Neural Networks

Saurabh Mathur

Machine Learning

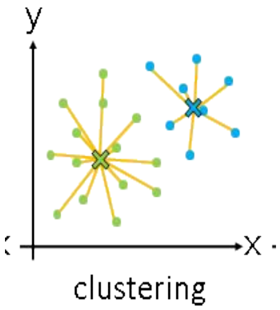
Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort.

- Arthur Samuel

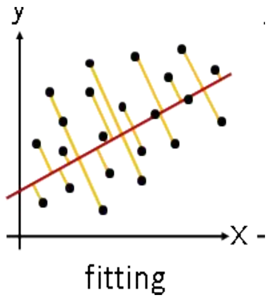
Machine learning research seeks to develop computer systems that auto matically improve their performance through experience.

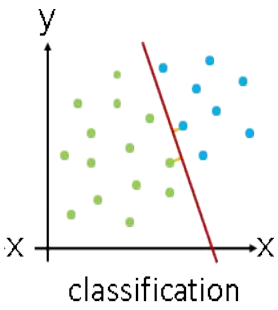
- Tom Mitchell

Problem formulation



Task	Experience	Performance
Predict house prices	House attributes and prices	Closeness to correct price (RMSE)
Grouping tweets together	Text content of tweets	Coherent groupings (Within group similarity)
Classify new images as either Cat or Dog	Images of cats & dogs	Accuracy of classification, Cross-entropy



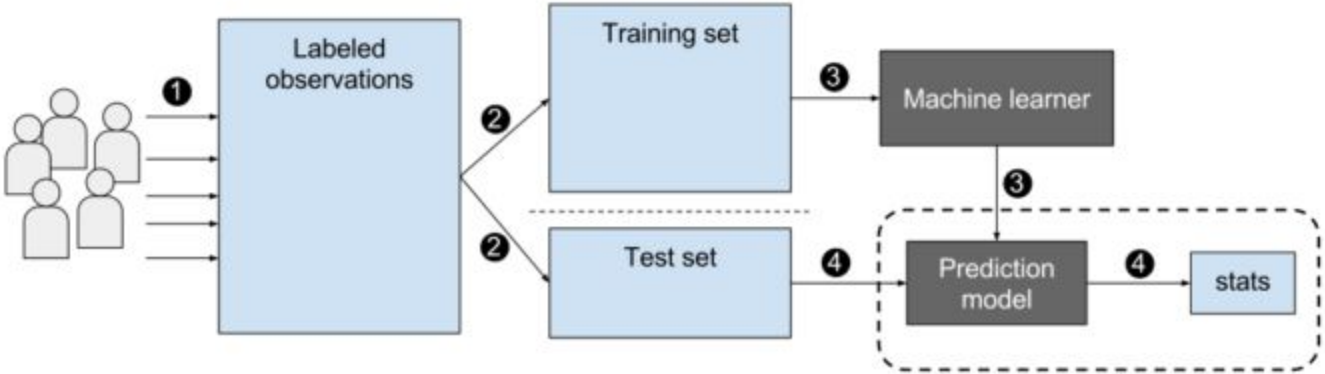


Source: https://factionsandfigures.netlify.com/2017/06/07/machine-learning/

Machine Learning Tasks

- Supervised Learning (We have examples and labels)
  - Classification
  - Regression
- Unsupervised Learning (We only have examples)
  - Topic modeling
  - Clustering
- Semi-Supervised Learning (Some examples have labels)

Supervised Learning



Source: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

Supervised learning

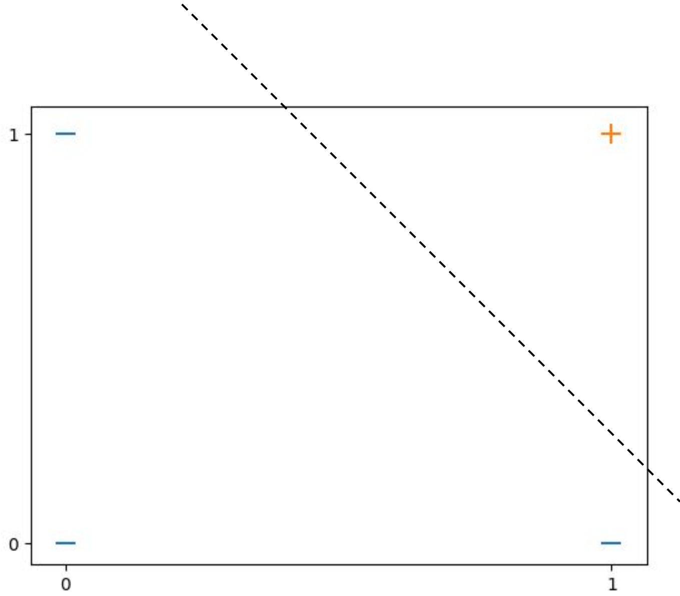
- Model
  - A simplified view of the world.
  - Encodes assumptions about the problem.
  - How was the data generated ?
- Learning Algorithm
  - Estimates specifics of the model from data.
- Measure of correctness
  - Loss Function
  - Log Likelihood

Example: Classification

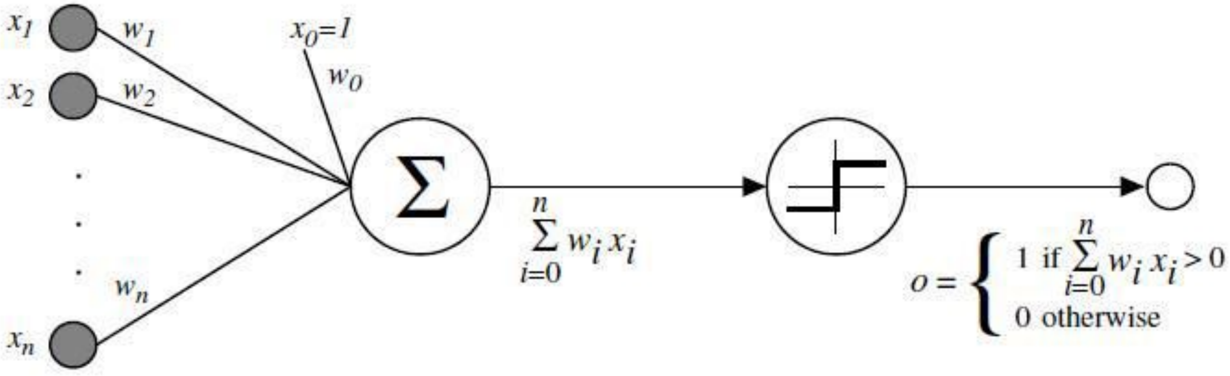
x_1	x_2	**y** (AND function)
0	0	-1
0	1	-1
1	0	-1
1	1	+1

Example: Classification

- Model
  - A line separates the examples : y = ax + b
  - Examples above the line are + others are -
- Measure of correctness
  - Accuracy = #correct / total
  - Error = sum(prediction - correct)
- Learning Algorithm ?



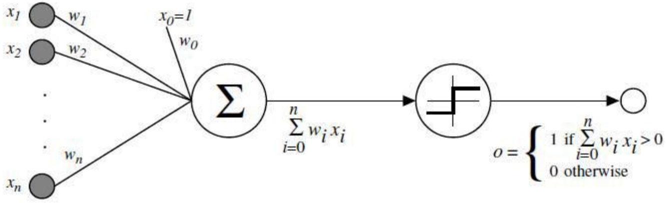
The Perceptron



Source: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

The Learning Algorithm

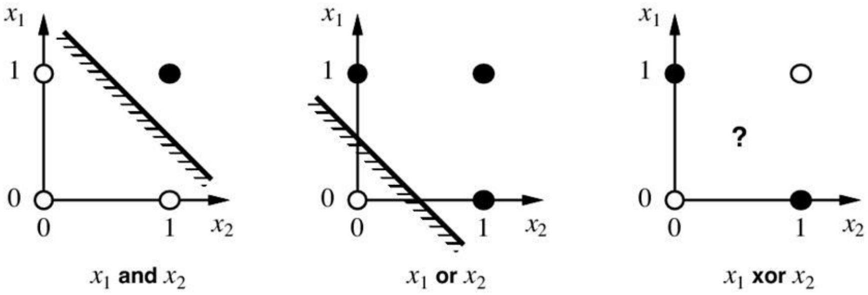
1. Set w to [0, 0, .., 0]
2. For each example x and output y in dataset:
    2.1. Compute prediction, p = Sign(w . x)
    2.2. If p ≠ y
        2.2.1. For i in 1 ,..., M
            Update, w = w + x * y



Limitations of perceptrons

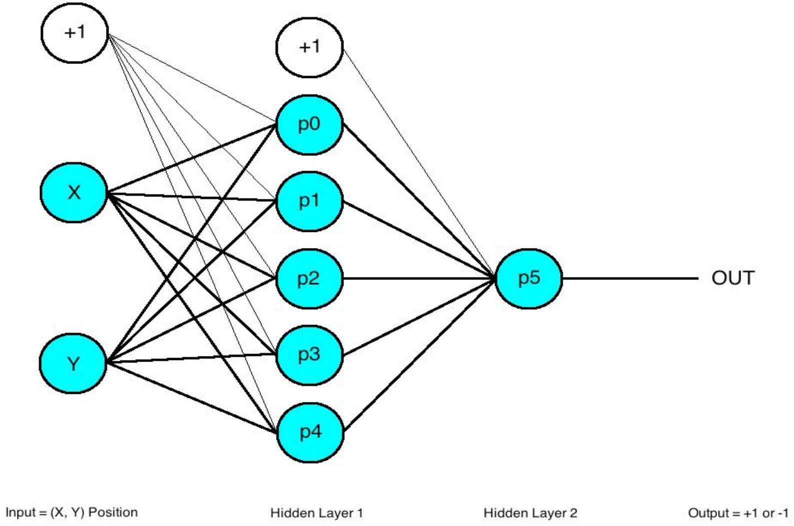
- Learning converges only if data is linearly separable.
- Does not model uncertainty - outputs either 0 or 1.
- Input features should be well-defined.

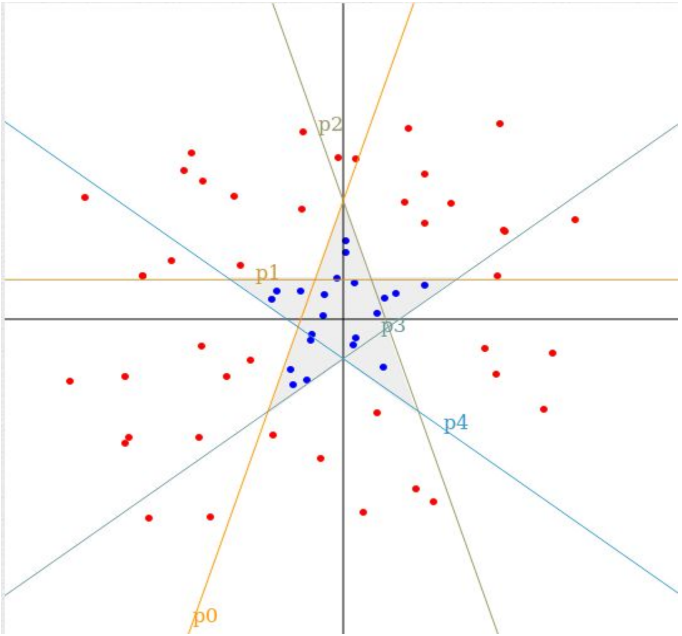
Linear separability



Source: https://towardsdatascience.com/radial-basis-functions-neural-networks-all-we-need-to-know-9a88cc053448

Multi-layered Perceptron

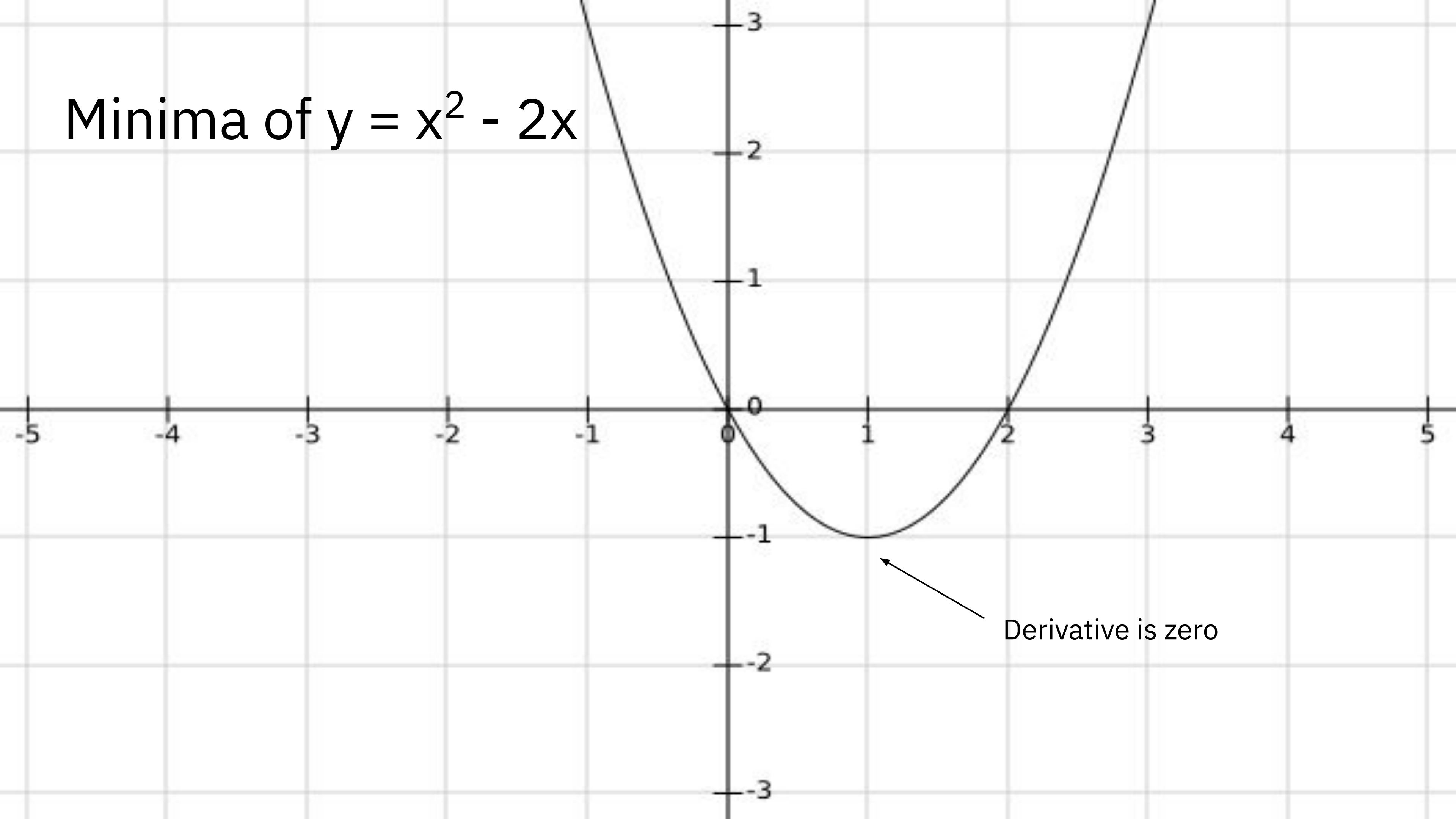




Source: https://www.cs.utexas.edu/~teammco/misc/mlp/

Learning as error-minimization

- Empirical risk minimization
- Hypothesis **h** : X -> y
- Find **h** by minimizing an error function over training data.



Minima of $y = x^2 - 2x$
3
2
1
0
-1
-2
-3
-5
-4
-3
-2
-1
0
1
2
3
4
5
Derivative is zero

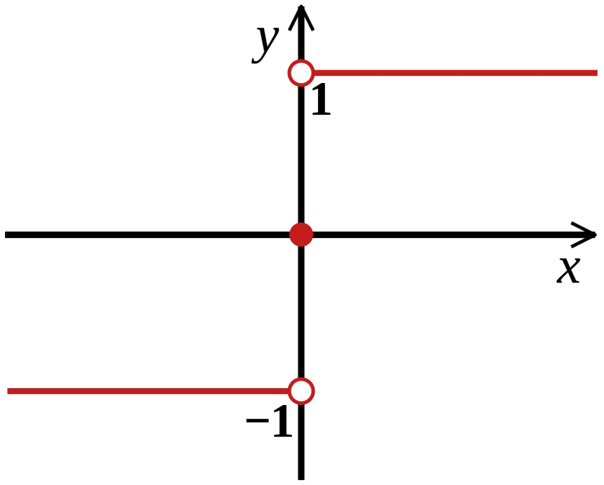
Requirements for direct error-minimization

- Differentiable expression for loss function
- Closed-form solution for minimum (derivative = 0)

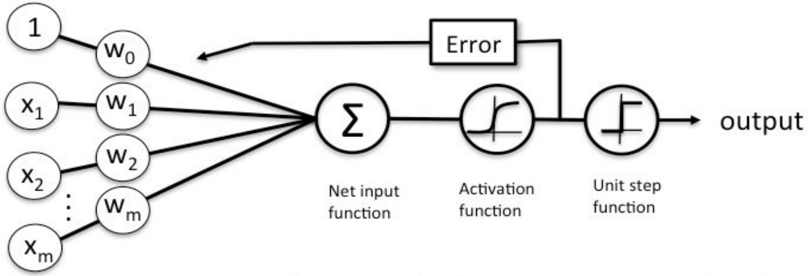
Perceptron Loss function

Perceptron uses the Sign function for prediction.

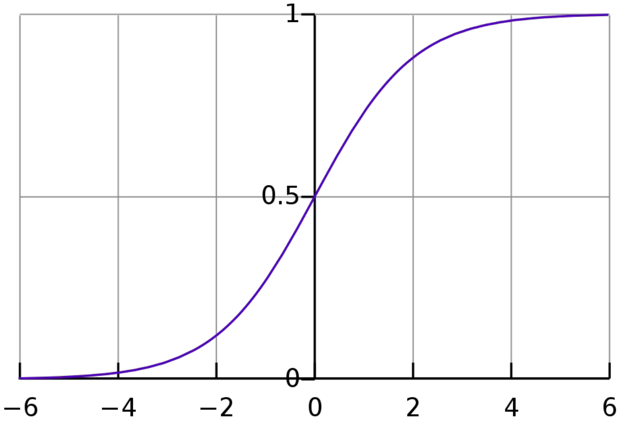
Is this differentiable ?



The Sigmoid Function



Schematic of a logistic regression classifier.



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

Source: https://en.wikipedia.org/wiki/Sigmoid_function#/

Logistic Regression

Replace Sign with Sigmoid.

$$Loss(w) = \sum_{i=1}^{m} y^{(i)} \log P(y=1) + (1 - y^{(i)}) \log P(y=0)$$

Differentiate and solve for w

$$Loss(w) = \sum_{i=1}^{m} y^{(i)} \log P(y=1) + (1 - y^{(i)}) \log P(y=0)$$

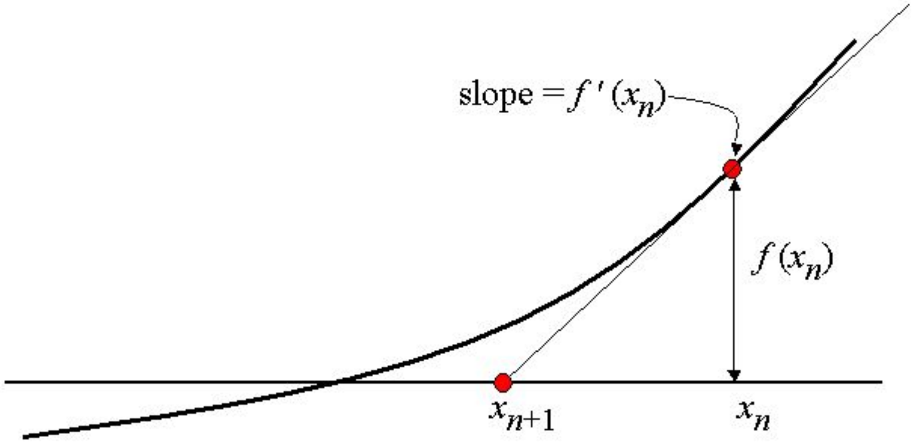
$$Loss(w) = \sum_{i=1}^{m} y^{(i)} \log\left(\frac{1}{1+e^{-w^T x}}\right) + (1 - y^{(i)}) \log\left(1 - \frac{1}{1+e^{-w^T x}}\right)$$

$$\frac{\partial Loss}{\partial w} = \sum_{i} \left[y_i - \frac{1}{1+\exp(x_i' w)}\right] x_i = 0$$

There is no closed-form solution

$$\frac{\partial Loss}{\partial w} = \sum_i \left[y_i - \frac{1}{1 + \exp(x_i' w)} \right] x_i = 0$$

Newton's Method to find zero-point



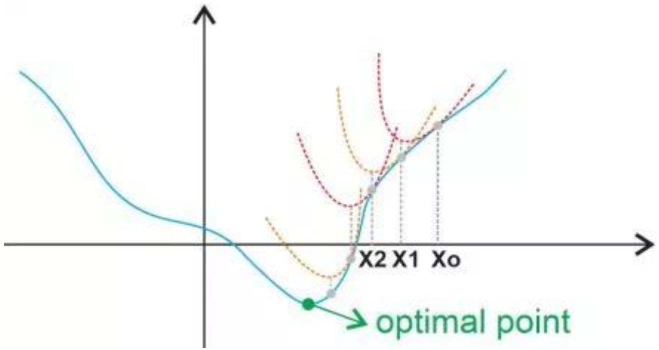
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Source: https://brilliant.org/wiki/newton-raphson-method/

Newton's Method for minimization

- Find zero-point of derivative.

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

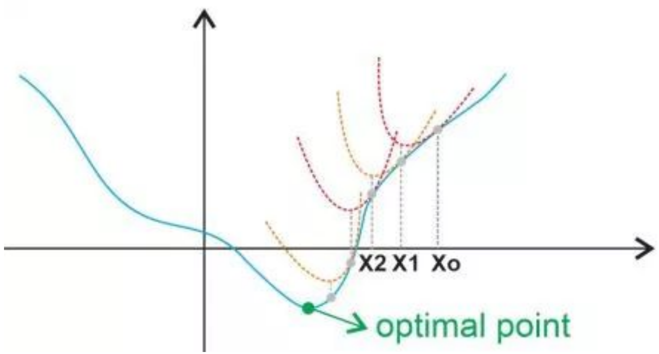


Gradient Descent for minimization

- Second derivative is expensive

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

$$x_{n+1} = x_n - \eta f'(x_n)$$

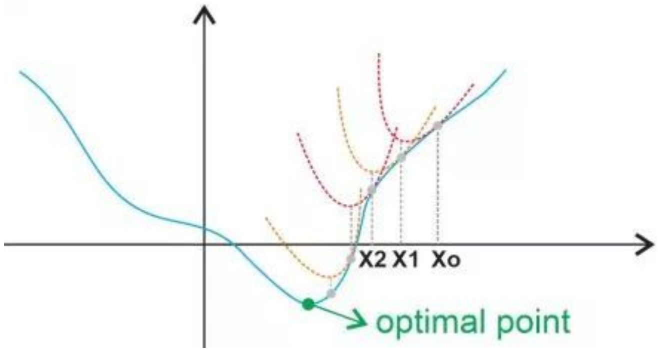


Stochastic Gradient Descent

- Approximate derivative using single example

$$x_{n+1} = x_n - \eta f'(x_n)$$

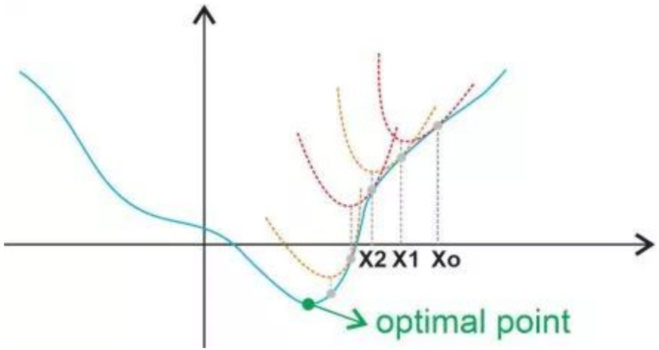
$$\sum_{i=1}^{N} \frac{\partial Loss(x_i, y, w_n)}{\partial w_n} \simeq N * \frac{\partial Loss(x_i, y, w_n)}{\partial w_n}$$



Extensions of SGD : Selecting LR

- ADAM
- RMSProp
- Adagrad
- Adadelta

http://ruder.io/optimizing-gradient-descent/index.html



"Shallow" Learning on images

Pixel values → Feature extraction → Shallow learning → Prediction

Histograms (HOG)	Logistic Regression	Object class
Feature Points (SIFT)	Perceptron	Object bounding box
Visual words	SVM	Depth

Activation Functions

	Propagation	Back-propagation
Sigmoid	$y_s = \frac{1}{1+e^{-x_s}}$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \frac{1}{(1+e^{x_s})(1+e^{-x_s})}$
Tanh	$y_s = \tanh(x_s)$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \frac{1}{\cosh^2 x_s}$
ReLu	$y_s = \max(0, x_s)$	$\left[\frac{\partial E}{\partial x}\right]_s = \left[\frac{\partial E}{\partial y}\right]_s \mathbb{I}\{x_s > 0\}$

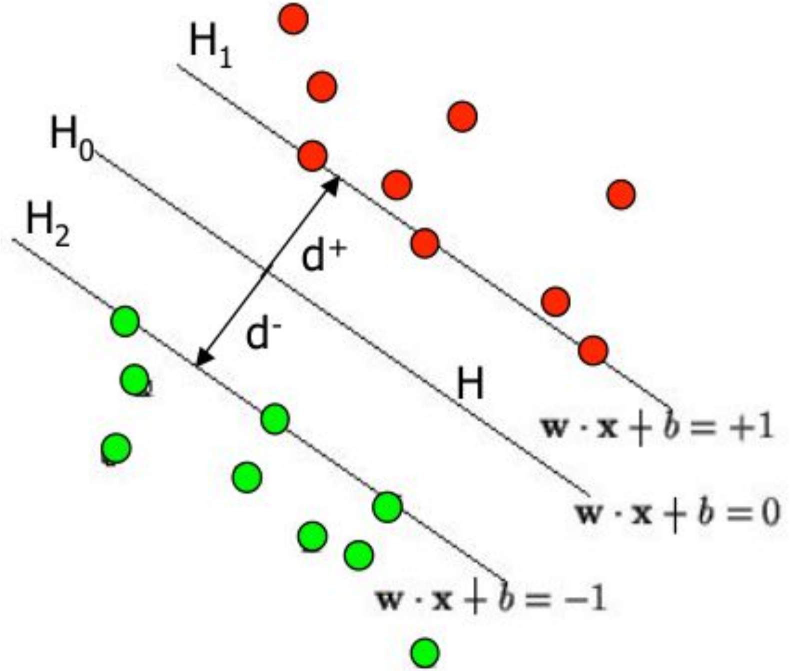
Source: http://www.marekrei.com/blog/26-things-i-learned-in-the-deep-learning-summer-school/

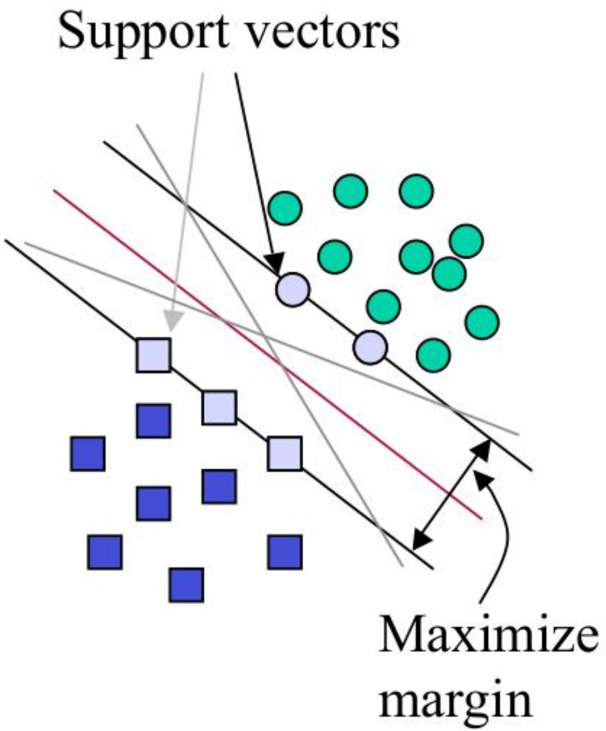
Loss Functions

Problem Type	Output Type	Final Activation Function	Loss Function
Regression	Numerical value	Linear	Mean Squared Error (MSE)
Classification	Binary outcome	Sigmoid	Binary Cross Entropy
Classification	Single label, multiple classes	Softmax	Cross Entropy
Classification	Multiple labels, multiple classes	Sigmoid	Binary Cross Entropy

Source: https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

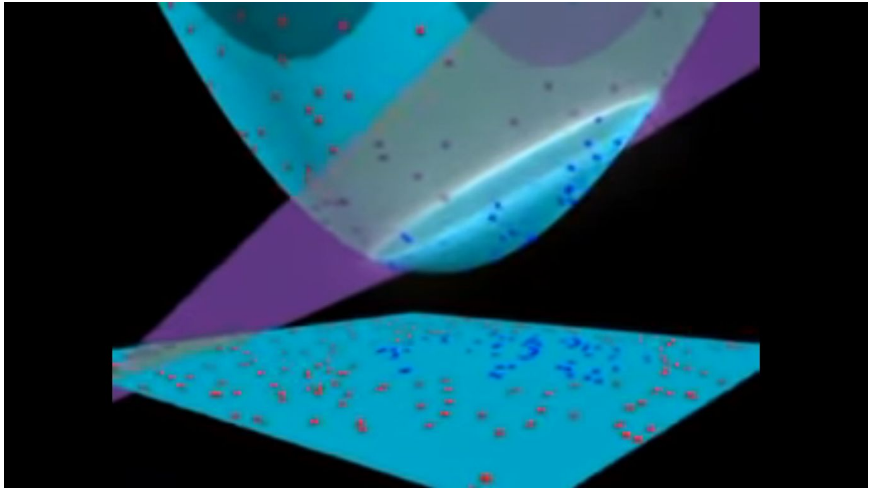
Support Vector Machines





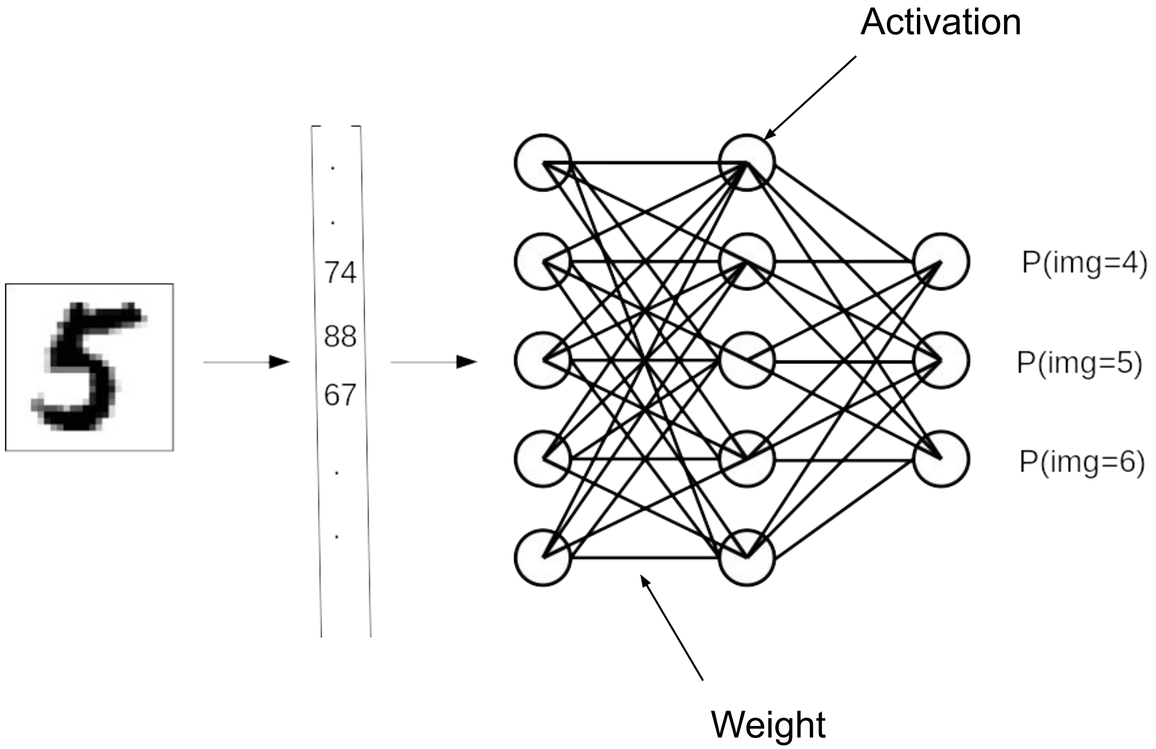
Source: http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Support Vector Machines: Kernels



Source: https://www.youtube.com/watch?v=3liCbRZPrZA

Neural Network for image classification

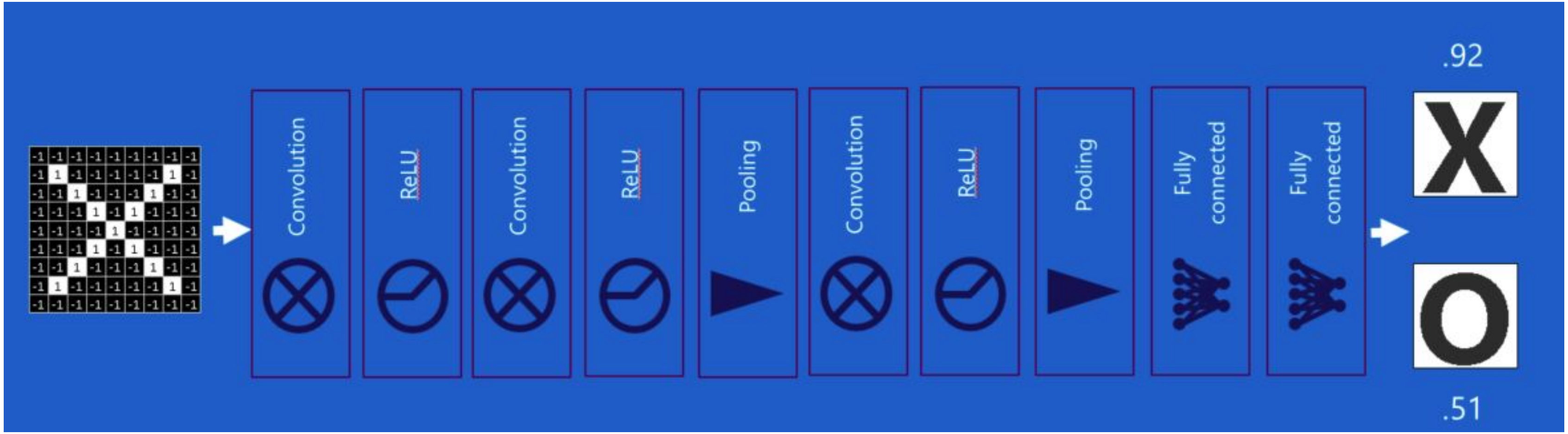


Backpropagation - Key Ideas

- Forward-propagation
- Use chain-rule to expand derivatives.
- Save intermediate derivatives (dynamic programming)

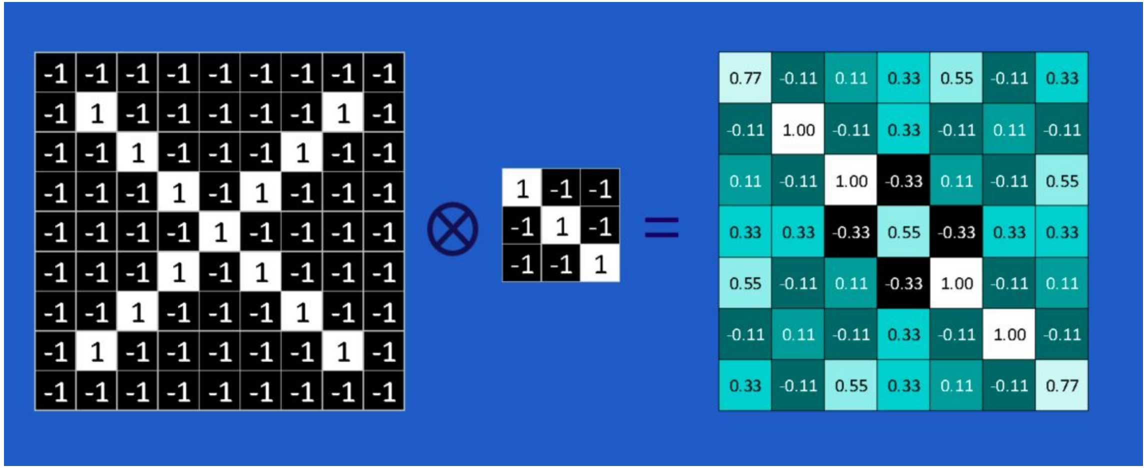
  https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/

Convolutional Neural Networks



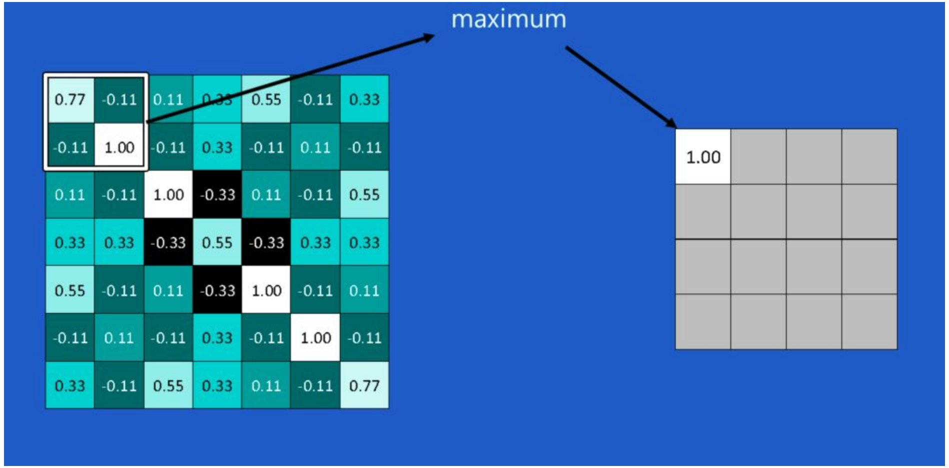
Source: http://brohrer.github.io/how_convolutional_neural_networks_work.html

Convolution Layer



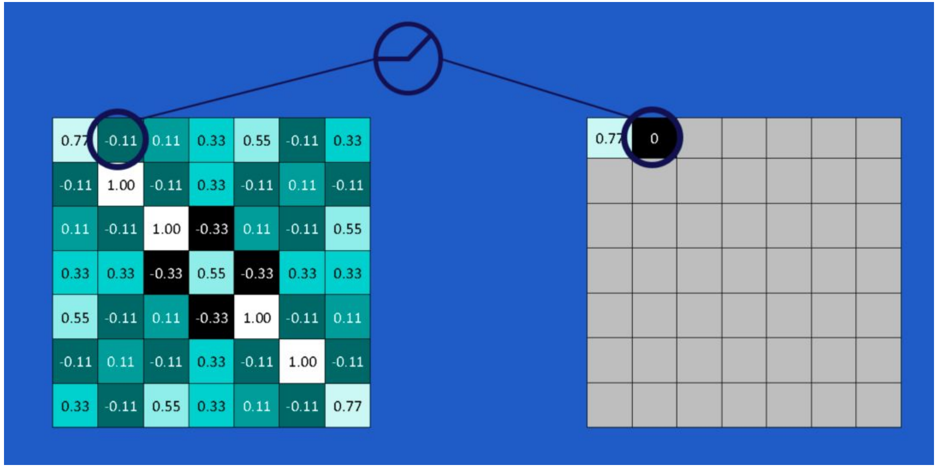
Source: http://brohrer.github.io/how_convolutional_neural_networks_work.html

Pooling Layer



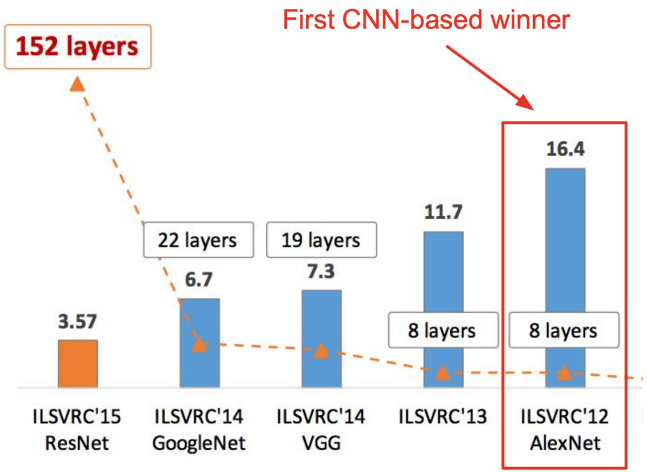
Source: http://brohrer.github.io/how_convolutional_neural_networks_work.html

Rectified Linear Unit



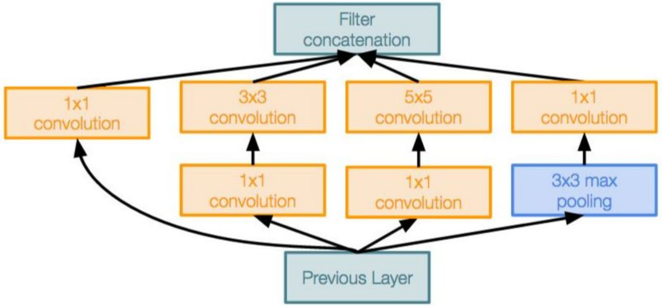
Source: http://brohrer.github.io/how_convolutional_neural_networks_work.html

Key Idea - More Layers



Source: Fei-Fei Li & Justin Johnson & Serena Young

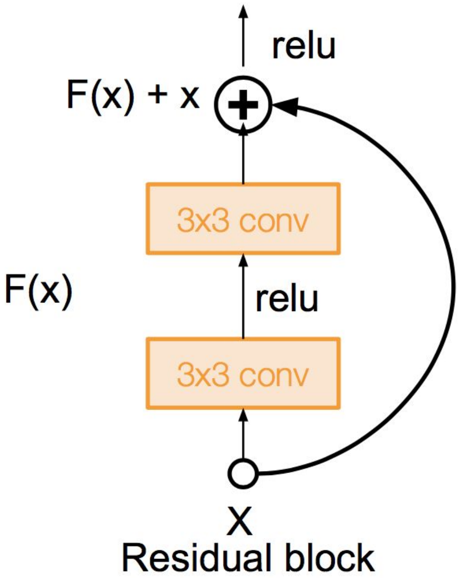
Key Idea - Parallel Layers



Inception module

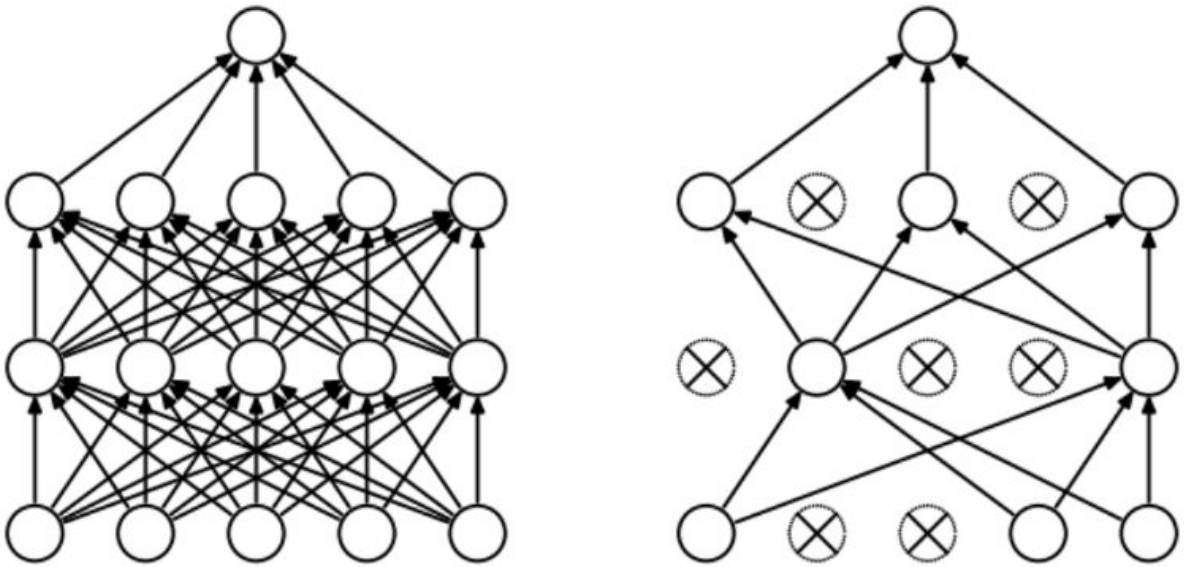
Source: Fei-Fei Li & Justin Johnson & Serena Young

Key Idea - Skip Connections



Source: Fei-Fei Li & Justin Johnson & Serena Young

Key Idea - Dropout

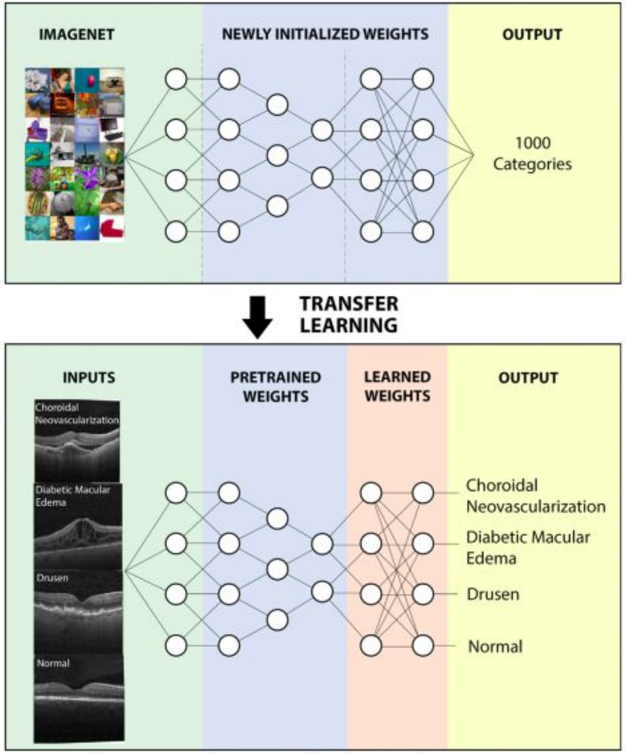


(a) Standard Neural Net

(b) After applying dropout.

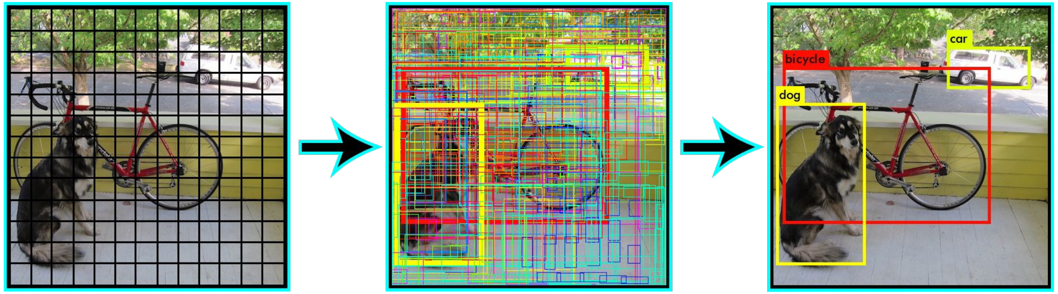
Source: Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from overfitting”, JMLR 2014

Key Idea - Transfer Learning



Source: https://software.intel.com/en-us/articles/use-transfer-learning-for-efficient-deep-learning-training-on-intel-xeon-processors

YOLO - Very Fast Object Detection

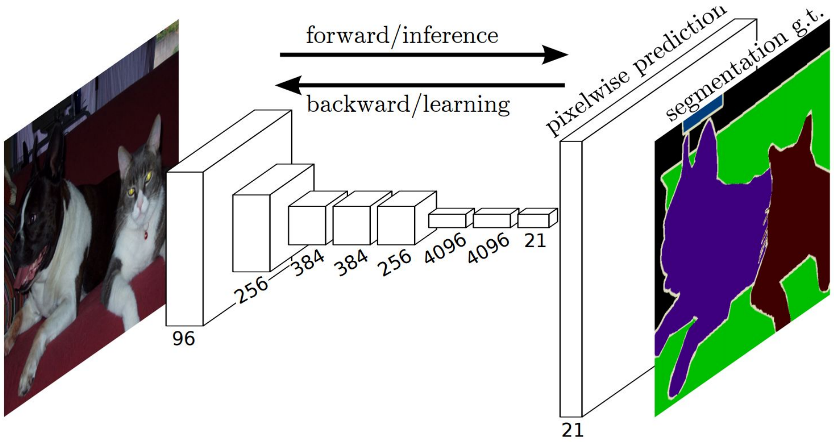


Quantization

Non-Maximal Suppression

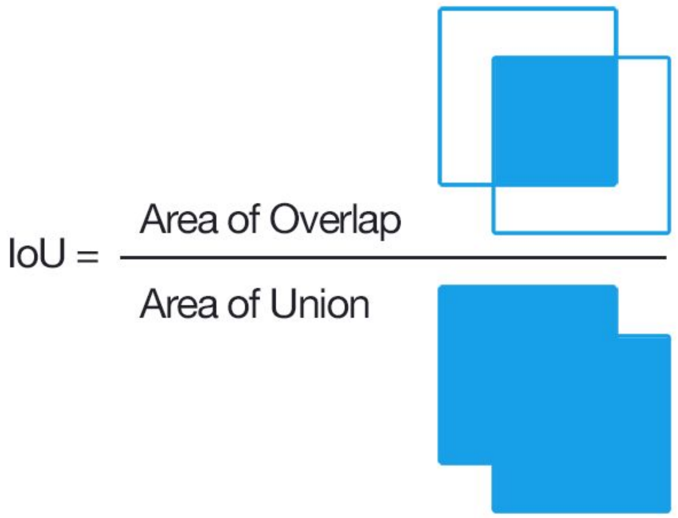
Source: https://pjreddie.com/darknet/yolov2/

FCN - Semantic Segmentation



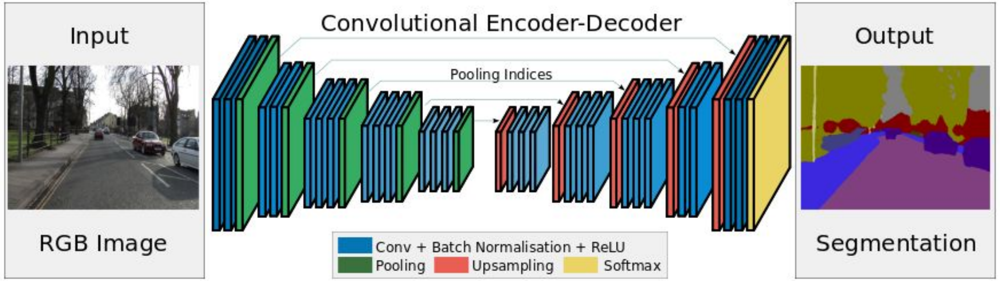
Source: http://deeplearning.net/tutorial/fcn_2D_segm.html

Intersection over Union



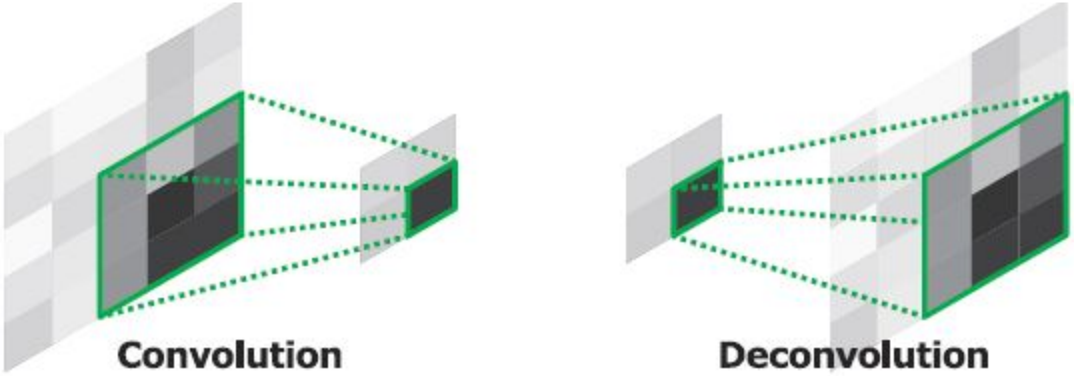
Source: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

SegNet : Trainable Encoder-Decoder Architecture



Source: http://mi.eng.cam.ac.uk/projects/segnet/

Key Idea - Transposed Convolutions for Upsampling



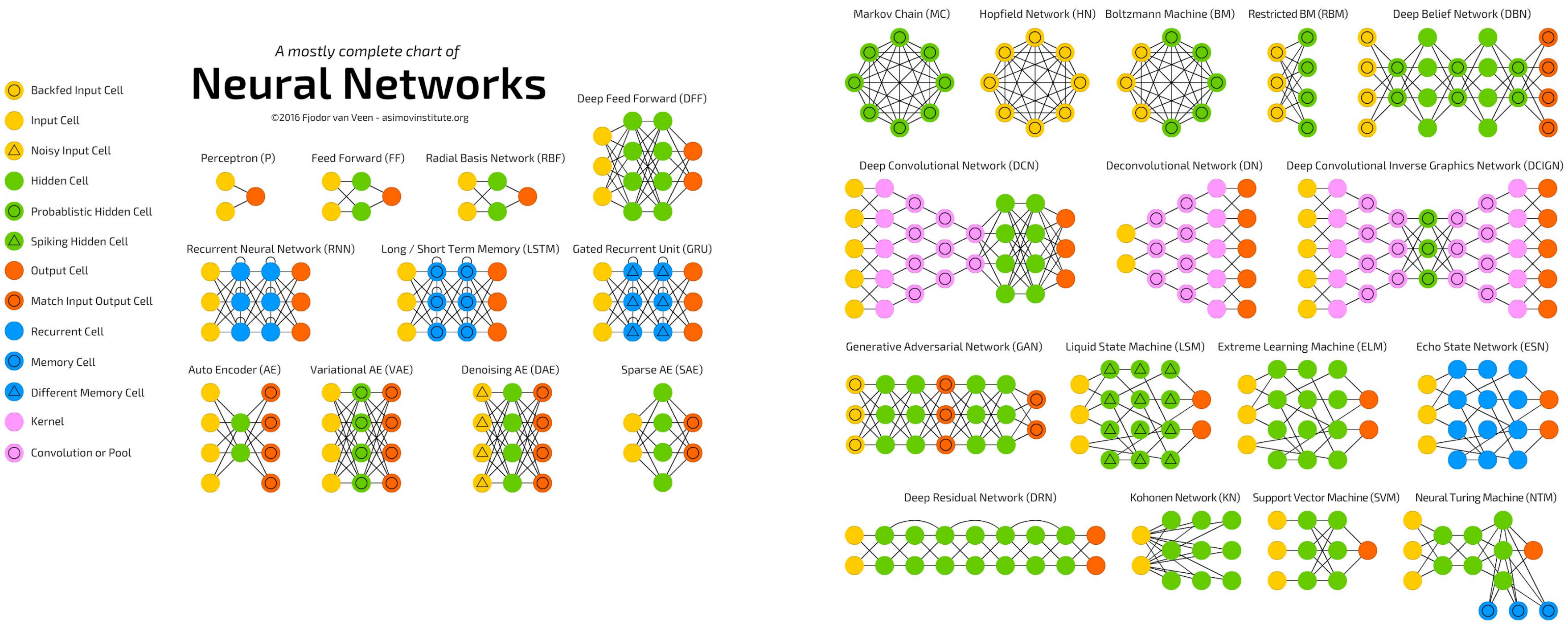
Source: https://towardsdatascience.com/review-deconvnet-unpooling-layer-semantic-segmentation-55cf8a6e380e

Key Idea - Transposed Convolutions for Upsampling

$$\begin{pmatrix}
w_{0,0} & 0 & 0 & 0 \\
w_{0,1} & w_{0,0} & 0 & 0 \\
w_{0,2} & w_{0,1} & 0 & 0 \\
0 & w_{0,2} & 0 & 0 \\
w_{1,0} & 0 & w_{0,0} & 0 \\
w_{1,1} & w_{1,0} & w_{0,1} & w_{0,0} \\
w_{1,2} & w_{1,1} & w_{0,2} & w_{0,1} \\
0 & w_{1,2} & 0 & w_{0,2} \\
w_{2,0} & 0 & w_{1,0} & 0 \\
w_{2,1} & w_{2,0} & w_{1,1} & w_{1,0} \\
w_{2,2} & w_{2,1} & w_{1,2} & w_{1,1} \\
0 & w_{2,2} & 0 & w_{1,2} \\
0 & 0 & w_{2,0} & 0 \\
0 & 0 & w_{2,1} & w_{2,0} \\
0 & 0 & w_{2,2} & w_{2,1} \\
0 & 0 & 0 & w_{2,2}
\end{pmatrix}^T$$

Source: http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html

Summary



Source: http://www.asimovinstitute.org/neural-network-zoo/