanhado de heurísticas, boas práticas e ferramentas.

- Hiperparâmetros:
  - Camadas e neurônios
  - Dropout
  - Learning Rate
  - Batch Size

- Escolher hiperparêmetros:
  - Manualmente: Necessita entendimento de como os o que os hiperparâmetros impactam no modelo.

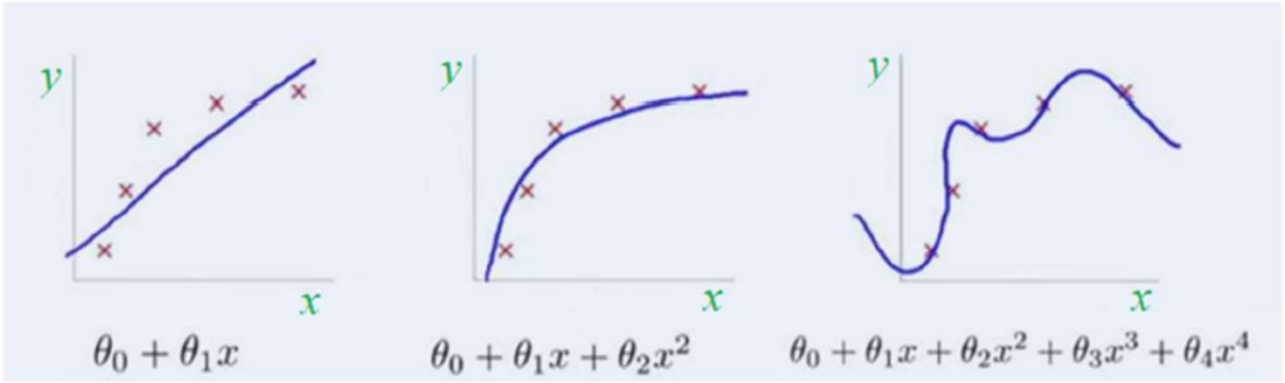
  - Automaticamente: Custo computacional elevado (requer máquina com muita capacidade).



A arquitetura de rede deve sempre ser decidida em última instância com experimentação e determinada pelo erro de validação!

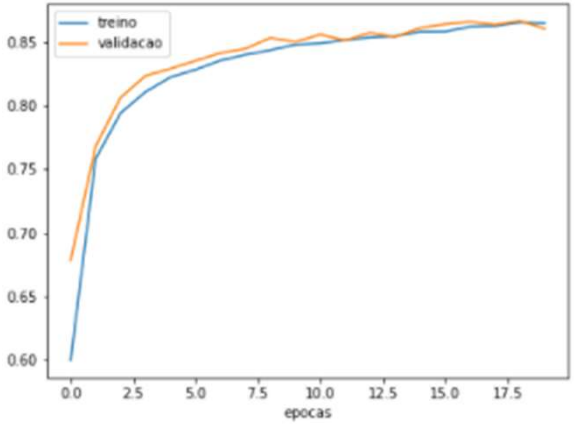
Camadas e neurônios

Premissa: Modelo mais simples possível (com menos camadas e neurônios).

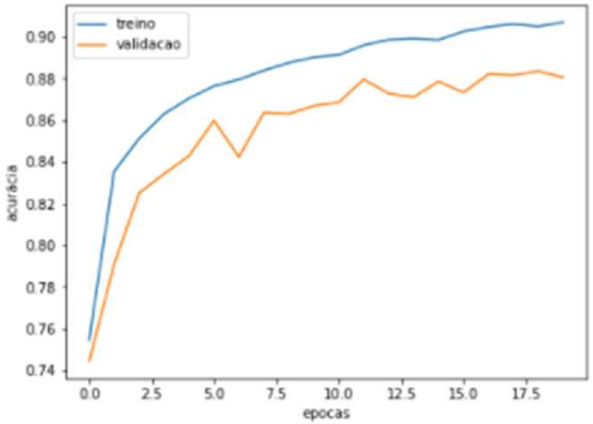


No nosso exemplo:

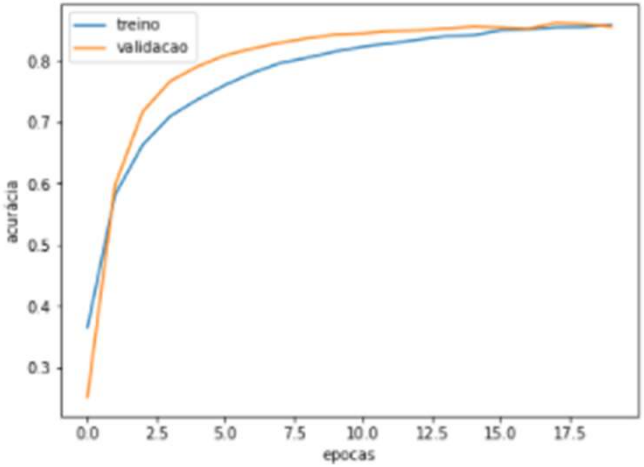
1 Hidden Layer (90)



1 Hidden Layer (900)



Com 4 Hidden Layers (30,25,20,15)



Camadas e neurônios

- Dados simples: Usar 1 ou 2 camadas ocultas.
- Dados compexos: Usar de 3 a 5 camadas ocultas
- O número de neurônios ocultos deve estar entre o tamanho da camada de entrada e o tamanho da camada de saída.
- O número de neurônios ocultos deve ser 2/3 do tamanho da camada de entrada, mais o tamanho da camada de saída.
- O número de neurônios ocultos deve ser menor que o dobro do tamanho da camada de entrada
- Número de neorônios nas camadas ocultas (N_h):

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

$$N_h = \sqrt{N_i * N_o}$$

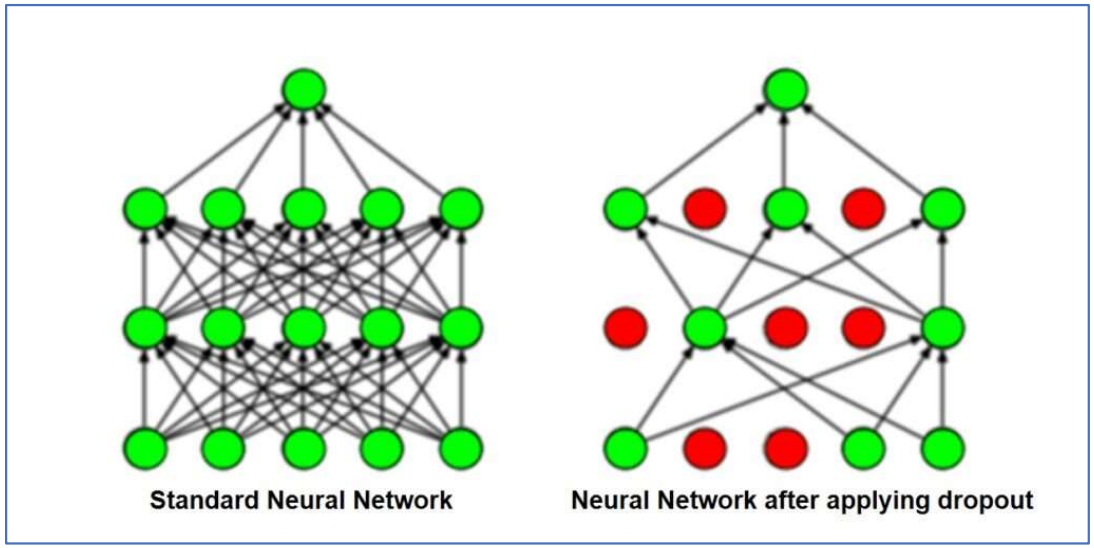
N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

Na maioria das vezes são suposições "arbitrárias"!

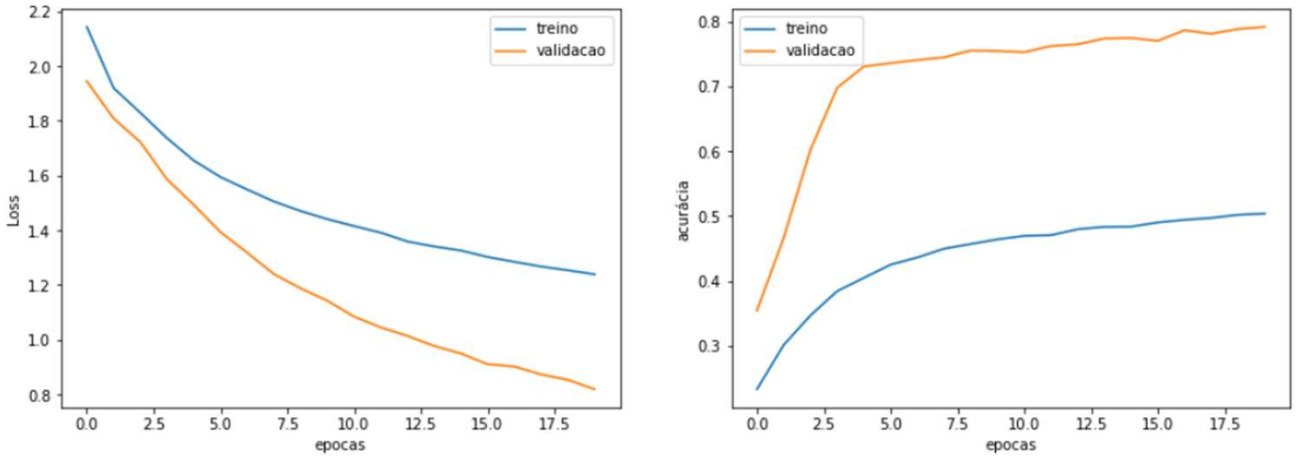
Algumas observações:

- Número de camadas está associado com a não linearidade dos dados.
- Dê preferência em ir reduzindo o número de neurônios ao longo das camadas.
- Comece sempre com o modelo mais simples (1 camada oculta).
- Encontre (na literatura) um problema similar ao seu para iniciar a definição dos hiperparâmetros.

Dropout

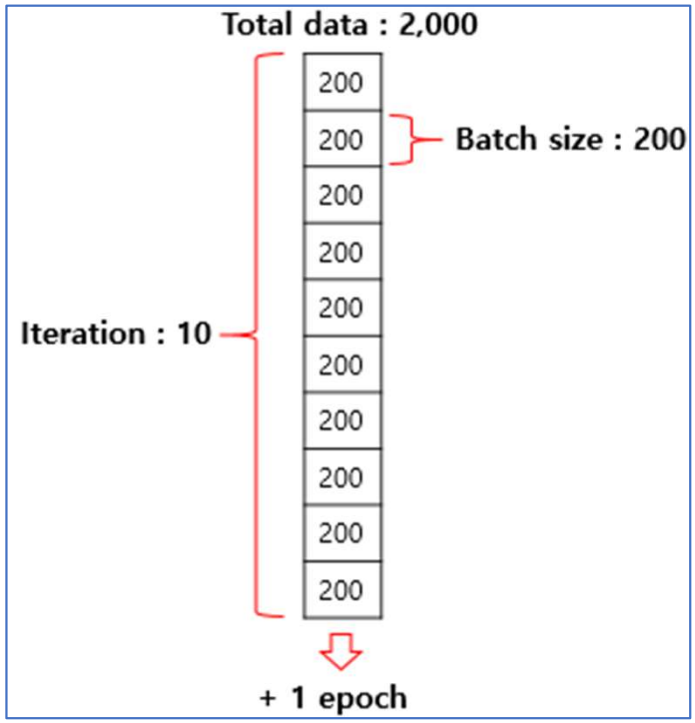


- Se o número de neurônios for muito pequeno, o dropout pode atrapalhar.
- Redes com dropout precisam de redes maiores e com mais iterações.



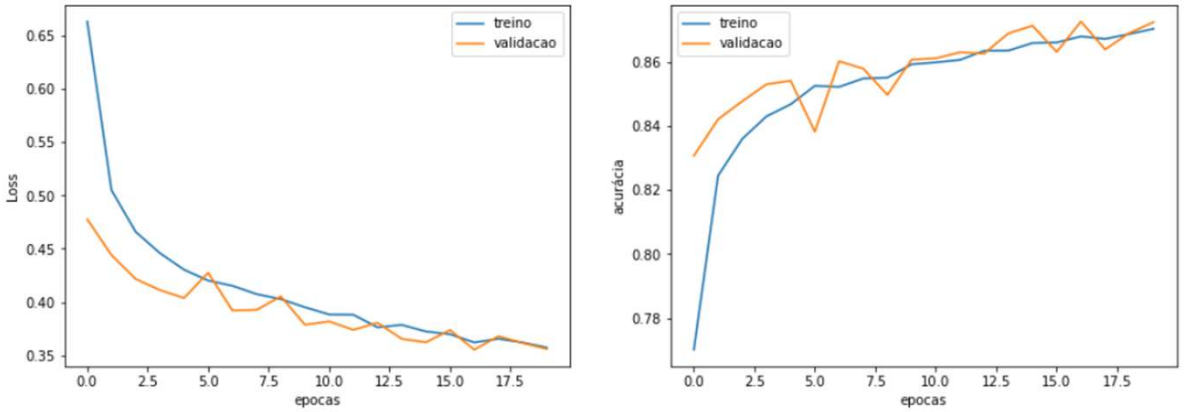
Exemplo de comportamento com excesso de dropout.

Batch size



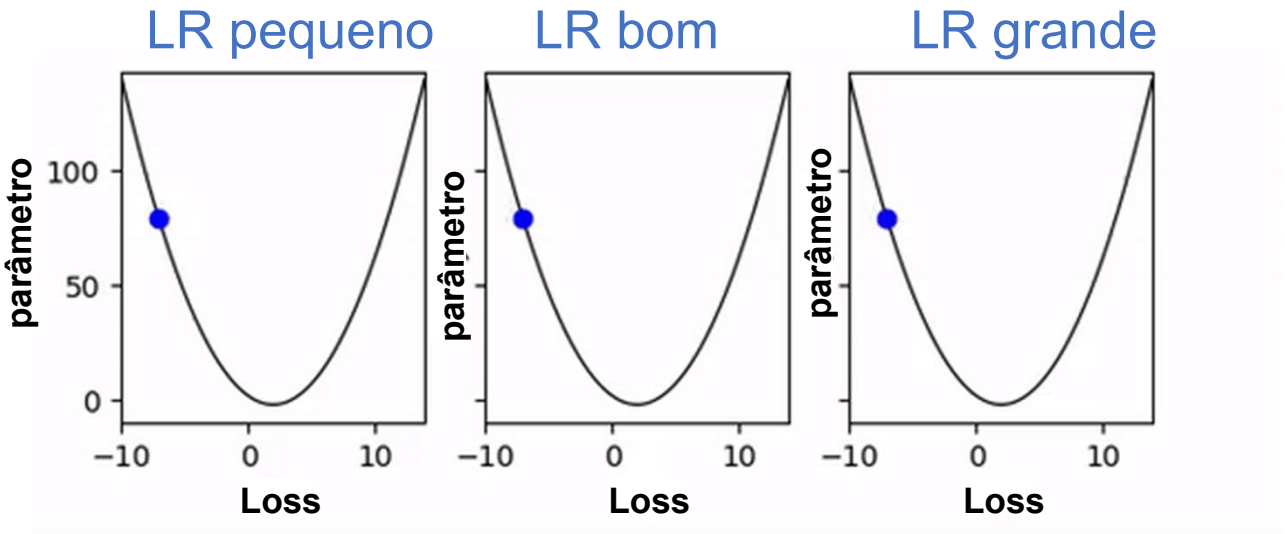
- Batch size grandes oferecem gradientes melhores, mas normalmente são limitados pela memória. Faça o batch size tão alto quanto sua memória pode suportar.
- Sempre avalie o número de amostras do seu conjunto de treinamento para escolher o batchsize.

Lembrando: Batch size default do Keras é 32 (quando o parâmetro não é informado).

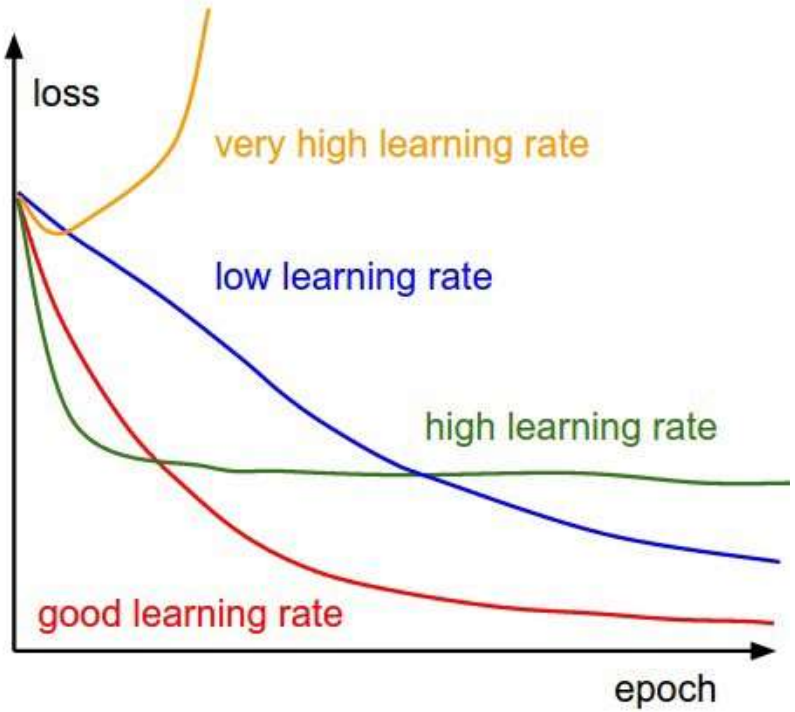


Exemplo de comportamento com batch size pequeno

Learning Rate



Geralmente, uma grande taxa de aprendizado permite que o modelo aprenda mais rápido, ao custo de chegar a um conjunto final de pesos abaixo do ideal. Uma taxa de aprendizado menor pode permitir que o modelo aprenda um conjunto de pesos mais ideal ou mesmo globalmente ideal, mas pode levar muito mais tempo para treinar.



* LR muito elevado pode também causar oscilação, similar ao comportamento do batch size

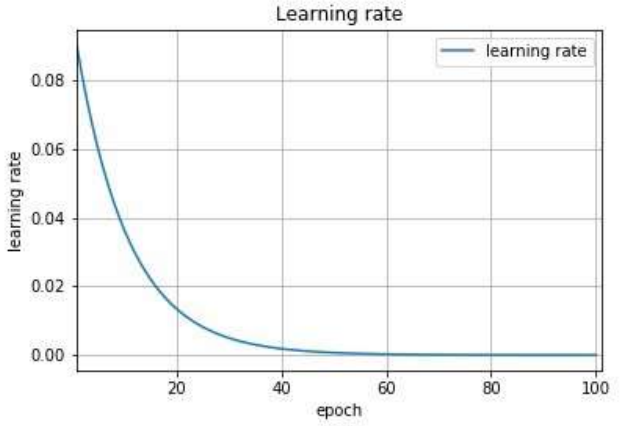
Learning Rate

- Usualmente, o LR varia entre 1 e 10^{-6}.
- Valores usuais: 0.1 e 0.01.
- LR pequenos requerem maior número de épocas.
- Batch Sizes pequenos requerem LR menores (pequenos passos a cada iteração).
- Com base em experiência de autores: *A taxa de aprendizado ideal é normalmente maior do que a taxa de aprendizado que produz o melhor desempenho após as primeiras 100 iterações, mas não tão alta que cause instabilidade.*

Learning Rate Schedule (LR adaptativo)

- LR que decai ao longo das épocas.

- Pode reduzir o tempo de treinamento.



Otimizadores **Adagrad, Adadelta, RMSprop, Adam**, fornecem uma alternativa adaptativa para o LR. Esses métodos de taxa de aprendizado por parâmetro fornecem uma abordagem heurística sem a necessidade de ajuste manual do Learning Rate Schedule.

Outras dicas

- Tente iniciar com a arquitetura de um problema **similar** ao seu.
- Se a qualidade do modelo é ruim mesmo após extensivos testes de ajuste então possivelmente o problema está nos dados (qualidade das features e quantidade).
- Se o modelo se ajusta bem aos dados de treino mas não se adequa aos dados de teste, tente simplificá-lo e/ou teste diferentes otimizadores e algoritmos de aprendizado. Se não funcionar mais dados são necessários.
- Algumas vezes transformações nos dados de entrada podem ajudar.
- ***features selection (VCII)***.

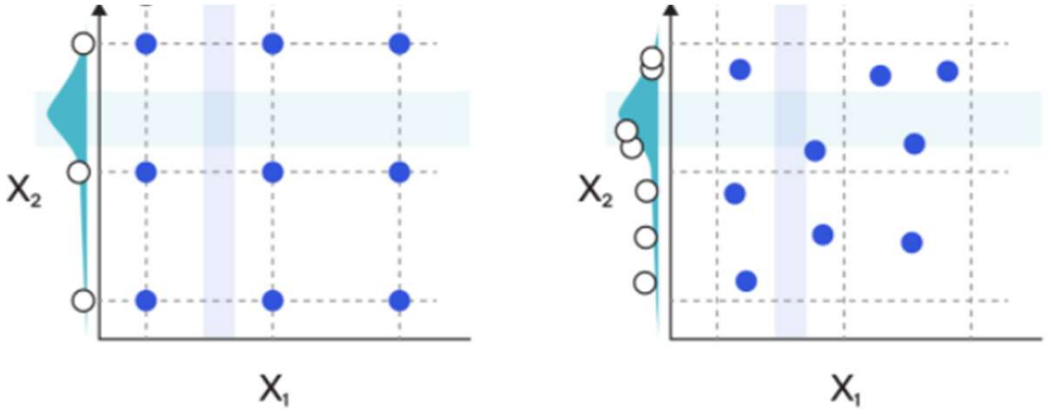
Métodos automáticos de ajuste

- Define-se um espaço de busca de hiperparâmetros. O algoritmo automaticamente treina e avalia os modelos com base nos dados de validação.

- Diversos métodos: exaustivos, Bayesianos, algoritmos genéticos....

Os dados de validação estão sendo usados indiretamente para o treinamento do modelo, pois são eles que norteiam a seleção da arquitetura. Assim, fica muito claro a necessidade dos dados de teste para avaliar o modelo final.

- Ferramentas do Scikit-Learn



(a) Standard Grid Search (b) Random Search

- Alternativa: Hyperas (https://github.com/maxpumperla/hyperas)

Referências

https://towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594

https://web.archive.org/web/20140721050413/http://www.heatonresearch.com/node/707

https://towardsdatascience.com/pruning-neural-networks-1bb3ab5791f9

https://jeffmacaluso.github.io//post/DeepLearningRulesOfThumb/

https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

https://keras.io/api/optimizers/adam/

https://machinelearningmastery.com/using-learning-rate-schedules-deep-learning-models-python-keras/

https://scikit-learn.org/stable/

https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fhagan.okstate.edu%2FNNDesign.pdf%23page%3D469&clen=11812273&chunk=true

https://www.linkedin.com/pulse/choosing-number-hidden-layers-neurons-neural-networks-sachdev

https://www.linkedin.com/pulse/choosing-number-hidden-layers-neurons-neural-networks-sachdev

https://www.baeldung.com/cs/neural-networks-hidden-layers-criteria

https://github.com/maxpumperla/hyperas