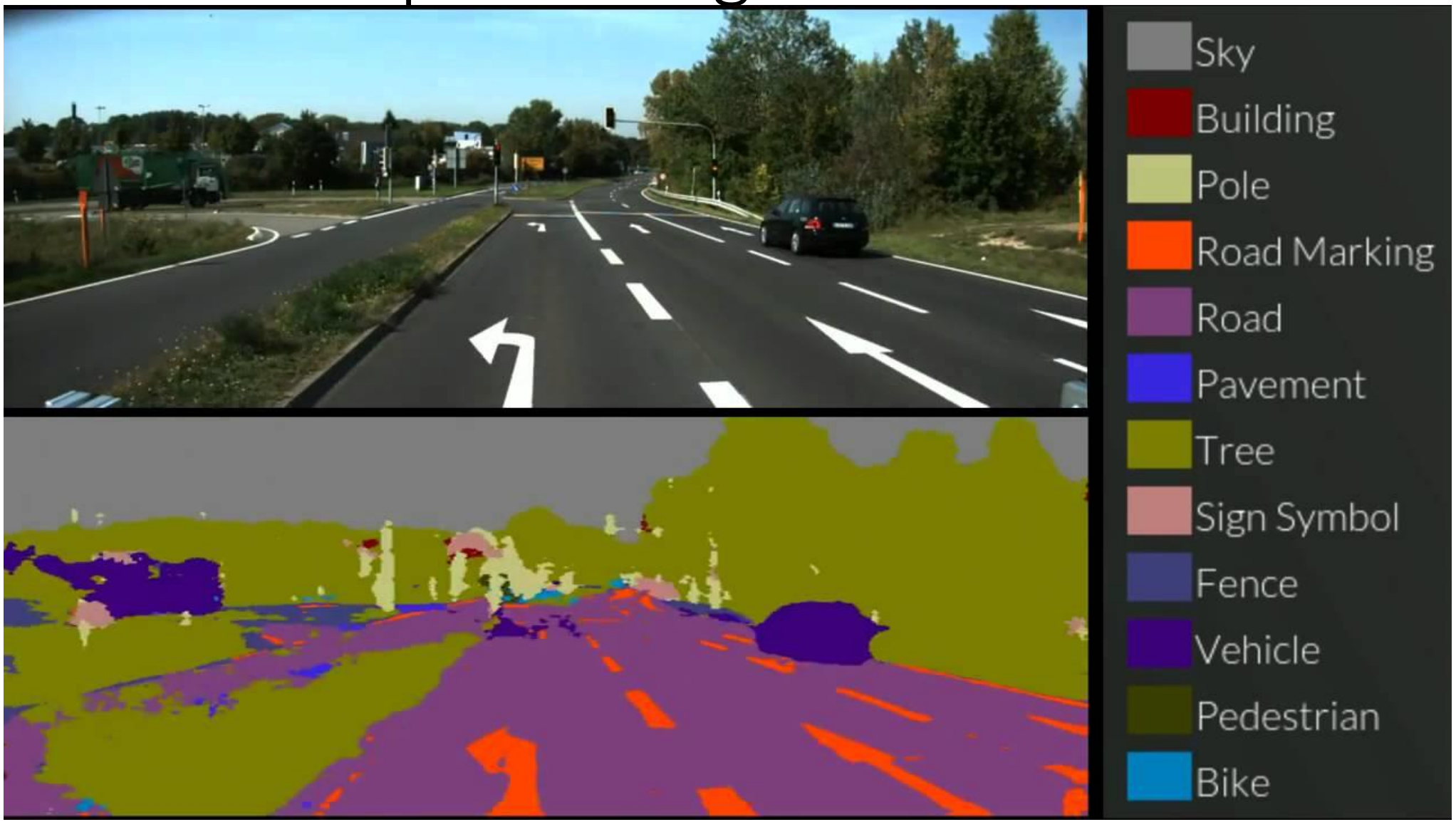
Introduction to Deep Learning: from linear regression to AlexNet.

Ruben Sanchez Garcia

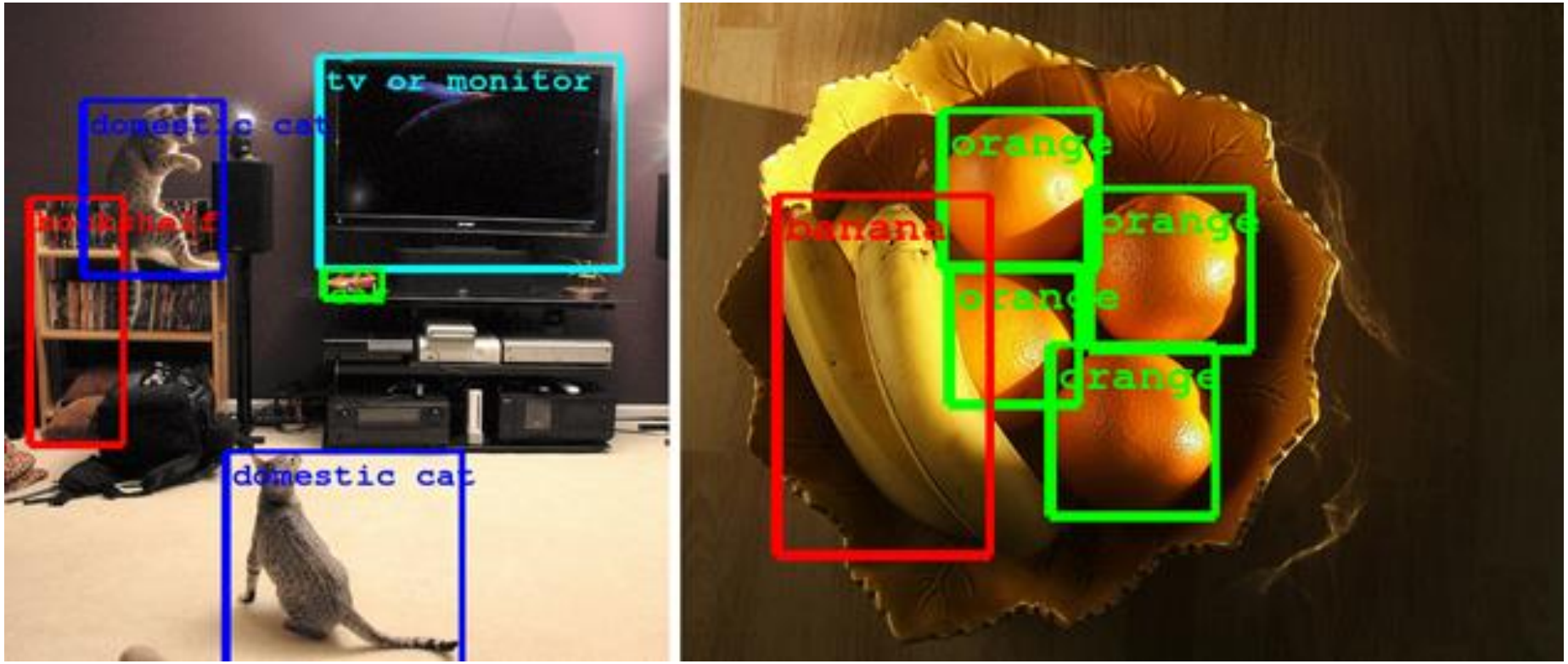
March 2018

What is Deep Learning?



What is Deep Learning?

What is Deep Learning?



What is Deep Learning?

ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go

LETTER

doi:10.1038/nature14236

Human-level control through deep reinforcement learning

Volodymyr Mnih[1]*, Koray Kavukcuoglu[1]*, David Silver[1]*, Andrei A. Rusu[1], Joel Veness[1], Marc G. Bellemare[1], Alex Graves[1], Martin Riedmiller[1], Andreas K. Fidjeland[1], Georg Ostrovski[1], Stig Petersen[1], Charles Beattie[1], Amir Sadik[1], Ioannis Antonoglou[1], Helen King[1], Dharshan Kumaran[1], Daan Wierstra[1], Shane Legg[1] & Demis Hassabis[1]

The theory of reinforcement learning provides a normative account[1], deeply rooted in psychological[2] and neuroscientific[3] perspectives on animal behaviour, of how agents may optimize their control of an environment. To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the

agent is to select actions in a fashion that maximizes cumulative future reward. More formally, we use a deep convolutional neural network to approximate the optimal action-value function

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s,\ a_t = a,\ \pi\right],$$

which is the maximum sum of rewards r_t discounted by γ at each time-

What is Deep Learning?

Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi
yonghui,schuster,zhifengc,qvl,mnorouzi@google.com

Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, Jeffrey Dean

Deep Learning for Forecasting Stock Returns in the Cross-Section

Masaya Abe[1] and Hideki Nakayama[2]

[1] Nomura Asset Management Co., Ltd., Tokyo, Japan
m-abe@nomura-am.co.jp
[2] The University of Tokyo, Tokyo, Japan
nakayama@nlab.ci.i.u-tokyo.ac.jp

Applied Energy 188 (2017) 56–70

Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apenergy

Deep learning based ensemble approach for probabilistic wind power forecasting

Huai-zhi Wang[a], Gang-qiang Li[a], Gui-bin Wang[b,*], Jian-chun Peng[a], Hui Jiang[c], Yi-tao Liu[a]

[a] The College of Mechatronics and Control Engineering, Shenzhen University, Shenzhen 518060, China
[b] Shenzhen Key Laboratory of Urban Rail Transit, The College of Urban Rail Transit, Shenzhen University, Shenzhen 518060, China

SCIENTIFIC REPORTS

OPEN

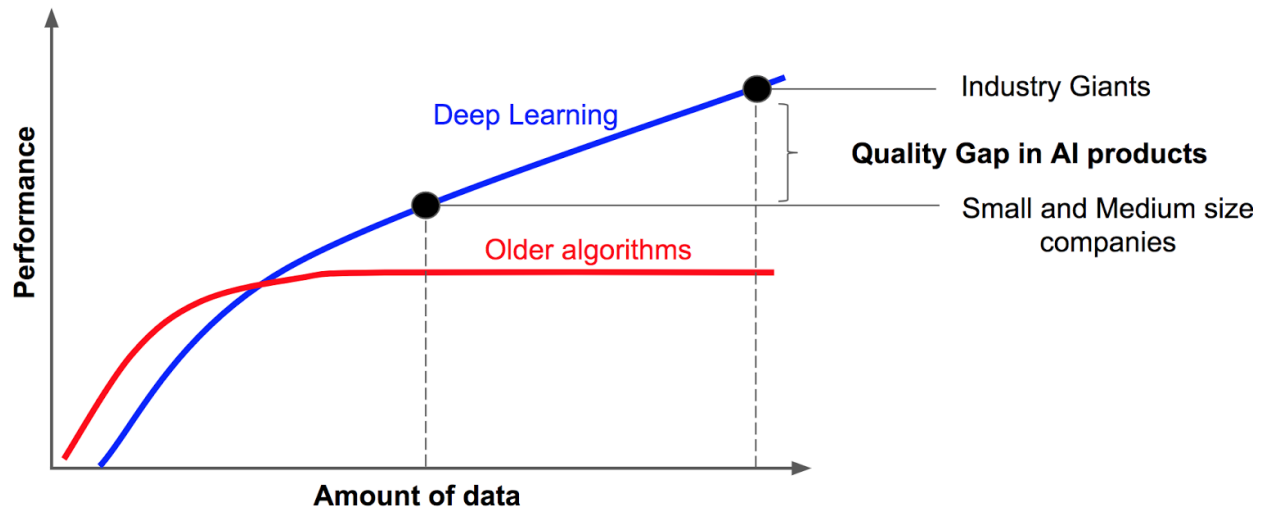
Multi-label Deep Learning for Gene Function Annotation in Cancer Pathways

ceived: 31 August 2017
cepted: 27 November 2017
olished online: 10 January 2018

Renchu Guan[1,3], Xu Wang[1], Mary Qu Yang[3], Yu Zhang[1,4], Fengfeng Zhou[1], Chen Yang[5] & Yanchun Liang[1,2]

What is Deep Learning?

- It is a machine learning subfield based on deep artificial neural networks.
- Contrary to regular machine learning, no feature engineering is needed.
- Tons of data are needed.



What is Machine Learning?

- It is an Artificial Intelligence field that aims to achieve intelligent systems that “learn” from data without the need of explicit programming.

- Learning what? To solve a task.

Tasks that can be “learned”

- Supervised Learning:
  - Training data: data points and label
  - We want to predict labels associated to that points (mapping).
    - Labels can be numerical o categorical.
- Unsupervised Learning:
  - No training data
  - We have data points and no labels and we want to draw conclusions from them.
    - Clustering

Supervised tasks that can be “learned”

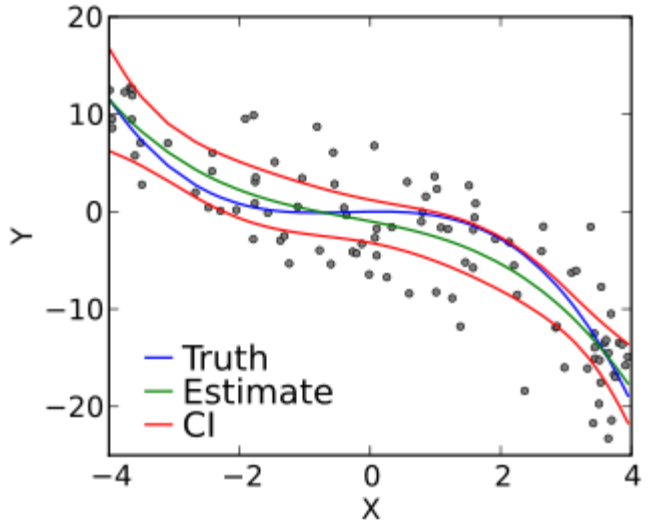
- Regression: Given a set of points we want to predict a (several) real value(s)

$$R^n \rightarrow R^m$$

Features

Labels

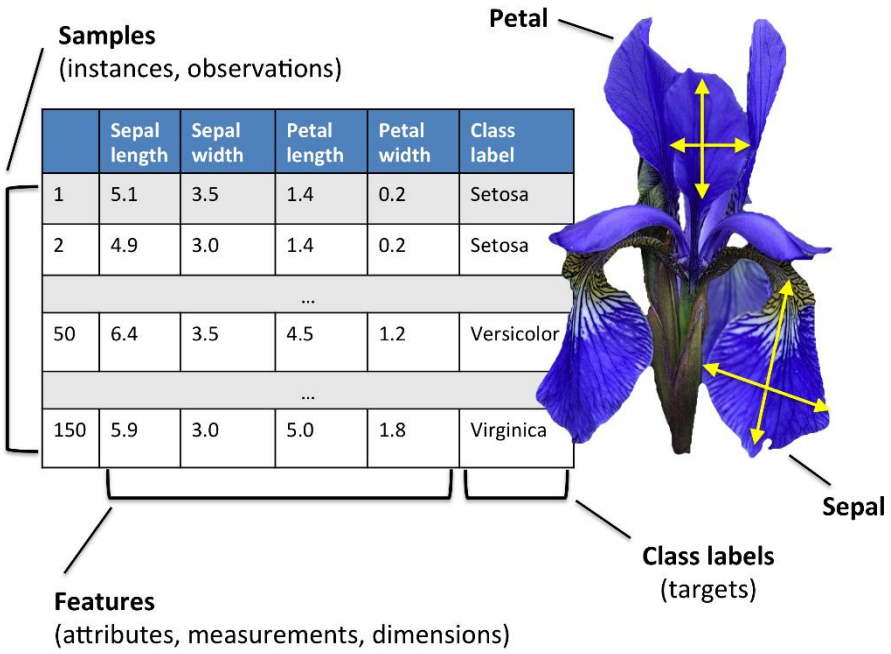
	Color	Quality	Price
4	7	5	65
5	3	7	38
6	5	8	51
7	8	1	38
8	9	3	55
9	5	4	43
10	4	0	25
11	2	6	33
12	8	7	71
13	6	4	51
14	9	2	49



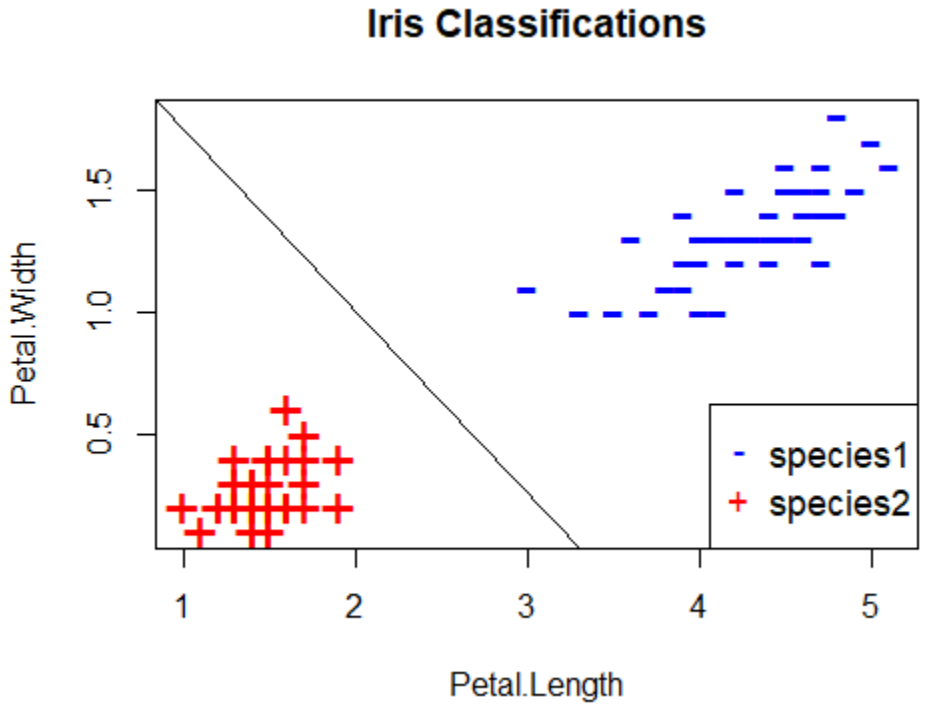
Supervised tasks that can be “learned”

- Classification: Given a set of points we want to predict a (several) category(s)

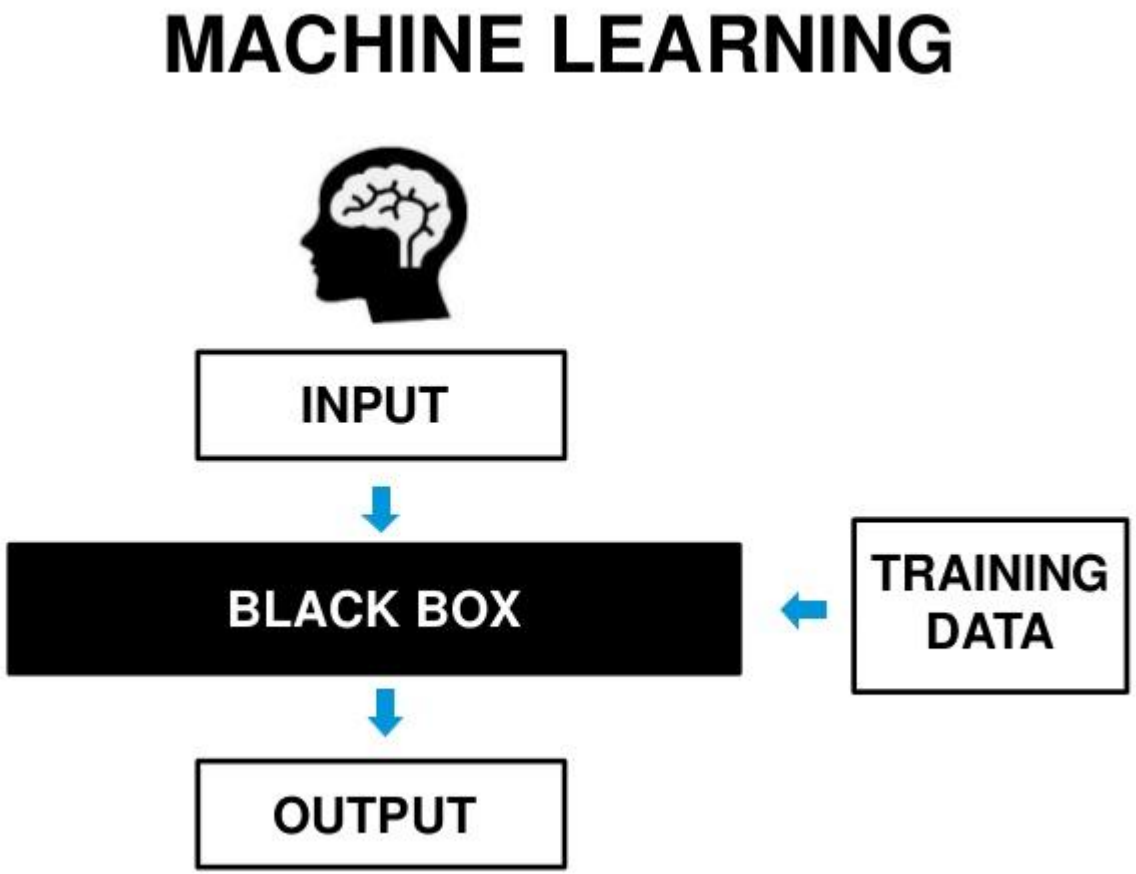
$$R^n \rightarrow [0,1]^m$$



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

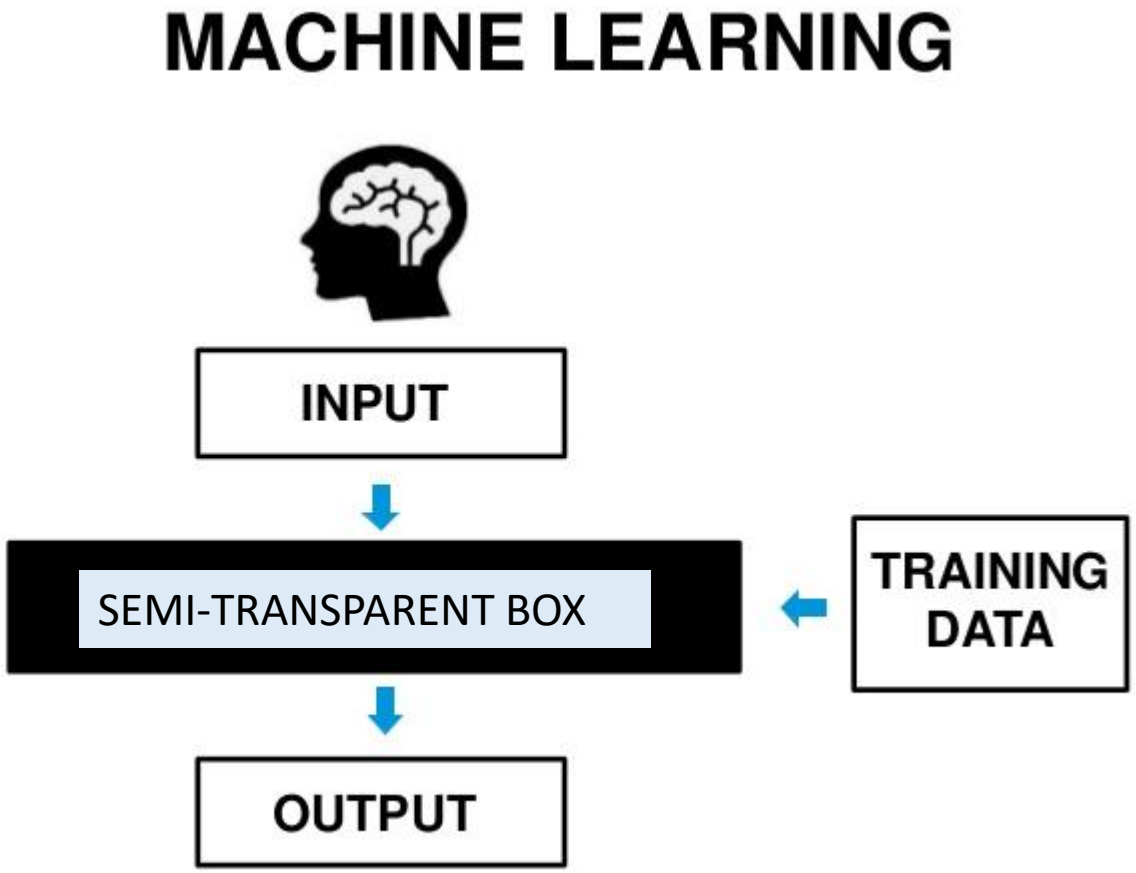


Some people view of supervised machine learning

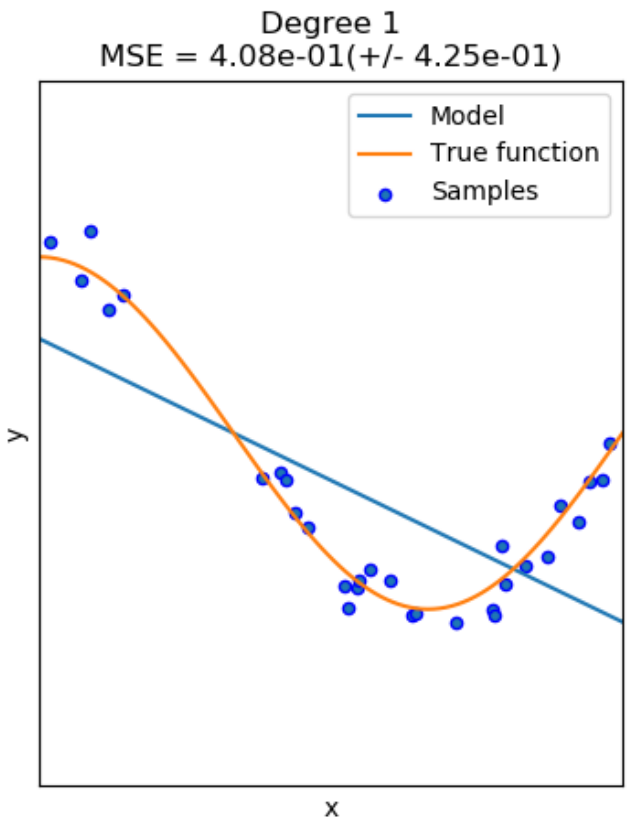


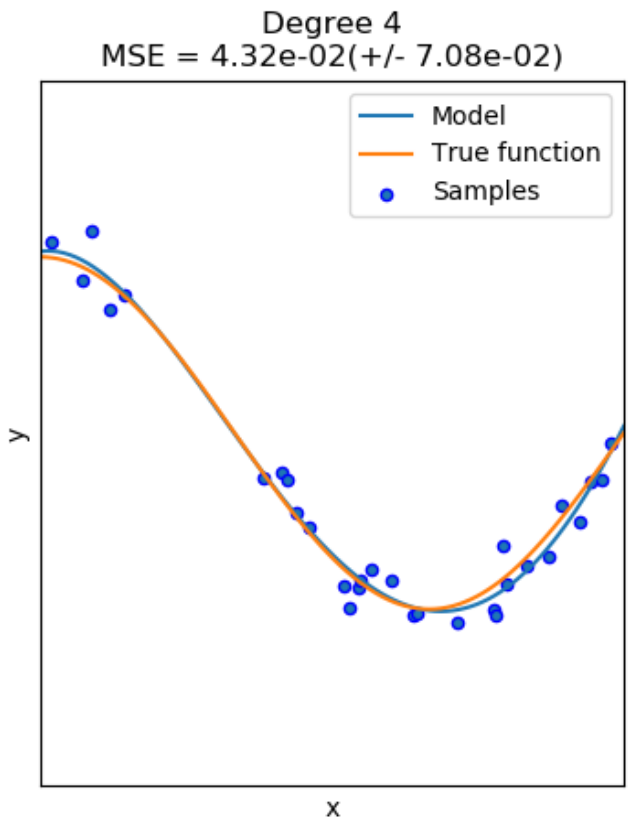
My personal view of supervised machine learning

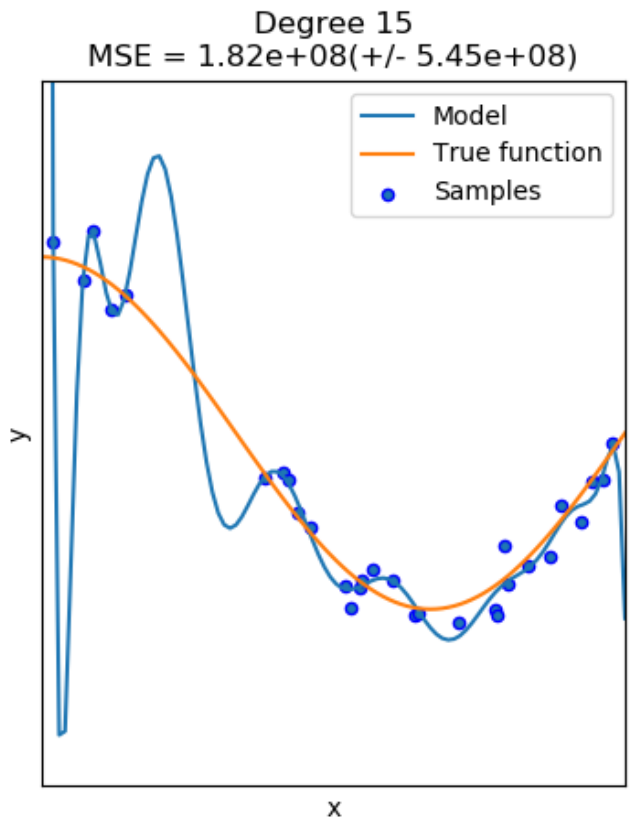
You do not to know how the algorithm works, but, you need to know how to set its hyper-parameters properly. Depending on them, your model will be prone to either overfitting or underfitting. Your task will be to choose the hyper-parameters such that the model exhibits no underfitting nor overfitting (ideally).



Underfitting and overfitting







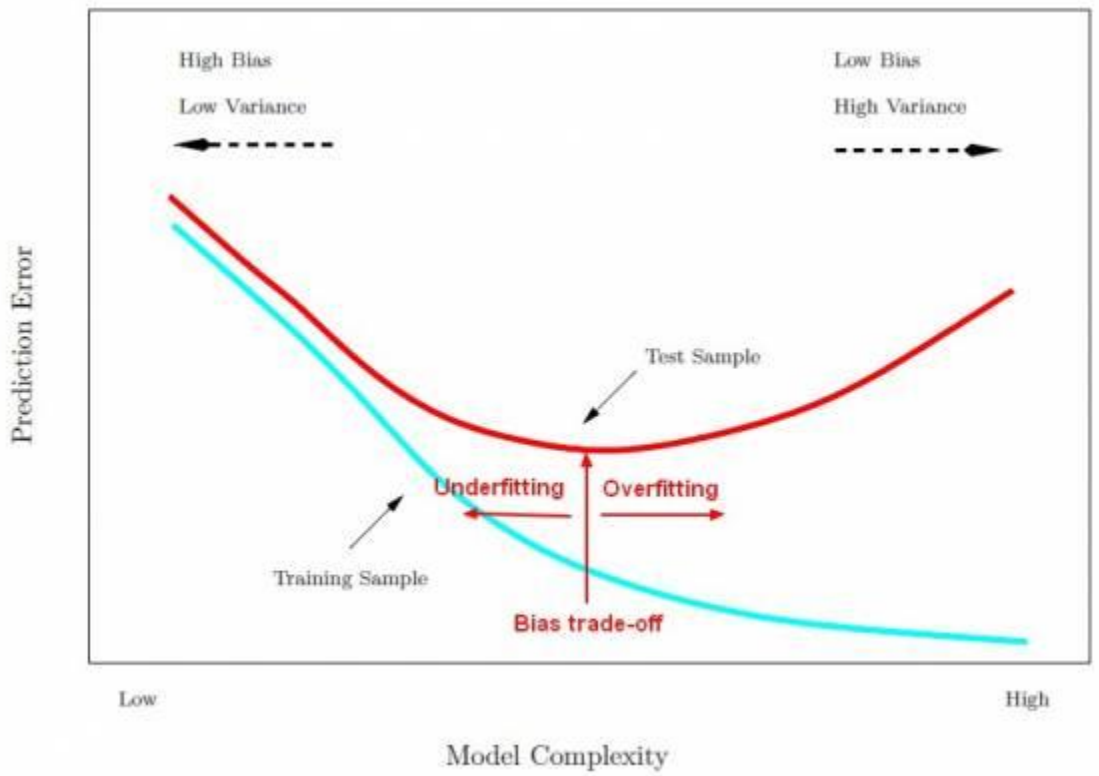
Underfitting and overfitting

In order to select proper hyper-parameters you should detect if your model is in the overfitting zone or in the underfitting zone:

- Split your data into 3 parts: training data set, test data set and validation data set (e.g. 70% 20% and 10%) or do cross-validation.

- Compare error on training set with error on test set.

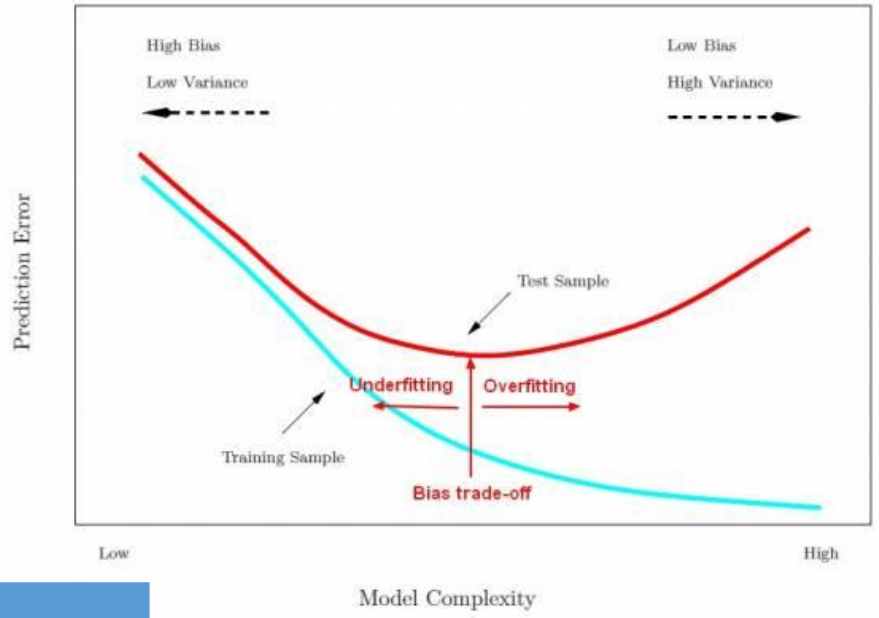
    - If your training error is low and the test error is high → overfitting
    - If your training error is high → underfitting



Underfitting and overfitting

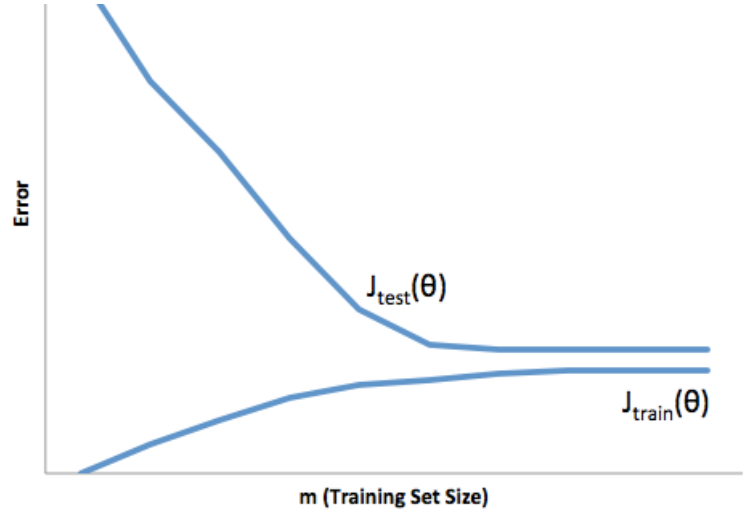
If the training error oscillates as crazy:

- Decrease learning rate (α)



If you are in:

Underfitting regime	Overfitting regime
Make your model more complex (p.e. increase the degree of the polynomial, more or bigger layers in neural net)	Make your model less complicate (decrease the degree of the polynomial, less layers or smaller in neural net
Look for new variables/ feature engineering	Get more training data
Decrease regularization strength	Apply regularization techniques
Tune hyper-parameters to make your model fit better your data	Tune hyper-parameters to make your model fit worse your data

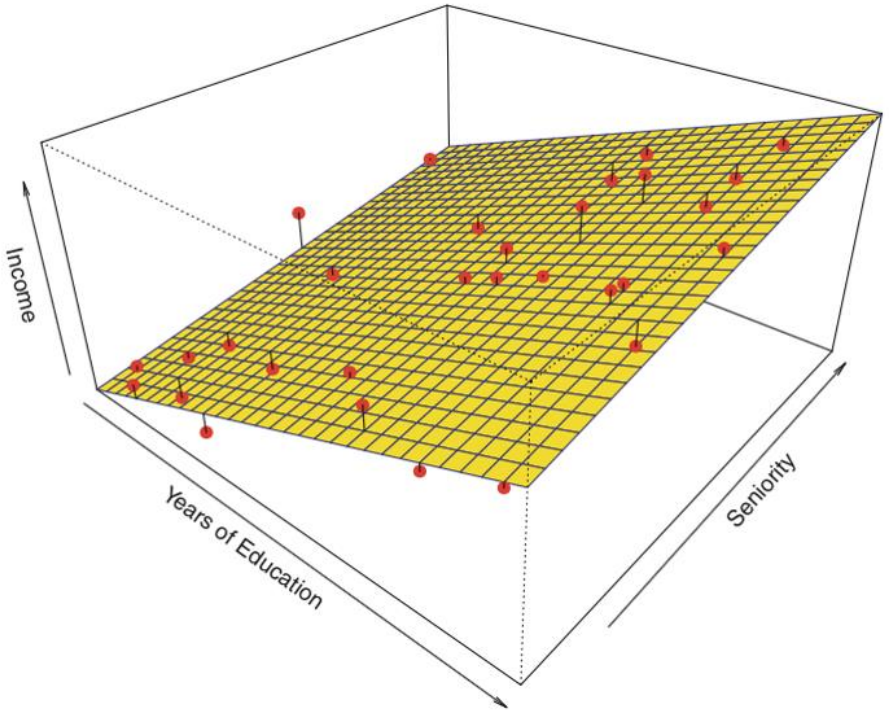


Easiest model: Linear regression.

$$y(x_1, x_2, \dots, x_n) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = W^T X$$

$$X = \begin{pmatrix} 1 \\ w_1 \\ \dots \\ w_n \end{pmatrix}$$

$$W = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{pmatrix}$$



Example number	Years of Education	Seniority	Income
1	2	2	24000
2	4	4	37000
…	…	…	…
N	6	1	31000

Easiest model: Linear regression.

$$X = \begin{pmatrix} 1 \\ w_1 \\ \cdots \\ w_n \end{pmatrix}$$

$$y(x_1, x_2, \ldots, x_n) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = W^T X$$

$$W = \begin{pmatrix} a_0 \\ a_1 \\ \cdots \\ a_n \end{pmatrix}$$

$$E(W) = RMSE = \frac{1}{2m} \sum_{i=1}^{m} \left(y\left(X^i\right) - l^i\right)^2$$

$E \rightarrow error\ of\ the\ model$
$l_i \rightarrow\ value\ of\ example\ i$ (label)
$x_i \rightarrow features\ of\ example\ x$
m$\rightarrow number\ of\ examples$

Find *W* such that E(*W)* is minimum

$$W = \underset{W}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(y\left(X^i\right) - l^i\right)^2$$

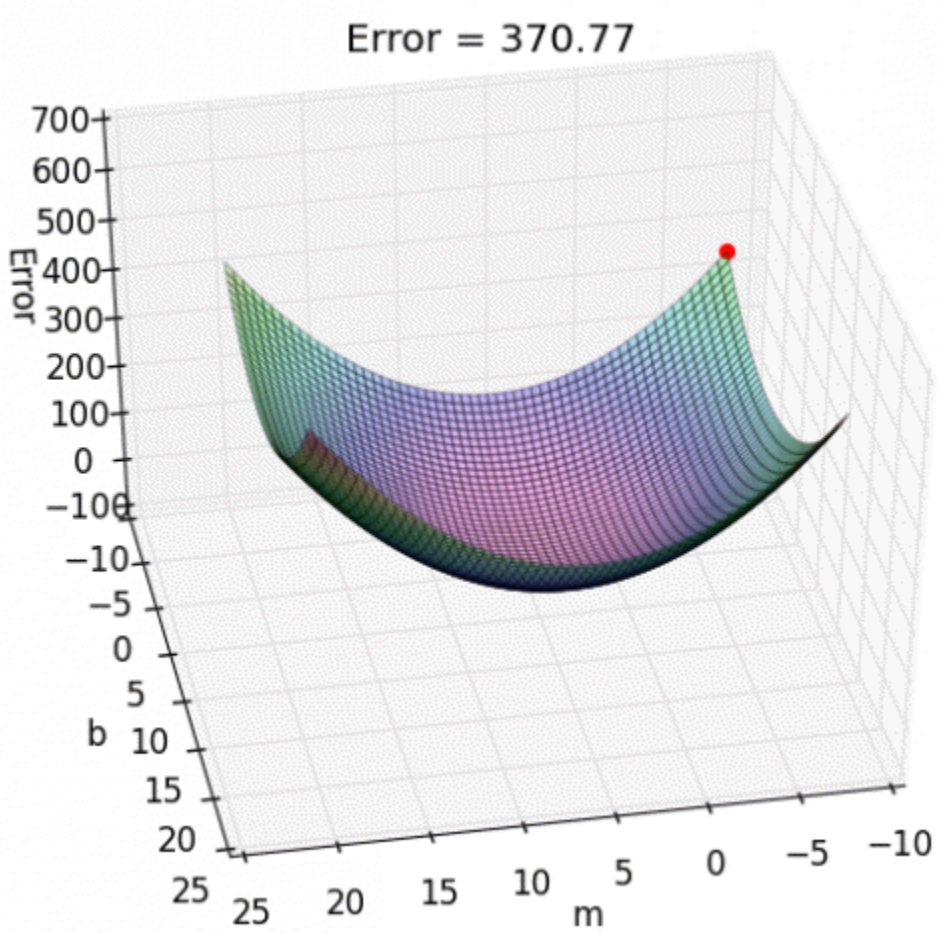
Easiest model: Linear regression.

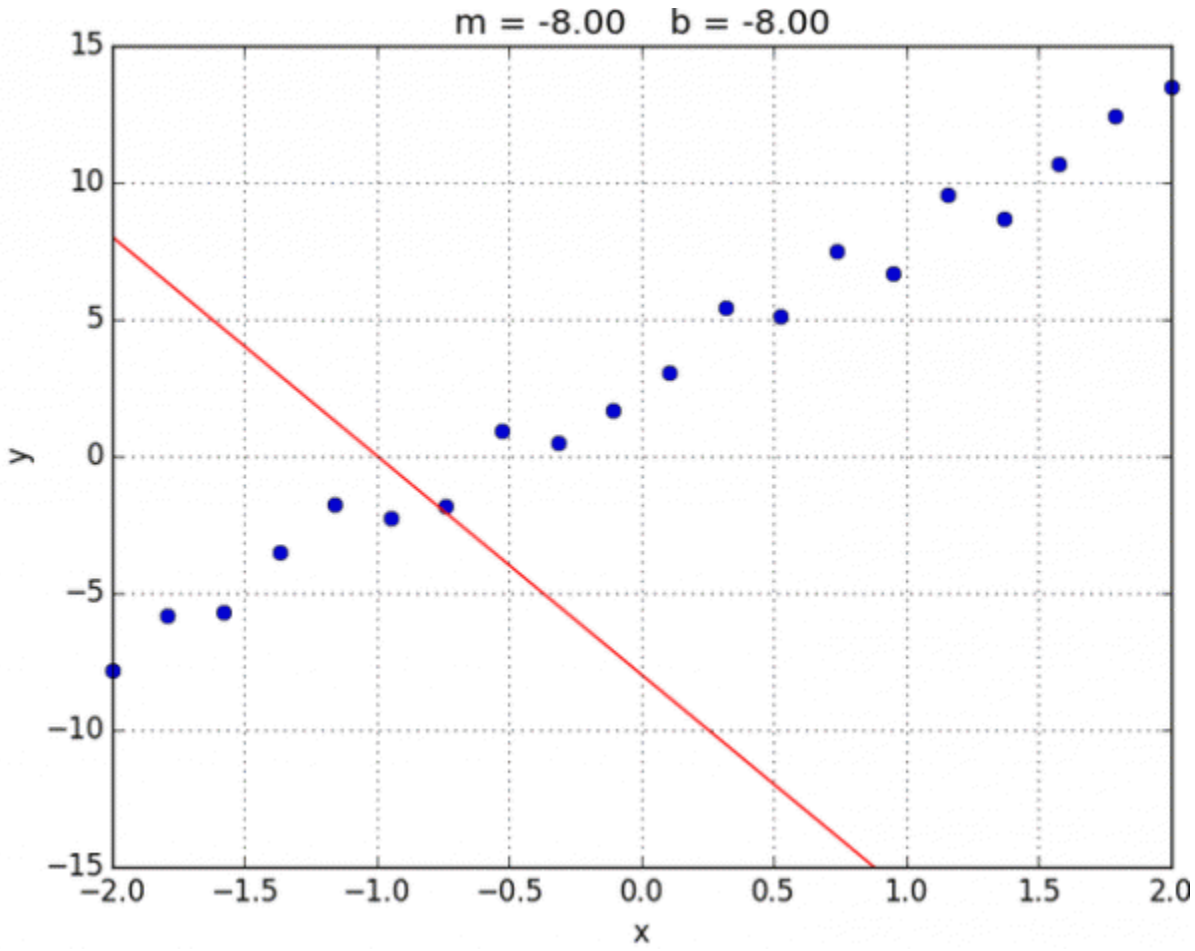
Find *W* such that E(*W*) is minimum $$W = \underset{W}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(y\left(X^i\right) - l^i\right)^2 = \underset{W}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(W^T X^i - l^i\right)^2$$

For linear regression, there is an analytic solution. But there is not for neural networks, instead **gradient descent** is used.

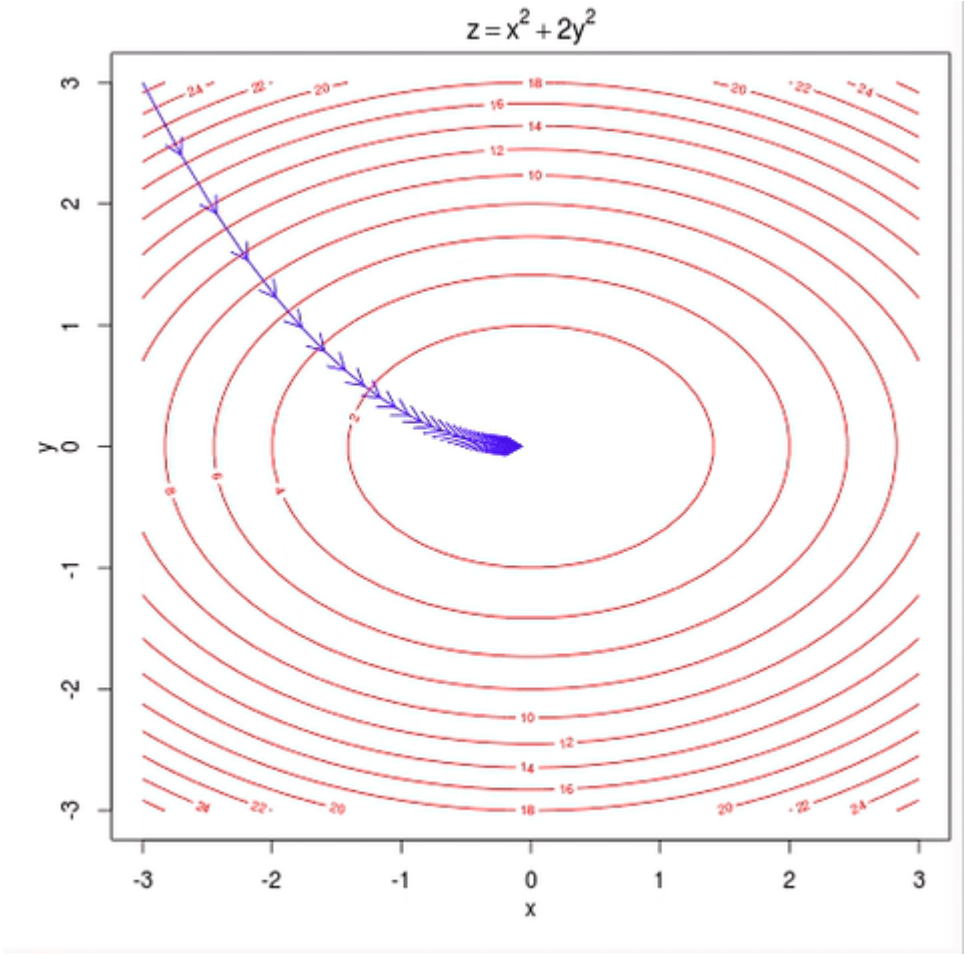
1. $Step\ 1,\ initialize\ W\ (generally\ randomly)\ and\ select\ \alpha$
2. $Step\ 2,\ w_j^t := w_j^{t-1} - \alpha \frac{\partial E(W)}{dw_j}$ For linear regression → $\frac{\partial E(W)}{dw_j} = \frac{1}{\mathrm{m}} \sum_{i=1}^{m} \left(W^T X^i - l^i\right) X_j^i$
3. $Go\ to\ Step\ 2\ \ until\ stop\ criteria\ (generally\ convergency\ or\ number\ of\ iterations)$

Gradient descent



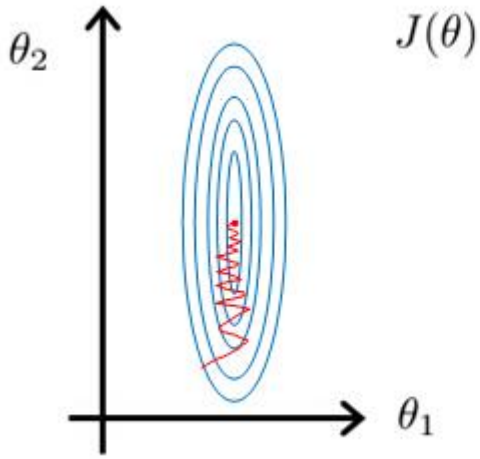


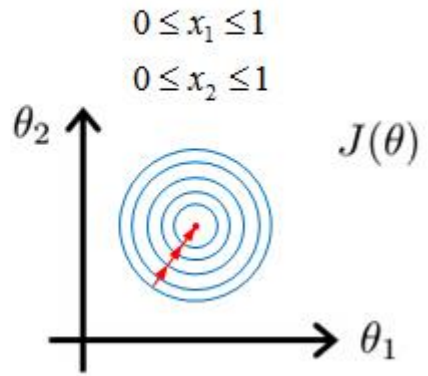
Gradient descent



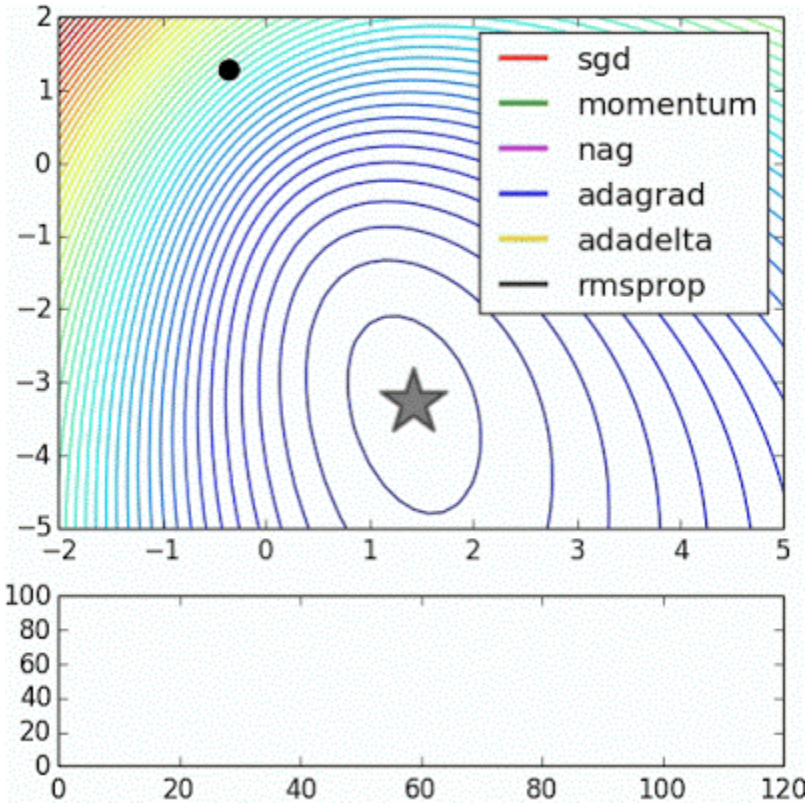
Important:

- Scale features before applying gradient descent (min-max normalization or mean-std normalization)
- Choose learning rate α wisely.





More advanced gradient descent



More advanced gradient descent: SGD

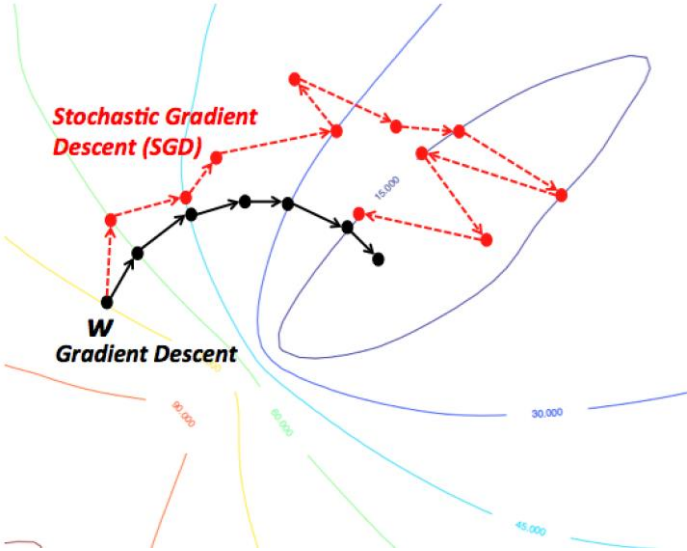
Instead of computing for the weights update

$$\frac{\partial E(W)}{dw_j} = \frac{1}{\mathrm{m}} \sum_{i=1}^{m} \left(W^T X^i - l^i\right) X_j^i$$

Use an estimation computed with a random sample of training examples. This random sample is named batch of training examples.

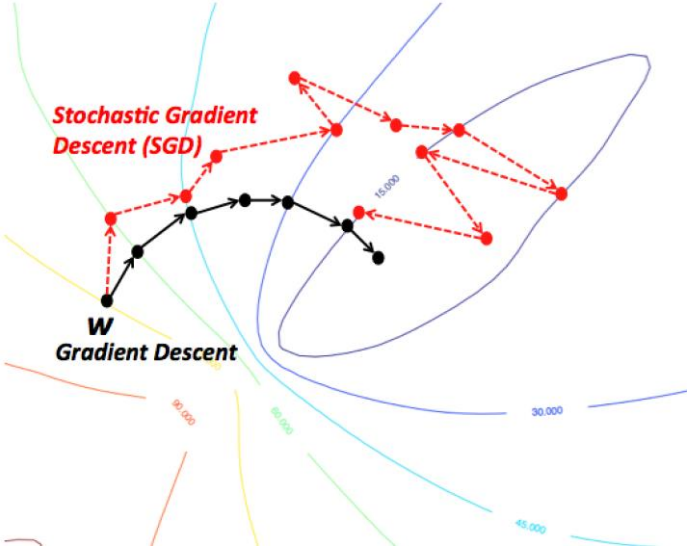
$$\frac{\partial E(W)}{dw_j} \approx \frac{1}{b} \sum_{i=1}^{b} \left(W^T X^i - l^i\right) X_j^i \qquad \mathrm{b} \ll m$$

And apply the regular gradient algorithm. It is generally convenient to slowly decrease learning rate as the number of iterations grows.



SGD vs GD

- Computing $\frac{\partial E(W)}{dw_j}$ can take long time if m is big, estimation is fast.
- SGD generally converges faster than GD.
- SGD is more difficult to tune, batch size is a new hyper-parameter.
- SGD is able to scape from some local minima due to randomness.



Easiest model for classification: Logistic regression. We are close to neural nets, I promise

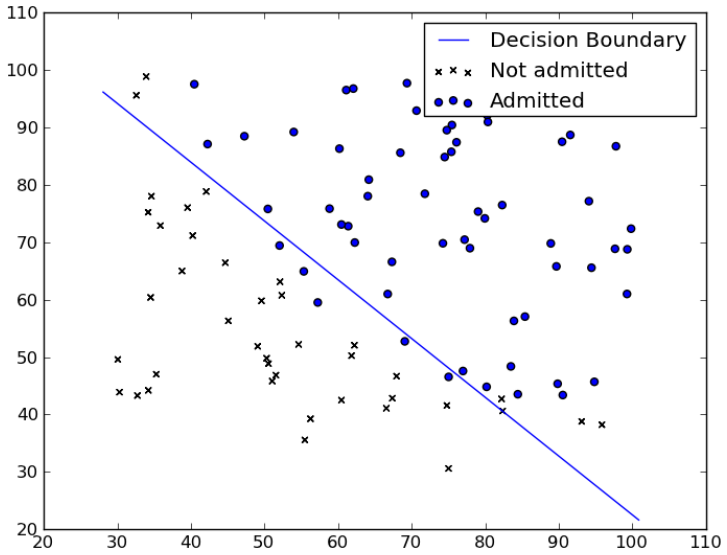
$$y(x_1, x_2, \dots, x_n) = g(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n) = \frac{1}{1 + e^{-W^T X}}$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$W = \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{pmatrix}$$

$$X = \begin{pmatrix} 1 \\ w_1 \\ \dots \\ w_n \end{pmatrix}$$

Example number	Exam 1 score	Exam 2 score	Admitted
1	20	20	0
2	40	90	1
…	…	…	…
N	60	60	1

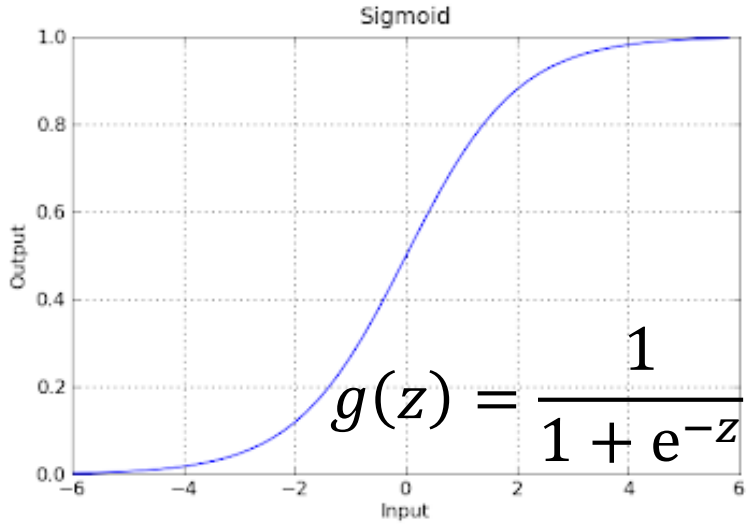


Easiest model for classification: Logistic regression.

$y(x_1, x_2, \dots, x_n)$

Can be interpreted as the probability that a given example will belong to the category 1 $P(y = 1 \mid X, W)$

In order to get categorical predictions, a decision threshold must be set. If no prior information 0.5 is chosen.



Easiest model for classification: Logistic regression.

$$y(x_1, x_2, \dots, x_n) = \frac{1}{1 + \mathrm{e}^{-W^T X}}$$

$$E(W) = cross - entropy = -\frac{1}{m}\left[\sum_{i=1}^{m} l^i \log\left(y\left(X^i\right)\right) + \left(1 - l^i\right)\log\left(1 - y\left(X^i\right)\right)\right]$$

Find *W* such that E(*W*) is minimum

$$W = \underset{W}{\operatorname{argmin}} -\frac{1}{m}\left[\sum_{i=1}^{m} l^i \log\left(y\left(X^i\right)\right) + \left(1 - l^i\right)\log\left(1 - y\left(X^i\right)\right)\right]$$

$E \rightarrow error\ of\ the\ model$
$l_i \rightarrow\ value\ of\ example\ i$ (label)
$x_i \rightarrow features\ of\ example\ x$
m$\rightarrow number\ of\ examples$

Easiest model for classification: Logistic regression.

$$W = \underset{W}{\operatorname{argmin}} -\frac{1}{m}[\sum_{i=1}^{m} l^i \log\left(y\left(X^i\right)\right) + \left(1 - l^i\right) \log\left(1 - y\left(X^i\right)\right)]$$

To obtain W, use gradient descent as before. For cross entropy with sigmoid activation, the derivative of the error is:

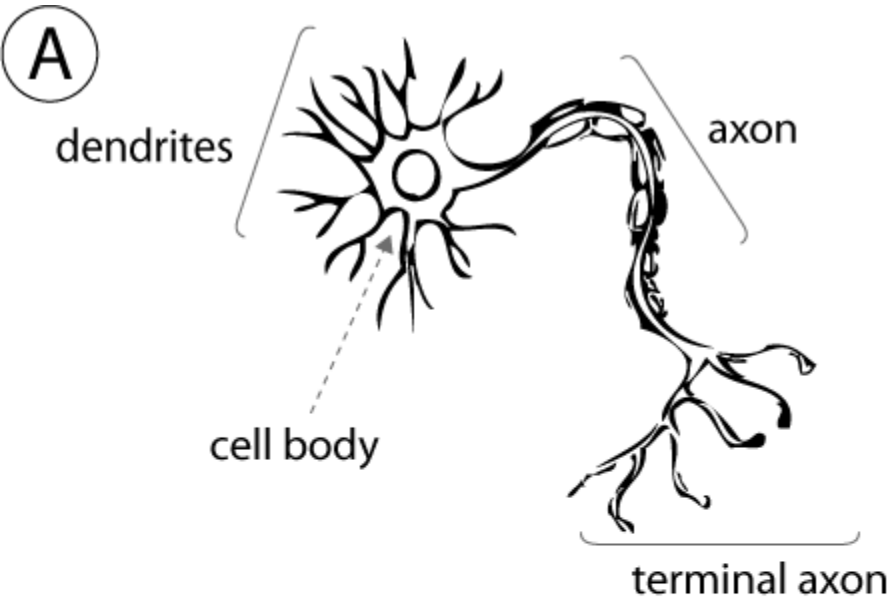
$$\frac{\partial E(W)}{dw_j} = \frac{1}{\mathrm{m}} \sum_{i=1}^{m} \left(\frac{1}{1 + \mathrm{e}^{-W^T X^i}} - l^i \right) X_j^i$$

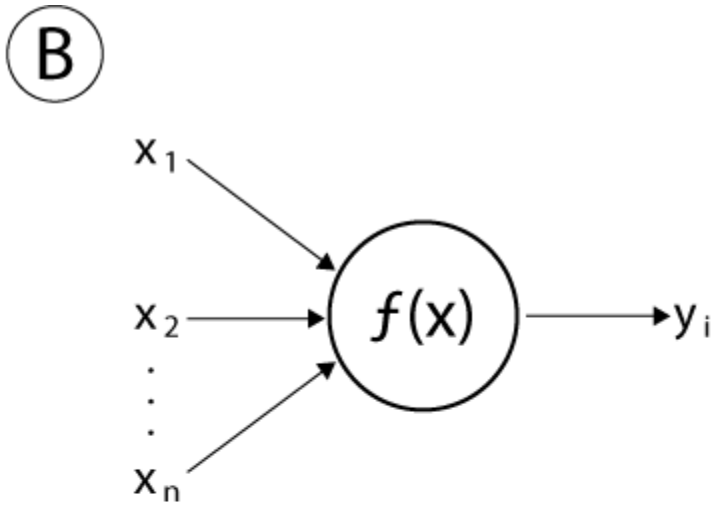
A simple way to diminish overfitting: regularization

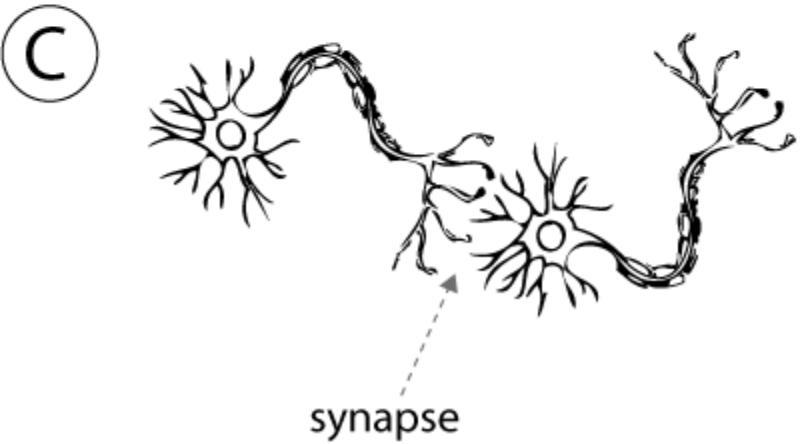
Add a regularization term to the error function. This term will prevent weights from growing to much, constraining the model.

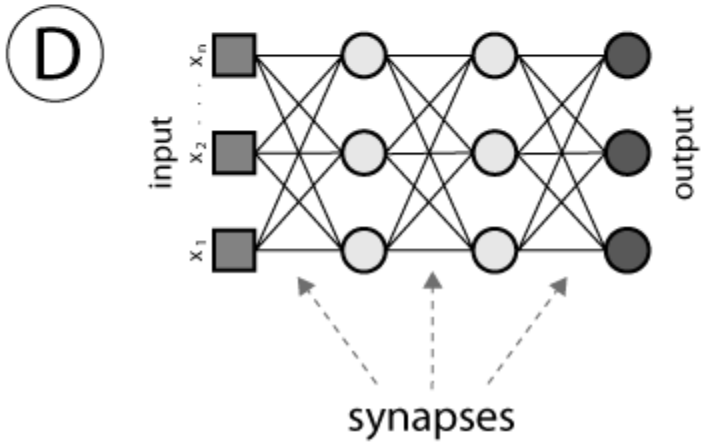
$$E(W) = -\frac{1}{m}[\sum_{i=1}^{m} l^i \log\left(y\left(X^i\right)\right) + \left(1 - l^i\right)\log\left(1 - y\left(X^i\right)\right)] + \lambda \sum_{j=1}^{n} w_j^2 \qquad \lambda \geq 0$$

ARTIFICIAL NEURAL NETWORKS

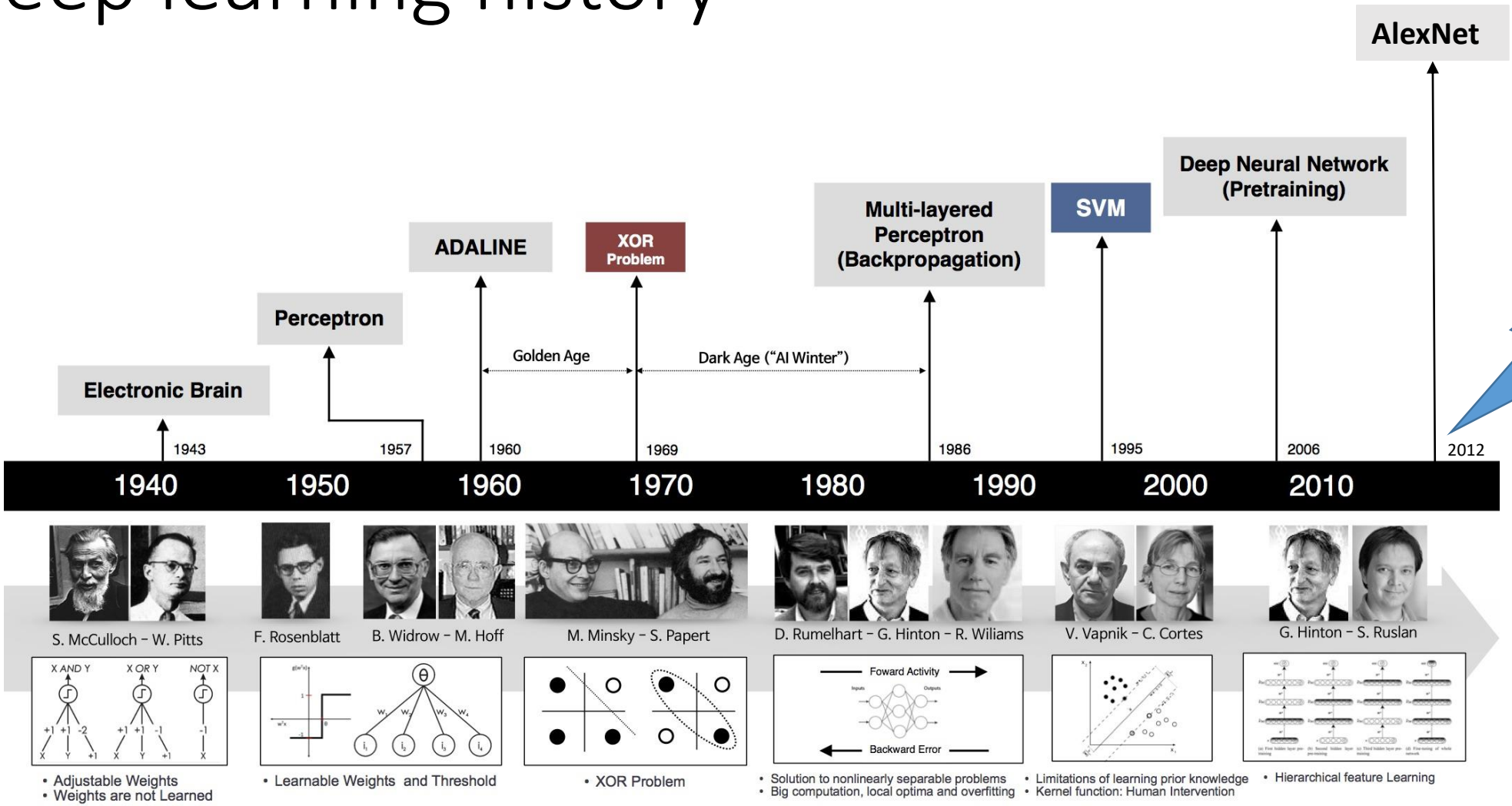








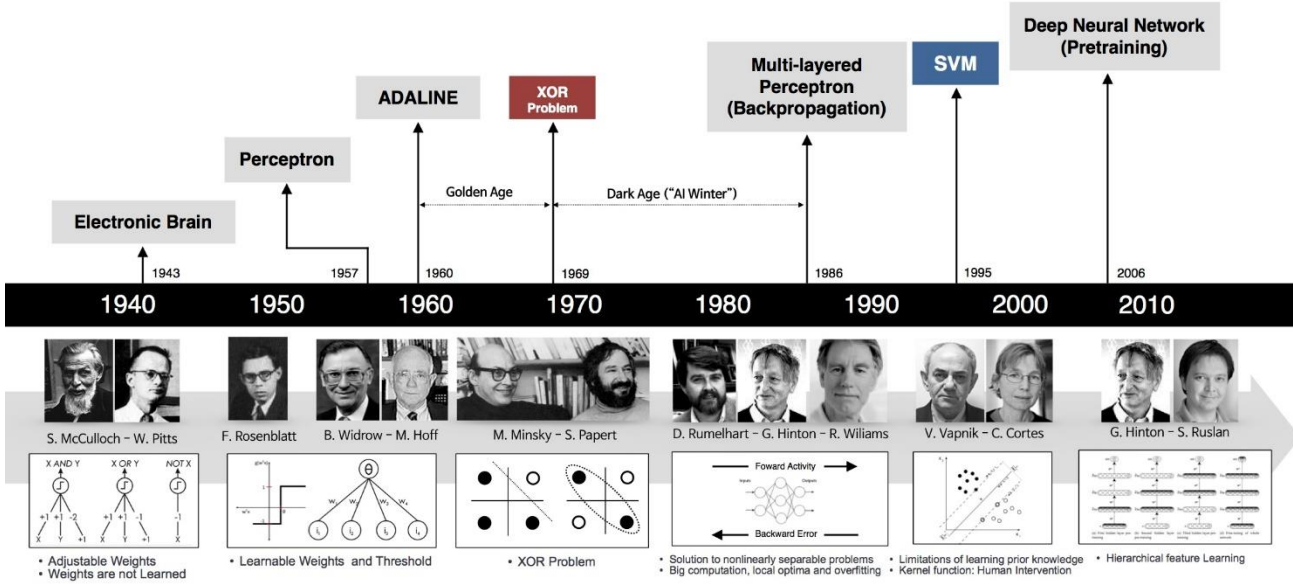
Deep learning history



Deep Learning become trendy

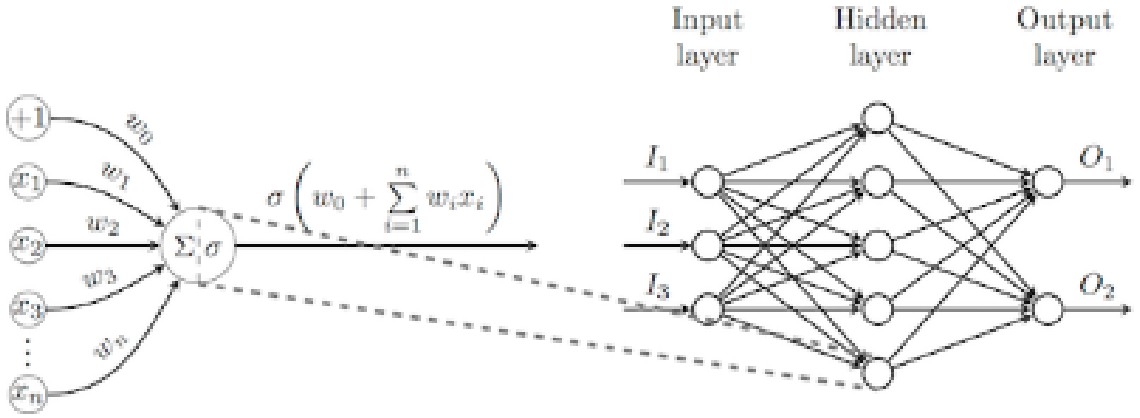
VGG-net
Inception
ResNet
U-net…

Deep learning history



- 1940- 1950. Neurons as weighted sum of inputs thresholded
- 1950-1960. Perceptron as combinations of several neurons. Still no perfect algorithm to train them
- 1969. Perceptron can not learn the XOR function. Neural net research became minoritarian for 15 years.
- 1980- 1990. **Multilayer perceptron** and **back propagation**. The root of modern neural networks.
- 1998. **Convolutional networks** were invented. LeNet.
- 2006 – 2012. Neural networks field was renamed to Deep Learning when Deep Networks were able to be trained by layer-wise pretraining.
- 2012-now. **AlexNet** showed that direct training of Deep Convolutional Networks was possible and they perform better than any other known method. Deep Learning age.

Multilayer Perceptron



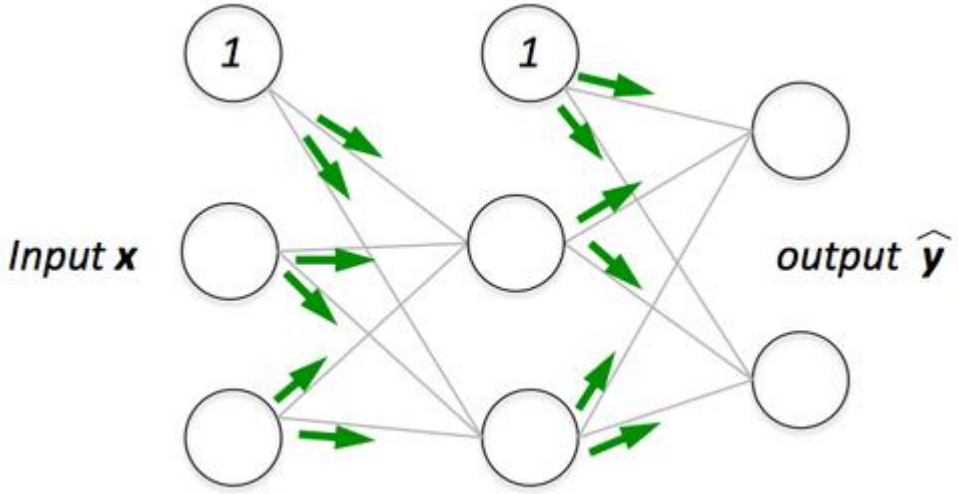
- Also called fully connected networks.

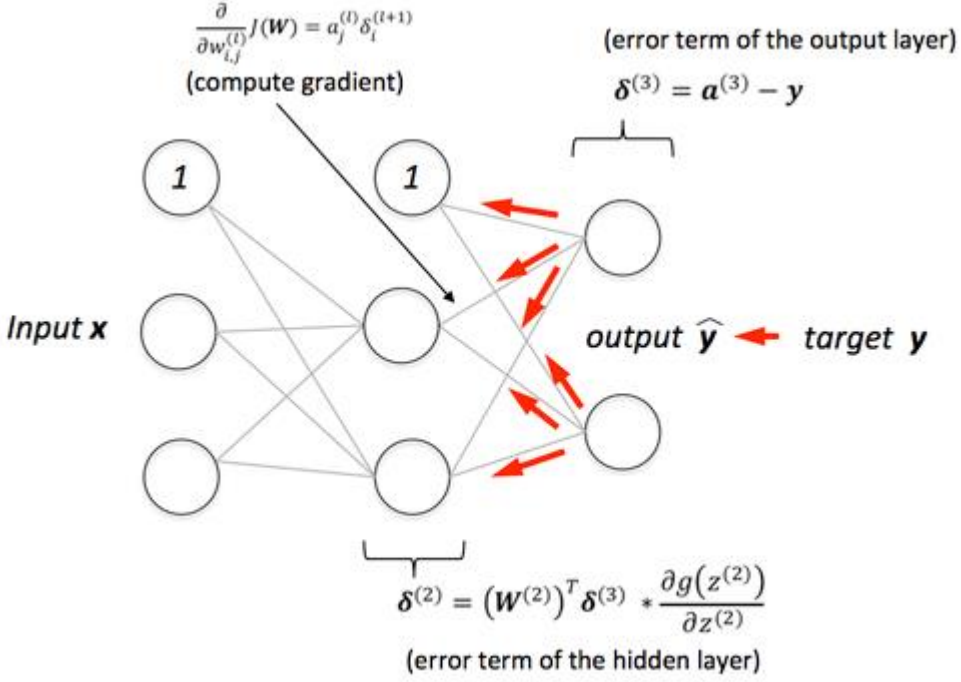
- At least three layers:
  - Input layer. As many units as dimensions the input have
  - Hidden layer(s).
    - As many hidden layers as desired with as many neurons as desired. Each neuron is equivalent to an individual logistic regression.
    - Each neuron of a hidden network is connected with all the neurons of the previous and the next layers.
  - Output layer . As many units neurons as dimensions the ouput have. For classification, softmax is generally applied in order to normalize probabilities.

$$s(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Multilayer Perceptron (MLP)

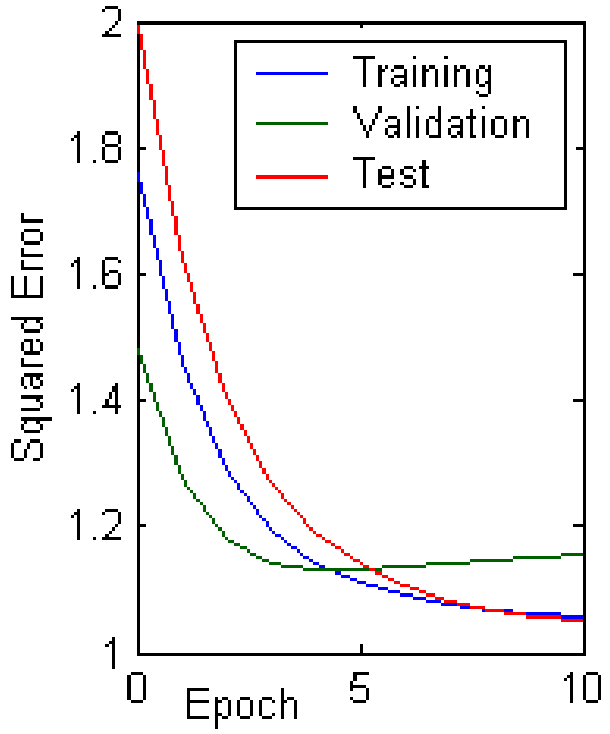
- Trained by gradient descent (better SGD).
  - The gradient of the error, needed for gradient descent is calculated by means of back-propagation, which it is nothing but the use of the calculus chain rule.
  - In back propagation, first a output for a sample is computed (forward pass)
  - Then the error is back-propagated





Multilayer Perceptron (MLP)

- Generally, weights are random initialized using a Gaussian distribution (0,0.1)
- Training a deep MLP is generally no so easy, as logistic functions can saturate.
  - Use other activation function.
- MLPs have so many parameters that they tend to overfit.
  - Inspect learning curves and use early stopping
  - Use regularization
  - Other tricks we will discuss later.



The first convolutional network: LeNet

- As there are substructure in images (a path of an image is still an image, and neighbor pixels are related) fully connected architectures are not needed to analyze them.
- Instead use locally connected networks that inspect patches instead the full image → equivalent to convolution.

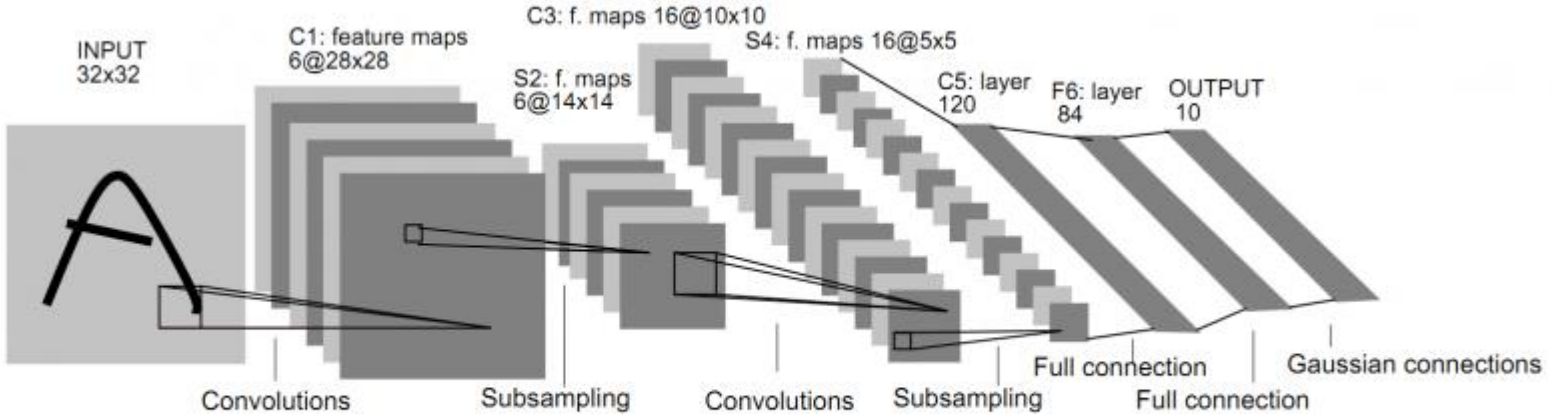


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

The first convolutional network: LeNet

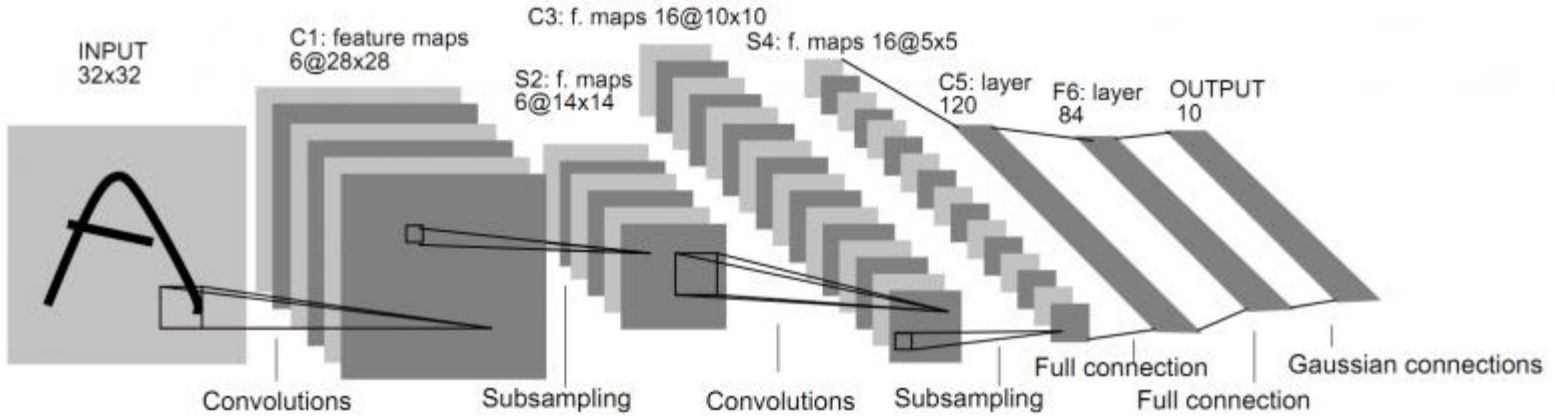


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429

Feature Map

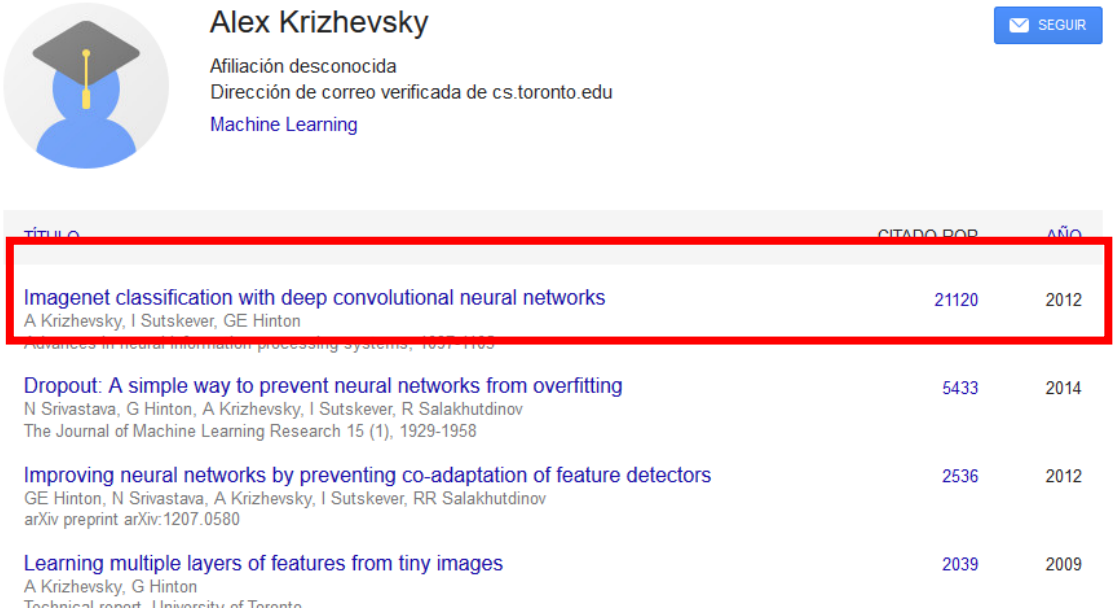
6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

Max-Pooling

The origin of modern Deep Learning: AlexNet

IMAGENET

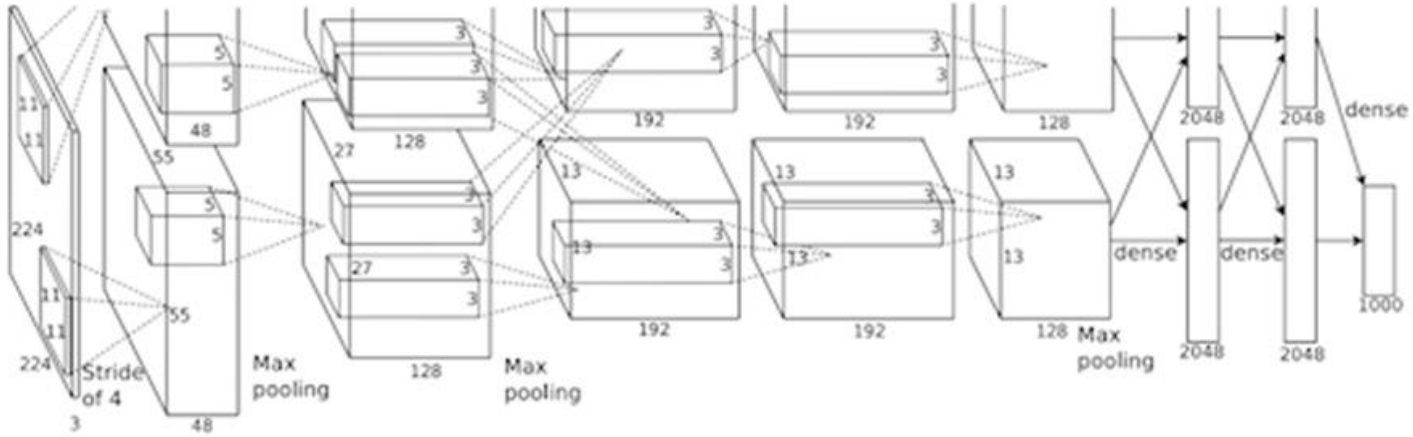
- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



AlexNet won 2012 ImageNet competition (1000 thousand categories several millions images) lowering error to 15.4 % (second best error was 26.2%). No neural net won ImageNet until AlexNet. From now on, Deep Learning solutions have always won the conquest.

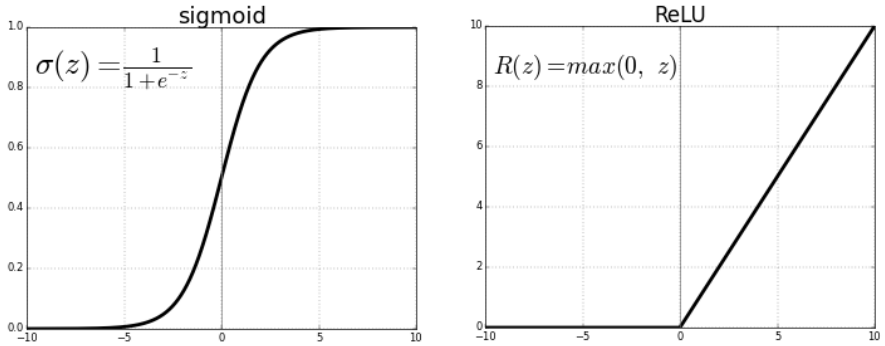
The origin of modern Deep Learning: AlexNet

- AlexNet is a convolutional neural network of 5 convolution layers combined with max-pooling and a final 2 fully connected layer-classifier.
- They employ revolutionary tricks that changed the field, such as:
  - Data augmentation
  - Stochastic gradient descent trained on GPU
  - ReLU as activation function (no need layer wise pretraining)
  - Dropout

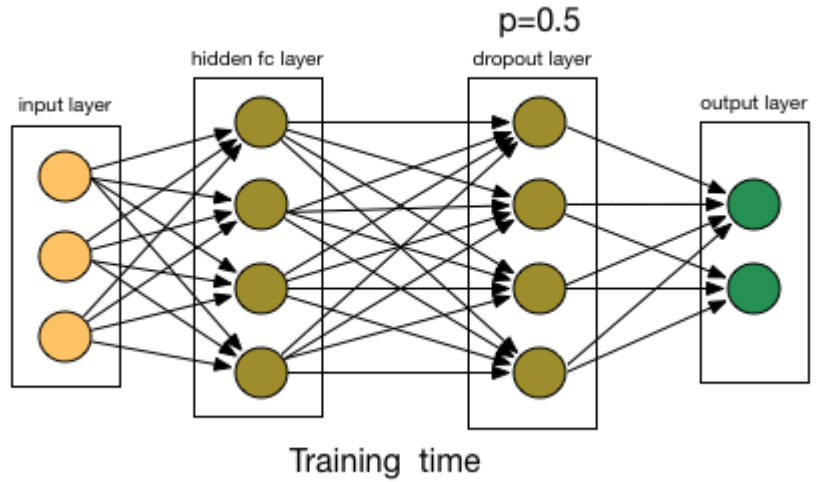


The origin of modern Deep Learning: AlexNet

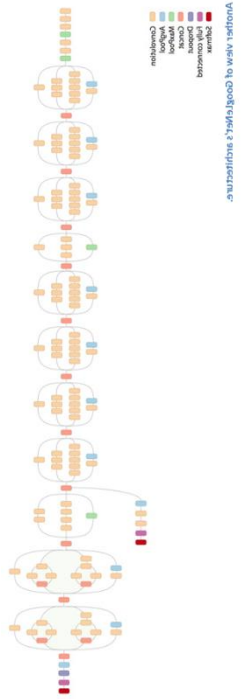
- Relu. Makes training easier an faster as it does not saturate



- Dropout: Regularization technique. Use it if you have overfitting

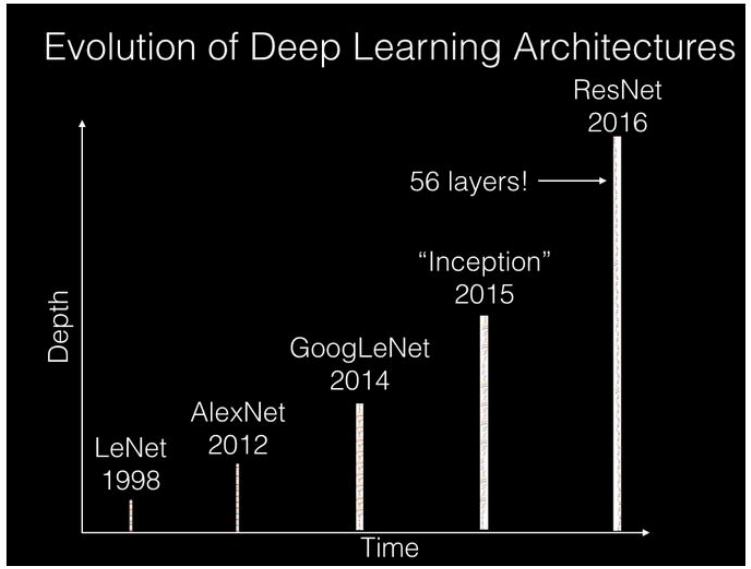


Other famous Neural Nets and tricks



- Neural Nets:
  - VGG net. Pretty simple convNet of 16 layers that employs only 3x3 filter size. Several convs before max pooling.
  - Google Net (Inception). It won ImageNet 2014. Very complicated convNet with branches.
  - ResNet. It won ImageNet 2015. Similar to VGG net but with skipping connections.
  - Faster R-CNN. Used to localize objects in images, not to classify full images.
  - LSTMs. Recurrent networks suited for sequence analysis.
  - Generative Adversarial Networks. One network is train to classify and other to confuse the first network.
  - …

- Tricks:
  - Batch Normalization.
  - 1d convolution.
  - Residuals (skipping connections).
  - Fully convolution networks for image segmentation.
  - …



Practical examples with Tensorflow and Keras

To be continued....

Modern architectures and tricks: INCEPTION, RESNET, batch normalization…

Deep Learning for other than classification tasks: image segmentation, time series analysis, clustering…

Suggested topics …

or not?