Introduction to Big Data

Zhuoyan Xu

University of Wisconsin-Madison

zxu444@wisc.edu

Feb 6,2020

Outline

Big Data

Figure: https://www.analyticsinsight.net/10-parameters-for-big-data

What is Big Data?

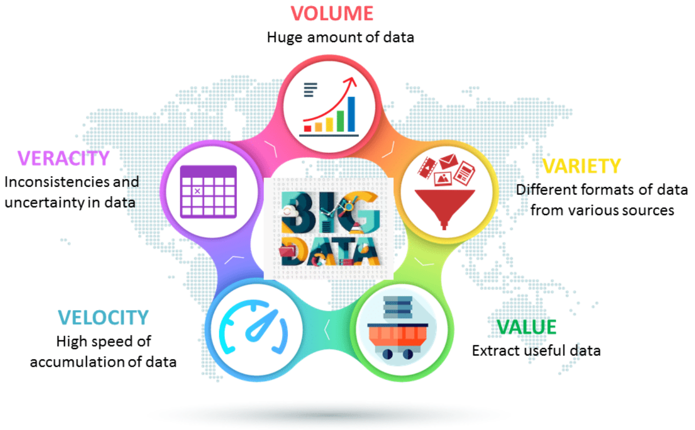


Figure: source:https://dlpng.com/png/5446068

Machine Learning

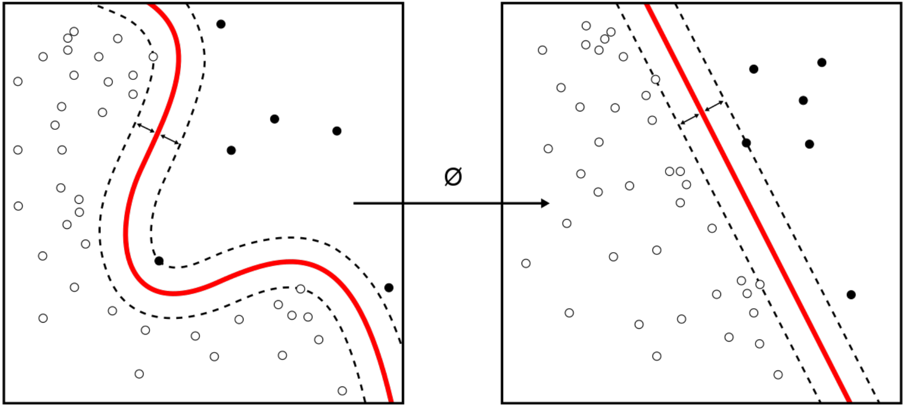


Figure: `en.wikipedia.org/wiki/`

Machine Learning

"Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed"

— Arthur L. Samuel, AI pioneer, 1959

Traditional way

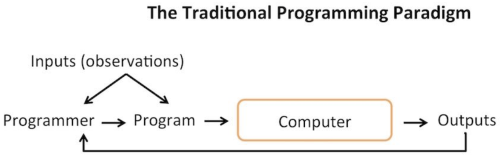


Figure: https://github.com/rasbt/stat479-machine-learning-fs18

e.g.

$$\texttt{Body Mass Index (BMI)} == \frac{\text{mass}_{\,\mathrm{kg}}}{\text{height}^{\,2}_{\,\mathrm{m}}} = \frac{\text{mass}_{\,\mathrm{lb}}}{\text{height}^{\,2}_{\,\mathrm{in}}} \times 703$$

Machine Learning

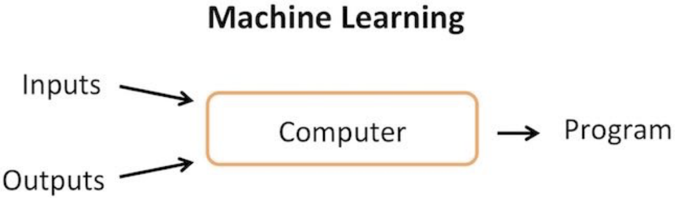


Figure: https://github.com/rasbt/stat479-machine-learning-fs18

Applications of Machine Learning

- Email spam detection
- Face recognition
- Self-driving cars
- Language translation
- Recommendation system

Categories of Machine Learning

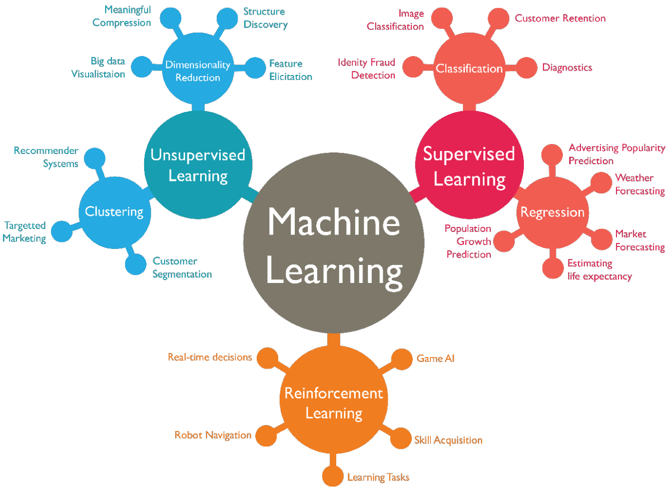


Figure:
https://www.wordstream.com/blog/ws/2017/07/28/machine-learning

Outline

1. Overview
2. Basic Terminology
3. Deep learning
4. Optimization
5. Convolutional Neural Network(CNN)

Probability

Definition

A numerical description of how likely an event is to occur.

Example

- Flipping a fair coin with the head up.
- Rolling a dice and get three points up.

(a) Head of coin

(b) Dice

Figure: e.g.

Random Variable

Definition

A variable whose values depend on outcomes of a random phenomenon.

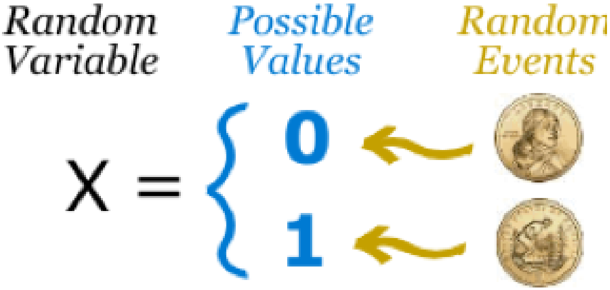
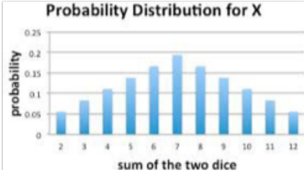


Figure: `https://www.mathsisfun.com/data/random-variables.html`

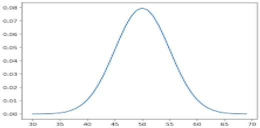
Probability distribution

Definition

A mathematical function that provides the probabilities of occurrence of different possible outcomes in an experiment.



(a) Head of coin



(b) Dice

Figure: e.g.

Expectation

Definition

The expected value of a discrete random variable is the probability-weighted average of all its possible values.

Example

Die-rolling game: \$30 for once

Rolled number	award (\$)
1	0
2	0
3	0
4	0
5	40
6	80

Estimator

Definition

A rule for calculating an estimate of a given quantity based on observed data.

Example

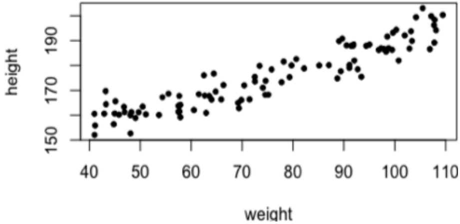
Q: What is mean depth of lakes in Wisconsin?

- Random guessing: 100m.
- Sample 100 lakes randomly from Wisconsin, take the sample mean as the estimator.

Linear Regression

Model Idea

- Consider the model of Y conditional on $X = x$.
- Predict response (y) based on predictors (x).



Linear Regression

Model

$$y_i = f(x_i) = \beta_0 + \beta_1 x_i + \epsilon_i, \quad \epsilon_i \sim i.i.d.N\left(0, \sigma^2\right)$$

Prediction

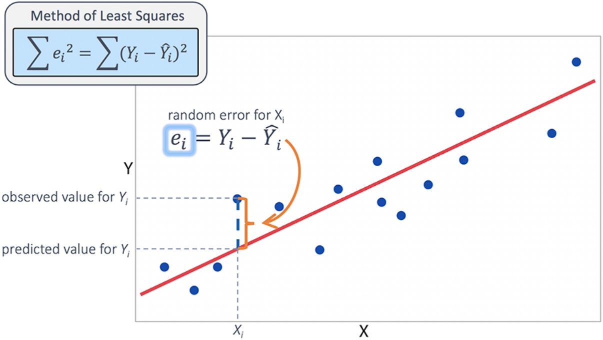
$$y_{new} = \beta_0 + \beta_1 x_{new}$$

Lost function

Residual sum of squares / Sum of Square error (SSE)

Let $\hat{y}_i = f(\hat{x}_i) = \beta_0 + \beta_1 \hat{x}_i$

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



Ordinary Least Squared Estimator (OLSE)

LSE

Estimator of parameter β_0, β_1 that minimize SSE.

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)\left(Y_i - \bar{Y}\right)}{\sum_{i=1}^{n}\left(X_i - \bar{X}\right)^2}$$

$$\hat{\beta}_0 = \frac{1}{n}\left(\sum_{i=1}^{n} Y_i - \hat{\beta}_1 \sum_{i=1}^{n} x_i\right) = \bar{Y} - \hat{\beta}_1 \bar{X}$$

Bias and Variance general definition

Bias

The bias of an estimator is the difference between this estimators expectation and the true value of the parameter being estimated.

$$\text{Bias}(\hat{\theta}) = E[\hat{\theta}] - \theta$$

Variance

Informally, it measures how far a set of (random) numbers are spread out from their average value.

$$\text{Var}(\hat{\theta}) = E\left[(E[\hat{\theta}] - \hat{\theta})^2\right]$$

Overfitting and underfitting

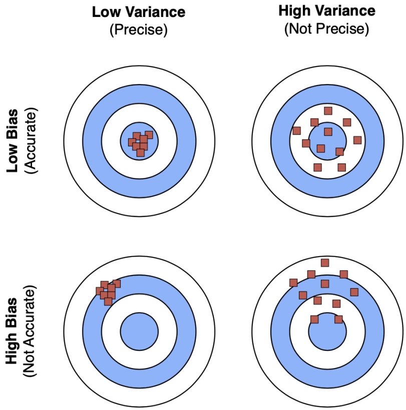


Figure 4: Bias-variance intuition.

Figure: `https://github.com/rasbt/stat479-machine-learning-fs18`

Overfitting and underfitting

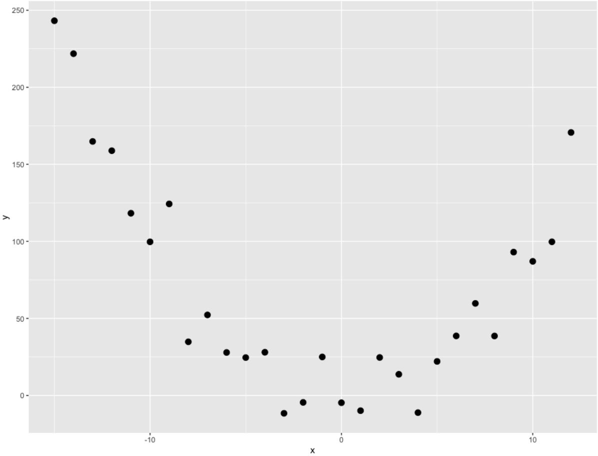


Figure: train data

Overfitting and underfitting

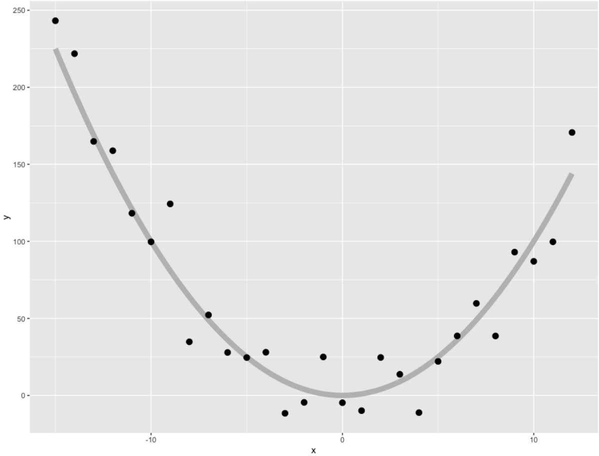


Figure: Good fit

Overfitting and underfitting

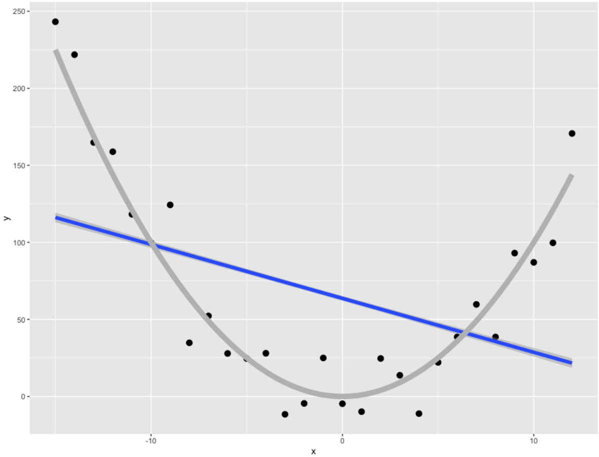


Figure: Underfitting; High bias

Overfitting and underfitting

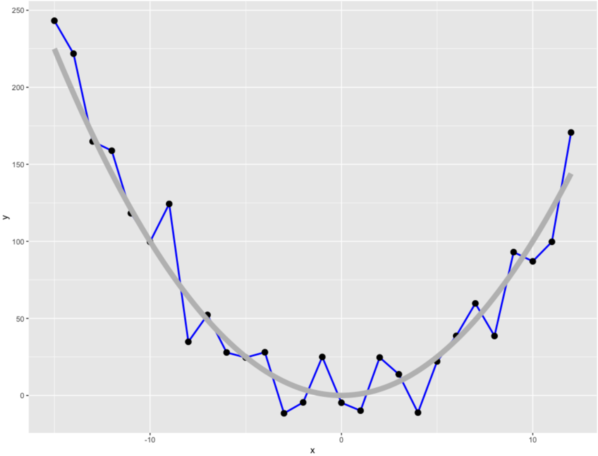


Figure: Overfitting; High variance

Outline

Neural Network

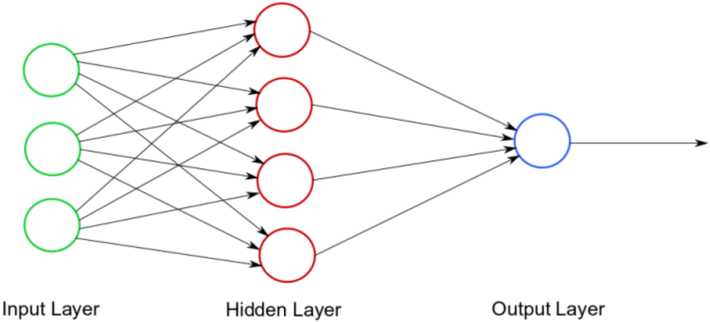


Figure: Neural Network

Perceptron

Inspired by Biological Brains and Neurons

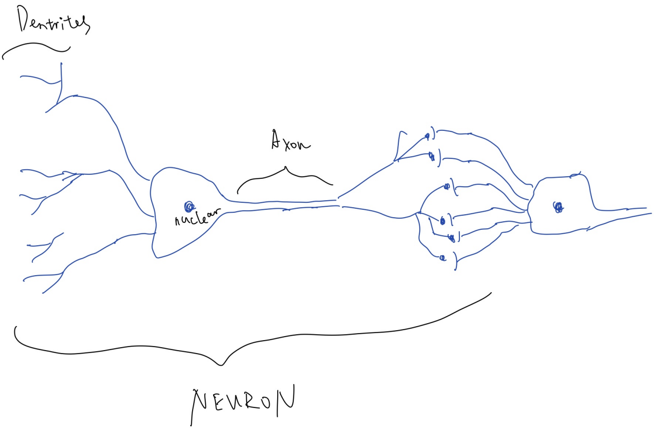


Figure: Biological Neuron

Perceptron

Inspired by Biological Brains and Neurons

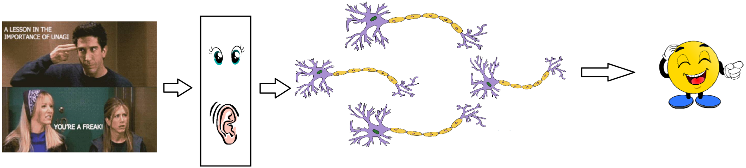
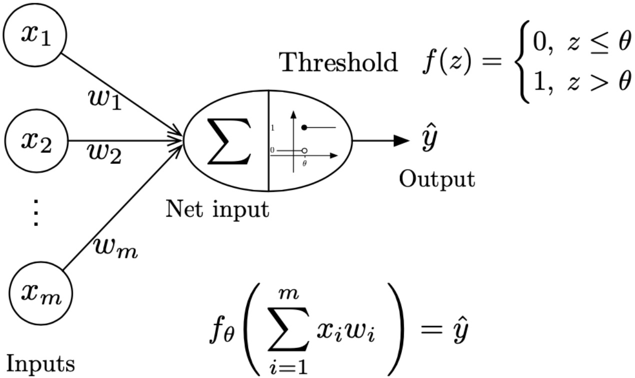


Figure: Biological Neuron

Perceptron

Terminology

- Net input = weighted inputs.
- Activations = activation function(net input).
- Label output = threshold(activations of last layer).



Perceptron Output

$$\hat{y} = \begin{cases} 0, z - \theta \leq 0 \\ 1, z - \theta > 0 \end{cases}$$

We call $-\theta$ bias.

Geometric Intuition

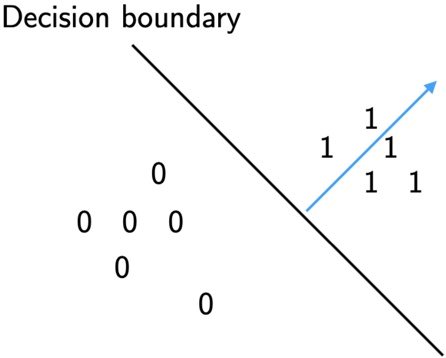
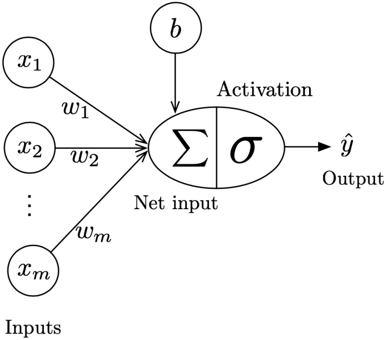


Figure: Single Layer Neural Network

$$\hat{y} = \begin{cases} 0, \mathbf{w}^T\mathbf{x} \leq 0 \\ 1, \mathbf{w}^T\mathbf{x} > 0 \end{cases}$$

Single Layer Neural network



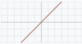
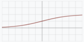
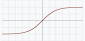
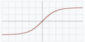
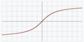
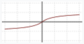
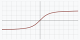
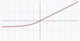
$$\sigma\left(\sum_{i=1}^{m} x_i w_i + b\right) = \sigma\left(\mathbf{x}^T \mathbf{w} + b\right) = \hat{y}$$

$$\sigma(z) = \begin{cases} 0, & z \leq 0 \\ 1, & z > 0 \end{cases}$$

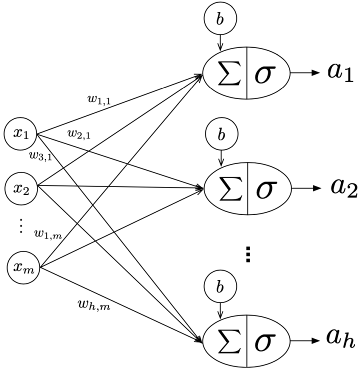
$$b = -\theta$$

Figure: Single Layer Neural Network

Activation function

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$ [1]
TanH		$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
SQNL[10]		$f(x) = \begin{cases} 1 & : x > 2.0 \\ x - \frac{x^2}{4} & : 0 \leq x \leq 2.0 \\ x + \frac{x^2}{4} & : -2.0 \leq x < 0 \\ -1 & : x < -2.0 \end{cases}$
ArcTan		$f(x) = \tan^{-1}(x)$
ArSinH		$f(x) = \sinh^{-1}(x) = \ln\left(x + \sqrt{x^2 + 1}\right)$
ElliotSig[11][12] Softsign[13][14]		$f(x) = \frac{x}{1 + \lvert x \rvert}$
Inverse square root unit (ISRU)[15]		$f(x) = \frac{x}{\sqrt{1 + \alpha x^2}}$
Inverse square root linear unit (ISRLU)[15]		$f(x) = \begin{cases} \frac{x}{\sqrt{1 + \alpha x^2}} & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Rectified linear unit (ReLU)[16]		$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$

Fully connected layer



where
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{h,1} & w_{h,2} & \cdots & w_{h,m} \end{bmatrix}$$

Layer activations for 1 training example

$$\sigma(\mathbf{Wx} + \mathbf{b}) = \mathbf{a}$$
$$\mathbf{a} \in \mathbb{R}^{h\times 1}$$

note that $w_{i,j}$ refers to the weight connecting the j-th input to the i-th output.

Figure: FC layer

Neural Network

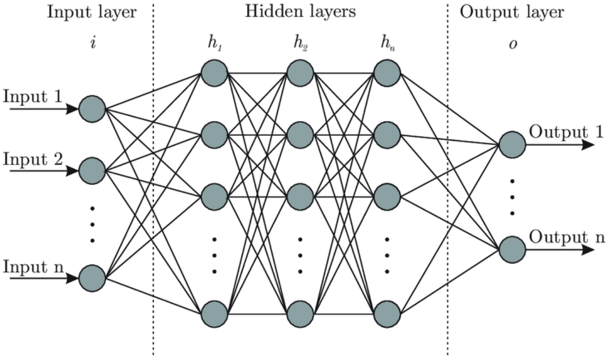


Figure: Neural Network

Why Neural Network can approximate any continunous function?

Universal approximation theorem

a feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n, under mild assumptions on the activation function.

Why sometimes we need deeper architectures?

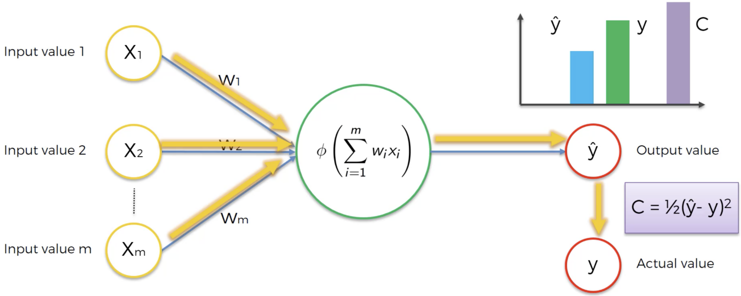
According to universal approximation theorem, we can put one layer and many nodes in the one hidden layer, the neural network can fit any function.

Why sometimes we need more than one hidden layers?

Breadth and depth

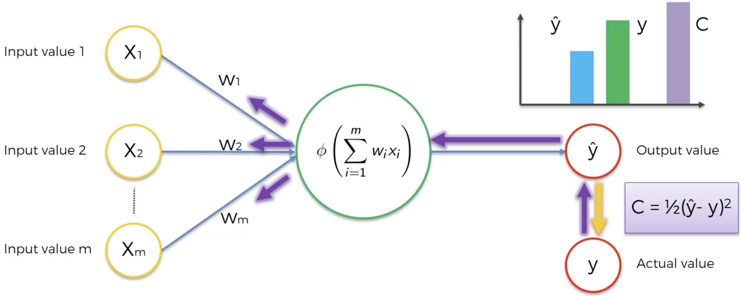
- Can achieve the same expressiveness with more layers but fewer parameters ; fewer parameters $\Longrightarrow$ less overfitting.
- having more layers provides some form of regularization: later layers are constrained on the behavior of earlier layers; regularization $\Longrightarrow$ less overfitting.
- However, more layers $\Longrightarrow$ vanishing/exploding gradients.

Forward propagation



Figure

Back propagation



Figure

Train a neural network

We want to find a model (neural net) that performs the best in some sense. A general way is minimizing the loss.

Loss:

- Squared error loss ($(y - \hat{y})^2$).
- Kullback Leibler divergence (KL divergence), known as cross entropy in classification problem.

Unfortunately, there is no closed form solution of weights for NN model.

Consider a object function

$$L(\beta) = \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$$

Goal:
find β_0, β_1 such that min $L(\beta)$.

Why not just try out?

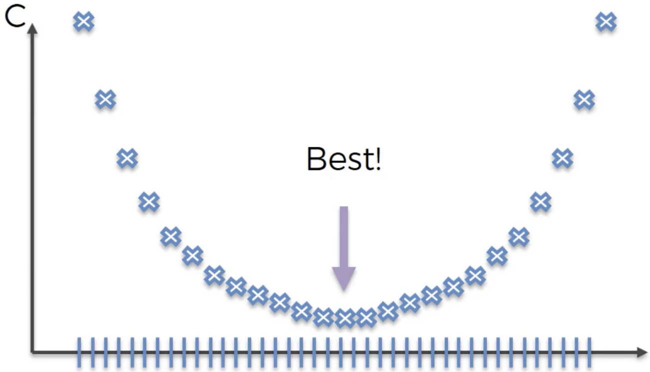
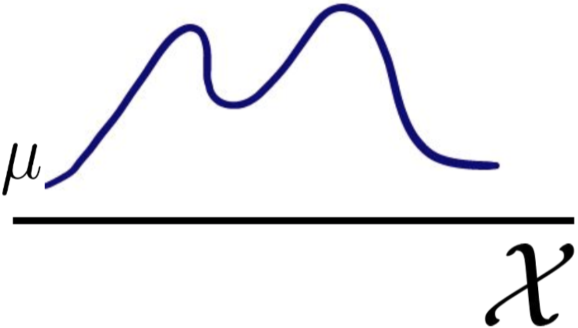


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6760390

Why not calculate directly?



Figure

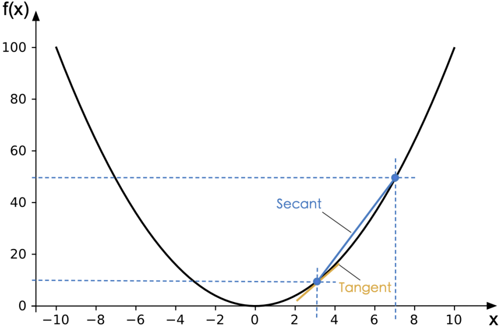
Gradient Descent

Recap:

Derivative = "slope"

$$f'(x) = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

tangent secant



Figure

Learning rate/stepsize

Learning rate determine how much we update at each step:

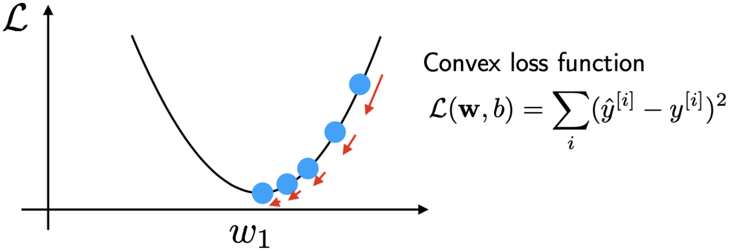


Figure: https://github.com/rasbt/stat479-deep-learning-ss19

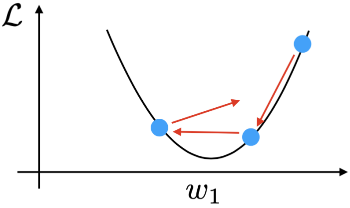
$$W_{k+1} = W_k - \eta \nabla L(W)$$

where η is learning rate.

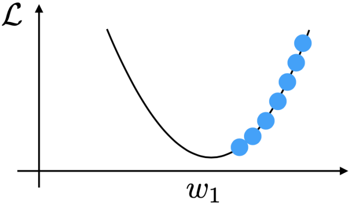
Learning rate

Choose learning rate:

If the learning rate is too large, we can overshoot



If the learning rate is too small, convergence is very slow



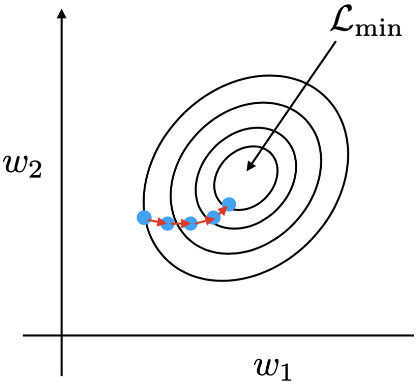
Stochastic gradient descent

Batch gradient descent: Update parameter after algorithm evaluating all the train sample.
SGD : Update parameter after algorithm evaluating one train sample.
Advantage:

- Some randomness to avoid the local minimum.
- Computation efficiency.

Some thing between these two:Mini-batch gradient descent.

Intuition



(a) Batch gradient descent (b) Stochatic gradient descent

Figure: e.g.

Paradigm

STEP 1: Randomly initialise the weights to small numbers close to 0 (but not 0).

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Regularization

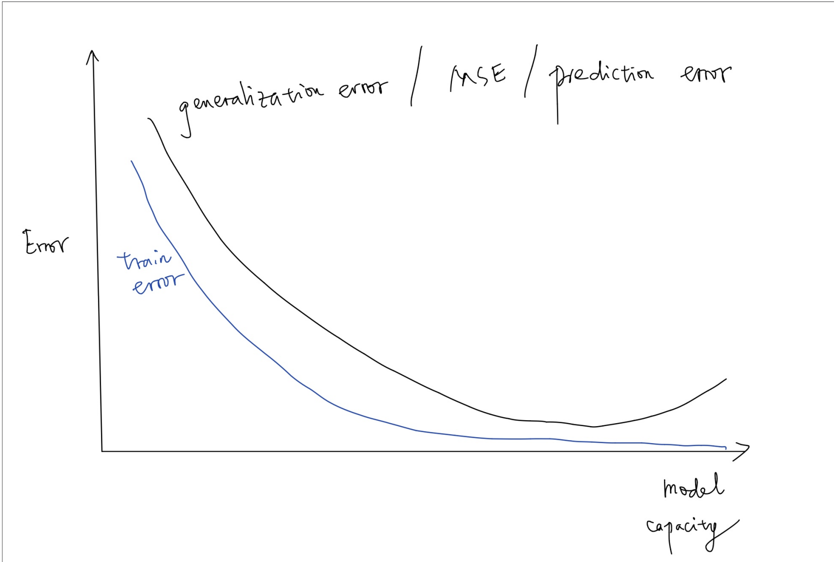
Goal: reduce overfitting

usually achieved by reducing model capacity and/or reduction of the variance of the predictions (as explained last lecture)

How to deal with overfitting?

- Collecting more data.
- If not possible, data augmentation is also helpful (e.g., for images: random rotation, crop, translation ...) – actually, this is always recommended (and easy to do).
- Additionally, reducing the model capacity (e.g., regularization) helps.

Overfitting vs capacity



Early stopping

1. Split your dataset into 3 parts: train set; validation set; test set (always recommended).
   - use test set only once at the end (for unbiased test of generalization performance).
   - use validation accuracy for parameter tuning.
2. Early stopping: reduce overfitting by observing the training/validation accuracy gap during training and then stop at the "right" point.

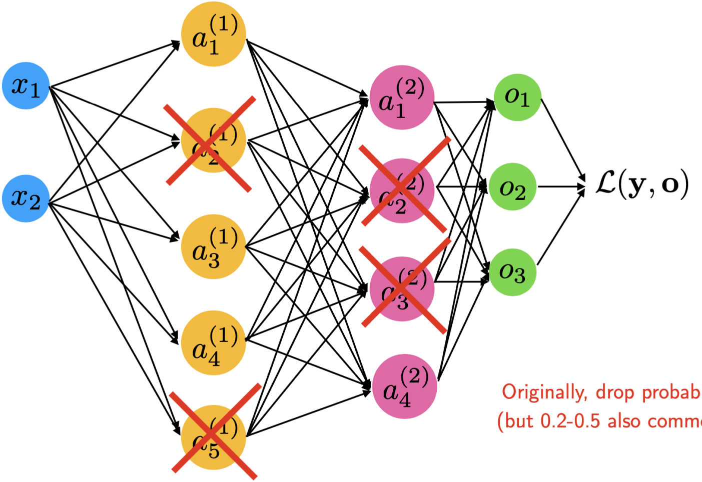
Dropout

Original research articles:

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

Dropout



Originally, drop probability 0.5
(but 0.2-0.5 also common now)

Figure: https://github.com/rasbt/stat479-deep-learning-ss19

Some techniques in optimization

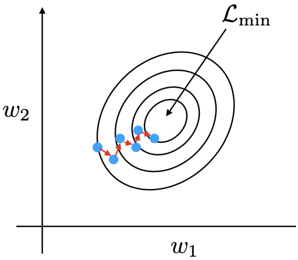


Figure: https://github.com/rasbt/stat479-deep-learning-ss19

Gradient is noiser:

- good: chance to escape local minimum.
- bad: can lead to extensive oscillation.

Learning rate decay

To dampen oscillations towards the end of the training, we can decay the learning rate:

$$
\begin{aligned}
\eta_t &:= \eta_0 \cdot e^{-k \cdot t} \\
\eta_t &:= \eta_{t-1}/2 \\
\eta_t &:= \frac{\eta_0}{1 + k \cdot t}
\end{aligned}
$$

Momentum method

Helps with dampening oscillations, also helps with escaping local minima traps

$$\Delta w_{i,j}(t+1) := \alpha \cdot \Delta w_{i,j}(t) + \eta \cdot \frac{\partial \mathcal{L}}{\partial w_{i,j}}(t)$$
$$w_{i,j}(t+1) := w_{i,j}(t) - \Delta w_{i,j}(t+1)$$

reference paper: *Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). Doklady ANSSSR (translated as Soviet.Math.Docl.), vol. 269, pp. 543– 547.*

Adaptive Learning Rates

- decrease learning if the gradient changes its direction
- increase learning if the gradient stays consistent.
- reference paper: *Igel, Christian, and Michael Hüsken. "Improving the Rprop learning algorithm." Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000). Vol. 2000. ICSC Academic Press, 2000.*

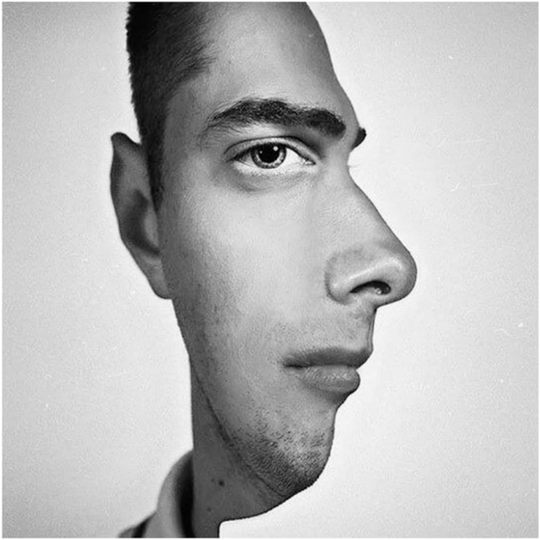
ADAM

Combination of momentum method and adaptive Learning Rates.
reference paper: *Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.*

ADAM (Adaptive Moment Estimation) is probably the most widely used optimization algorithm in DL as of today.

https://bl.ocks.org/EmilienDupont/aaf429be5705b219aaaf8d691e27ca87

Convolutional Neural Network



Convolutional Neural Network

Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Convolutional Neural Network

Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Overview of CNN

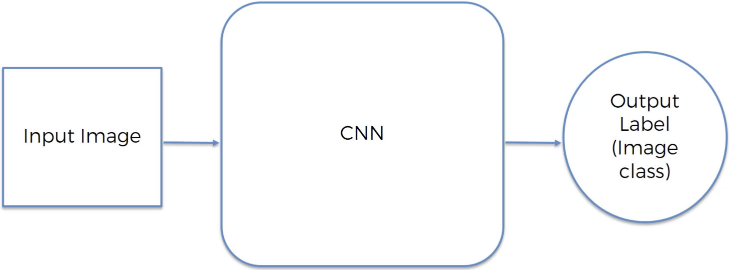


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Overview of CNN

Image Source: twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

Image Source: https://www.pinterest.com/pin/244742560974520446

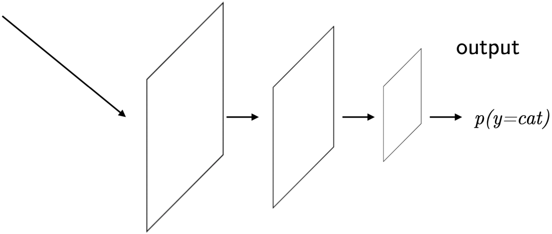


Figure: https://github.com/rasbt/stat479-deep-learning-ss19

Applications of CNN

Object Detection

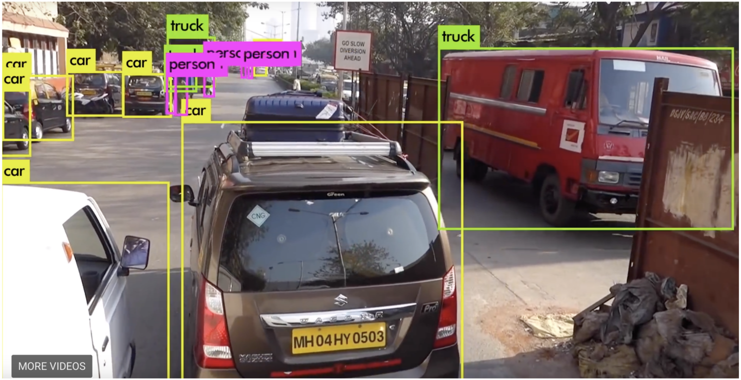


Figure: Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788).

Applications of CNN

Object Segmentation

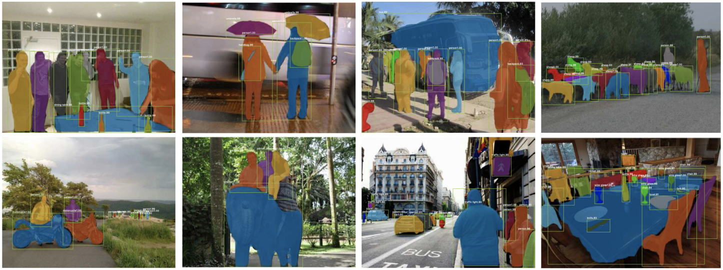


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [15], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Figure: He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN." In Proceedings of the IEEE International bigger gains under stricter localization metrics. Second, we Conference on Computer Vision, pp. 2961-2969. 2017

Pixels in Image

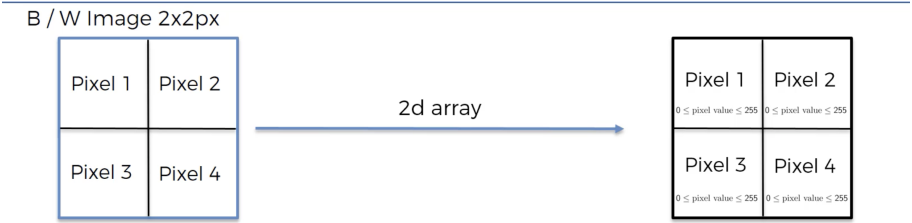


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pixels in Image

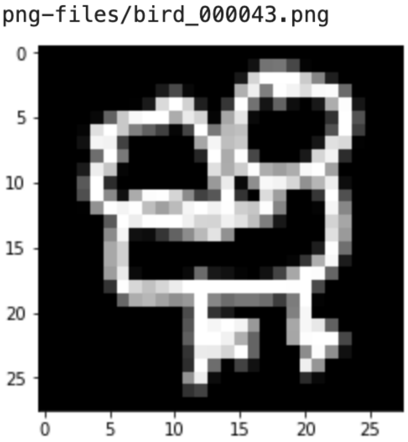


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pixels in Image

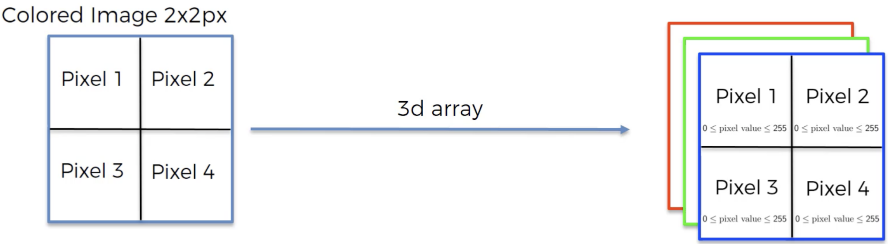
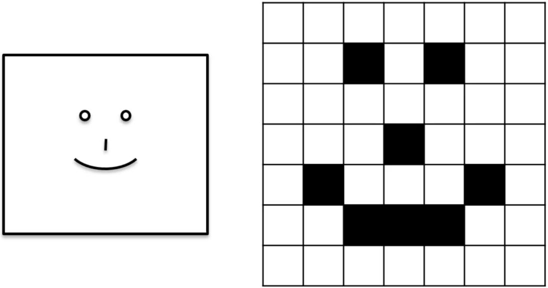


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pixels in Image



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

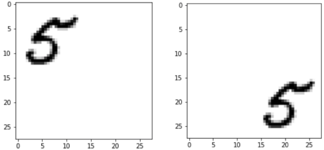
Why Image Classification is Hard

Different lighting, contrast, viewpoints, etc.

Image Source:
twitter.com%2Fcats&psig=AOvVaw30_o-PCM-K21DiMAJQimQ4&ust=1553887775741551

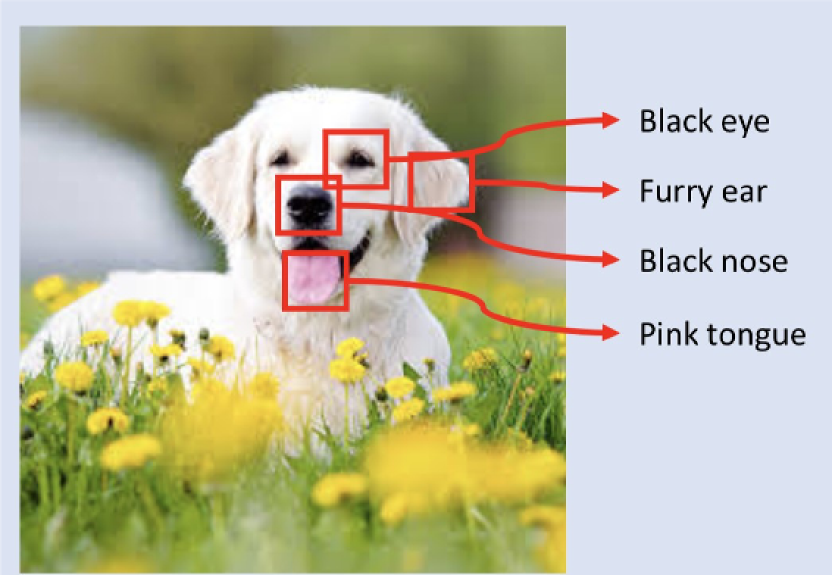
Image Source: https://www.123rf.com/photo_76714328_side-view-of-tabby-cat-face-over-white.html

Or even simple translation



This is hard for traditional methods like multi-layer perceptrons, because the prediction is basically based on a sum of pixel intensities

detect feature



Convolution

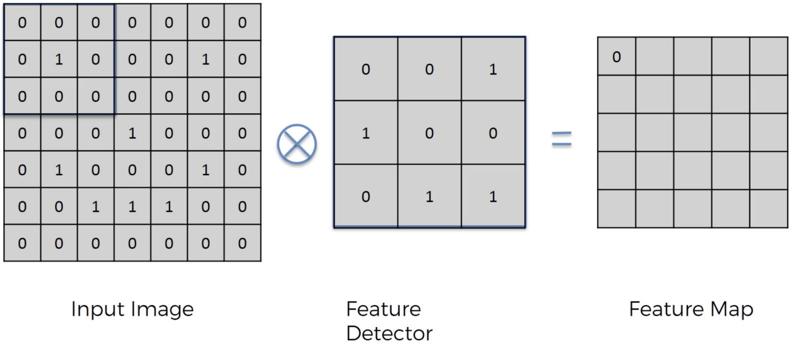


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

feature detector/filter/kernel feature map

Convolution

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

$\otimes$

0	0	1
1	0	0
0	1	1

Feature Detector

$=$

0	1			

Feature Map

Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

feature detector/filter/kernel feature map

Convolution

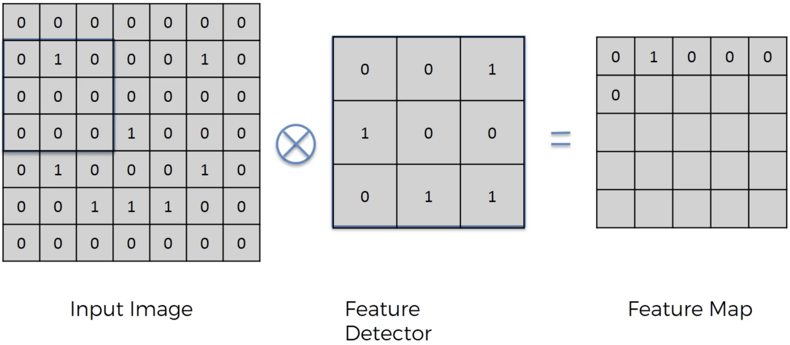


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

feature detector/filter/kernel feature map

Convolution

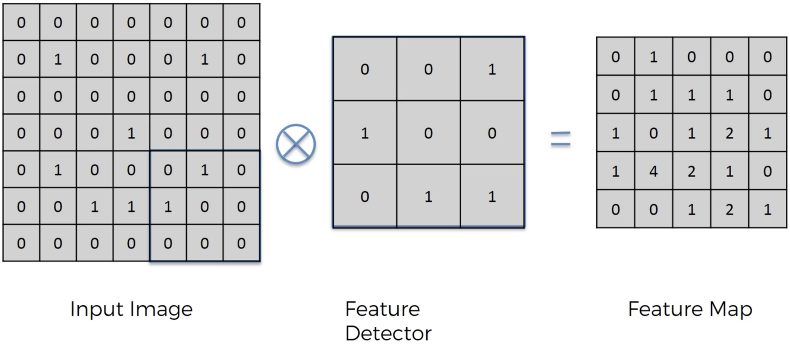


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

feature detector/filter/kernel feature map

Convolution layer

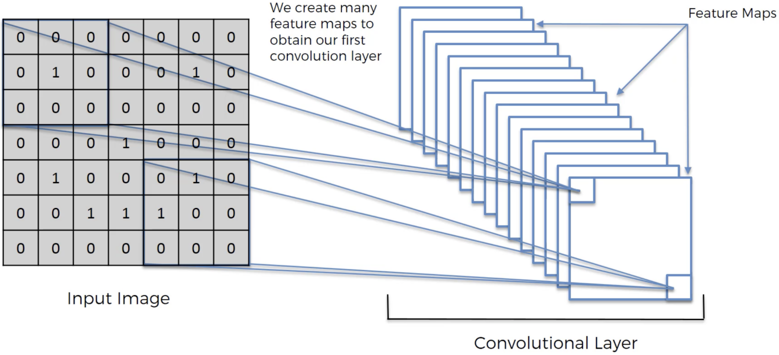


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pooling layer

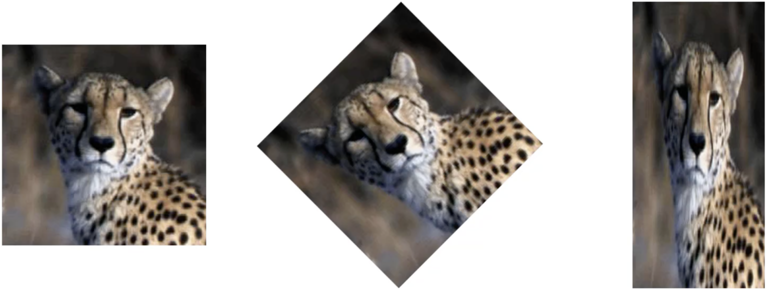


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pooling layer

Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pooling layer

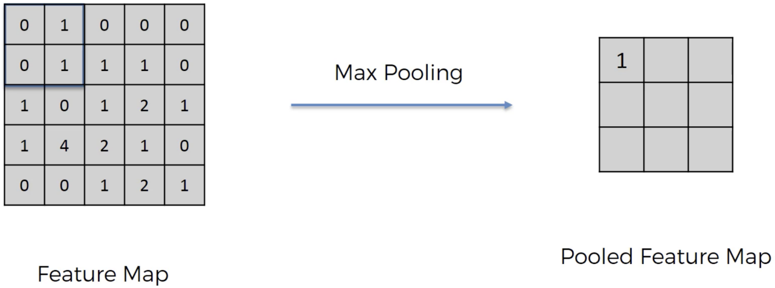


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pooling layer

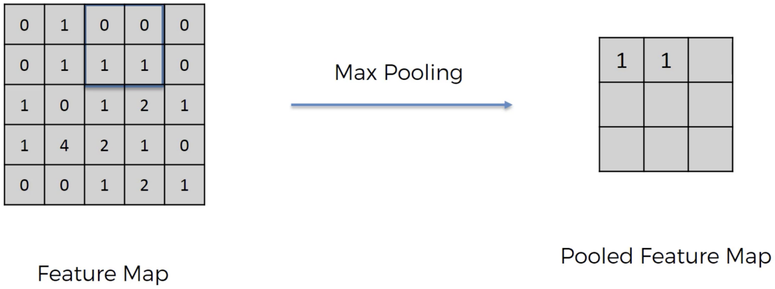


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Pooling layer

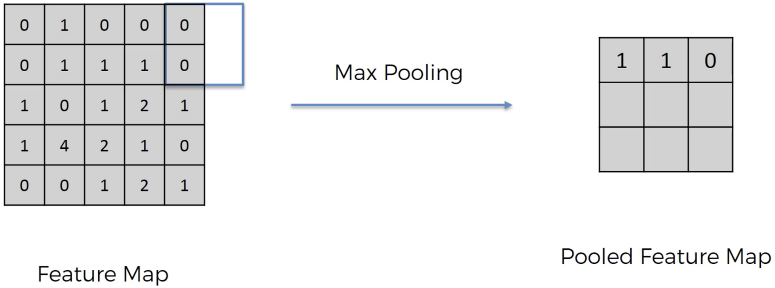


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

flattening/fully connection

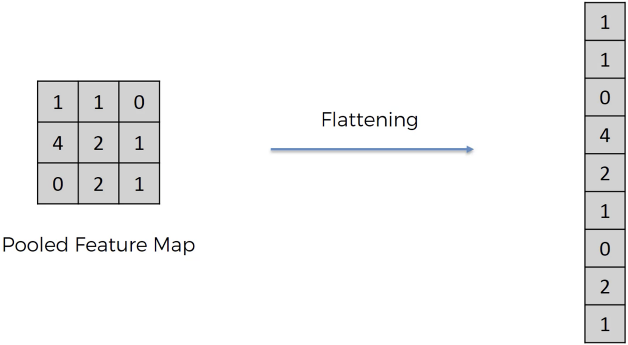


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

flattening/fully connection

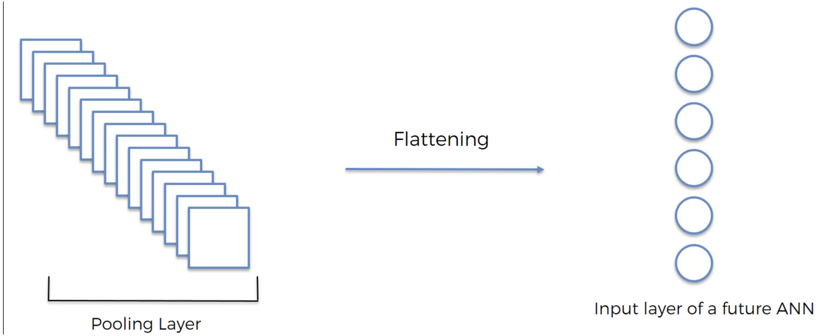


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

whole view

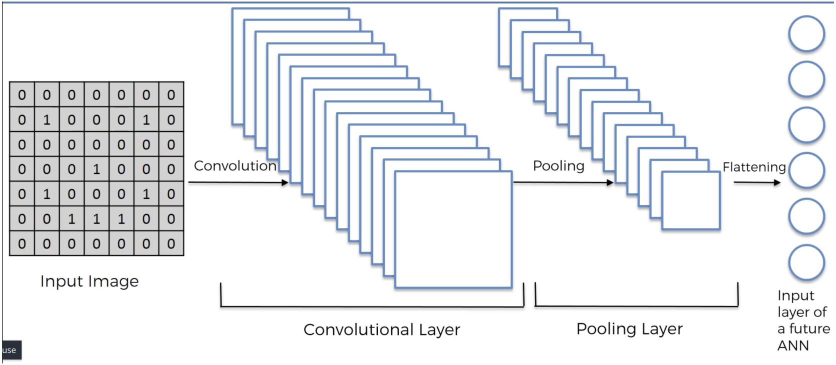


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

Whole view

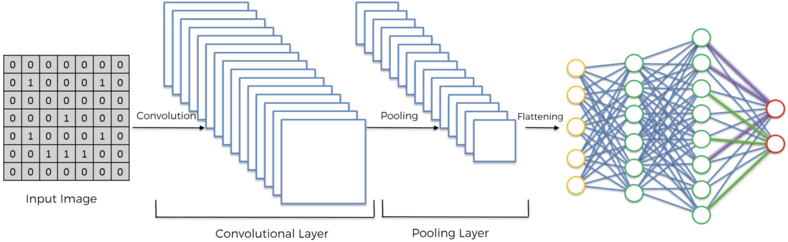


Figure: https://www.udemy.com/course/machinelearning/learn/lecture/6683196

small example

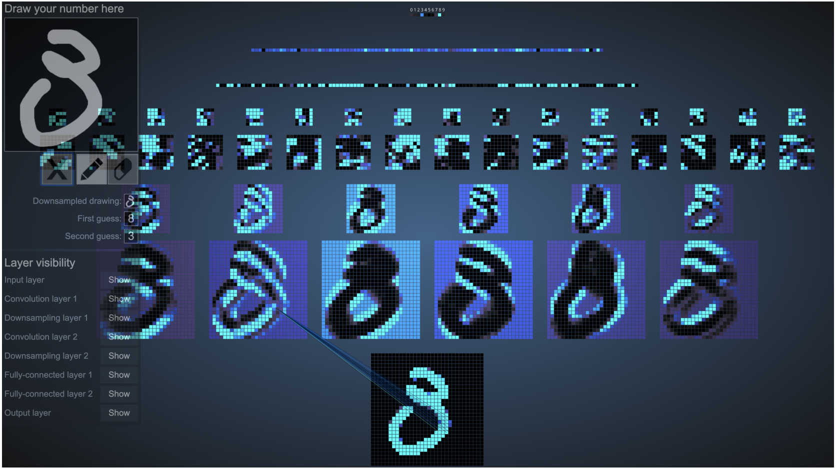


Figure: https://www.cs.ryerson.ca/ aharley/vis/conv/flat.html

AlexNet: Milestone for CNN

Main Breakthrough for CNNs: AlexNet & ImageNet

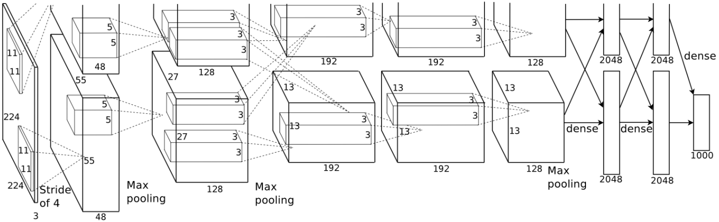
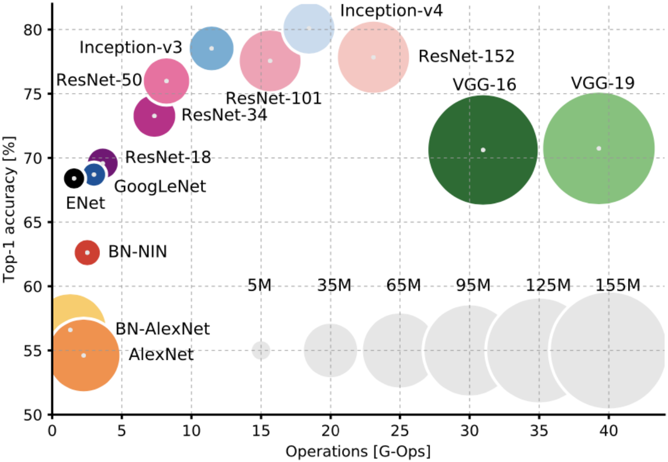


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Common architecture



Figure