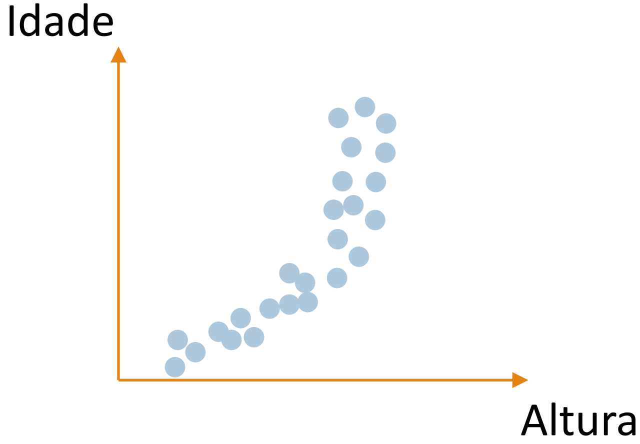
Redes Neurais para Processamento de Imagens

PROF. CESAR HENRIQUE COMIN

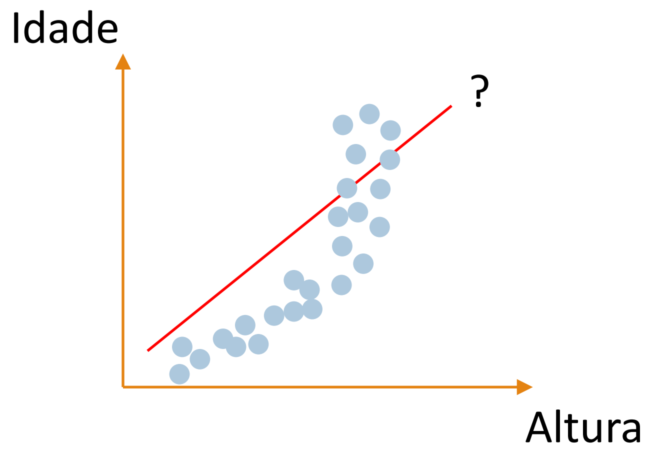
Regressão linear

Vamos supor que queremos prever a idade de uma pessoa com base em algumas de suas características



Regressão linear

Vamos supor que queremos prever a idade de uma pessoa com base em algumas de suas características



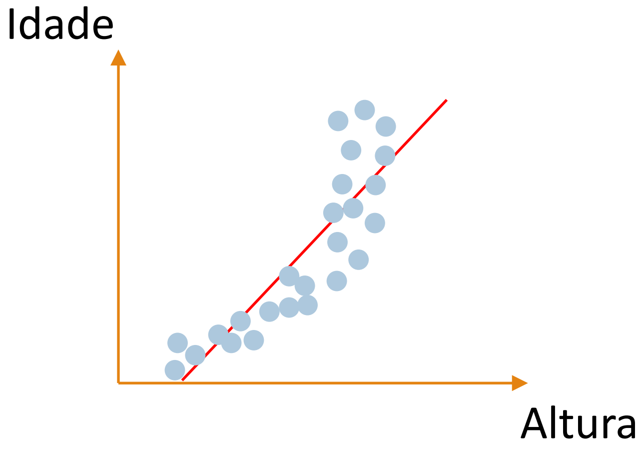
Uma possibilidade: relação linear

$$\text{Idade} = w_1 * \text{Altura} + w_0$$

- Precisamos encontrar w_0 e w_1 que forneça o melhor ajuste de linha reta para a relação entre idade e altura.

Regressão linear

Vamos supor que queremos prever a idade de uma pessoa com base em algumas de suas características



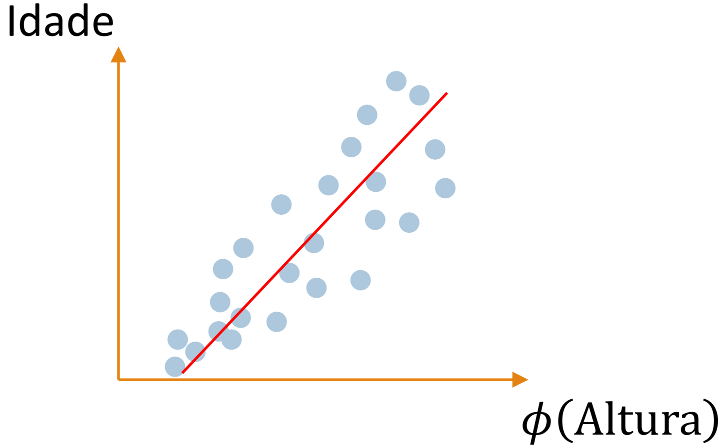
Uma possibilidade: relação linear

$$\text{Idade} = w_1 * \text{Altura} + w_0$$

- Precisamos encontrar w_0 e w_1 que forneça o melhor ajuste de linha reta para a relação entre idade e altura.
- Usamos regressão de mínimos quadrados para encontrar o melhor w_0e w_1!

Regressão linear

A relação não parece perfeitamente linear, portanto, podemos mapear o valor de altura para um novo valor $\phi(\text{Altura})$, o que, com sorte, tornará o relacionamento linear

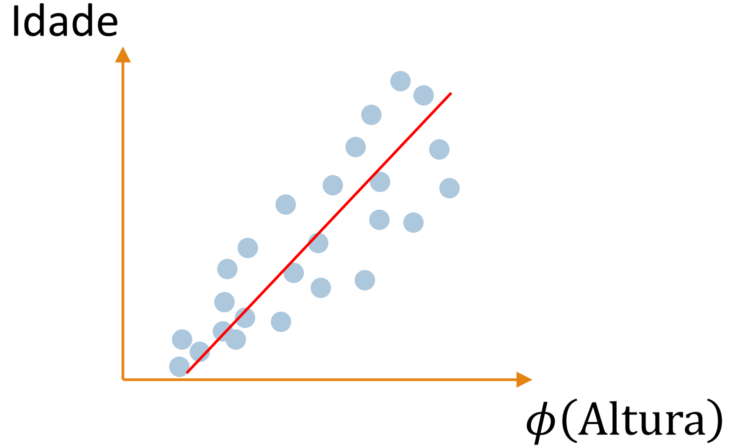


Por exemplo

$$\phi(\text{Altura}) = \text{Altura}^2$$

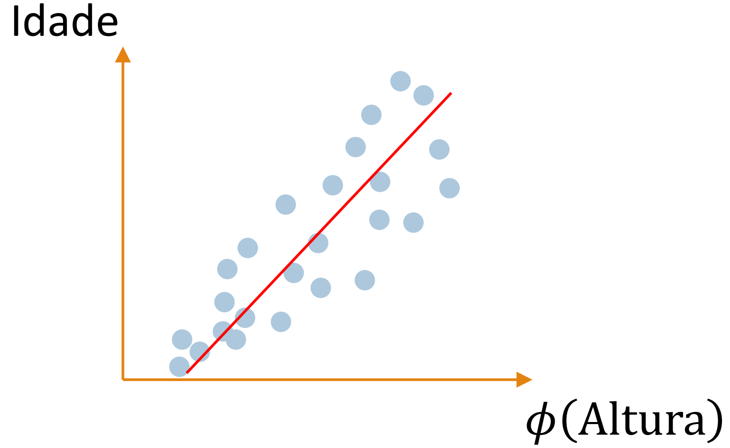
Regressão linear

Temos um ajuste melhor, mas ainda assim, a previsão não é muito boa. Para uma dada altura, a pessoa pode ter uma grande variedade de idades diferentes. Como podemos melhorar isso?



Regressão linear

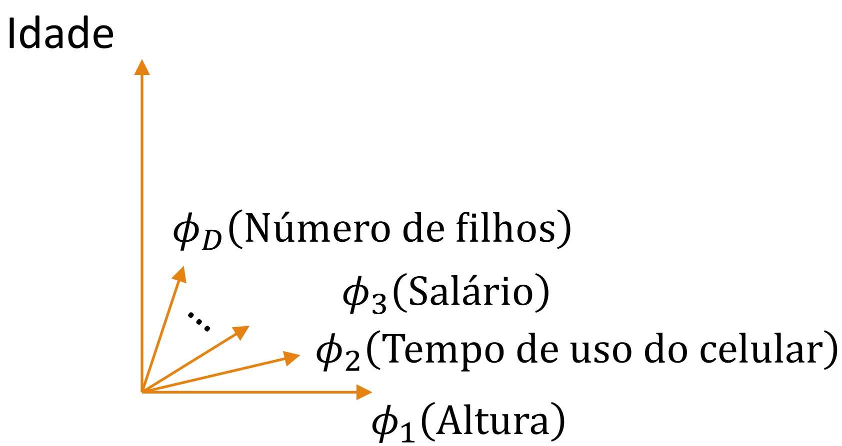
Temos um ajuste melhor, mas ainda assim, a previsão não é muito boa. Para uma dada altura, a pessoa pode ter uma grande variedade de idades diferentes. Como podemos melhorar isso?



Adicionar mais atributos!

Regressão linear

Temos um ajuste melhor, mas ainda assim, a previsão não é muito boa. Para uma dada altura, a pessoa pode ter uma grande variedade de idades diferentes. Como podemos melhorar isso?



$$\text{Idade} = w_1 * \phi_1(\text{Altura}) + w_2 * \phi_2(\text{Tempo de uso do celular}) + w_3 * \phi_3(\text{Salário}) + \cdots + \phi_D(\text{Número de filhos}) + w_0$$

Regressão linear

Então, nosso modelo é esse

$$\text{Idade} = w_0 + \sum_{i=1}^{D} w_i \phi_i(x_i)$$

Mas o modelo pode ser ainda mais flexível. Cada função ϕ_i pode depender não apenas de x_i, mas de todas as variáveis. Escrevemos $\phi_i(x_1, x_2, \ldots, x_D)$ como $\phi_i(\boldsymbol{s})$. Nosso modelo agora é

$$\text{Idade} = w_0 + \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{s})$$

* Observe que podemos definir $\phi_i(\boldsymbol{s}) = x_i$. Nesse caso, temos as características originais

Regressão linear

$$\text{Idade} = w_0 + \sum_{i=1}^{M} w_i \phi_i(\boldsymbol{s})$$

Podemos escrever o modelo de uma forma mais compacta

$$\text{Idade} = \boldsymbol{w\phi}$$

Onde $\boldsymbol{w} = (w_0, w_1, \dots, w_M)$, $\boldsymbol{\phi} = (\phi_0, \phi_1, \dots, \phi_M)$ e $\phi_0 = 1$.

Ainda podemos usar a regressão de mínimos quadrados para encontrar $w_0, w_1, \dots, w_M$!

Regressão logística

$$\text{Idade} = \boldsymbol{w\phi}$$

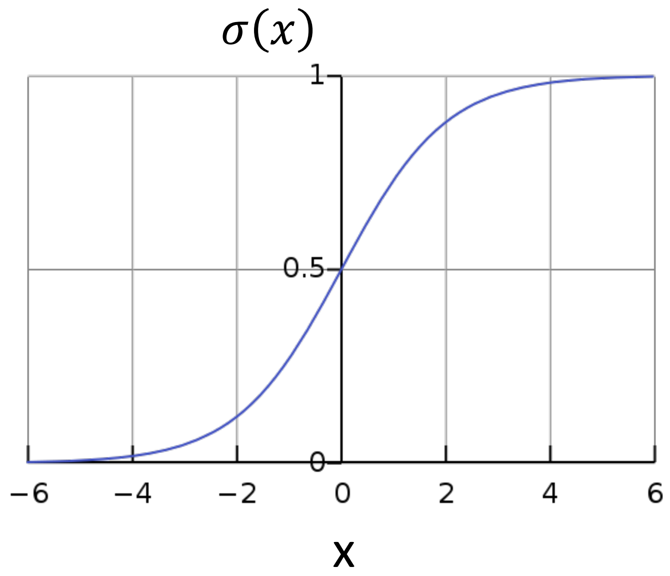
E se quisermos classificar uma pessoa em *jovem* ou *idosa* , dependendo de um conjunto de características?

Podemos utilizar a **regressão logística**

Regressão logística

Na regressão logística, mapeamos a combinação linear entre os atributos usando uma função não linear definida entre [0,1], denominada função logística.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Regressão logística

A vantagem desse mapeamento é que podemos relacionar o resultado da função logística à probabilidade de uma determinada pessoa $\boldsymbol{s}_i$ pertencer à classe jovem ou idosa:

$$P(\text{Idosa}|\boldsymbol{s}_i) = \sigma(\boldsymbol{w}\boldsymbol{\phi}(\boldsymbol{s}_i))$$

Regressão logística

A vantagem desse mapeamento é que podemos relacionar o resultado da função logística à probabilidade de uma determinada pessoa $\boldsymbol{s}_i$ pertencer à classe jovem ou idosa:

$$P(\text{Idosa}|\boldsymbol{s}_i) = \sigma(\boldsymbol{w}\boldsymbol{\phi}(\boldsymbol{s}_i))$$

$$P(\text{Jovem}|\boldsymbol{s}_i) = 1 - P(\text{Idosa}|\boldsymbol{s}_i)$$

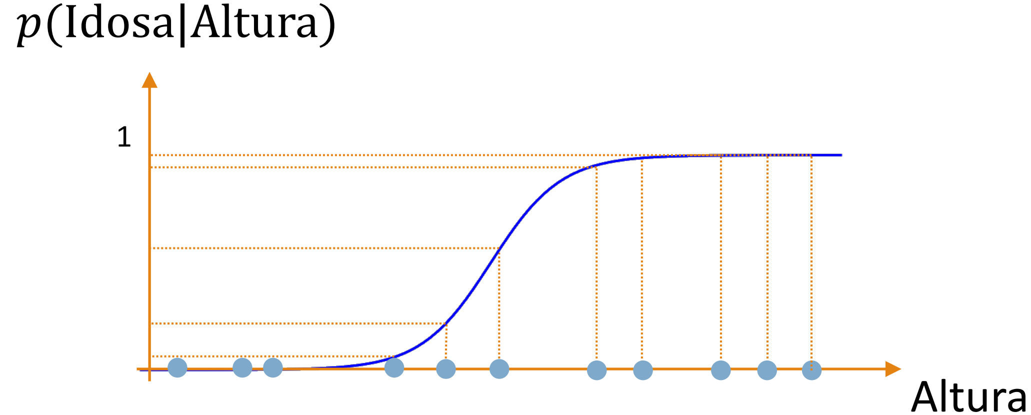
Regressão logística

A vantagem desse mapeamento é que podemos relacionar o resultado da função logística à probabilidade de uma determinada pessoa $\boldsymbol{s}_i$ pertencer à classe jovem ou idosa:

$$P(\text{Idosa}|\boldsymbol{s}_i) = \sigma(\boldsymbol{w}\boldsymbol{\phi}(\boldsymbol{s}_i))$$

$$P(\text{Jovem}|\boldsymbol{s}_i) = 1 - P(\text{Idosa}|\boldsymbol{s}_i)$$

Por exemplo, quando temos apenas uma variável



Regressão logística

$$P(\text{Idosa}|\boldsymbol{s}_i) = \sigma(\boldsymbol{w}\boldsymbol{\phi}(\boldsymbol{s}_i))$$

Portanto, podemos definir o seguinte critério de decisão:

Se $\sigma\left(\boldsymbol{w}\boldsymbol{\phi}(\boldsymbol{s}_i)\right) > 0.5$, a pessoa é idosa
Caso contrário, a pessoa é jovem

Regressão logística

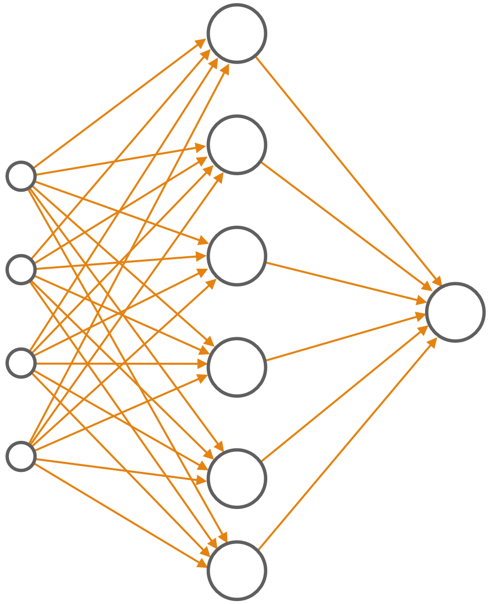
- Treinamos a regressão logística encontrando os parâmetros $\boldsymbol{w}$ que maximizam a função de verossimilhança do nosso modelo, utilizando os dados que temos sobre um conjunto de pessoas.
- Observe que o treinamento **não maximiza a precisão.**

Regressão logística

- E as funções de mapeamento $\phi_i(\boldsymbol{s})$?
- Até agora, assumimos que funções de mapeamento apropriadas são conhecidas
- Mas geralmente elas não são conhecidas

Perceptron Multicamadas

- Provavelmente o modelo de rede neural mais tradicional
- Também chamada de rede neural feed-forward, pois não há loops

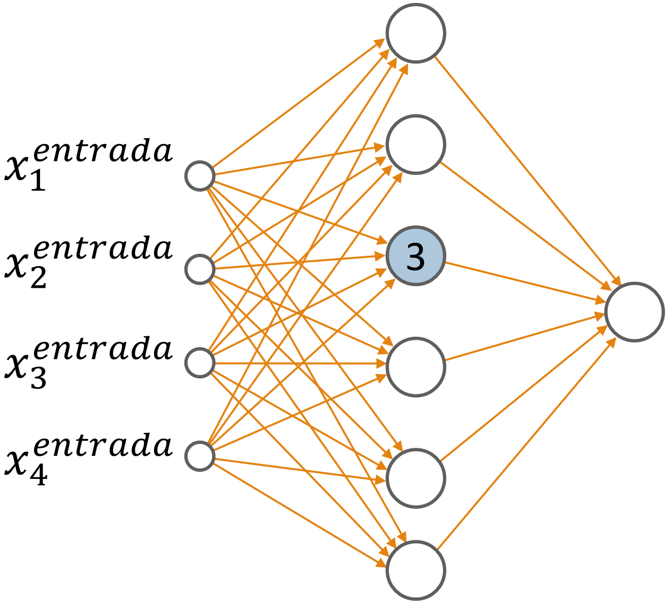


Camada
de entrada
(propriedades)

Camada
escondida

Camada de saída
(probabilidade
jovem/idosa)

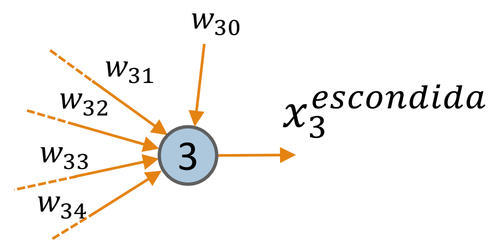
Perceptron Multicamadas



Camada de entrada (propriedades)

Camada escondida

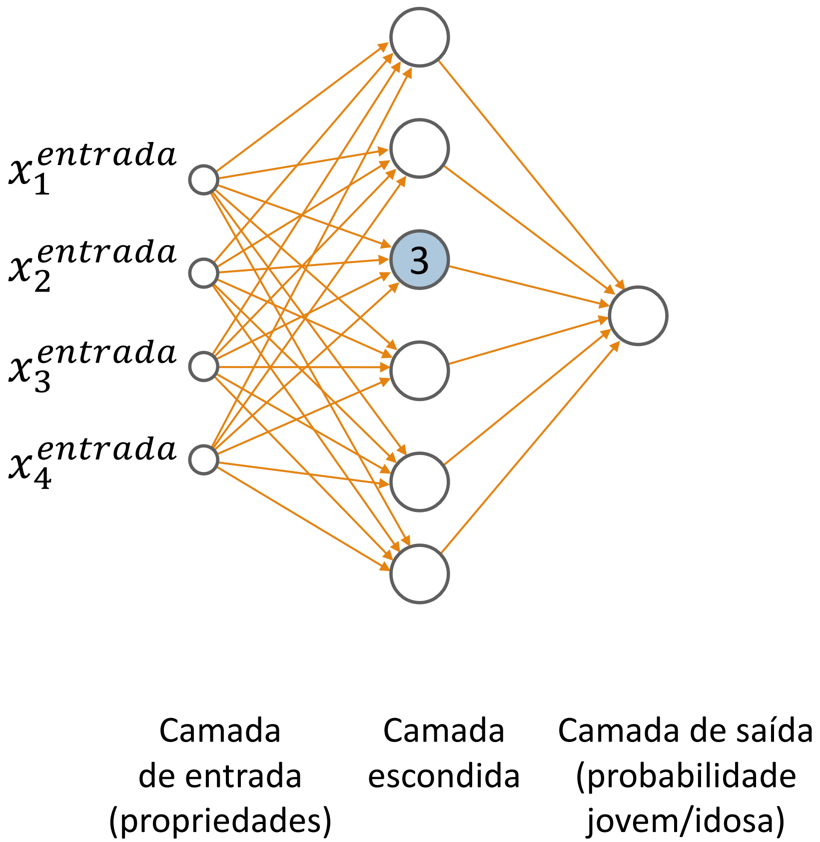
Camada de saída (probabilidade jovem/idosa)

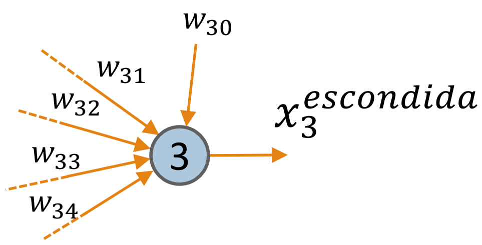


Para um dado neurônio j na camada escondida:

$$x_j^{escondida} = S\left(w_{j0} + \sum_{i=1}^{4} w_{ji} x_i^{entrada}\right)$$

Perceptron Multicamadas





Para um dado neurônio j na camada escondida:

$$x_j^{escondida} = S\left(w_{j0} + \sum_{i=1}^{4} w_{ji} x_i^{entrada}\right)$$

Ativação de um neurônio de saída :

$$P(\text{Idosa}) = S\left(w_{o0} + \sum_{j=1}^{6} w'_{oj} x_j^{escondida}\right)$$

Perceptron Multicamadas

- Efetivamente, estamos definindo novas variáveis $x_j^{escondida}$, que são uma função das variáveis de entrada
- Isto é, $x_j^{escondida} = \phi_j(\boldsymbol{x}^{entrada})$

$$x_j^{escondida} = S\left(w_{j0} + \sum_{i=1}^{4} w_{ji} x_i^{entrada}\right)$$

$$P(\text{Idosa}) = S\left(w_{o0} + \sum_{j=1}^{6} w'_{oj} x_j^{escondida}\right)$$

Perceptron Multicamadas

- Efetivamente, estamos definindo novas variáveis $x_j^{escondida}$, que são uma função das variáveis de entrada
- Isto é, $x_j^{escondida} = \phi_j\left(\boldsymbol{x}^{entrada}\right)$
- Assim, estamos fazendo um mapeamento não linear das variáveis de entrada em, idealmente, variáveis mais apropriadas
- Então, $P(\text{Idosa})$ é dado por uma combinação linear das variáveis ocultas, mapeadas no intervalo [0,1] pela função logística

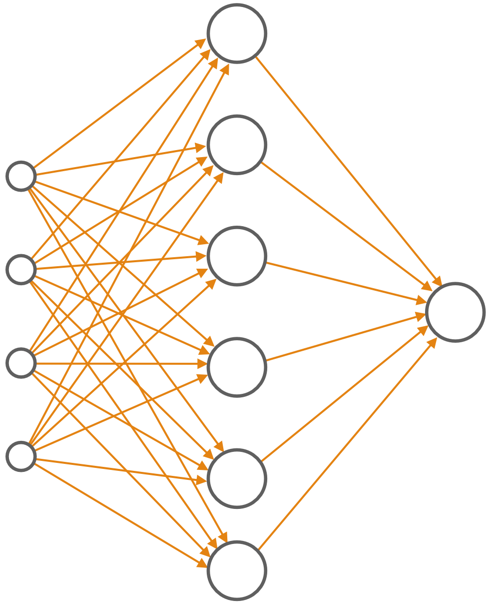
$$x_j^{escondida} = S\left(w_{j0} + \sum_{i=1}^{4} w_{ji} x_i^{entrada}\right)$$

* Se quisermos fazer regressão em vez de classificação, removemos S() da saída

$$P(\text{Idosa}) = S\left(w_{o0} + \sum_{j=1}^{6} w'_{oj} x_j^{escondida}\right)$$

Perceptron Multicamadas

Observe que temos que ajustar 37 parâmetros!



Camada de entrada (propriedades)

Camada escondida

Camada de saída (probabilidade jovem/idoso)

$$x_j^{escondida} = S\left(w_{j0} + \sum_{i=1}^{4} w_{ji} x_i^{entrada}\right)$$

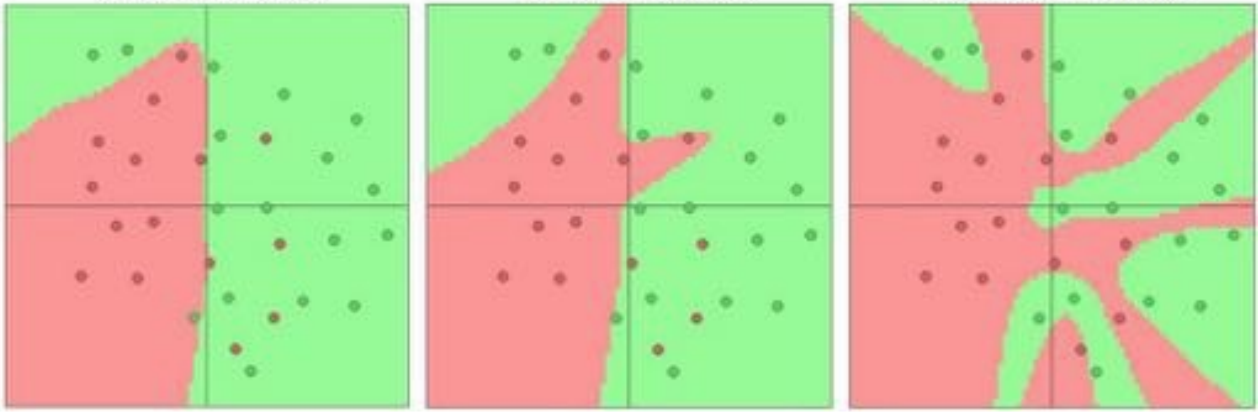
$$P(\text{Idosa}) = S\left(w_{o0} + \sum_{j=1}^{6} w'_{oj} x_j^{escondida}\right)$$

Perceptron Multicamadas

O número de camadas ocultas e o número de neurônios nessas camadas está associado à complexidade do modelo representado pela rede neural.

Menos neurônios

Mais neurônios



Perceptron Multicamadas - Teorema da Aproximação Universal

Um resultado importante sobre redes neurais é o teorema da aproximação universal, que afirma:

Uma rede feedforward com uma única camada oculta pode aproximar todas* as funções contínuas.

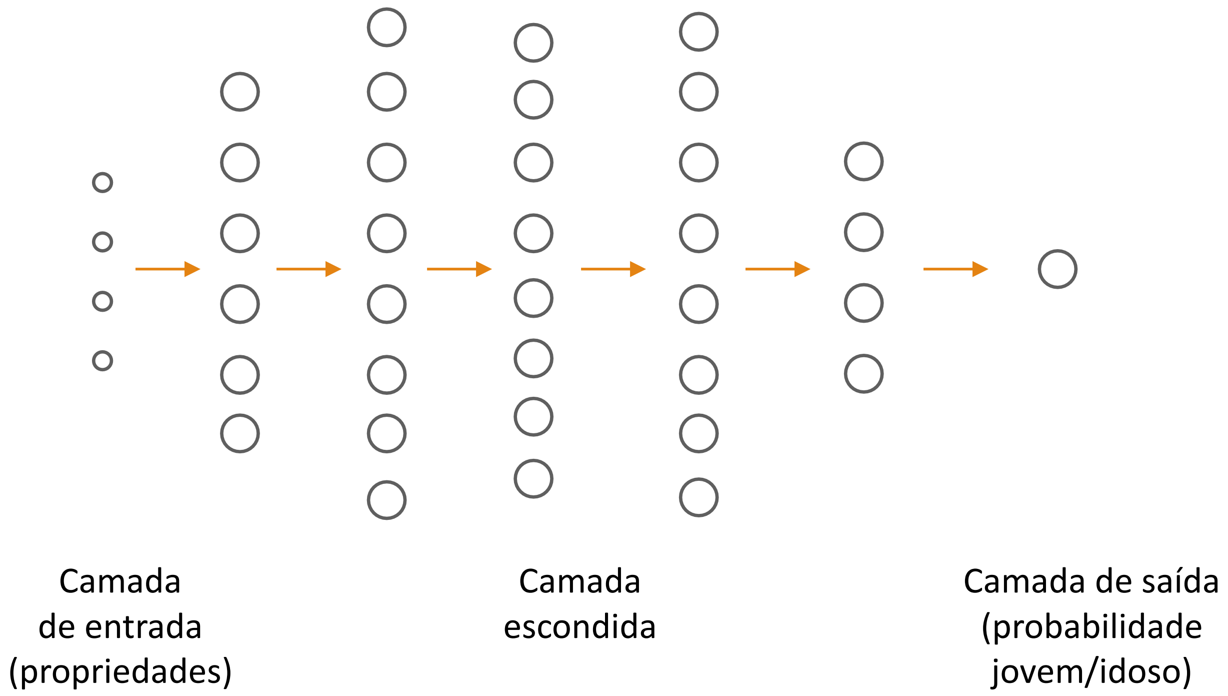
Isso significa que qualquer função da forma

$$y = f(\boldsymbol{x})$$

pode ser representada como uma rede neural.

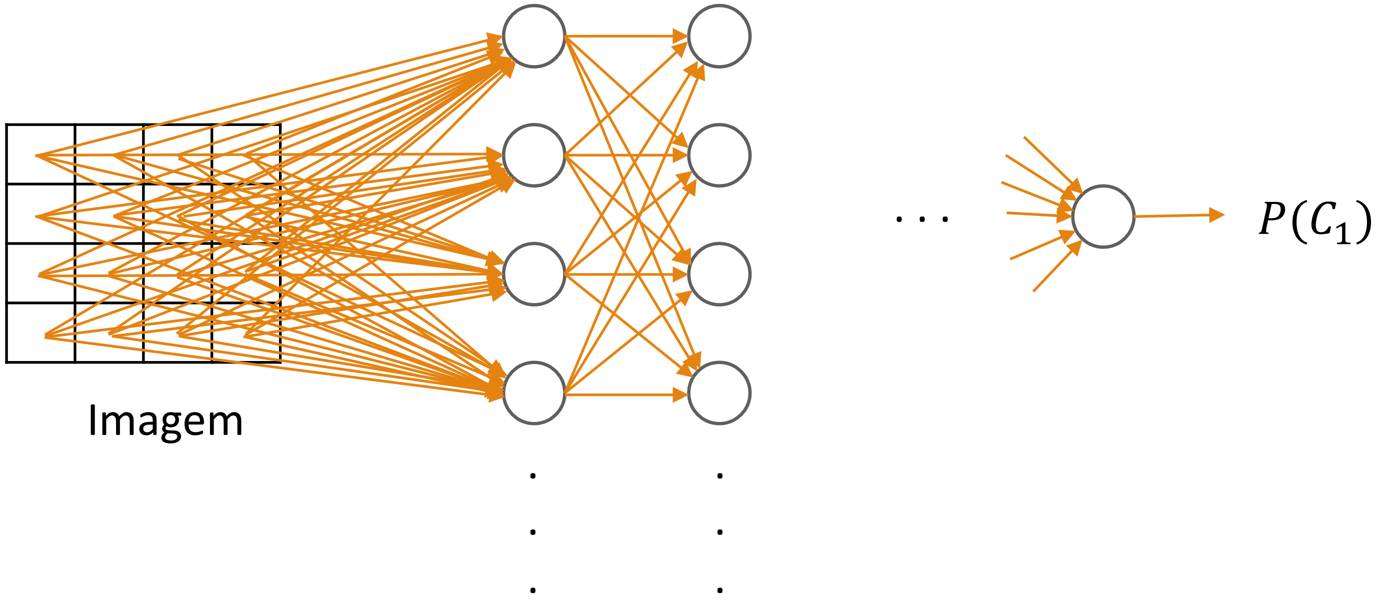
Aprendizado profundo

- Um perceptron multicamadas com muitas camadas pode ser chamado de rede profunda. O processo de treinamento de uma rede profunda é chamado de Deep Learning.



Aprendizagem profunda em imagens

- Podemos aplicar uma rede neural diretamente em uma imagem!

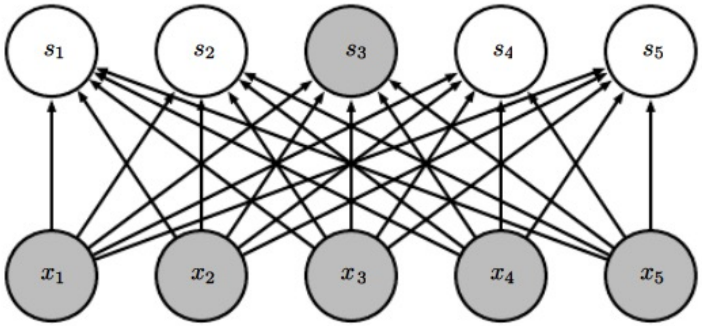


Problema: explosão do número de parâmetros que precisam ser ajustados!

Solução: redes neurais convolucionais

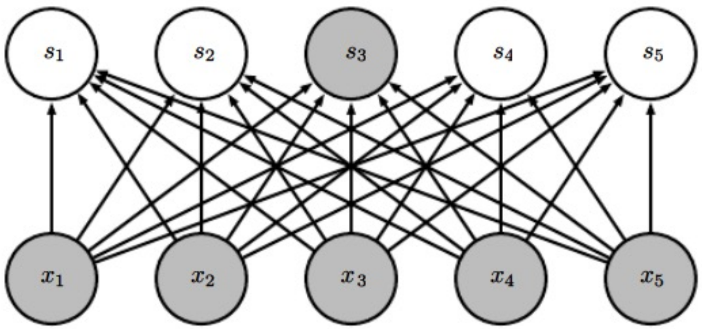
Redes neurais convolucionais

Camadas totalmente conectadas

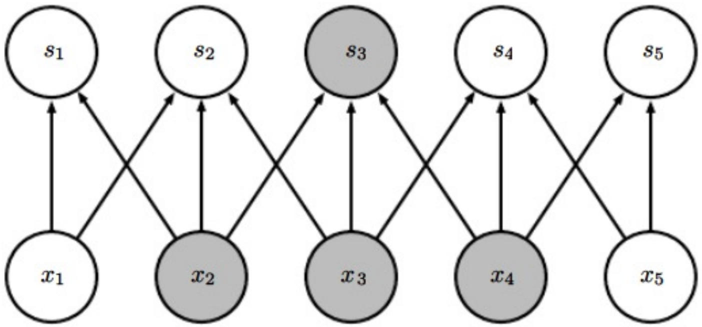


Redes neurais convolucionais

Camadas totalmente conectadas

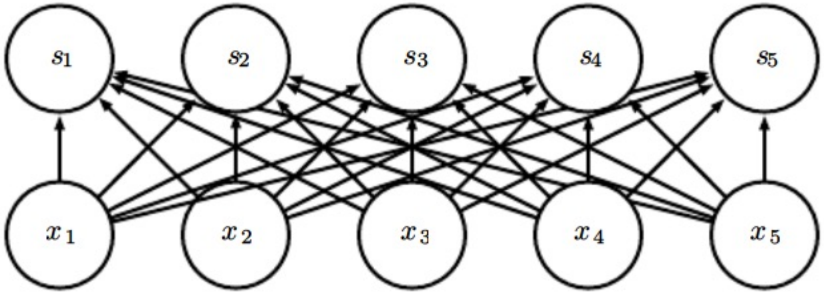


Camadas localmente conectadas

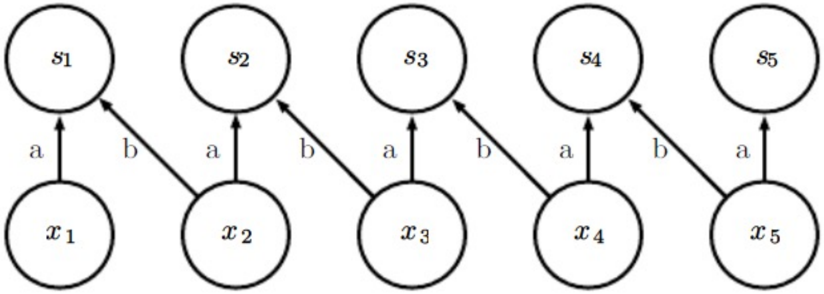


Redes neurais convolucionais

Camadas totalmente conectadas

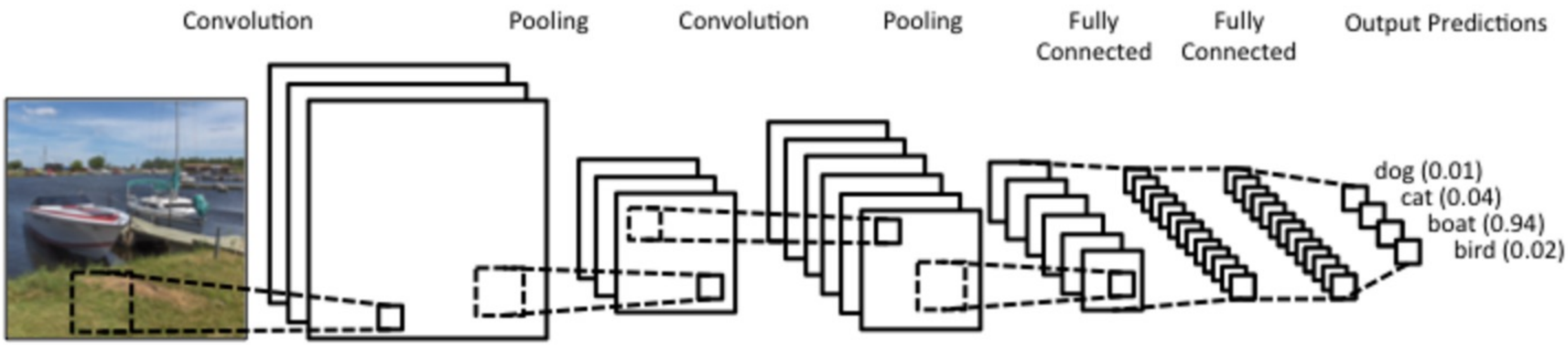


Camadas localmente conectadas com compartilhamento de parâmetros (convolução)



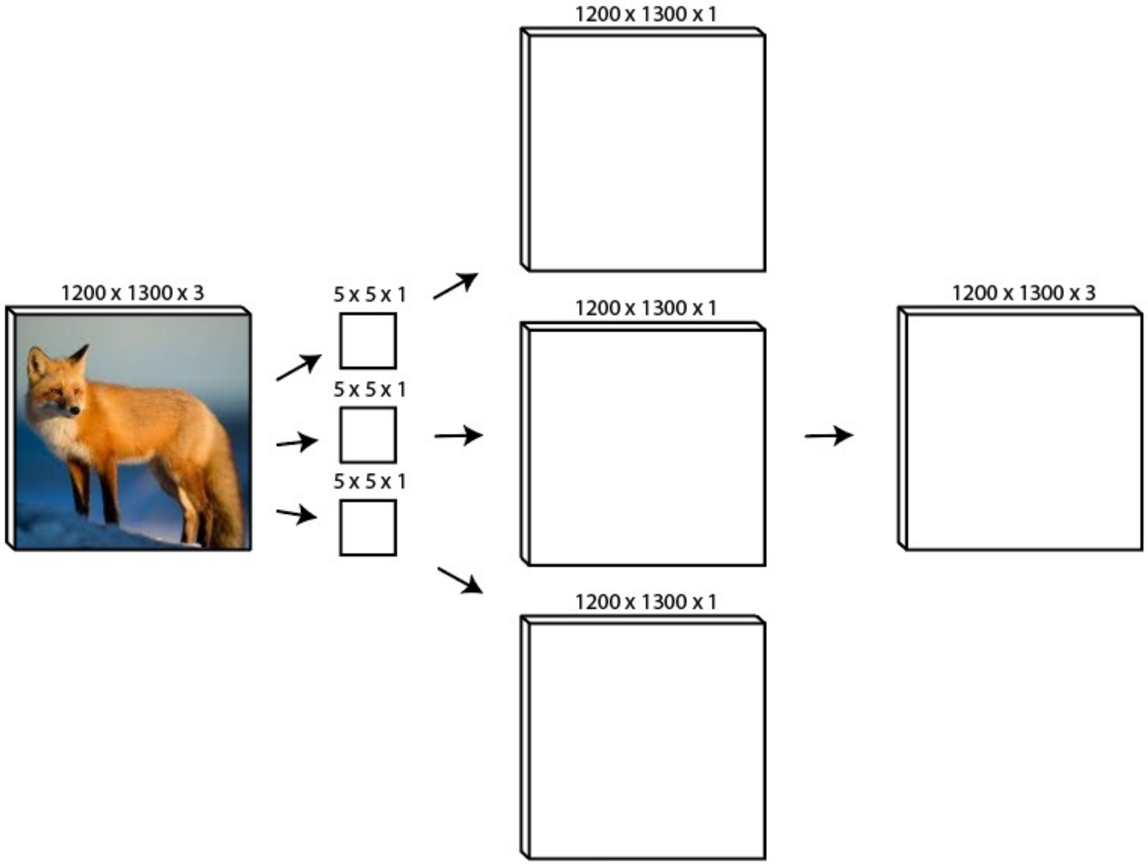
Redes neurais convolucionais (CNN)

Arquitetura típica de uma CNN:



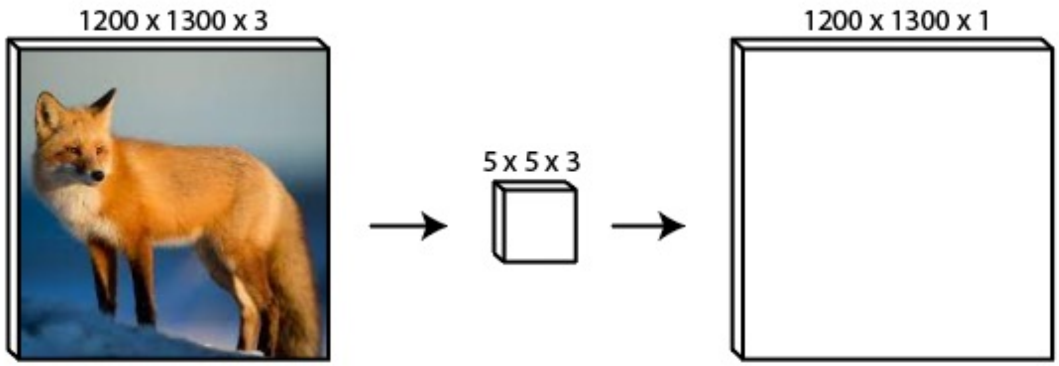
Redes neurais convolucionais (CNN)

Convolução de uma imagem colorida utilizando 3 filtros:



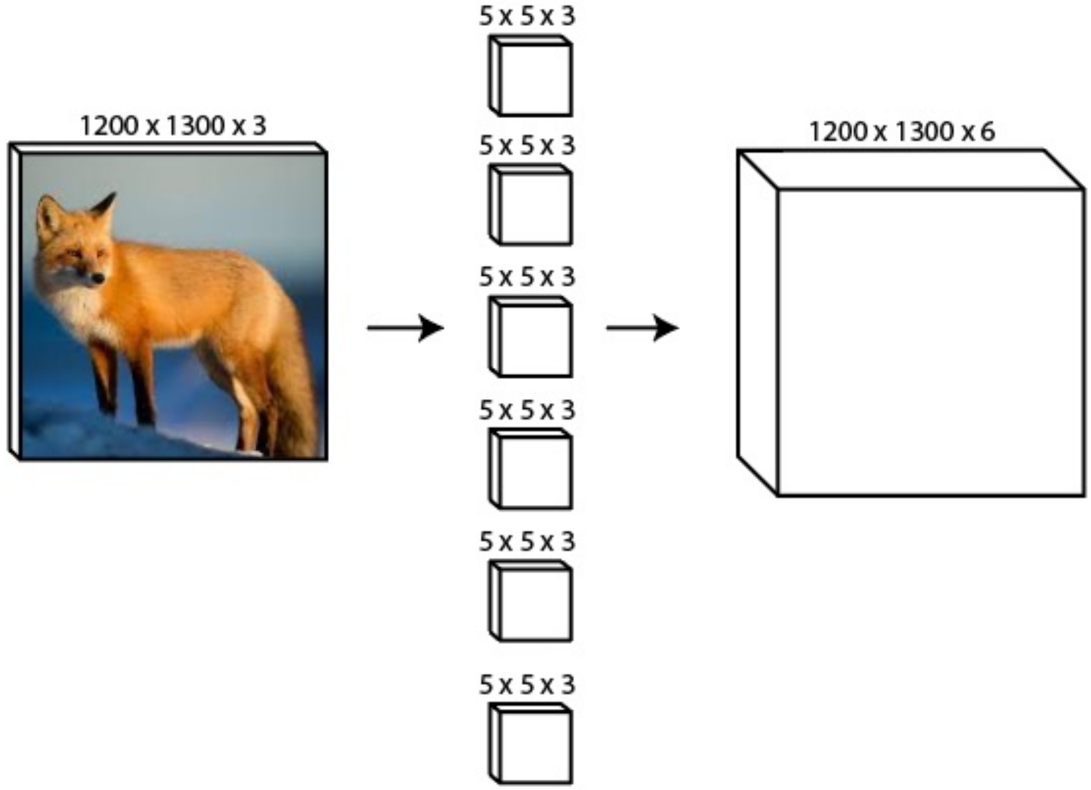
Redes neurais convolucionais (CNN)

- A convolução realizada por uma CNN é um pouco distinta
- O filtro também possui três canais, e o resultado é uma imagem de 1 canal



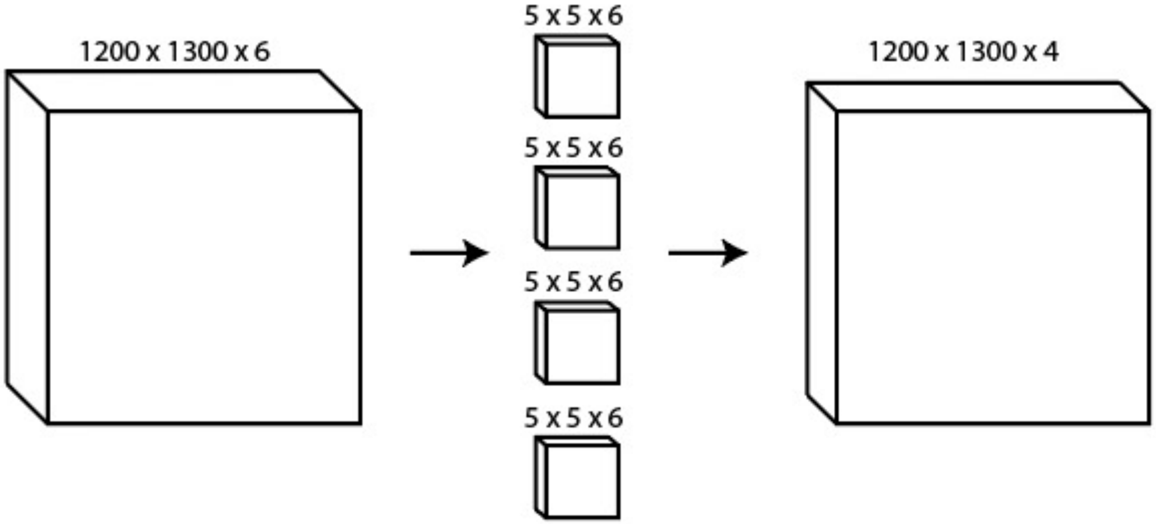
Redes neurais convolucionais (CNN)

Podemos convoluir com diversos filtros, e obter várias imagens de resultado



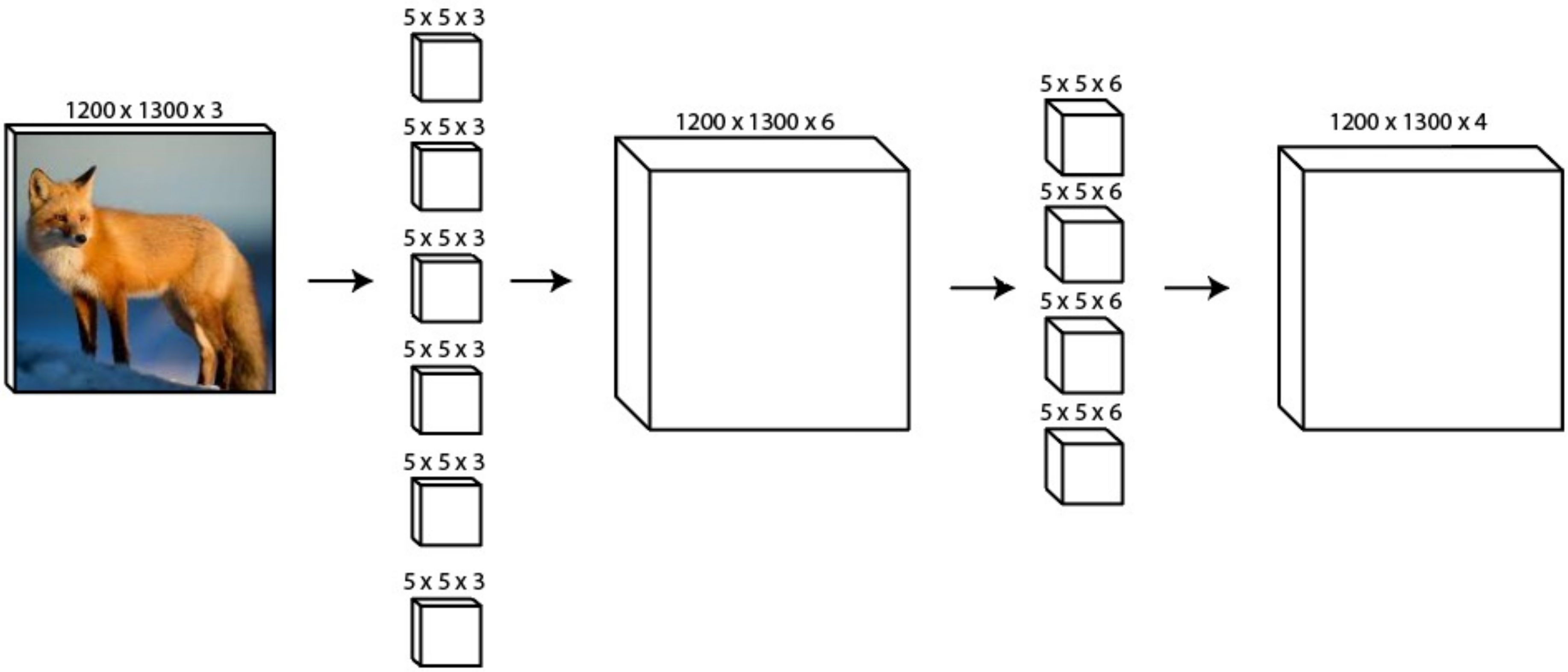
Redes neurais convolucionais (CNN)

- Novos filtros são aplicados ao resultado da camada anterior
- Note que os filtros possuem um número de canais igual ao número de canais do resultado anterior



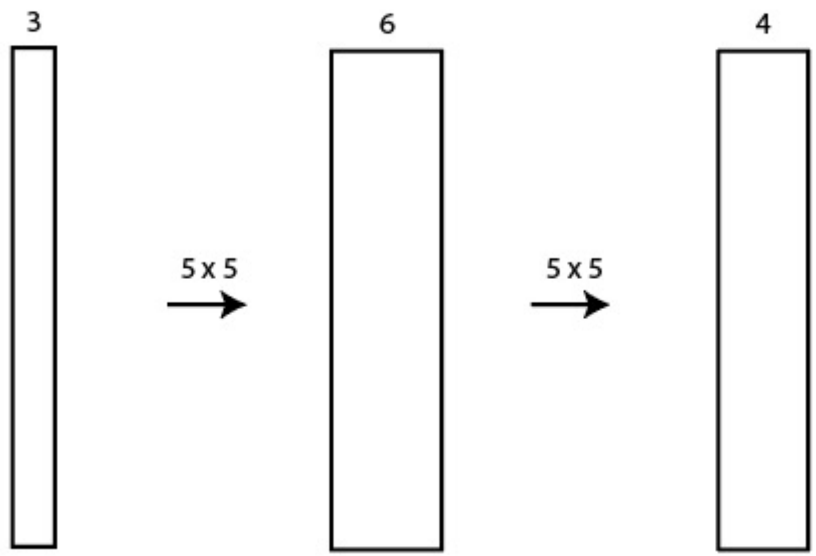
Redes neurais convolucionais (CNN)

Usualmente, uma rede é representada visualmente de forma bem enxuta…



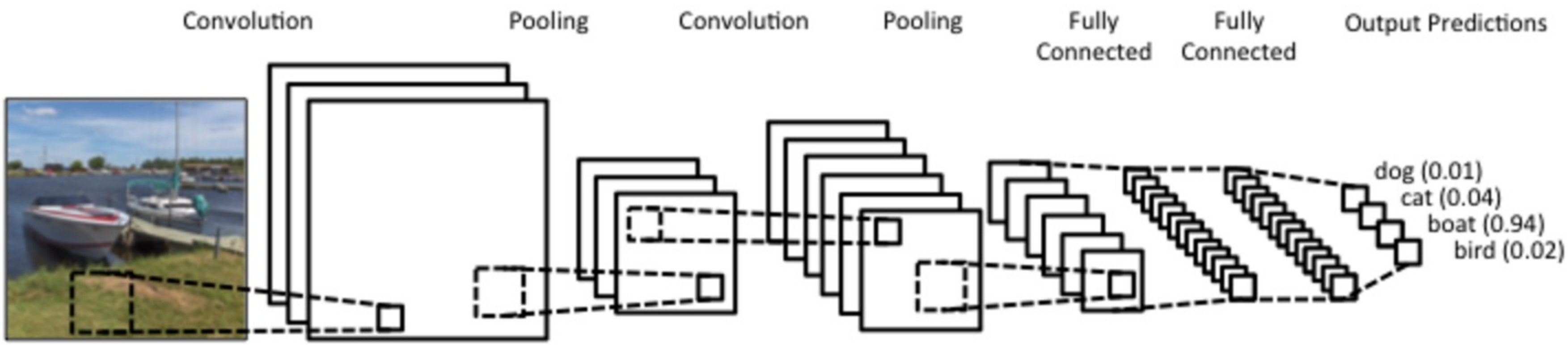
Redes neurais convolucionais (CNN)

…como retângulos que indicam apenas o número de canais utilizados em cada camada, e o tamanho dos filtros.

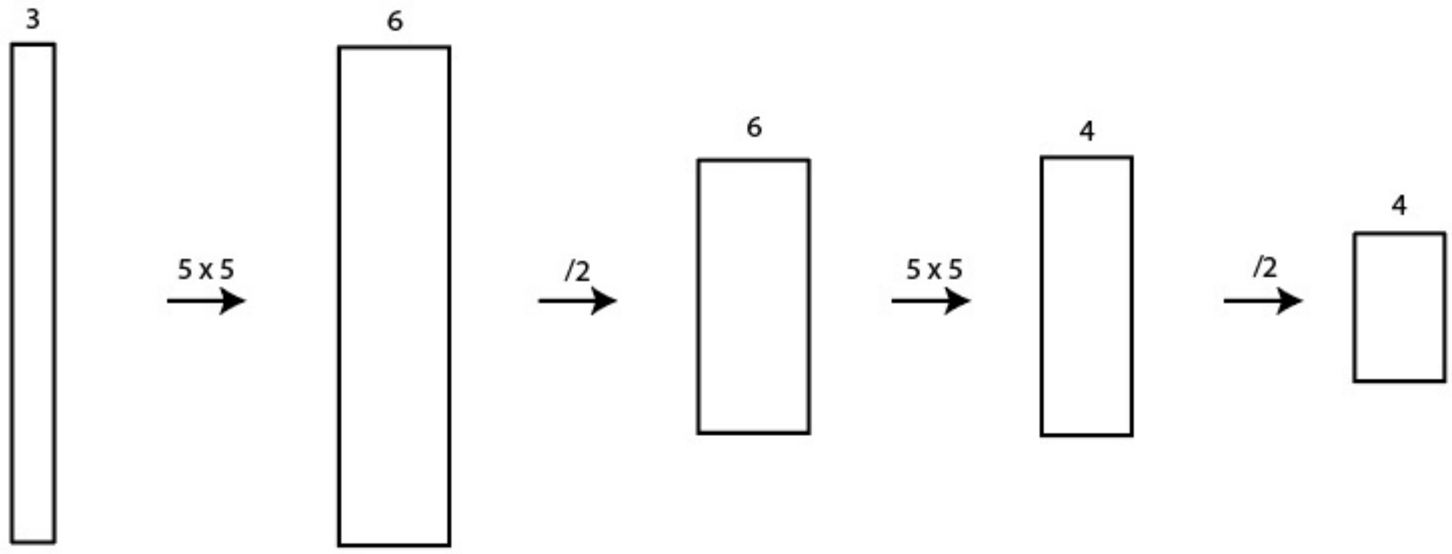


Redes neurais convolucionais (CNN)

Arquitetura típica de uma CNN:



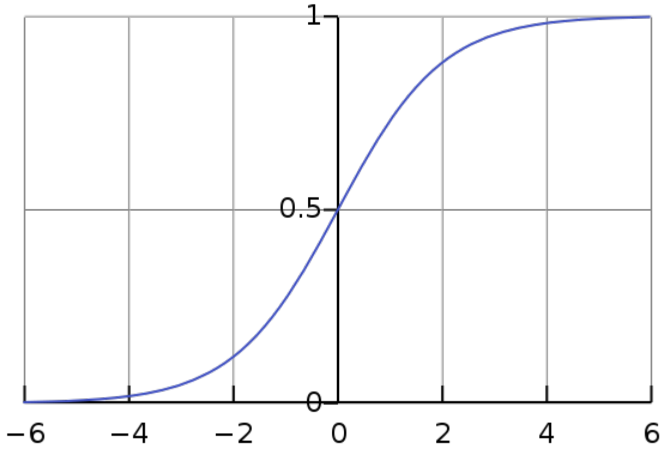
Redes neurais convolucionais (CNN)



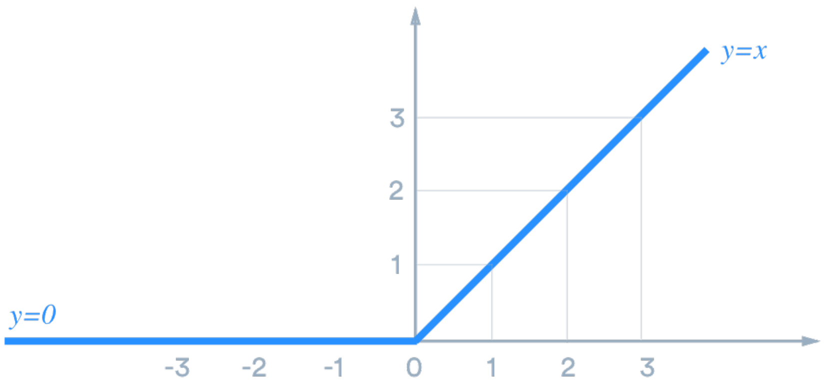
Redes neurais convolucionais

Percebeu-se que ReLu funciona melhor que sigmoide

sigmoide



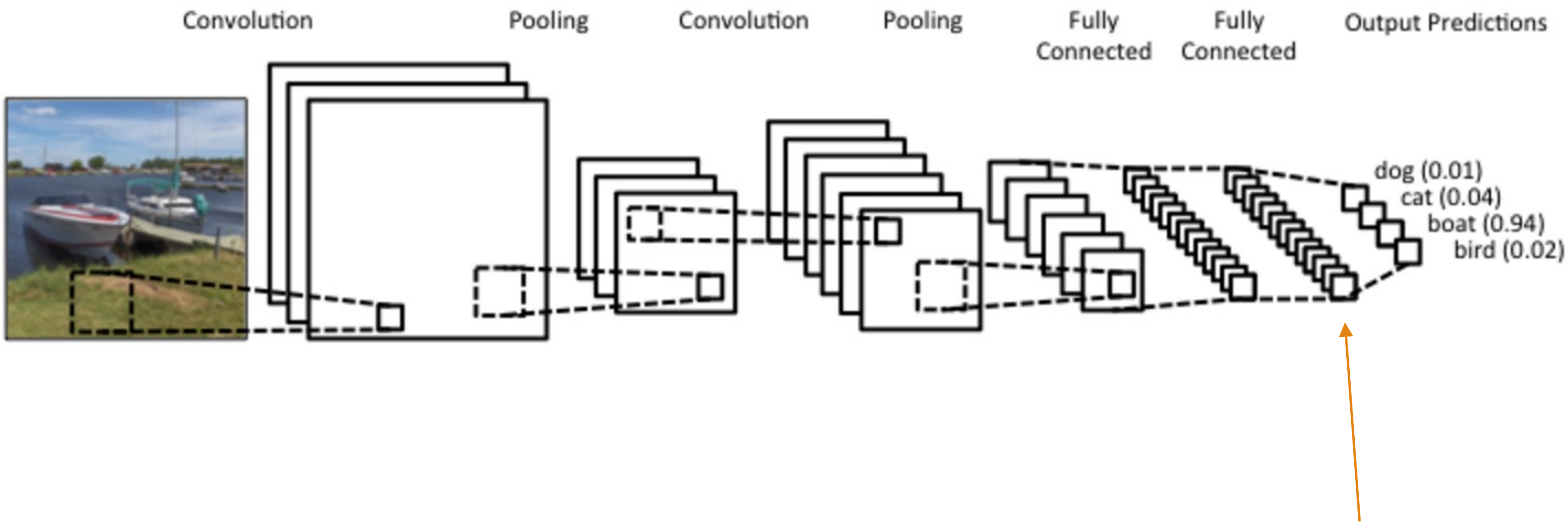
Unidade linear retificada (ReLu)



Redes neurais convolucionais

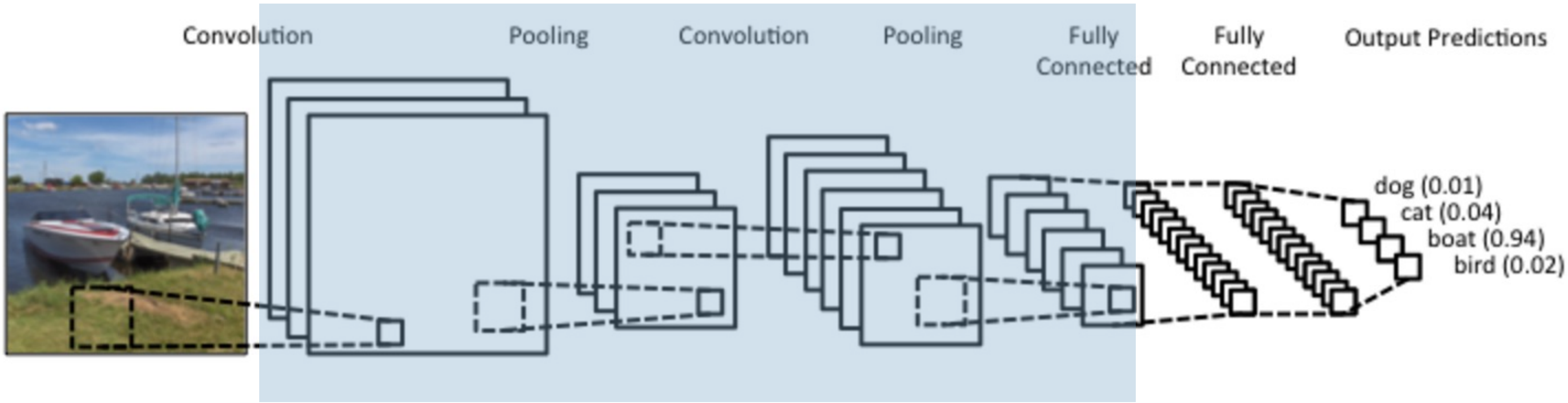
- Treinar uma pequena rede neural convolucional do zero não é difícil
- Treinar uma arquitetura de rede estado da arte é mais complicado
- Usualmente é utilizado aprendizado por transferência

Transferência de aprendizagem



Esta camada contém atributos que são muito bons para classificar imagens

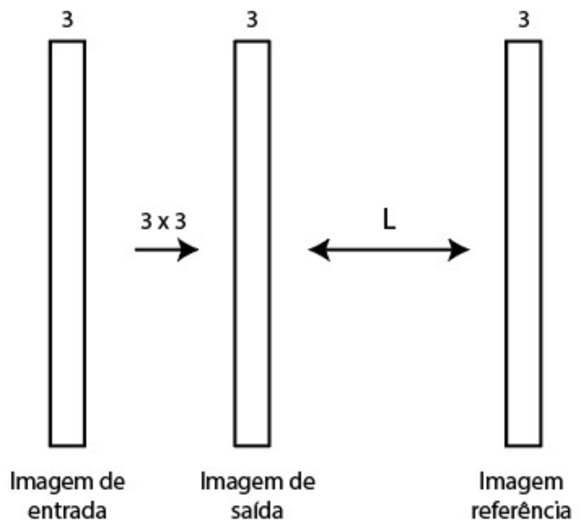
Transferência de aprendizagem



Congelamos as camadas profundas da rede e treinamos apenas as últimas camadas em novas imagens

Treinando a rede

- Para treinar uma rede neural utilizamos gradiente descendente para minimizar uma métrica de qualidade
- Vamos supor que temos uma rede simples composta por apenas 3 filtros de tamanho 3x3
- Nosso objeto é que a saída da rede seja igual a uma outra imagem, ou seja, queremos encontrar filtros que transformem uma imagem em outra



Treinando a rede

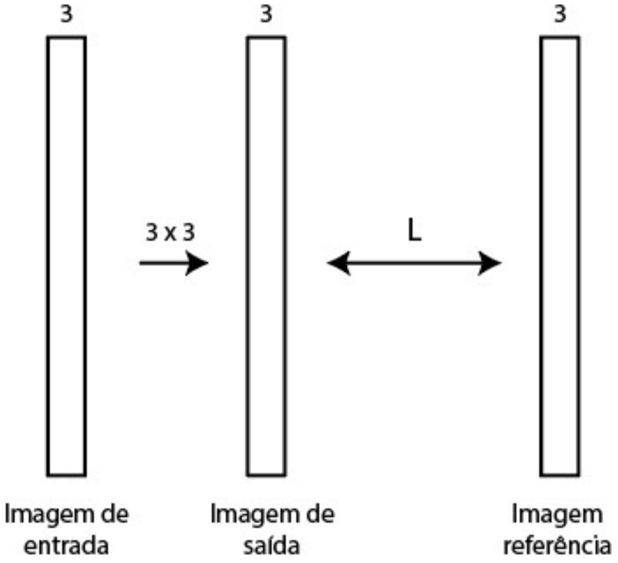
- L é uma função, chamada de função de perda, que quantifica a similaridade entre duas imagens;
- Por exemplo, o erro quadrático médio:

$$L = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_s(i,j) - I_r(i,j)\right)^2$$

onde I_s é a saída da rede e I_r a imagem de referência;

- L depende de todos o pesos da rede:

$$L(w_1, w_2, \ldots, w_{81})$$



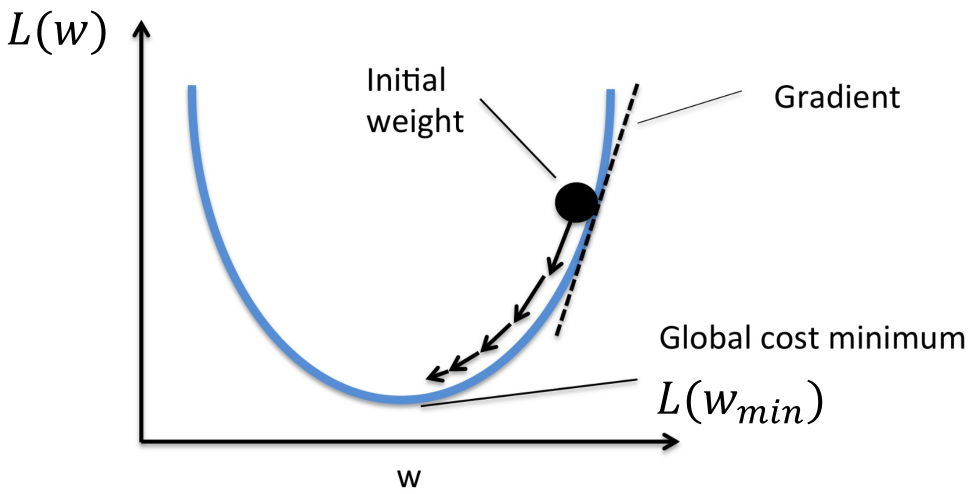
Gradiente descendente

- Vamos analisar o caso no qual a perda depende apenas de um único peso

$$L(w)$$

- Uma boa estratégia é atualizar o peso como

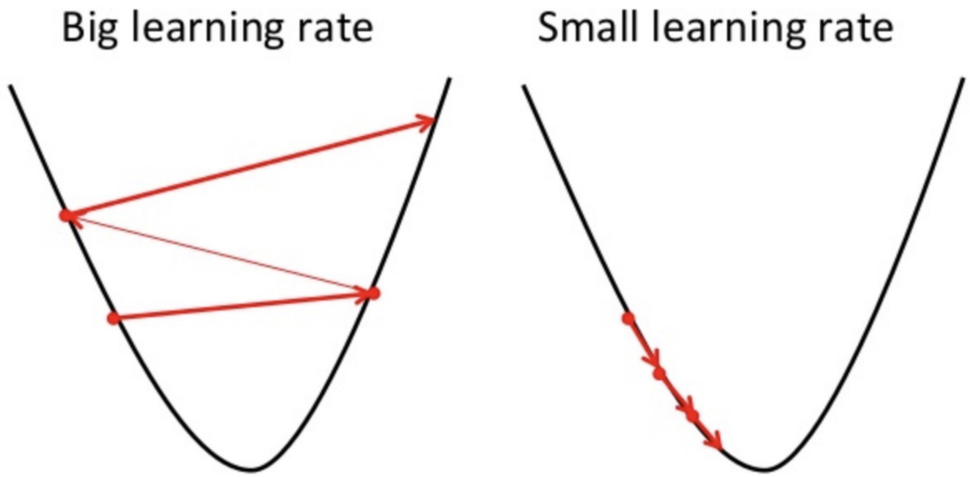
$$w_{t+1} = w_t - \alpha \frac{dL}{dw}$$



Gradiente descendente

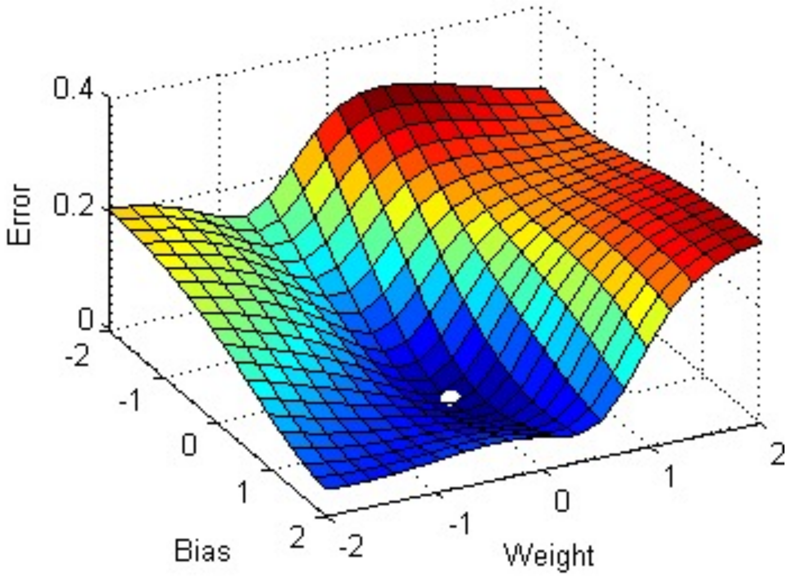
$$w_{t+1} = w_t - \alpha \frac{dL}{dw}$$

α é a taxa de aprendizado



Gradiente descendente

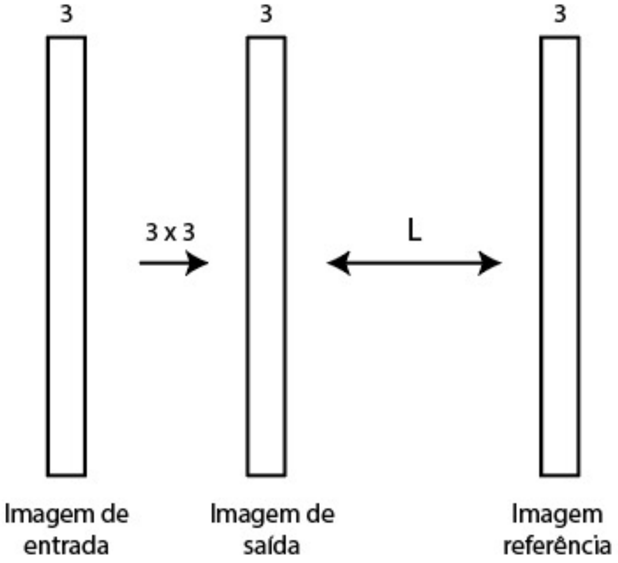
Se L dependesse de duas variáveis :



Treinando a rede

Como a nossa rede possui 81 parâmetros:

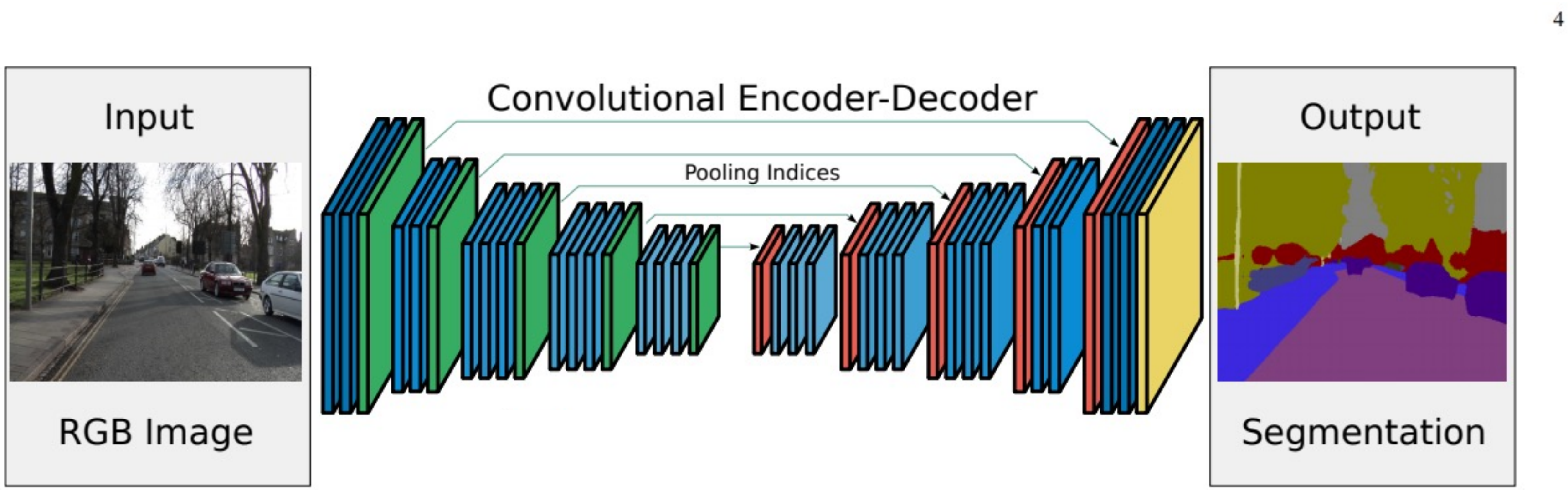
$$w_i^{t+1} = w_i^t - \alpha \frac{dL}{dw_i}$$



Treinando uma rede neural usando o PyTorch

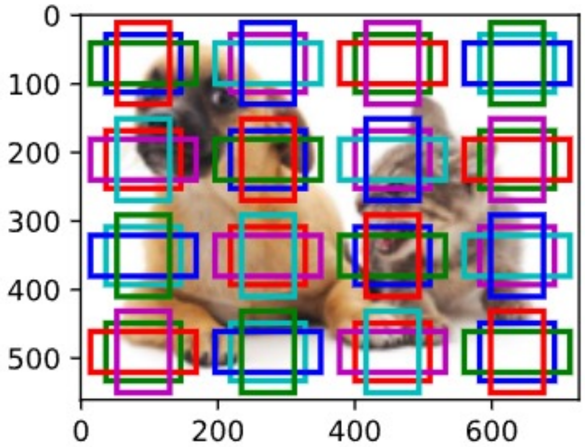
Notebooks “**Otimização de filtros**”

Segmentação de imagens



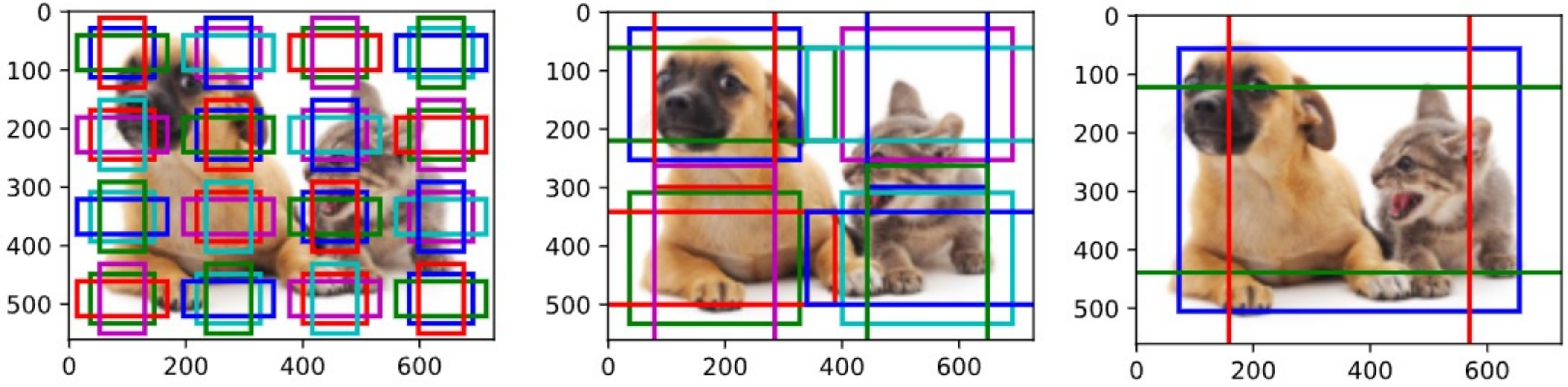
Detecção de objetos

Várias janelas são criadas em toda a imagem



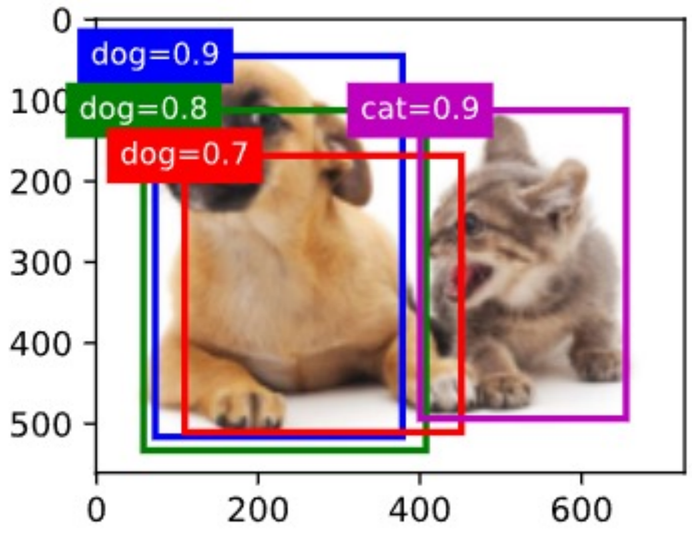
Detecção de objetos

Várias janelas são criadas em toda a imagem



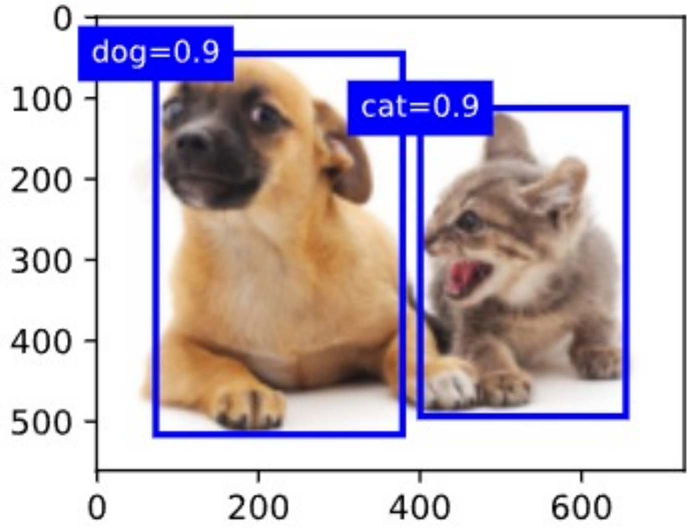
Detecção de objetos

As janelas com alta probabilidade de conter objetos são armazenadas



Detecção de objetos

Janelas com menor probabilidade são removidas



Classificando e segmenatndo imagens usando o PyTorch

Notebooks “**Classificação de imagens usando redes neurais**” e “**Detecção de objetos e segmentação**”