Treinamento de Redes Profundas (2)

ROSELI APARECIDA FRANCELIN ROMERO
ICMC/Universidade de São Paulo
www.icmc.usp.br/~rafrance - rafrance@icmc.uspbr
São Carlos, Brazil

SUMÁRIO

Estratégias de treinamento

Função de custo e gradiente
Otimizadores
Normalização de Dados
Convergência empírica
Suposições para convergência e aprendizado
Estratégias para melhor generalização

Transferência de aprendizado

COMO TREINAR? OTIMIZAÇÃO

Machine Learning e Deep Learning depende de entender otimização e conhecer bem:

- Função de custo/perda e intuição de seus valores
- (Intuição) do gradiente da função
- Inicialização
- Algoritmo de otimização
- Taxa de aprendizado
- Tamanho do batch
- Convergência ao longo do treinamento

FUNÇÃO CUSTO OU PERDA

Métrica que indique o custo de escolher o modelo atual

- Idealmente deve ser convexa e produzir um gradiente com boa magnitude
- Difícil, considerando todas as direções do hiper-espaço de parâmetros

FUNÇÃO CUSTO OU PERDA

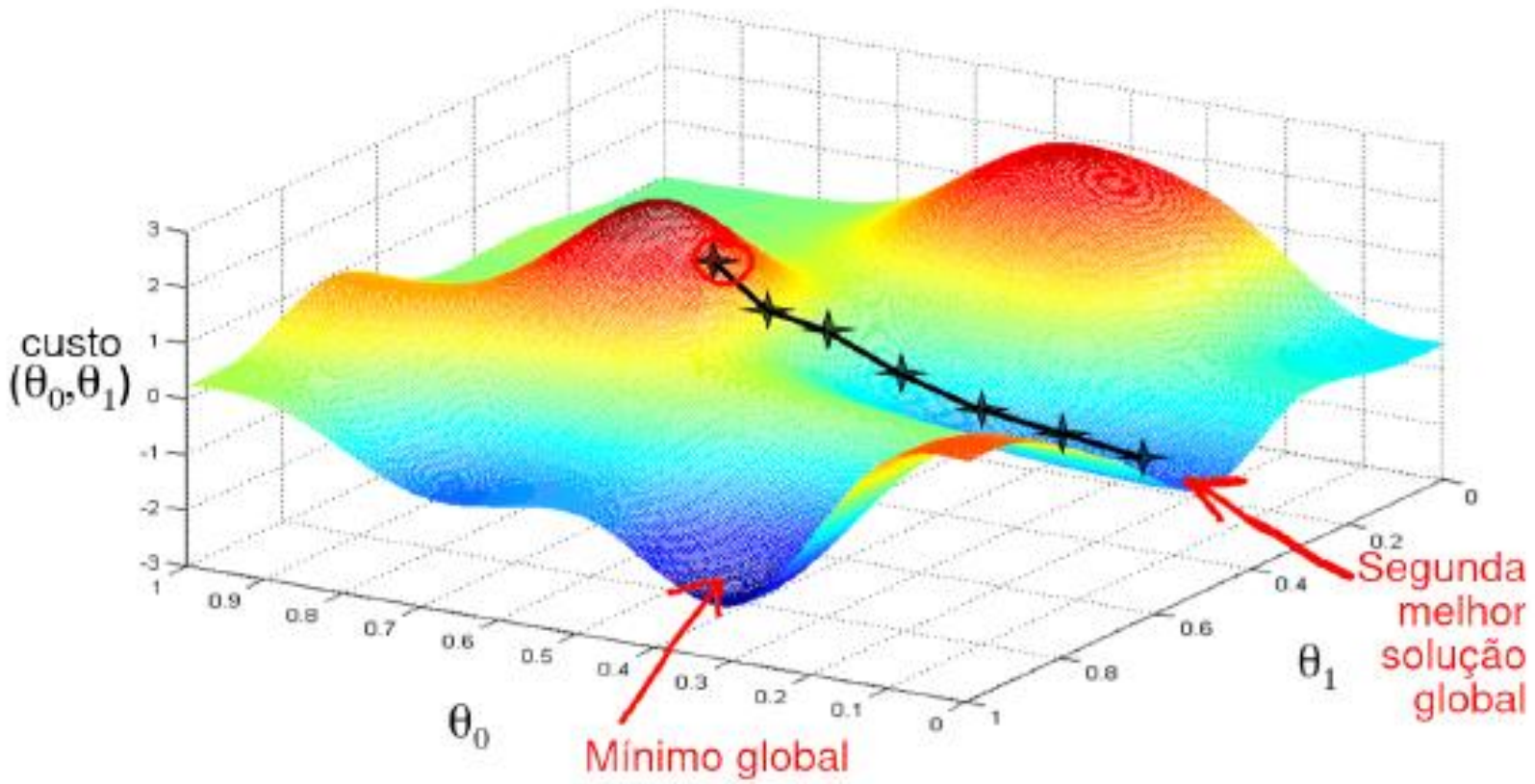
Destaques

- **Mean-squared-error**: erro médio quadrático/perda quadrática
  - utilizada para valores contínuos,
  - mede a divergência quadrática de cada valor de entrada com relação à saída
- **Cross-entropy**: entropia cruzada
  - mais comum e recomendada para probabilidades
  - teoria da informação
  - intuição: o numero de bits adicionais necessários para representar o evento de referência ao invés do predito.

O GRADIENTE DA FUNÇÃO ERRO

Codifica as taxas de alteração no espaço de parâmetros

- queremos andar na direção do vale, em busca do mínimo global



Qual o papel do GRADIENTE no treino

Backpropagation

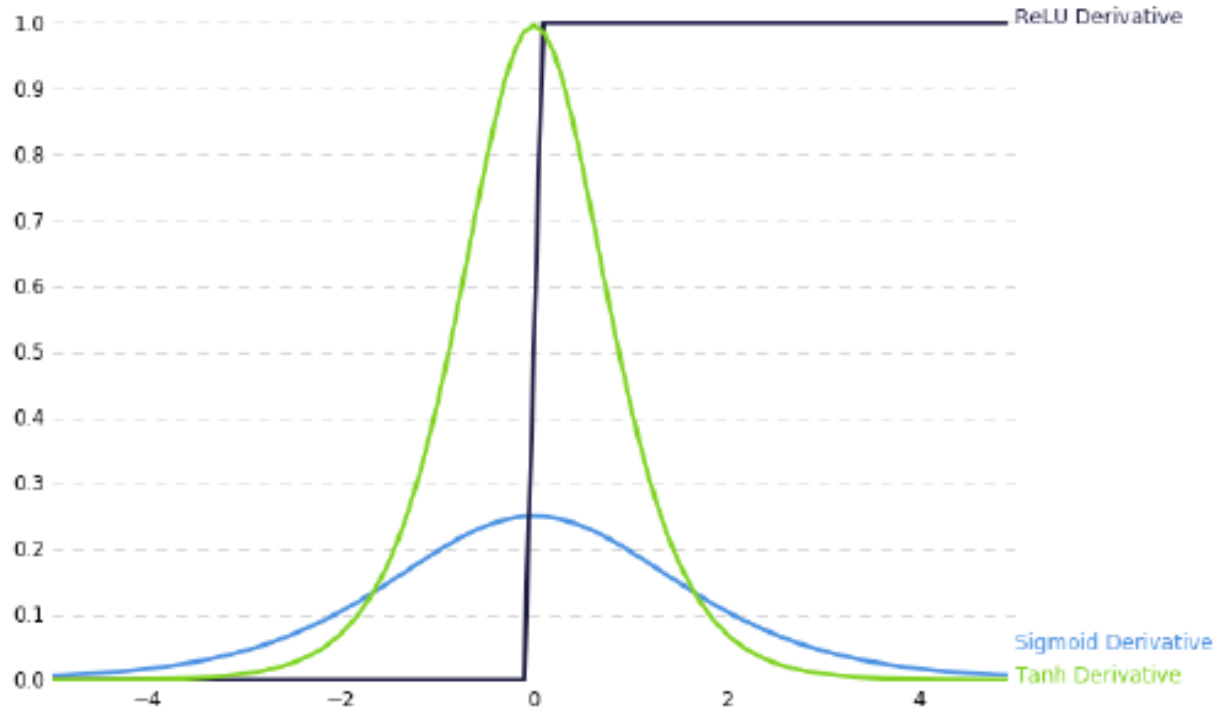
- utiliza a derivada ao longo das camadas para adaptar os pesos
- as funções de custo e de ativação tem que produzir derivada útil

Vanishing gradient

- se ativações geram valores baixos não é possível adaptar
- usar precisão dupla (double) e escalar funções é possibilidade
- motivador de ReLU ao invés de Sigmóides como ativação

Derivadas

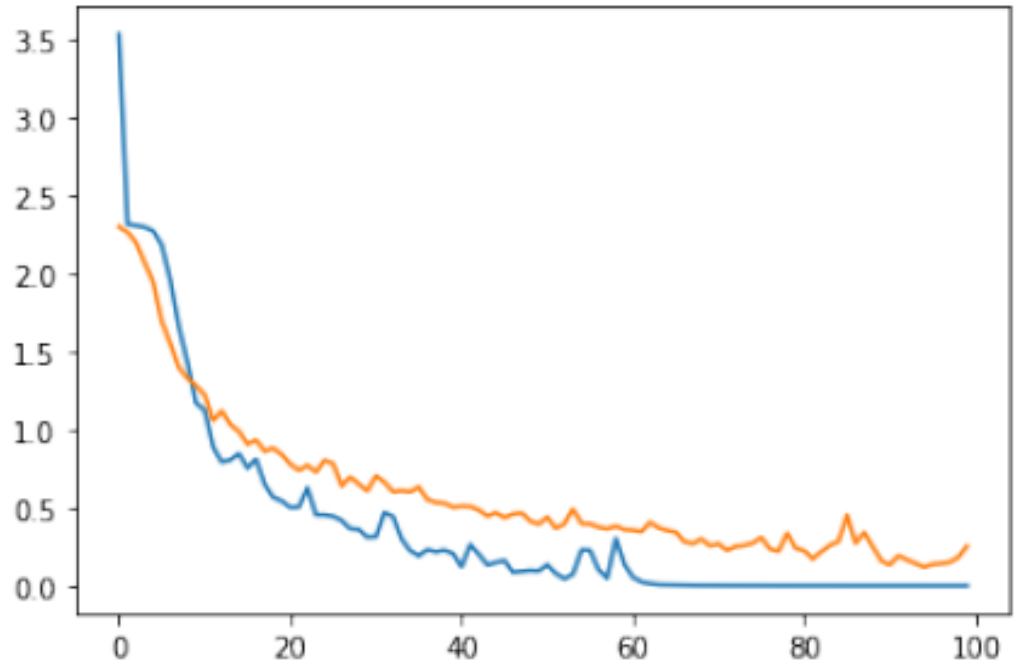
Derivada da sigmóide, ReLU e tangente hiperbólica



Agradecimentos a Harini suresh (http://harinisuresh.com) pelos gráficos

Função Custo vai sendo minimizada

Ao longo do treinamento a rede adapta os pesos cada vez mais devagar, convergindo para uma solução



Inicialização

Aleatória portanto o resultado é diferente a cada execução.

Escolhas comuns

- Pesos: valor aleatório pela distribuição normal entre 0-1
- Bias: 0 (zero)

A complexidade do treinamento dificulta múltiplas execuções

- Importante fazer experimentos piloto em pequenos subconjuntos de dados

SUMÁRIO

- **Estratégias de treinamento**

  - Função de custo e gradiente
  - Otimizadores
  - Normalização de Dados
  - Convergência empírica
  - Suposições para convergência e aprendizado
  - Estratégias para melhor generalização

- **Transferência de aprendizado**

Otimizadores

Stochastic Gradient Descent (SGD)

Formulação original (atualização por instância)

$$\begin{aligned}\theta_{k+1} &= \theta_k - \alpha_k \nabla_\theta \ell(y, f(x; \theta_k)), \\ &= \theta_k - \alpha_k g(x, \theta_k),\end{aligned}$$

α_k é a taxa de aprendizado (learning rate) na iteração k

Batch Stochastic Gradient Descent (SGD)

computando o gradiente da função de custo de um lote de instâncias X_k na iteração k

$$\theta_{k+1} = \theta_k - \alpha_k g(X_k, \theta_k),$$

Tamanho do Batch e taxa de aprendizado

Há uma relação entre tamanho de batch e taxa de aprendizado.

Batch

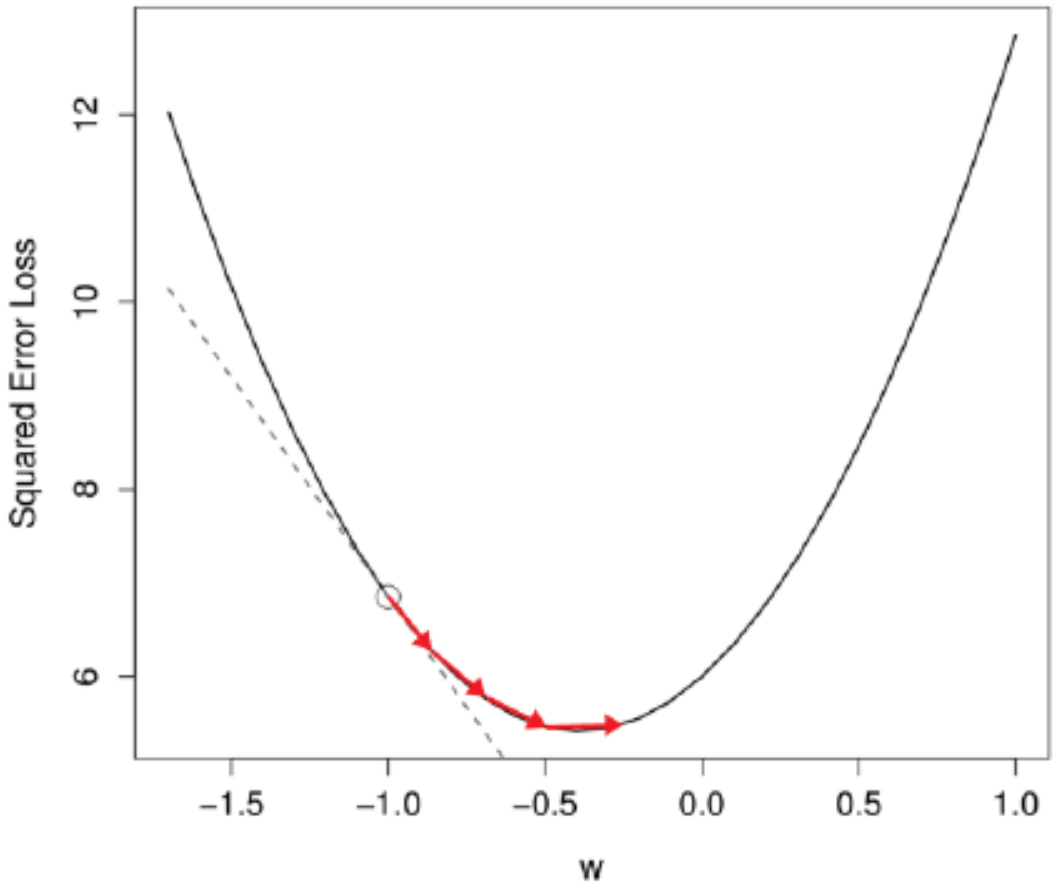
- Padrão é 32
  - batches maiores: estimativas mais suaves, difícil manter na memória, exige ajustar bem a taxa de aprendizado,
  - batches menores: estimativas mais ruidosas, mas que mostraram vantagens em encontrar melhores mínimos.

Tamanho do Batch e taxa de aprendizado

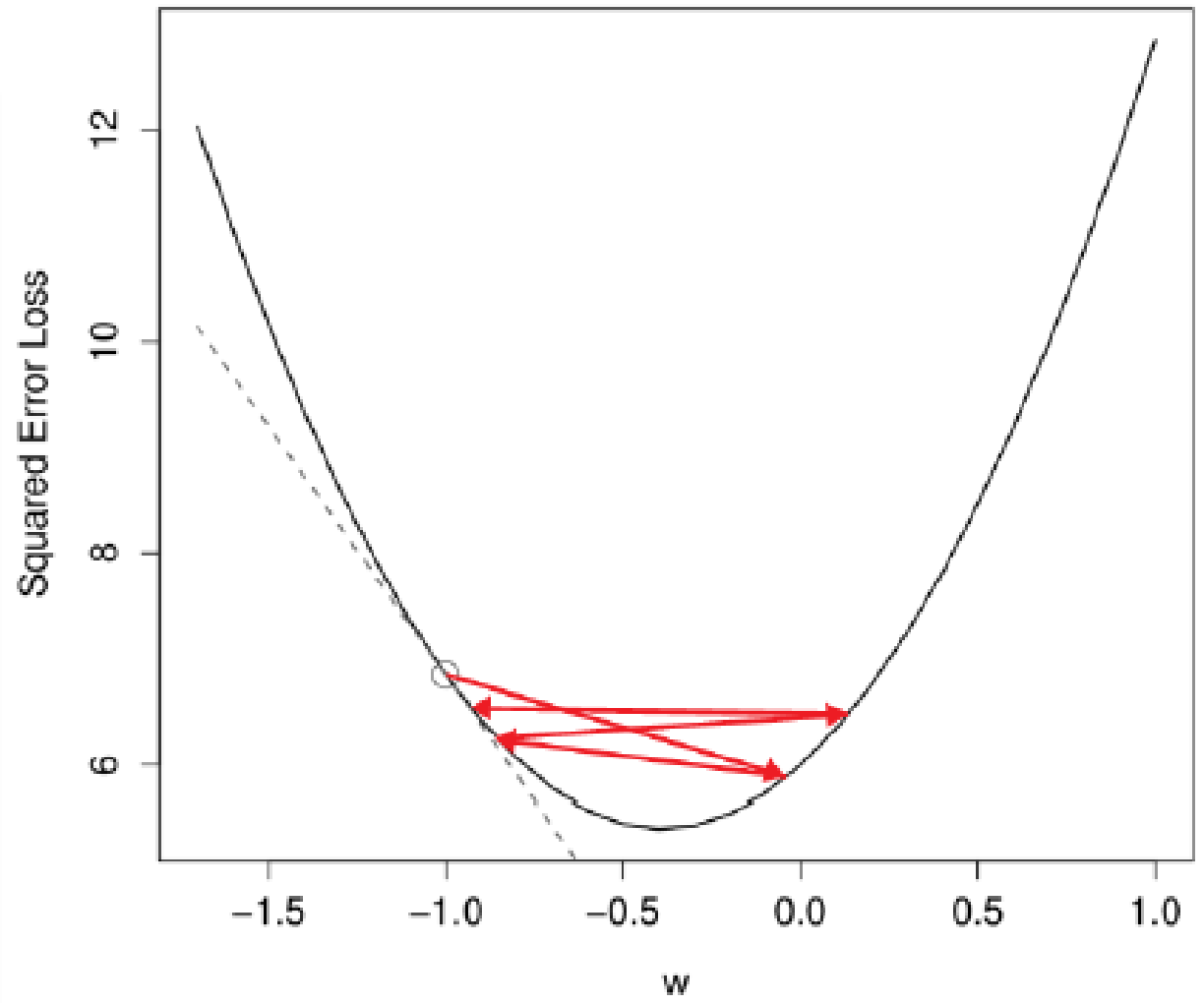
Taxa de aprendizado

- Padrão é 0.01
  - pode ser pouco adequado para alguns otimizadores
  - pode ser pouco adequado para batchs maiores (ou muito pequenos)
- É recomendado iniciar com um valor maior, e reduzir a taxa progressivamente (learning rate scheduling).

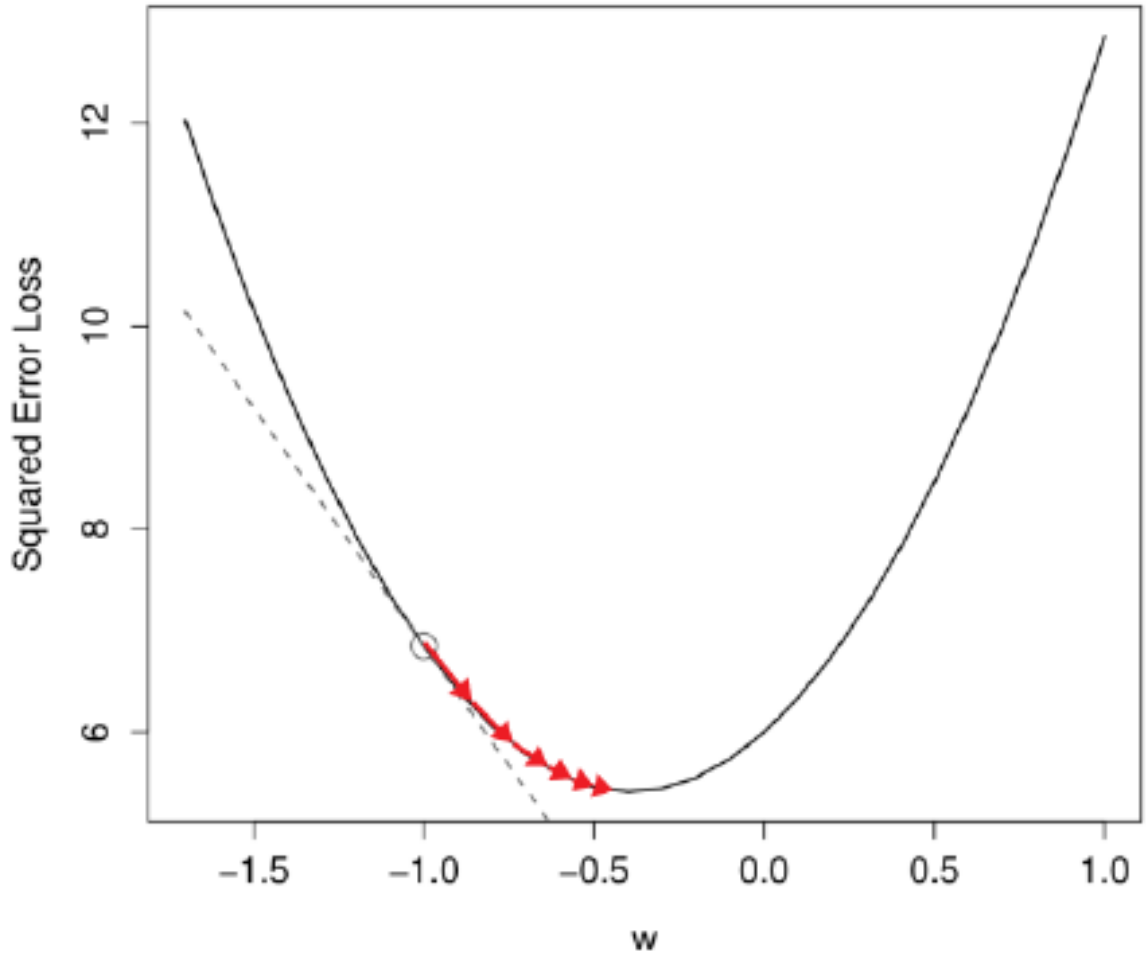
Taxa de aprendizado: com valor pequeno



Taxa de aprendizado: com valor excessivamente grande



Taxa de aprendizado: com decaimento



Taxa de Aprendizado

Recomendação 1:

→ Utilizar decaimento de taxa de aprendizado

Otimizadores

Momentum

Interpreta o custo como um terreno montanhoso.

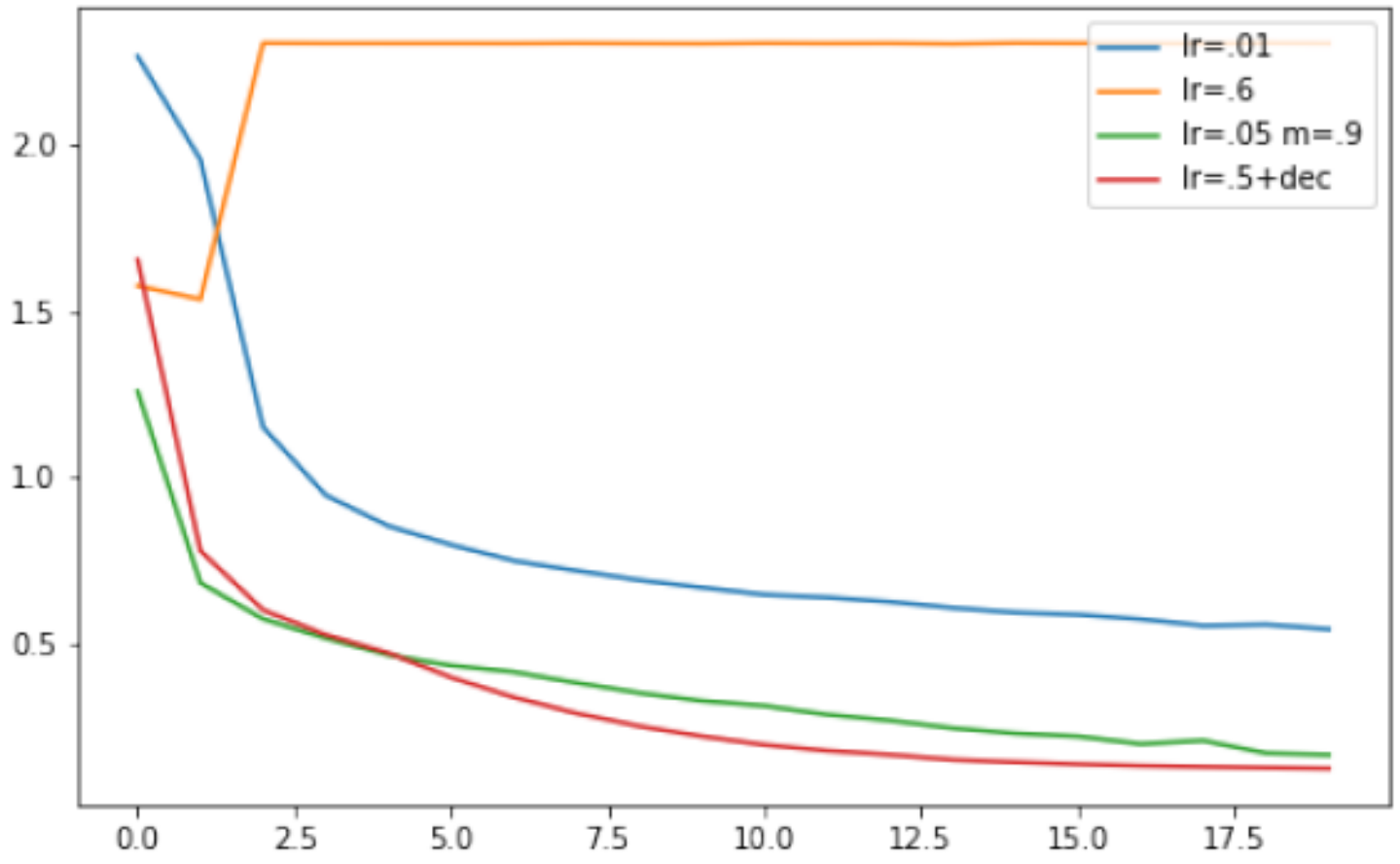
- Inicializar: posicionar partícula com velocidade zero no terreno
- Otimização: rolar partícula, considerando a aceleração.
- Consequência: velocidade ajustada pela a magnitude de atualizações anteriores

$$\theta_{k+1} = \theta_k + m \cdot v - \alpha_k g(x, \theta_k),$$

v é o momentum, inicialmente 0;
m peso (hiperparâmetro): quanto menor funciona como atrito que reduz a energia cinética do sistema

- Investigar $m \in [0.5, 0.9, 0.95, 0.99]$
- ou iniciar com valor menor e aumentar ao longo das épocas

Taxa de aprendizado



Outros Otimizadores

Adam

Utiliza momentos do gradiente: o segundo momento é usado para normalizar o primeiro, evitando outliers/pontos de inflexão

$$\theta_{k+1} = \theta_k - \alpha_k \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}$$

$\hat{m}$ e $\hat{v}$ são estimativas corrigidas do primeiro e segundo momentos do gradiente.

- $\hat{m}_k$ é a soma do gradiente atual com o acúmulo de gradientes anteriores $\hat{m}_{k-1}$ (similar a momentum).
- $\hat{v}_k$ é a soma do quadrado do gradiente em k com o acúmulo de valores anteriores $\hat{v}_{k-1}$ (taxa de aprendizado adaptativa).
- Funciona melhor com passo **menor** do que SGD

Otimizadores

RECOMENDAÇÃO 2

→Utilizar SGD (+ Momentum) ou
→Adam

Exemplo 4
Adaptação da Taxa de Aprendizado

SUMÁRIO

- **Estratégias de treinamento**

  - Função de custo e gradiente
  - Otimizadores
  - Normalização de Dados
  - Convergência empírica
  - Suposições para convergência e aprendizado
  - Estratégias para melhor generalização

- **Transferência de aprendizado**

Normalização de dados

- Exemplos de técnicas:
  - **Normalização (ou padronização) z-score**: valores com média zero e desvio padrão 1;
  - **Normalização min-max**: valores no intervalo 0-1.
- Objetivo: facilitar otimização ao normalizar/padronizar a magnitude dos valores utilizados no treinamento:
  - suaviza as ativações dos neurônios, reduzindo a variância do gradiente;
  - ataca o problema de "desaparecimento" do gradiente (vanishing gradient) em particular para redes profundas.

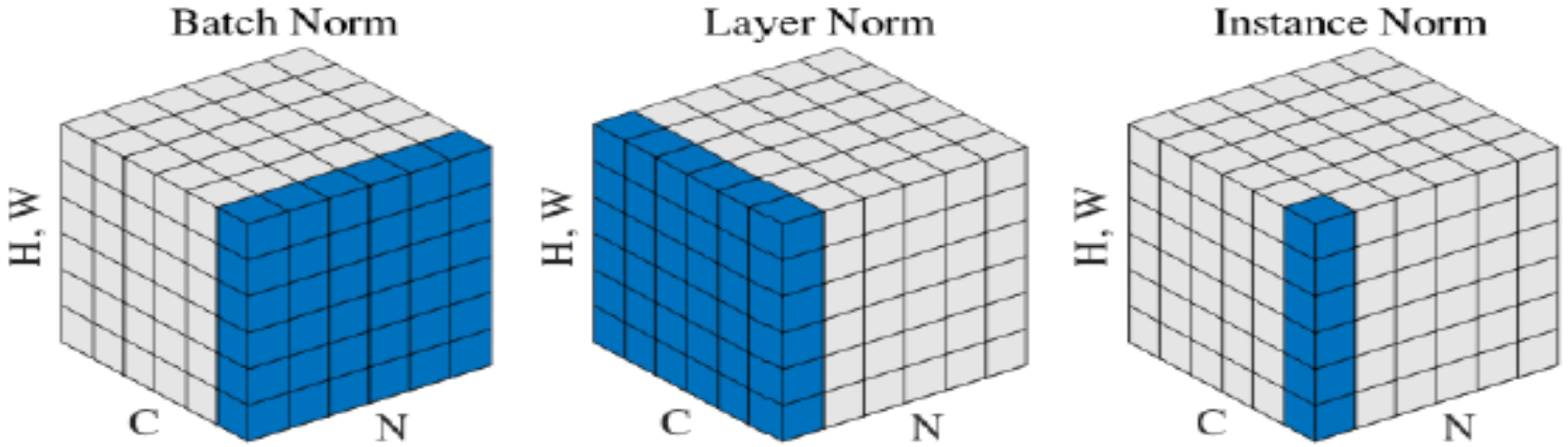
Normalização de dados

- Podemos usar como pré-processamento considerando todos os dados de treinamento
- Mas durante o treinamento pode também ser aplicado ao batch ou às camadas

Tipos de normalização baseada em camadas!

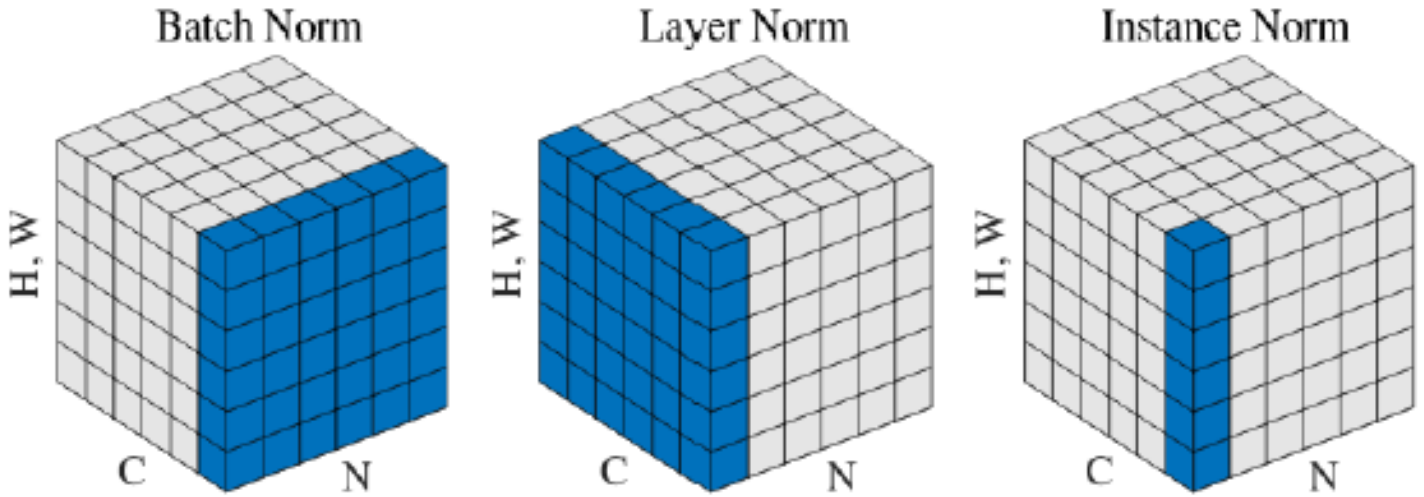
Batch
Camada
Instância

Normalização de dados: Batch



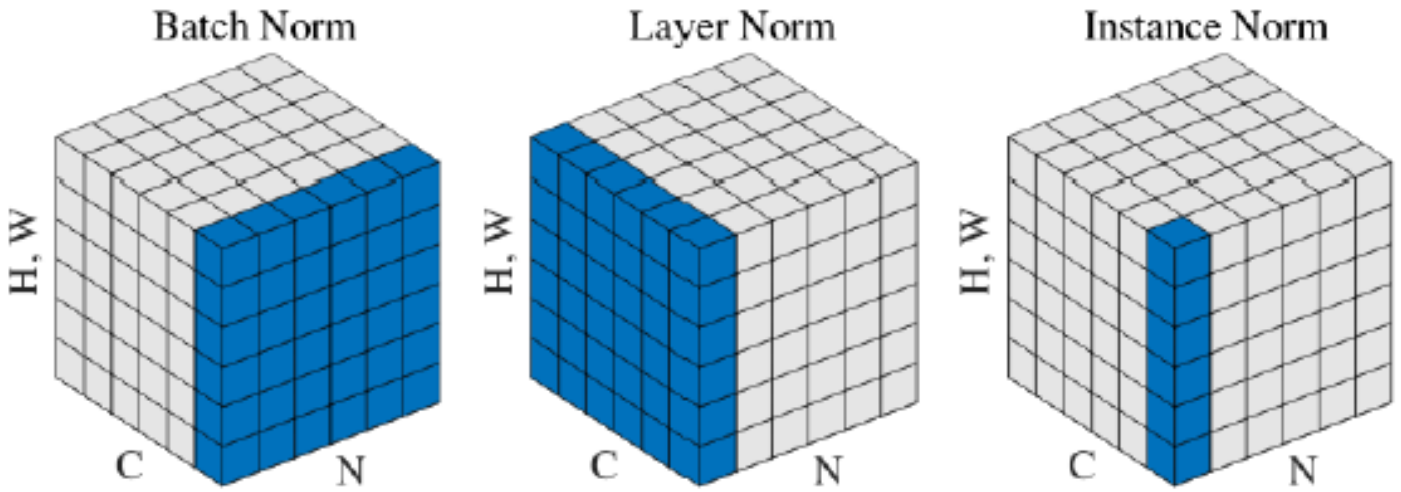
- **Batch normalization** (BN): para cada batch
  - média e desvio calculados por canal (total C)
  - normalização nos canais das N instâncias no batch
- Funciona melhor com batchsize ≥ 32

Normalização de dados: Layer



- **Layer normalization** (LN):
  - média e desvio calculados por instância (total N)
  - normalização de cada instância ao longo de todas as ativações de todos os canais
- Independe do tamanho do batch, mais comum em redes recorrentes e adversariais

Normalização de dados: Instance



- **Instance normalization** (IN):
  - média e desvio por instância e canal (total $N \times C$)
  - normalização de cada instância ao longo de cada canal
- Independe do tamanho do batch,
  mais comum em redes recorrentes e adversariais

SUMÁRIO

Estratégias de treinamento

Função de custo e gradiente
Otimizadores
Normalização de Dados
Convergência empírica

- Suposições para convergência e aprendizado
- Estratégias para melhor generalização

Transferência de aprendizado

Convergência ao longo do treinamento

O gráfico do custo diz muito sobre o aprendizado

Recomendação 3
➢ Acompanhe o custo ao longo de épocas, se possível com conjunto de validação (idealmente não deve ser o teste!) ➢ Inicie com experimentos com poucos exemplos ➢ Explore os hiperparâmetros tentando obter “overfitting” para um subconjunto de exemplos, atingindo custo abaixo de uma tolerância, e depois refine a busa num conjunto maior.

Suposições que foram feitas

Dados de treinamento

- Limpos
- Representativos e bem definidos com relação à tarefa: classes, valores da regressão, etc.
- Baixa taxa de erros de rótulo
- Quantidade de dados é suficiente

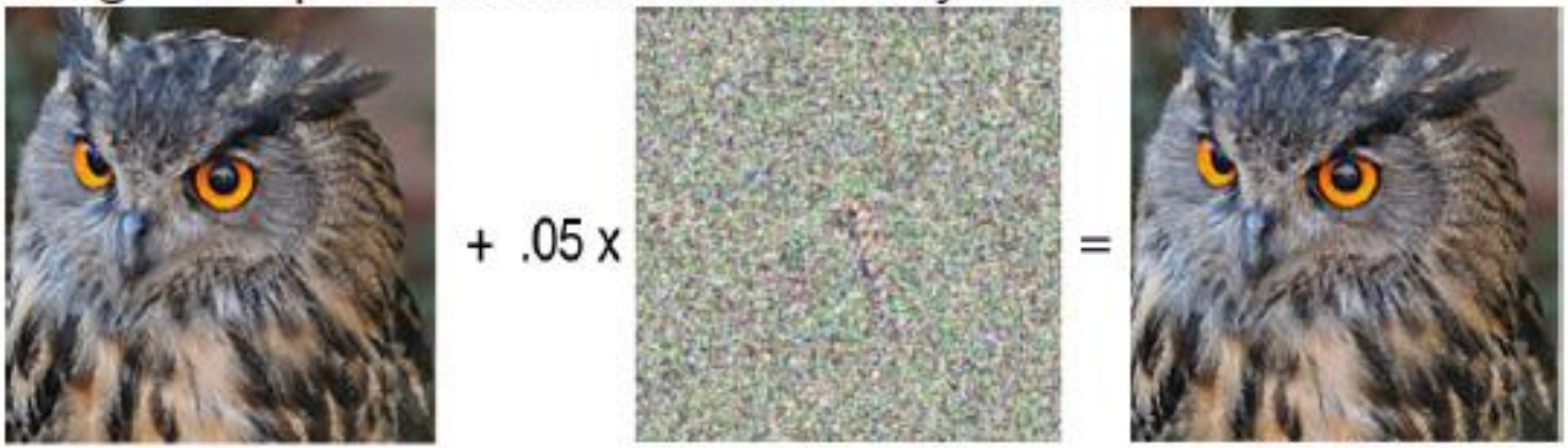
- E se não for possível?
  Riscos:
  – *overfitting*, baixa generalização,
  – dificuldade na convergência.

Controvérsias

Marcus (2018) em "Deep Learning: a critical appraisal":
"... sistemas que se baseiam em Deep Learning frequentemente devem generalizar para além de dados específicos... mas a garantia de performance em alta qualidade nesses cenários é mais limitada."

Ataques Adversariais

Artigo "Deep Neural Networks are easily fooled"



Zhang et al (2017)

"... nossos experimentos estabeleceram que redes convolucionais profundas do estado da arte (...) *facilmente ajustam rótulos aleatórios* nos dados de treinamento."

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

ABSTRACT

Despite their massive size, successful deep artificial neural networks can exhibit a

- **Estratégias de treinamento**

- Função de custo e gradiente
- Otimizadores
- Normalização de Dados
- Convergência empírica
  - Suposições para convergência e aprendizado
  - Estratégias para melhor generalização

- **Transferência de aprendizado**

I - Regularização

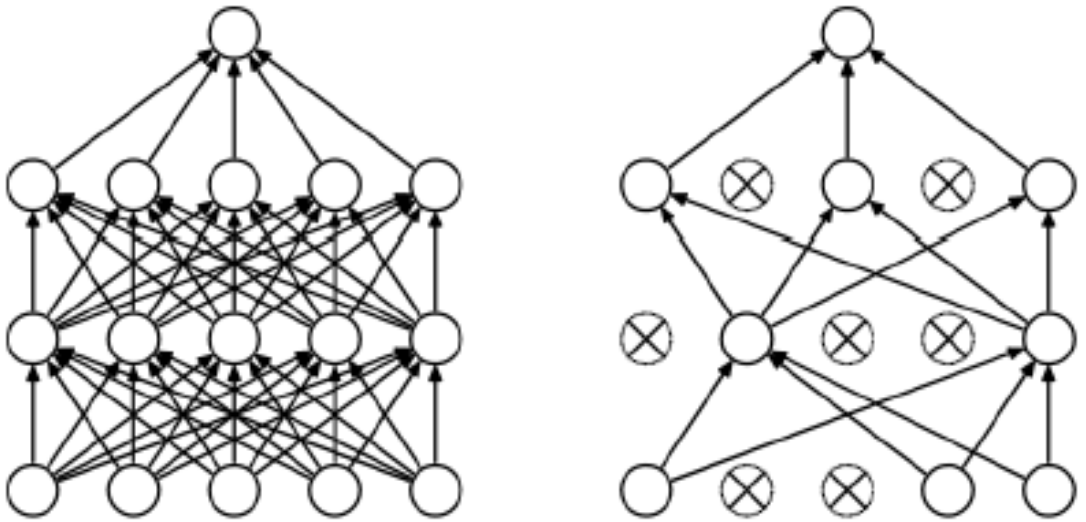
Relembrando a regularização L2 (ou de Tikhonov)

$$\ell(\Theta) = \frac{1}{N}\sum_{i=1}^{N} \ell_i(x_i, y_i, \Theta) + \lambda\frac{1}{2}||\Theta||^2$$

- Objetivo: limitar a capacidade do modelo de se especializar demais nos dados
- Formas:
  - Global: na função de perda ponderada por λ
  - Definindo λ_l por camada (ou grupos de camadas)
- Interpretação: vê cada entrada como sendo de maior variância

II - DROPOUT

- Objetivo: limitar a capacidade de certos parâmetros e memorização de dados
- Implementado na forma de "camada"
- A cada iteração, desliga ativações de neurônios aleatoriamente com probabilida
- Interpretação: treinamento com técnica "Bagging" por camada



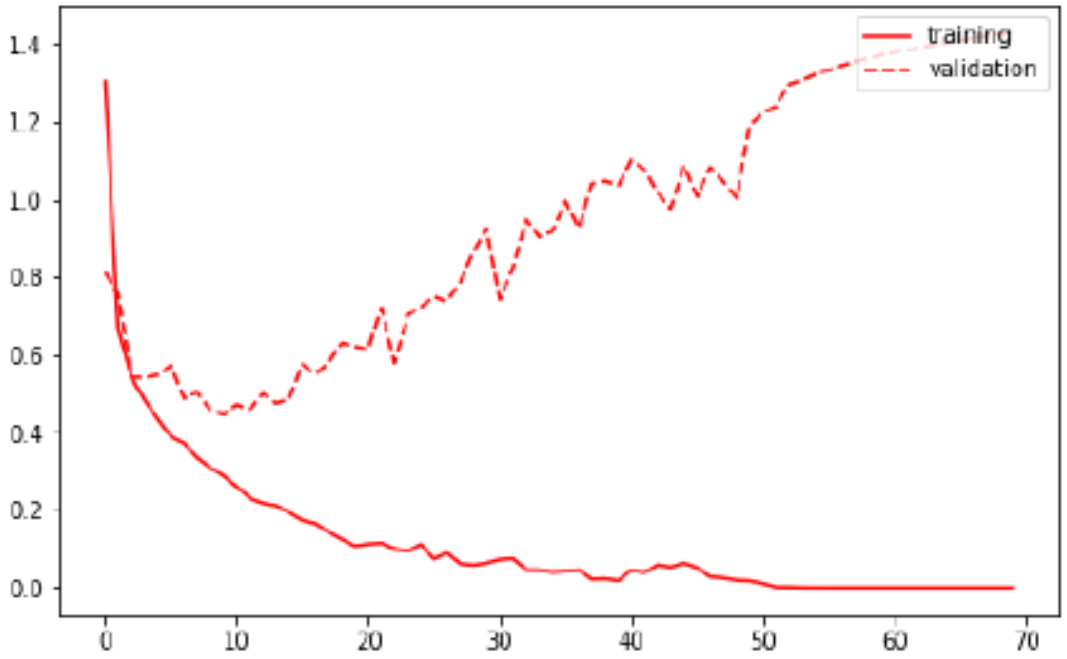
(a) Standard Neural Net

(b) After applying dropout.

Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting

III – Parada Precoce

- Objetivo: evitar que o modelo memorize os dados de treinamento ao treinar por muitas épocas
- Acompanhar um conjunto de validação e interromper de acordo com a relação do custo no treinamento e validação



IV – Coletar mais dados

- Objetivo: impedir que o treinamento considere apenas um conjunto limitado de exemplos
- Baseado na lei dos grandes números, quanto maior a amostra, teremos um melhor estimador

V – Data Augmentation

- Objetivo: gerar exemplos artificiais na esperança de que melhore as propriedades de convergência
- Implementado por meio da manipulação de exemplos existentes, ou sua combinação
- Exemplos:
  - Dados estruturados: SMOTE
  - Não estruturados: rotação, corte, injeção de ruído, e outros que não descaracterizem os dados
  - Dropout na camada de entrada:
    eliminando features aleatoriamente a cada iteração.

V – Data Augmentation (cont.)

Dica para melhoria de performance final

- Para cada exemplo de teste:
  1. Gerar m exemplos com aumentação de dados
  2. Predizer o resultado para os m exemplos
  3. Combinar as predições: por média, maioria ou outro método

SUMÁRIO

- **Estratégias de treinamento**

- Função de custo e gradiente
- Otimizadores
- Normalização de Dados
- Convergência empírica
  - Suposições para convergência e aprendizado
  - Estratégias para melhor generalização

- **Transferência de aprendizado**

Transferência de aprendizado

Utilizar um modelo treinado em uma determinada tarefa ou domínio, aproveitando o aprendizado em outra tarefa ou domínio alvo.

Transferência de Aprendizado

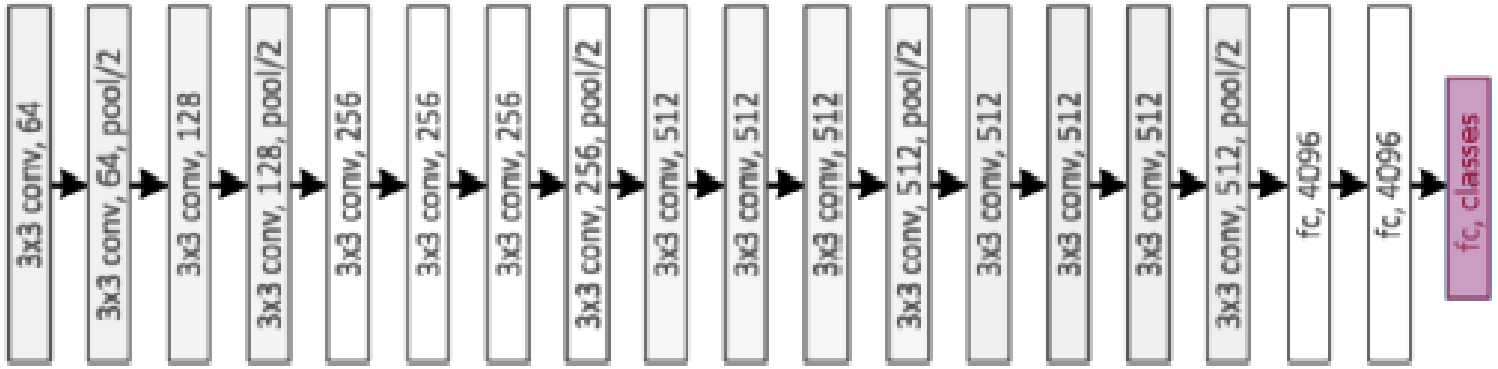
- Modos mais comuns:
  - ajuste fino/ adaptação dos parâmetros
  - Extração de características

Ajuste fino / adaptação de parâmetros

Exemplo: classificação de imagens

Modelo fonte: CNN treinada na ImageNet

- Transferência de aprendizado
  - Redefinir a última camada (inicialização aleatória) com o número de classes desejado
  - Realizar treinamento a partir dos pesos pré-treinados
  - Permitir adaptação apenas da últimas camadas, congelando as demais

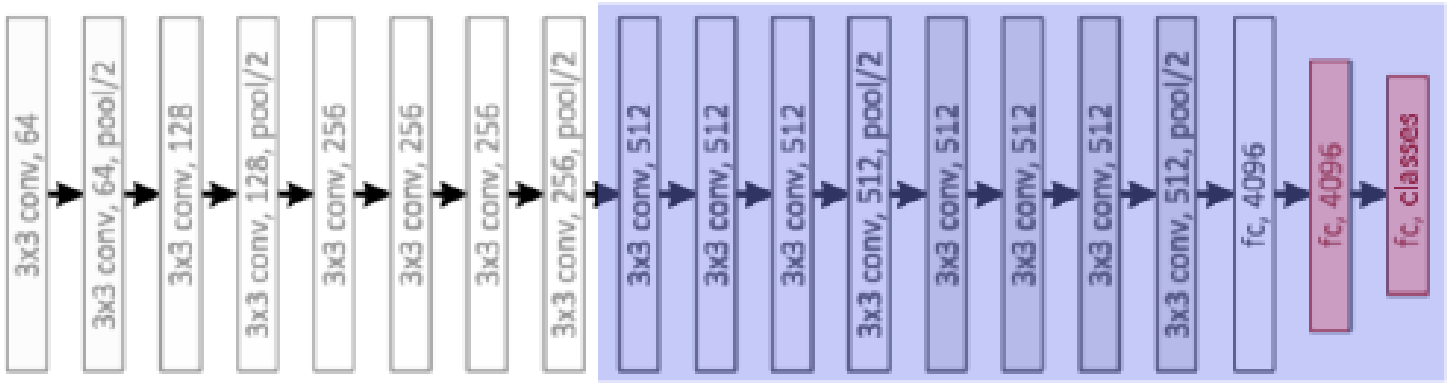


Ajuste fino / adaptação de parâmetros

Exemplo: classificação de imagens

Modelo fonte: CNN treinada na ImageNet

- Ajuste-fino:
  - Comumente feito após o anterior, em que já temos o classificador treinado
  - (Opcionalmente) Re-inicializar ou inserir novas camadas densas ocultas
  - Permitir adaptação de camadas do meio da rede (sem reinicializar seus pesos), congelando algumas iniciais



Transferência de Aprendizado

Dicas

- CNNs com menos parâmetros costumam generalizar melhor para dados muito diferentes do treinamento
- Exemplos: MobileNet, SqueezeNet, etc. funcionam melhor em imagens médicas do que ResNet e Inception.
- Ajuste-fino pode não convergir se tivermos poucos dados, ex. menos de 100 instâncias por classe.

Dicas

1 - Novo conjunto de dados é pequeno e parecido com o original.

- Nesta situação o *fine-tuning* da CNN pode gerar *overfitting*. Como os dados são similares, é esperado que as características já aprendidas sejam relevantes no novo problema.
- Se os dados forem semelhantes aos dados originais, esperamos que os recursos de nível superior no ConvNet sejam relevantes para esse conjunto de dados.

Dicas

1 - Novo conjunto de dados é pequeno e parecido com o original.

- Por exemplo, se a rede original foi treinada para reconhecer carros, você ainda pode usá-la para reconhecer vans, já que os recursos de nível superior, como rodas, portas etc., seriam comuns.
- Portanto, a melhor idéia pode ser treinar um classificador linear usando os recursos extraídos da CNN.
- Assim, é recomendado treinar um classificador como o SVM que recebe como entrada as *features* extraidas pela CNN.

Dicas

2- Novo conjunto de dados é grande e similar ao original.

- Com mais dados, há mais garantias que o modelo não fará *overfitting*. Tornando o *fine-tuning* viável.
- Isso significa que você pode pegar a CNN pré-treinada e executar a retropropagação novamente com os novos exemplos para ajustar os pesos para uma melhor precisão.

Dicas

3 - Novo conjunto é pequeno e muito diferente do original

- Neste cenário, o mais recomendado é treinar um classificador utilizando features extraídas de uma das camadas iniciais onde as *features* são mais genéricas.
- Se você tiver um pequeno conjunto de dados muito diferente do conjunto de dados no qual a CNN foi treinada, talvez não seja melhor treinar o classificador na parte superior da rede, que conterá recursos mais específicos do conjunto de dados no qual a rede foi treinado.
- Em vez disso, talvez seja melhor treinar o novo classificador de algum lugar anterior da rede, que deve ter recursos mais simples e genéricos.

Dicas

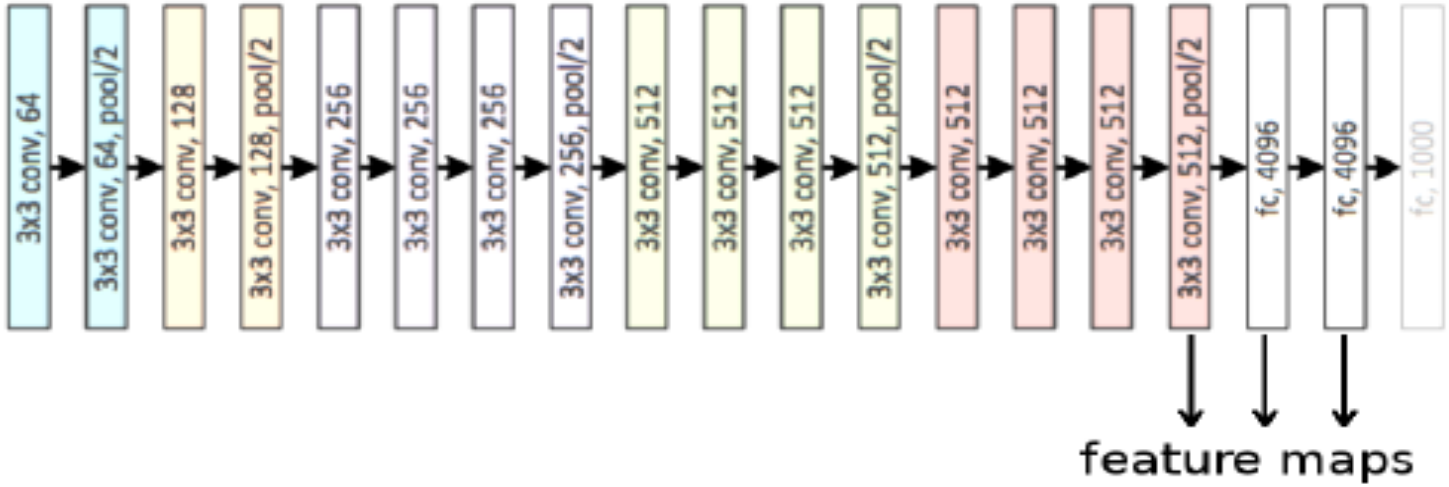
4 - Novo conjunto de dados é grande e muito diferente.

- Com muitos dados podemos treinar a rede do zero.
- Porém, na prática ainda podemos nos beneficiar da inicialização da rede com pesos de um modelo pré-treinado.

Extração de características

Características para dados não estruturados

- Carregar rede neural treinada em grande base de dados
- Passar exemplos de sua base de dados pela rede para predição (não treinamento!
- Obter os mapas de ativação de alguma camada



Extração de características

Dicas

- Aplicar redução de dimensionalidade baseada em PCA, Product Quantization ou outra projeção
- Treinar modelo de aprendizado raso com maiores garantias de aprendizado com poucos dados: SVM, árvore de decisão, etc.
- Características também efetivas para recuperação baseada em conteúdo

Transferência de Aprendizado

O aprendizado de transferência é uma técnica poderosa e muitas personalidades importantes da indústria de IA e ML, incluindo Andrew Ng, acreditam que o aprendizado de transferência será o próximo principal fator de sucesso comercial do ML.

Classificação de frutos de cacau

A total of 1243 images of un/ripe pods was collected.

- ImageNet (online, open source image database).
- Frames extracted from several video clips, which provided substantial variation.

Three types of Transfer Learning were applied.

- **Total training**: Training of all layers using the weights of the checkpoint as the starting point.
- **Partial training**: Freezing the weights of most hidden layers, training the last convolutional reduction and upper layers (*dropout*, and dense output layer).
- **Minimum training**: Freezing all hidden layers, only training the upper layers (*dropout*, and dense output layer).

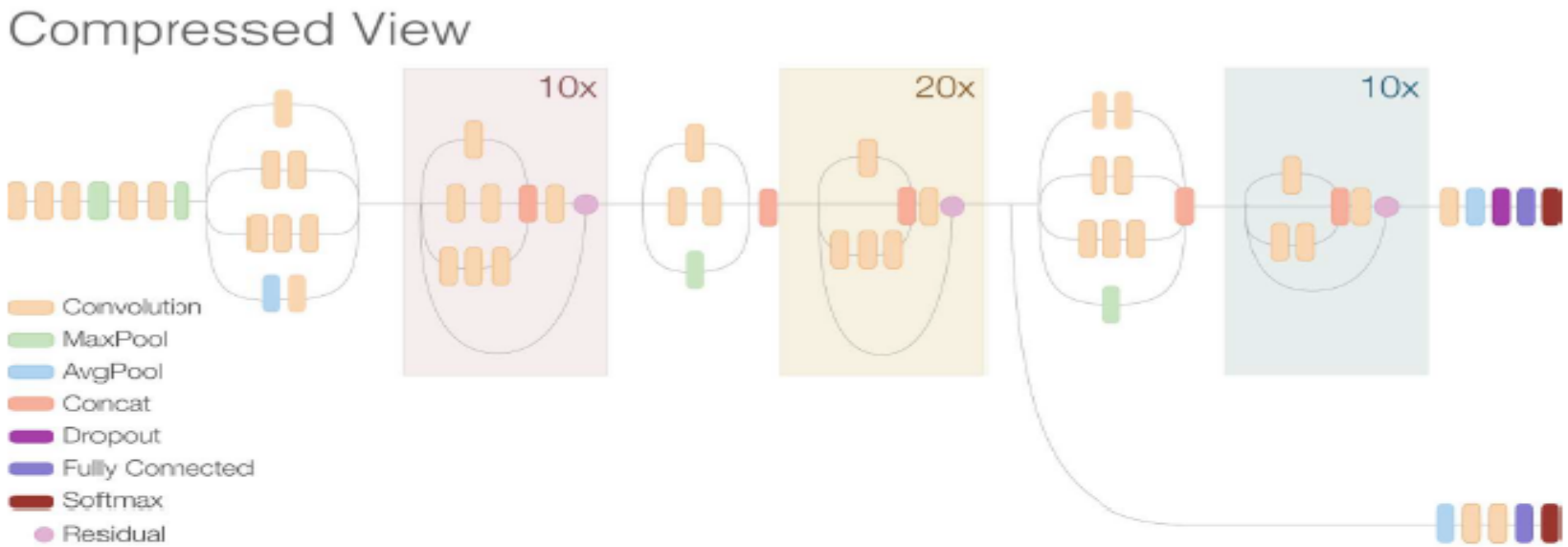


Figure: Inception-Resnet-v2 model architecture. Source: A. Alemi, Improving inception and image clas- sification in tensorflow, 2016. [Online]. Available: https://research.googleblog.com/2016/08/improving-inception-and-image.html

Acurácia por epoch

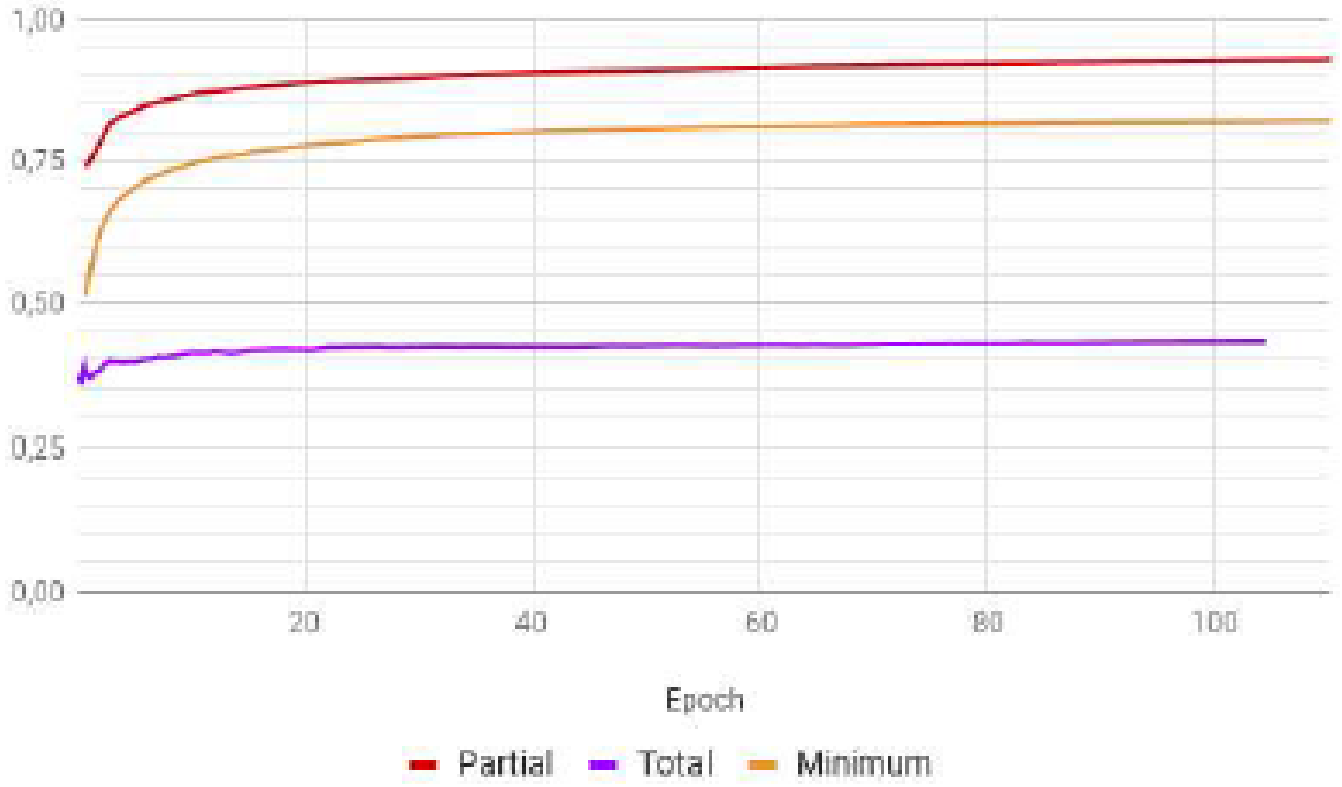


Tabela comparativa

Table: Accuracy comparison by each type of training.

	Total	Partial	Minimum
Epochs	105	975	434
Training	0.4328	0,9743	0,8363
Evaluation	0.3846	0,8956	0,8297

Mensagem ....

- Deep Learning não pode ser tratado como panacéia;
- Há ainda preocupações sobre sua capacidade de generalização e fragilidade a ataques;
- Sua grande utilidade está no aprendizado de representações, em particular para dados não estruturados...
  - representações que parecem ter excelente capacidade de transferência