Numerical Computation and Machine Learning Basics

https://github.com/timestocome/DeepLearning-Talks

Overflow and Underflow

Overflow -+/- infinity, NaN, wrapping

Underflow - numbers fade to zero

vanishing gradients [clip gradients, add epsilon]

exploding gradients [clip gradients, use regularization]

- Pentium FDIV Bug
- GPU truncation causing non-linearity
- https://blog.openai.com/nonlinear-computation-in-linear-networks/

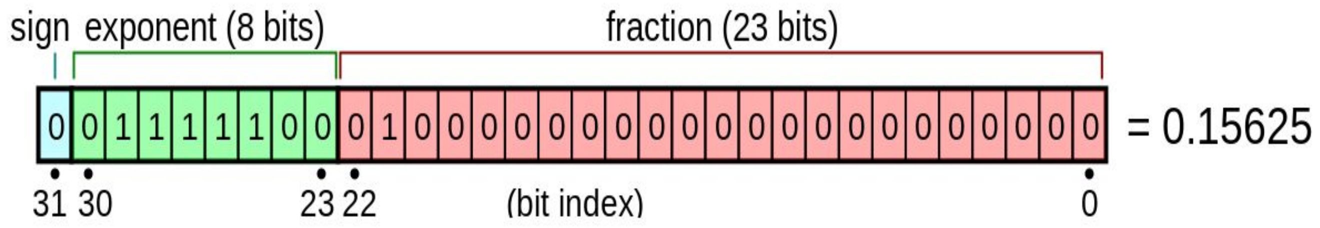
Most ML libraries will handle NaN, -/+ inf, div by zero for you, or add epsilon (a very small number) to denominator

Wrapping: use a 16 bit int to count in a loop and it'll wrap at 65,536 - 1

- don't use loops
- use a very large loop index

IEEE Standard

(-1 * s) x (fraction) x (2 ^ exponent)



(* fraction is usually named mantissa)

Size of ints, longs, floats, doubles on your computer

https://www.programiz.com/c-programming/examples/sizeof-operator-example

http://cstl-csm.semo.edu/xzhang/Class%20Folder/CS280/Workbook_HTML/FLOATING_tut.htm

http://www.boundedfloatingpoint.com/PressRelease_011718.pdf (end of floating point errors ?)

Poor Conditioning

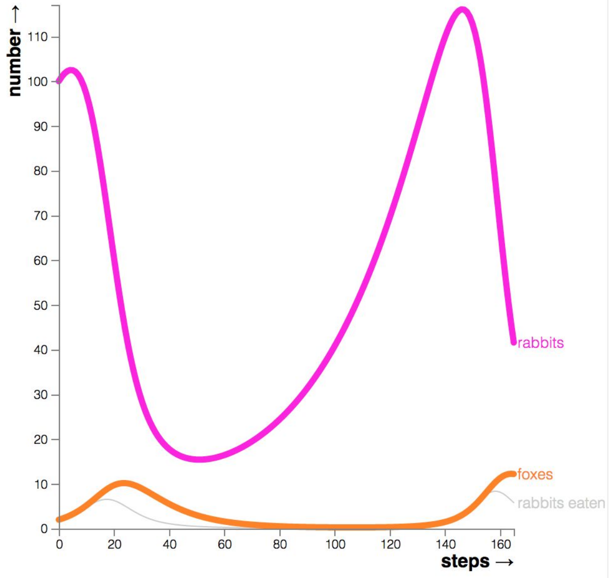
Conditioning is how quickly a function changes

Even a small learning rate increases the error causing the network to become stuck

In a small network this is a local minima, in a deep network it's a saddle point

Fix: use an adaptive learning rate (Adagrad, AdaDelta, RMSProp, Adam, Momentum ..

Adaptive learning rates are built into Theano, TensorFlow ...

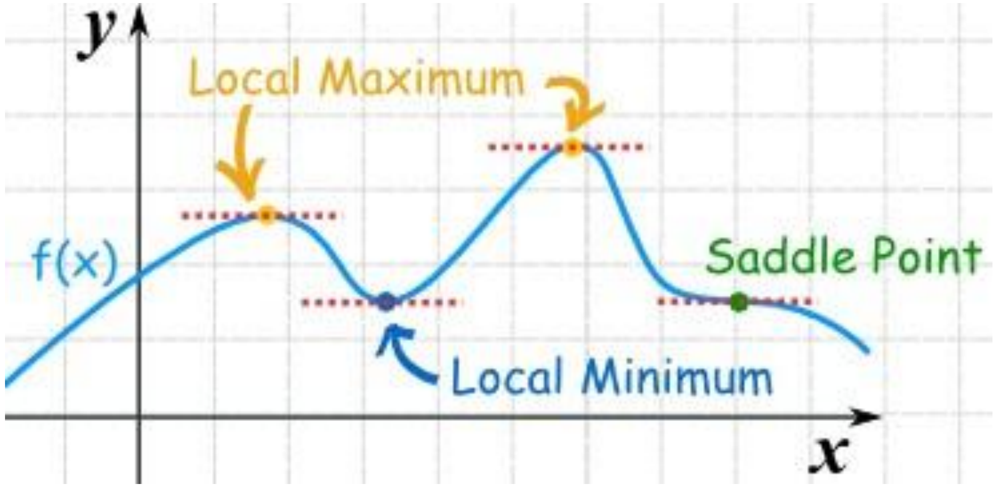


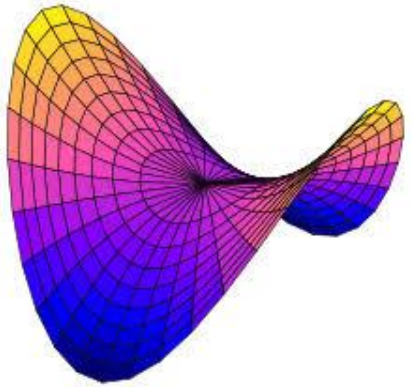
Gradient Based Optimization

Goal is to minimize or maximize a cost function (loss function, error function)

Using a small learning rate, take a small step in the direction of steepest decrease

Gradient = 0 at minimum, max, saddle point





Derivative

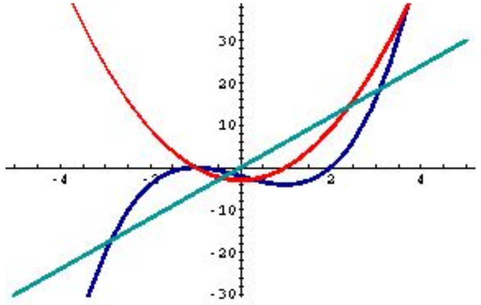
Measures the tangent line on a slope which gives you the direction of greatest increase. In neural networks we use the negative of it to get steepest descent.

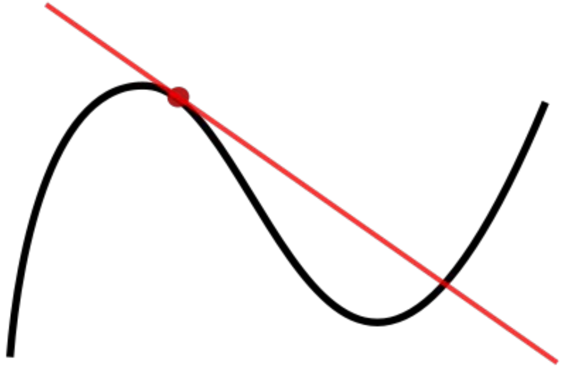
The 2nd derivative can tell you if you are at a max or min

f" < 0 max

f" == 0 no information

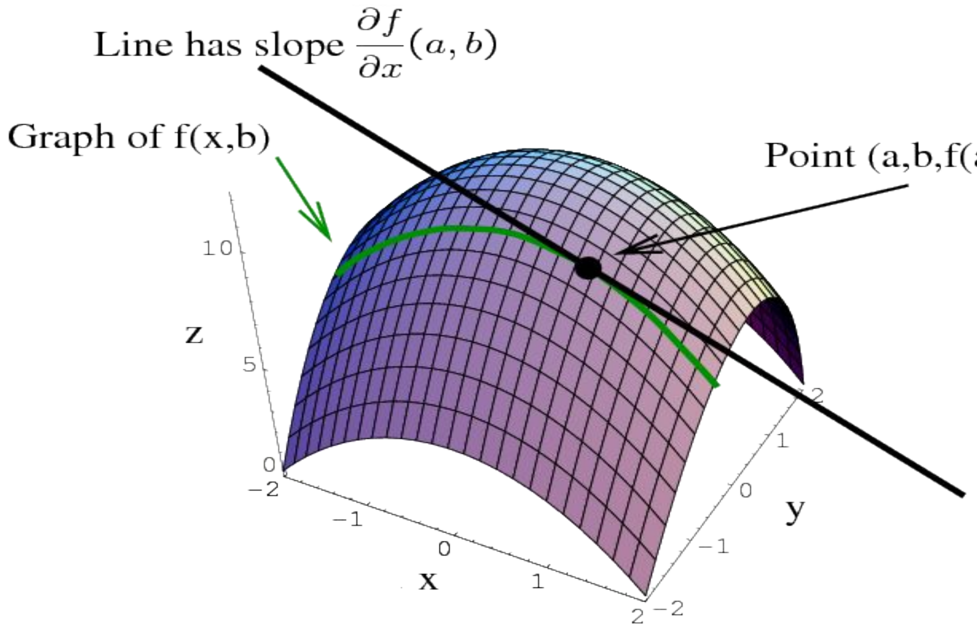
f" > 0 minimum





Partial Derivative

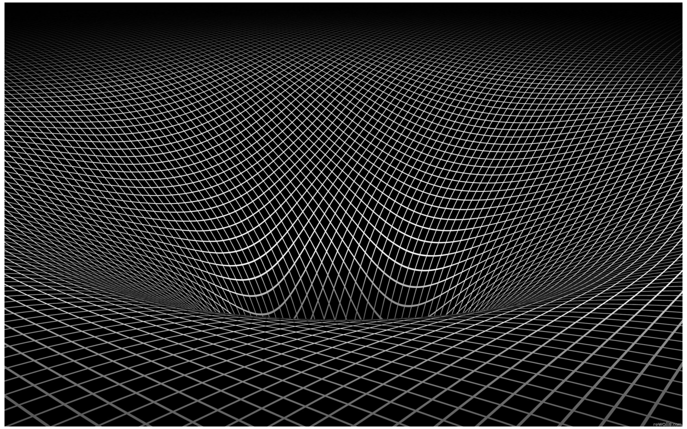
Derivatives calculated on vectors (gradient)



Jacobian Matrix

Take the first derivative of a vector and output a vector to find steepest descent

$$\frac{\partial(x,y)}{\partial(u,v)} = \begin{vmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{vmatrix} = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u}$$



Hessian Matrix

Used to find the 2nd derivative of a vector

Measures the curvature

Use to adjust learning rate (step size)

negative -> maxes negative curvature is down,

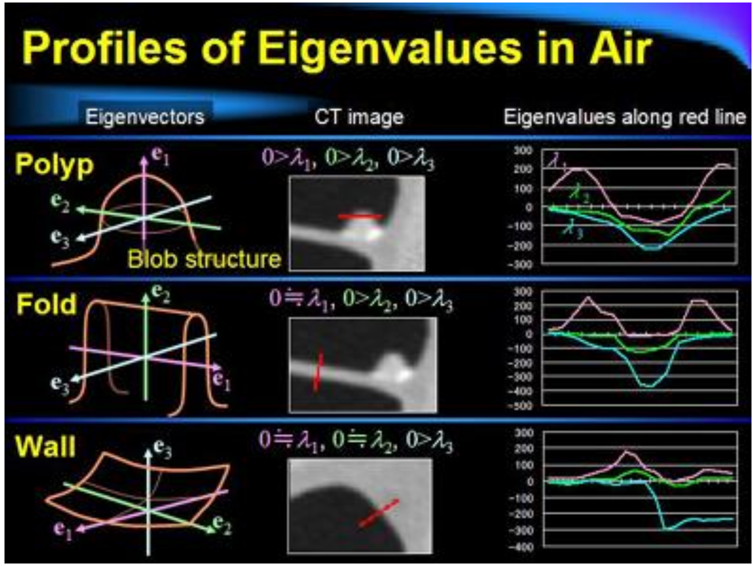
positive -> mins curvature is up

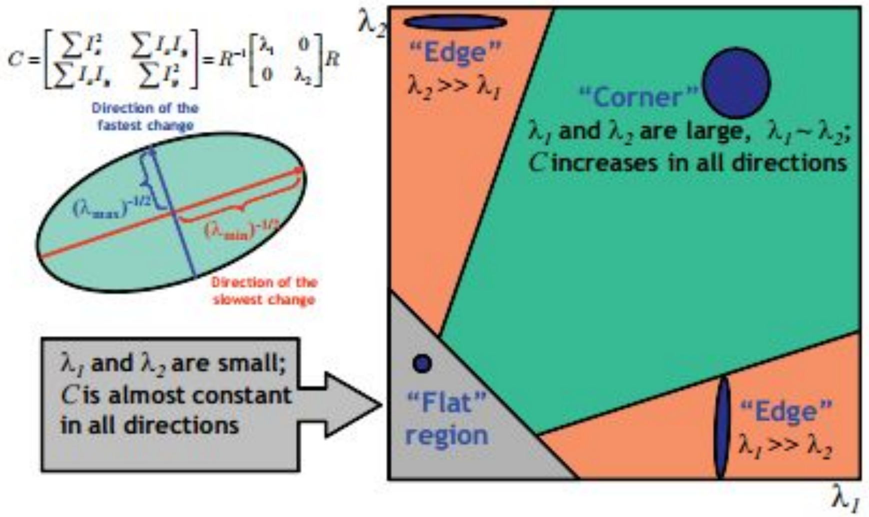
zero -> flat space or a saddle point

$$\text{Gradient} = \begin{bmatrix} \frac{\delta H}{\delta z_1} \\ \frac{\delta H}{\delta z_2} \\ \frac{\delta H}{\delta z_3} \end{bmatrix}, \text{Hessian} = \begin{bmatrix} \frac{\partial^2 H}{\partial {z_1}^2} & \frac{\partial^2 H}{\partial z_1 \partial z_2} & \frac{\partial^2 H}{\partial z_1 \partial z_3} \\ \frac{\partial^2 H}{\partial z_2 \partial z_1} & \frac{\partial^2 H}{\partial {z_2}^2} & \frac{\partial^2 H}{\partial z_2 \partial z_3} \\ \frac{\partial^2 H}{\partial z_3 \partial z_1} & \frac{\partial^2 H}{\partial z_3 \partial z_2} & \frac{\partial^2 H}{\partial {z_3}^2} \end{bmatrix}$$

Eigenvalues of Hessian

Find ridges (fingerprints, CAT scans); Motion capture (character animation)


Profiles of Eigenvalues in Air
Eigenvectors
CT image
Eigenvalues along red line
Polyp
$0>\lambda_1,\ 0>\lambda_2,\ 0>\lambda_3$
Blob structure
Fold
$0\fallingdotseq\lambda_1,\ 0>\lambda_2,\ 0>\lambda_3$
Wall
$0\fallingdotseq\lambda_1,\ 0\fallingdotseq\lambda_2,\ 0>\lambda_3$


$C=\begin{bmatrix}\sum I_x^2 & \sum I_xI_y\\ \sum I_xI_y & \sum I_y^2\end{bmatrix}=R^{-1}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}R$
Direction of the fastest change
Direction of the slowest change
$(\lambda_{max})^{-1/2}$
$(\lambda_{min})^{-1/2}$
λ_1 and λ_2 are small; C is almost constant in all directions
"Edge" $\lambda_2 >> \lambda_1$
"Corner" λ_1 and λ_2 are large, $\lambda_1 \sim \lambda_2$; C increases in all directions
"Flat" region
"Edge" $\lambda_1 >> \lambda_2$
λ_1
λ_2

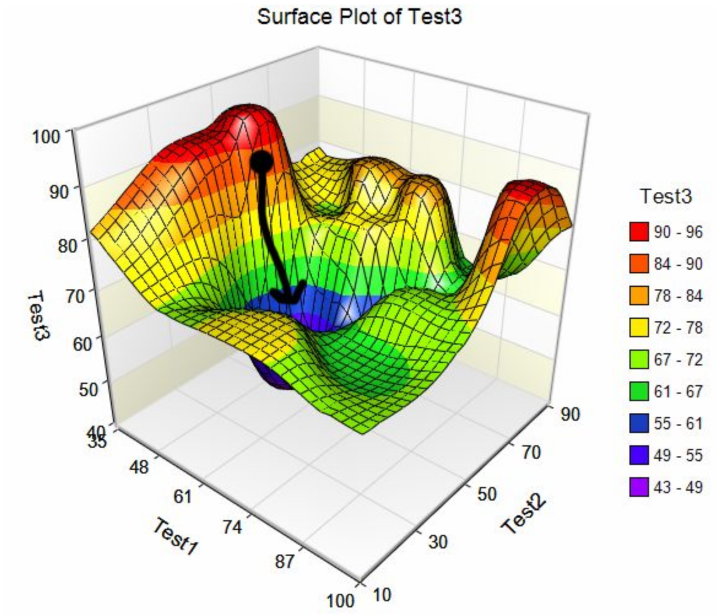
Stochastic Gradient Descent

Repeat until convergence

{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

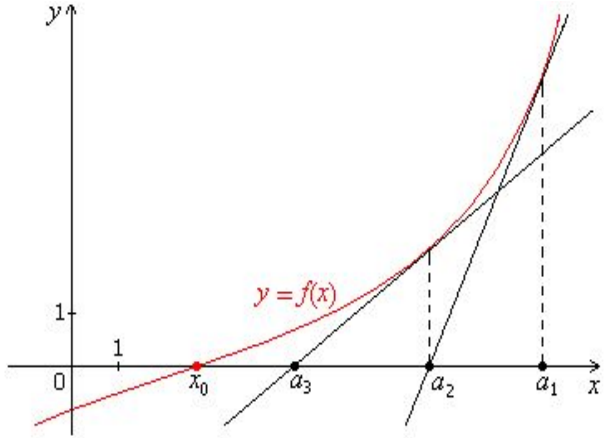
}

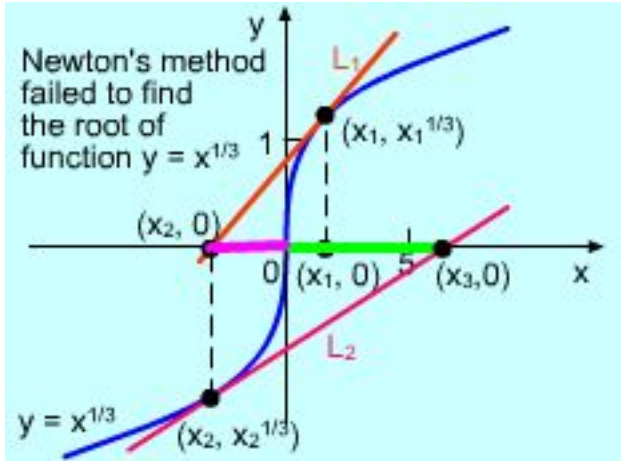


Newton's Method

Approximate f(x) = 0

Keep guessing, split the difference between guesses until f(x) = 0 Usually fails spectacularly, step size missed zero





Code example: http://www.aip.de/groups/soe/local/numres/bookcpdf/c9-4.pdf

Constrained Optimization

More than one cost in the problem

i.e. Find production number that maximizes income using only factory available

Or maximize f(x) subject to g(x)

Often just add g(x) to cost function as a penalty, but a large penalty can create ill conditioned Hessians causing the network to fail to converge

Lagrangian Multiplier

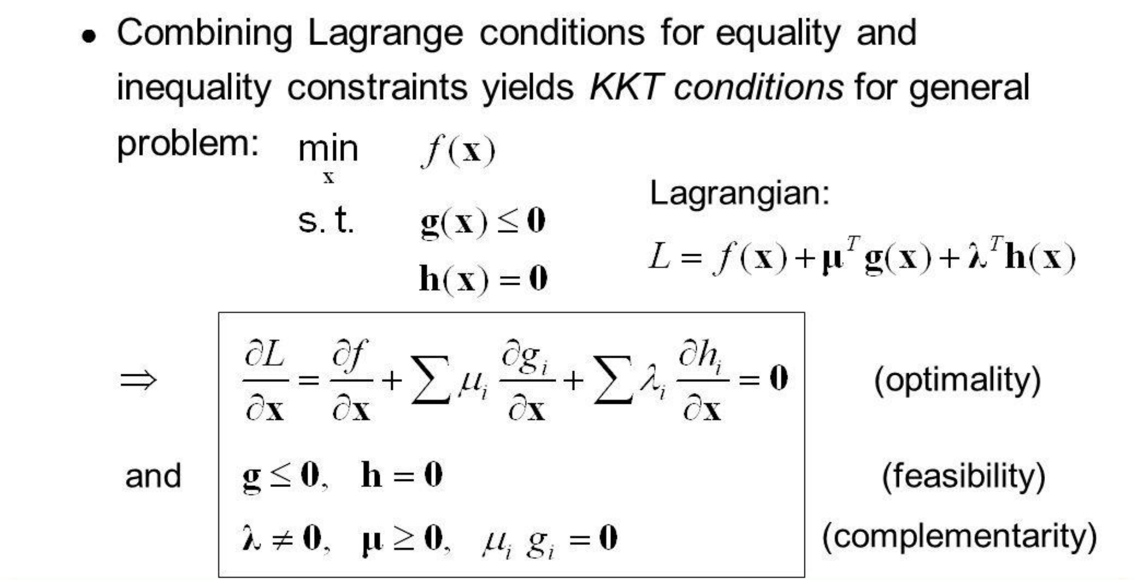
solve:	$\max \log(x) + \log(y)$ given: $x + y = m$
Combine equations, set g(x) to zero	$\max \log(x) + \log(y) + L(m - x - y)$
Set derivatives = 0	$1/x - z = 0,\ 1/y - z = 0,\ m - x - y = 0$
Remove L from x, y	$1/x - z = 1/y - z \implies x = y$
2 equations, 2 unknowns	$x = y = m/2$
Solve for Lagrangian multiplier	$z = m/2$

http://home.uchicago.edu/~vlima/courses/econ201/pricetext/chapter2.pdf

http://adl.stanford.edu/aa222/Lecture_Notes_files/constrainedOptimization.pdf

https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/constrained-optimization/a/lagrange-multipliers-single-constraint

Karush-Kuhn-Tucker Approach



- Combining Lagrange conditions for equality and inequality constraints yields *KKT conditions* for general problem:

$$\min_{\mathbf{x}} \quad f(\mathbf{x})$$
$$\text{s. t.} \quad \mathbf{g}(\mathbf{x}) \le \mathbf{0}$$
$$\mathbf{h}(\mathbf{x}) = \mathbf{0}$$

Lagrangian:

$$L = f(\mathbf{x}) + \boldsymbol{\mu}^T \mathbf{g}(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x})$$

$$\Rightarrow \quad \frac{\partial L}{\partial \mathbf{x}} = \frac{\partial f}{\partial \mathbf{x}} + \sum \mu_i \frac{\partial g_i}{\partial \mathbf{x}} + \sum \lambda_i \frac{\partial h_i}{\partial \mathbf{x}} = \mathbf{0} \quad \text{(optimality)}$$

and $\mathbf{g} \le \mathbf{0}, \quad \mathbf{h} = \mathbf{0}$ (feasibility)

$\boldsymbol{\lambda} \neq \mathbf{0}, \quad \boldsymbol{\mu} \ge \mathbf{0}, \quad \mu_i \, g_i = \mathbf{0}$ (complementarity)

http://slideplayer.com/slide/4991139/

Linear Least Squares with Constraint

Solve:	$f(x) = \frac{1}{2} \|\|Ax - b\|\|^2$
Constraint:	$g(x) = x.Tx <= 1$
Combine the equations:	$\frac{1}{2}\|\|Ax-b\|\|^2 + z(x.Tx - 1)$
set derivative = 0	$A.TAx - A.Tb + 2zx = 0$
Solve for Lagrangian Multiplier	$z = (A.TAx - A.Tb) * x/2$
and the constraint	$dL/dz = x.Tx - 1$

Learning Algorithms

Able to learn from data

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance in tasks at tasks in T, as measured by P, improves with experience E."

E: data, sensor input, images,

T: goals

P: error

ML is the field of study that gives computers the ability to learn without being explicitly programmed - Arthur Sammuel

Experience (data, sensor inputs...)

Missing values

- Use most common value, average or median
- Interpolation between values (time series)
- Drop that feature
- Infer from other features

Curse of dimensionality

If you have 2 equations and 3 unknowns there are an infinite number of solutions.

If you have a lot of features and few training samples there will be many paths through a network, they may validate and test okay, but will often randomly fail in real world use.

Categories Use one hot vectors

Sensors Allow for noise

Independent Identically Distributed

Garbage in, garbage out

Independent

- features cannot be correlated

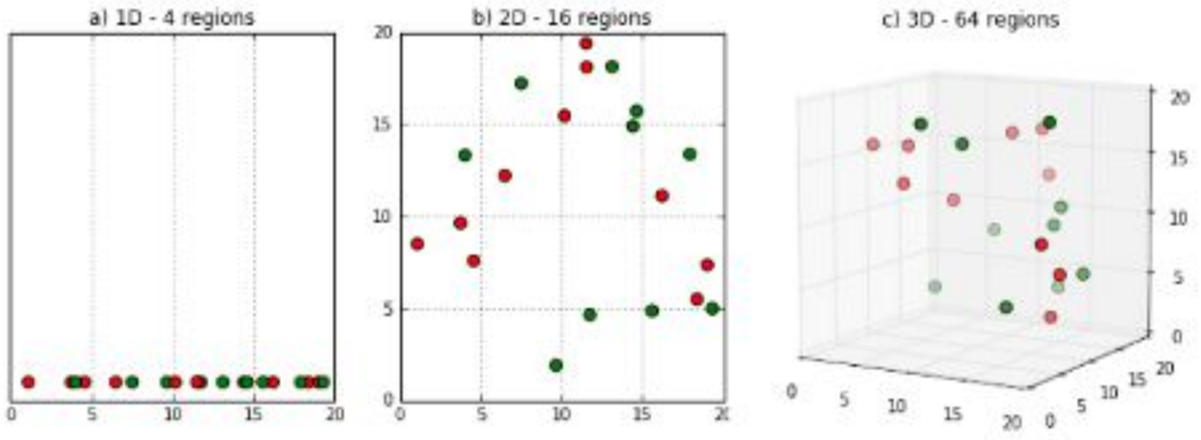
Identically Distributed

- training, validation, test data
- outputs (if 90% of the output is true, the network will always guess true and you'll have 90% accuracy - check the confusion matrix)

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Curse of dimensionality



The Task (Goals)

This is what you want your neural network to solve, the goal

Think carefully, you can get very different answers depending on how you frame your task.

-- First self driving cars were trained with human drivers and learned to drive on their own.

-- Test car is successfully driven down the road, then down a hill into a lake

The Performance Measure (Cost function)

~ 80% of data is used for training

~ 10% of data to validate

~ 10% of data is held out to be used as a test after training is done

```
While accuracy less than ( pick a number )%:
    shuffle data
    run training data in batches
    run validation every 100th epoch or so:
```

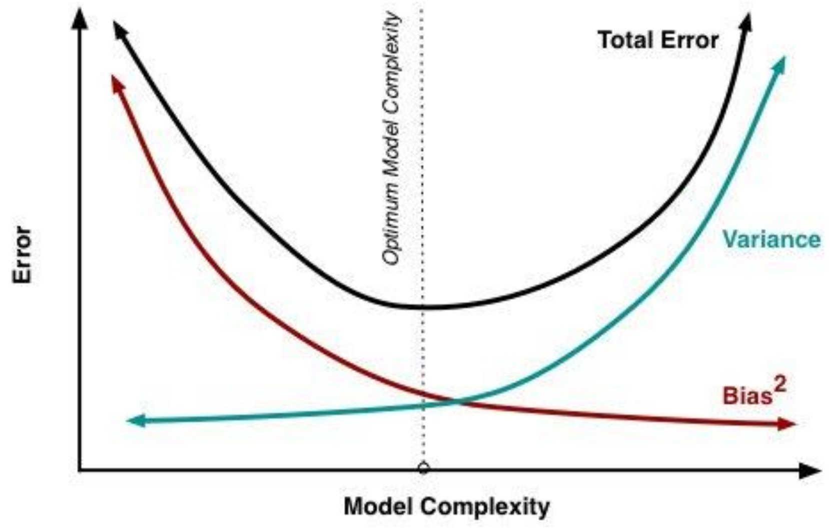
Then test on hold out data after training is complete (* never let subcontractors, contest entrants see hold out data)

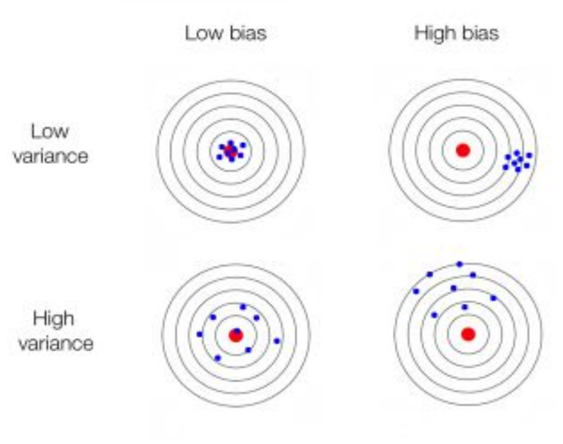
Training, testing, validation data must be statistically similar

Bias vs Variance

High bias == underfitting   High variance == overfitting

MSE = (target - predicted)^2 = Bias^2 + Variance + noise





Entropy

Entropy = how many bits are needed to encode information

(* information theory uses Log2, not e, not 10)

$$\text{Entropy} = -\sum_{i} P_i \operatorname{Log}_2 P_i$$

Coin toss: 2 possibilities (heads, tails) == 1 bit

Entropy = - Sum((½) * log(½) + (½) * log(½))

Entropy = - (½ * -1) + (½ * -1) = -(-½ - ½) = 1

Coin toss: 2 possibilities (heads ⅓, tails ⅔)

Entropy = -(⅓ * log(⅓) + ⅔ * log(⅔)) = 0.89

A Mathematical Theory of Communication, Shannon

http://affect-reason-utility.com/1301/4/shannon1948.pdf

KL Divergence

Kullback-Leibler Divergence tells how much information is lost between the neural network prediction and the actual values

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) \cdot (\log p(x_i) - \log q(x_i))$$

p is probability distribution (actual), q is approximation (predicted)

$$\text{Entropy} = -\sum_{i} P_i \operatorname{Log}_2 P_i$$

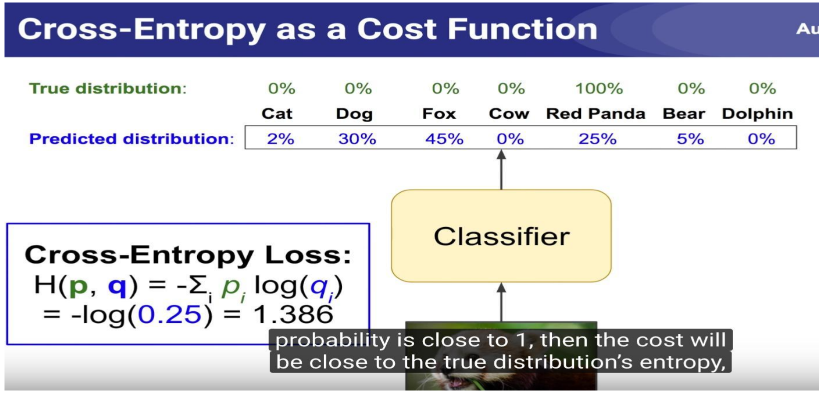
The principle of maximum likelihood says that given the training data, we should use as
gives the greatest possible probability to the training data

- not a distance measure !

https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained

http://cseweb.ucsd.edu/~elkan/250B/logreg.pdf

Cross entropy as cost function



Cross-Entropy as a Cost Function Au

	Cat	Dog	Fox	Cow	Red Panda	Bear	Dolphin
True distribution:	0%	0%	0%	0%	100%	0%	0%
Predicted distribution:	2%	30%	45%	0%	25%	5%	0%

Cross-Entropy Loss:
$H(\mathbf{p}, \mathbf{q}) = -\Sigma_i\, p_i \log(q_i)$
$= -\log(0.25) = 1.386$

probability is close to 1, then the cost will be close to the true distribution's entropy,

Cross entropy = entropy + KL Divergence

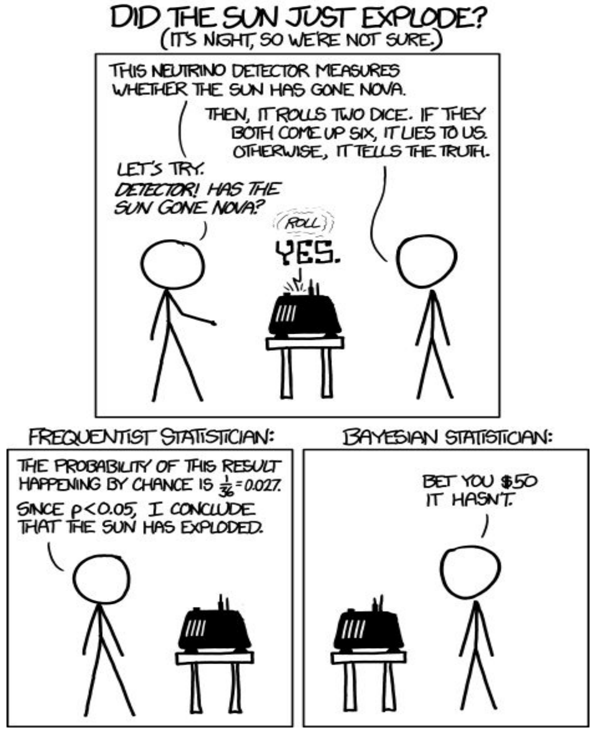
entropy = minimum bits needed to transmit actual information

KL Divergence = difference between predicted and actual information

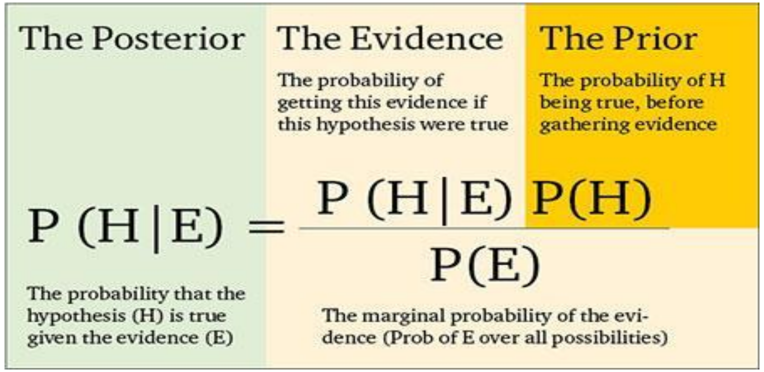
https://www.youtube.com/watch?v=ErfnhcEV1O8&feature=youtu.be

Bayesian vs Frequentist

Bayesian	Frequentist
Prior beliefs	Repeatable random sample
Fixed parameters	Fixed data
mu = mx + b	y = mx + b
y = N(mu, std)	
N - normal distribution	
std - standard deviation	
mu - mean	


DID THE SUN JUST EXPLODE?
(IT'S NIGHT, SO WE'RE NOT SURE.)
THIS NEUTRINO DETECTOR MEASURES WHETHER THE SUN HAS GONE NOVA.
THEN, IT ROLLS TWO DICE. IF THEY BOTH COME UP SIX, IT LIES TO US. OTHERWISE, IT TELLS THE TRUTH.
LET'S TRY. DETECTOR! HAS THE SUN GONE NOVA?
ROLL
YES.
FREQUENTIST STATISTICIAN:
THE PROBABILITY OF THIS RESULT HAPPENING BY CHANCE IS $\frac{1}{36}=0.027$. SINCE $p<0.05$, I CONCLUDE THAT THE SUN HAS EXPLODED.
BAYESIAN STATISTICIAN:
BET YOU $50 IT HASN'T.

Bayesian Statistics



The Posterior

The Evidence

The probability of getting this evidence if this hypothesis were true

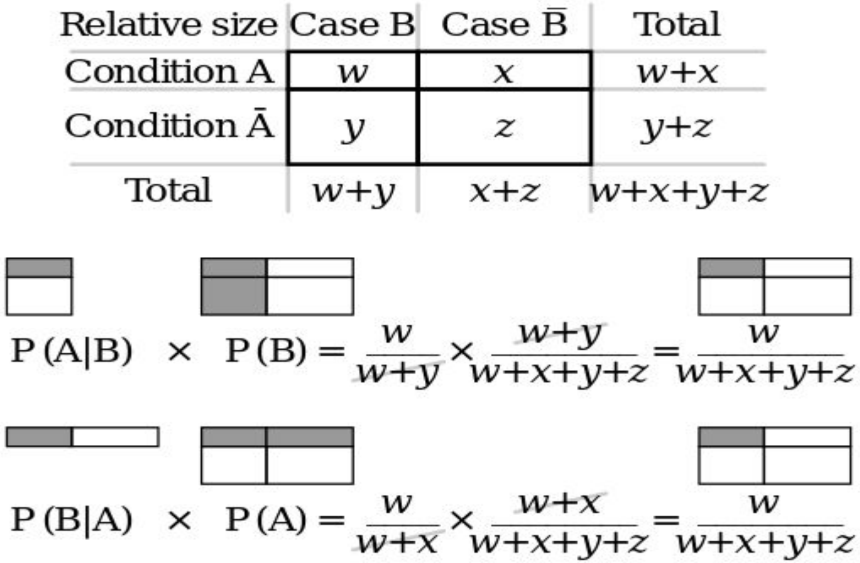
The Prior

The probability of H being true, before gathering evidence

$$P(H|E) = \frac{P(H|E)\,P(H)}{P(E)}$$

The probability that the hypothesis (H) is true given the evidence (E)

The marginal probability of the evidence (Prob of E over all possibilities)



Relative size	Case B	Case $\bar{B}$	Total
Condition A	w	x	$w+x$
Condition $\bar{A}$	y	z	$y+z$
Total	$w+y$	$x+z$	$w+x+y+z$

$$P(A|B) \times P(B) = \frac{w}{w+y} \times \frac{w+y}{w+x+y+z} = \frac{w}{w+x+y+z}$$

$$P(B|A) \times P(A) = \frac{w}{w+x} \times \frac{w+x}{w+x+y+z} = \frac{w}{w+x+y+z}$$

Intuitive understanding of Bayes

https://arbital.com/p/bayes_frequency_diagram/?l=55z&pathId=28771

http://scikit-learn.org/stable/modules/naive_bayes.html

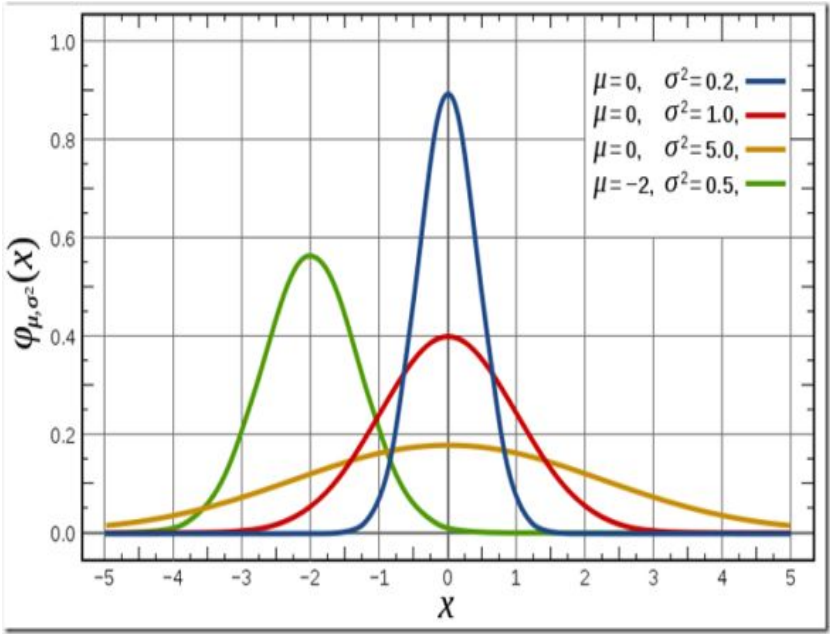
Maximum Log Likelihood (frequentist)

probability of observing the given data as a function of theta

$$\theta_{MLE} = \arg\max_{\theta} \log P(X|\theta)$$

$$= \arg\max_{\theta} \log \prod_i P(x_i|\theta)$$

$$= \arg\max_{\theta} \sum_i \log P(x_i|\theta)$$



Conditional Log Likelihood (Bayesian version)

Weights are trained to maximize the conditional data likelihood

$$W^* = \underset{W}{\operatorname{argmax}} \prod_{d \in D} P(Y^d \mid X_1^d \ldots X_n^d)$$

which is the same as maximizing the conditional log likelihood

$$W^* = \underset{W}{\operatorname{argmax}} \sum_{d \in D} \ln P(Y^d \mid X_1^d \ldots X_n^d)$$

http://cseweb.ucsd.edu/~elkan/250B/logreg.pdf

Mean Squared Error

If the data examples are i.i.d.

independent and identically distributed, then the Conditional Log Likelihood can be reduced to the Mean Squared Error

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Linear Regression as Maximum Likelihood

Dot product as a measure of similarity between vectors

MSE ~ Conditional log likelihood

As n_samples -> infinity convergence increases

- True distribution of data must be in model family
- True distribution must correspond to one set of weights

Let's take the Con out of Econometrics

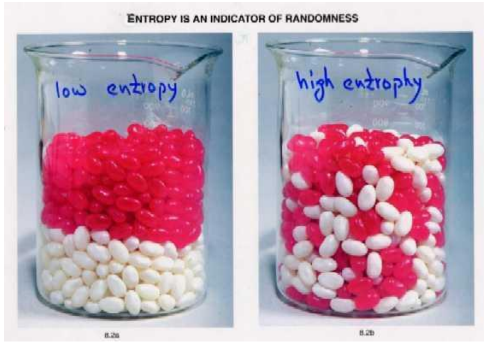
http://www.econ.ucla.edu/workingpapers/wp239.pdf

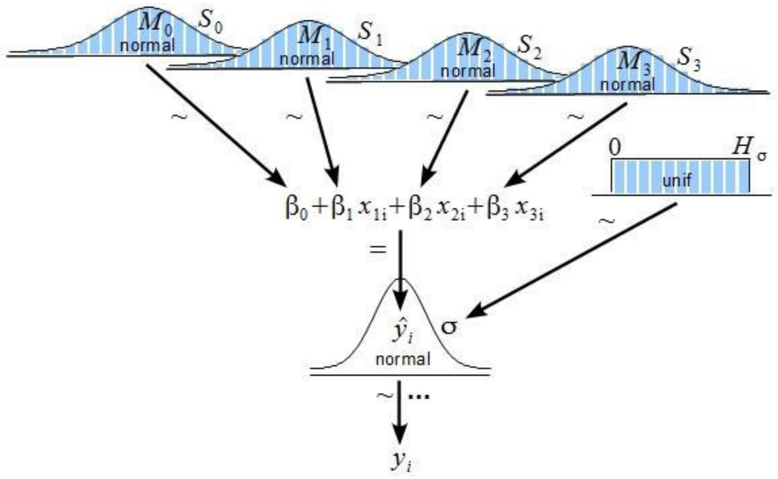
Bayesian Linear Regression

- Prior is uniform or Gaussian distribution with high entropy
- Observation reduces entropy and drives weights to single values

Using a known prior probability

Prior distribution shifts weights toward simpler smoother curves

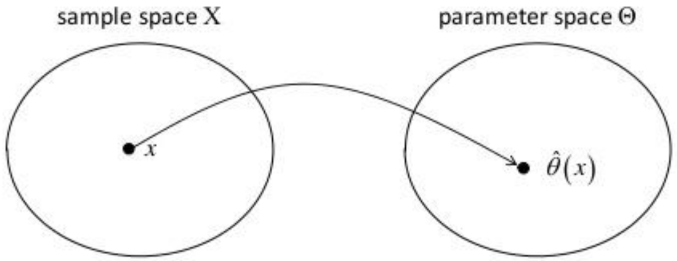




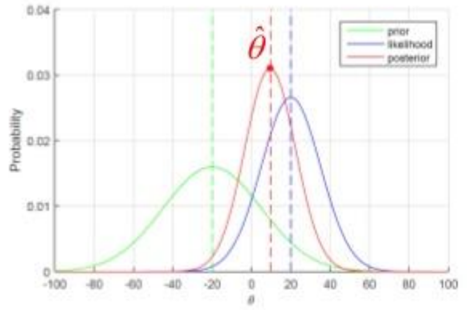
https://stats.stackexchange.com/questions/252577/bayes-regression-how-is-it-done-in-comparison-to-standard-regression

Maximum a Posteriori (MAP) Estimation

Bayesian estimation: Maximum A Posteriori (MAP).



$$\begin{aligned}\hat{\theta}(x) &= \arg\max_{\theta} P(\theta \mid x) \\ &= \arg\max_{\theta} \frac{P(x \mid \theta)P(\theta)}{P(x)} \\ &= \arg\max_{\theta} P(x \mid \theta)P(\theta)\end{aligned}$$



MLE vs MAP

Both compute a single value, not a distribution

MLE - fit a Gaussian to the data

MAP - uses posterior distribution

MLE is a specific case of MAP

$$\begin{aligned}\theta_{MAP} &= \arg\max_\theta \sum_i \log P(x_i|\theta)P(\theta) \\ &= \arg\max_\theta \sum_i \log P(x_i|\theta)\ const \\ &= \arg\max_\theta \sum_i \log P(x_i|\theta) \\ &= \theta_{MLE}\end{aligned}$$

$$\begin{aligned}\theta_{MLE} &= \arg\max_\theta \log P(X|\theta) \\ &= \arg\max_\theta \log \prod_i P(x_i|\theta) \\ &= \arg\max_\theta \sum_i \log P(x_i|\theta)\end{aligned}$$

https://wiseodd.github.io/techblog/2017/01/01/mle-vs-map/

Cross validation

Split the dataset into 3 sections:

Train on one section
Test on one section
Validate on one section

Shuffle data, split and repeat
Average error gives you an approximate idea of how well an algorithm will perform

Especially useful on small datasets.

http://scikit-learn.org/stable/modules/cross_validation.html

Hyperparameters

Variables used to control the algorithm

- max, min, depth of branches on a decision tree
- number of layers, size of layers in network
- learning rate, complexity constant

Usually a grid search (nested loops over several options) is used with cross validation

http://scikit-learn.org/stable/modules/grid_search.html

A recent development is that randomly picking values for a grid search is more effective than stepping through all values.

http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Regularization

Training adjusts the weights to minimize the error, regularization forces the weights to stay small which forces the network to generalize

Used to prevent overfitting: To be useful a network must be able to generalize to new data.

- L1 regularization
- L2 regularization
- Early stopping
- Drop out
- Batch normalization

L1 Regularization

aka Lasso

Minimize the sum of the absolute differences

Drives some weights to zero which has the effect of pruning features which acts as a feature selector. Almost never used in real problems because it is not rotationally invariant

L1 = complexity_parameter * sum(abs(W))

L2 Regularization

aka Ridge regression

Limits weight size without driving any to zero, minimizes variance, increases bias

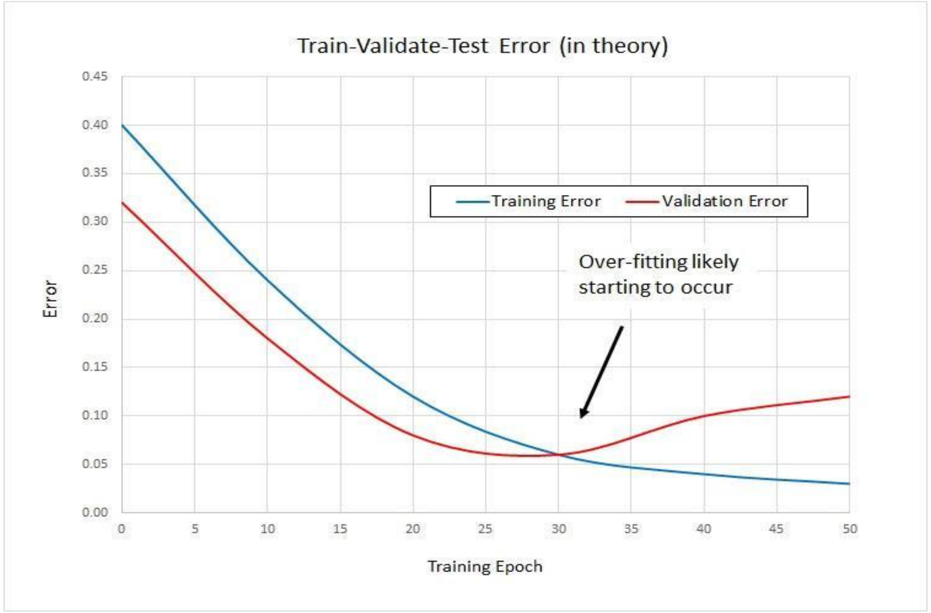
Rotational invariance is the main reason it is used instead of L1

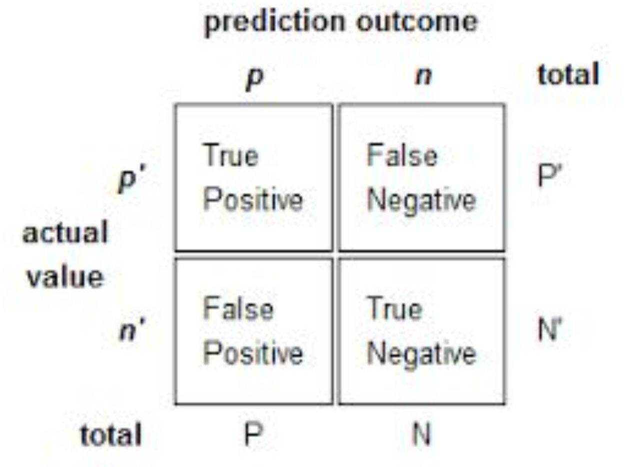
L2 = complexity_constant * sum(W)^2

- L1 and L2 have the effect of choosing a different prior in Bayesian solutions
- Elastic net uses both L1 and L2 https://web.stanford.edu/~hastie/Papers/B67.2%20(2005)%20301-320%20Zou%20&%20Hastie.pdf

Capacity, Overfitting, Underfitting

Early stopping





		prediction outcome		
		p	n	total
actual value	p'	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
	total	P	N	

Dropout

Averages predictions over all weights

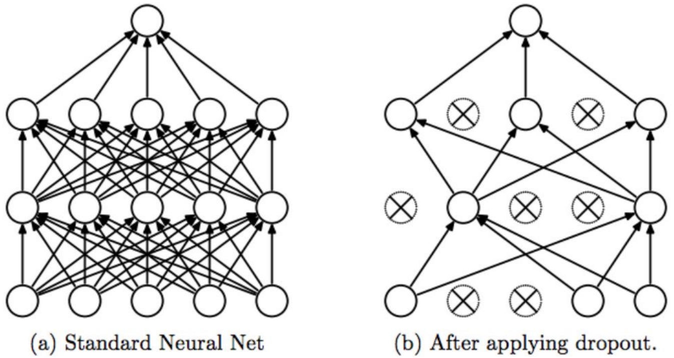
Randomly exclude a percent of hidden nodes from the neural network, change and randomly exclude a different group on next batch during training.

How to:

Create a matrix the size of the weights matrix

Randomly fill with ones and zeros

Multiply weights by the matrix



Dropout: A Simple Way to Prevent NN from Overfitting

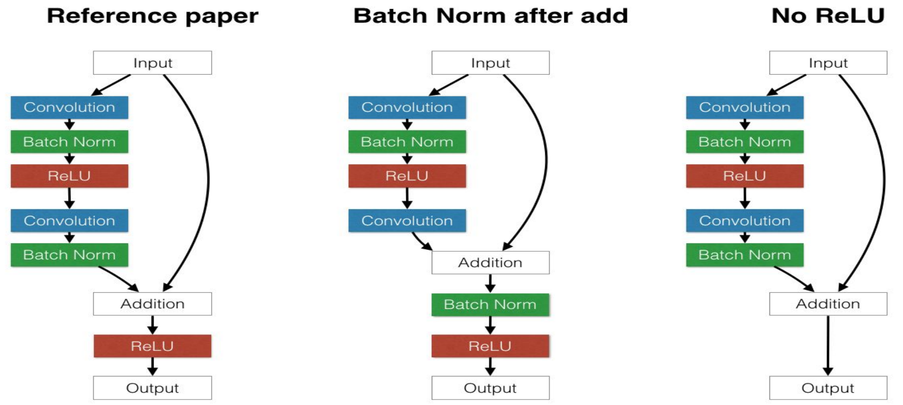
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

Batch Normalization

Is often used in place of DropOut, can use higher learning rates, networks train faster.

For each batch of training examples recenter the features.

In practice, the original input is fed to later layers



https://arxiv.org/pdf/1502.03167.pdf

Probabilistic Supervised Learning

Normal (Gaussian Distribution)

Probability of y given x

The activation functions are used to create non-linearity

- sigmoid, oldest, may cause vanishing gradients
- tanh, steeper than sigmoid, still can cause vanishing gradients
- relu, not linear, combinations or Relu also nonlinear, can approximate any function, drives some weights to zero
-     fast to compute
- leaky relu (use when relu drives all your weights to zero)
-
- softmax (used in last layer to scale values so they sum up to 1)

Linear Regression

prediction = dot(w.T, x)

MSE = 1/n_samples * Sum(predicted - actual)^2

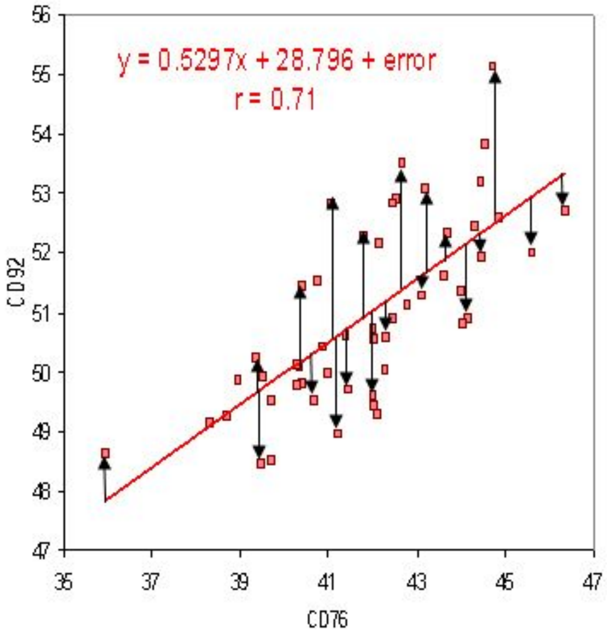
Solve for unknown f(x)

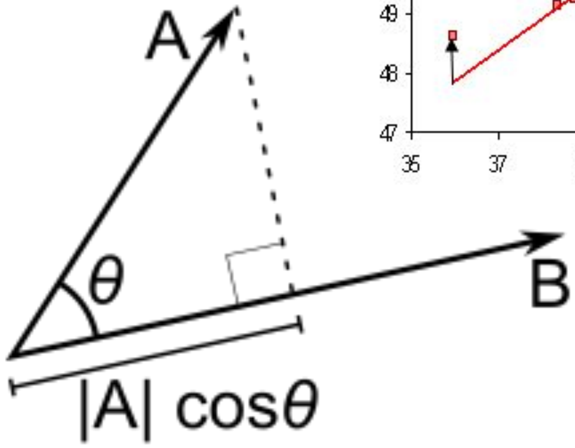
J(w) = ||y - dot(w.T, x)||^2

Use derivative to find direction to change weights

dJ/dw = 2 * XTXw -2 * XTy

w = w - learning_rate * 2 * (XTXw - XTy)





Classification

Classification: image recognition, medical symptoms, marketing, voters ....

Features are turned into vectors and the distance between vectors is used to determine group membership. (city block, bird's eye)

Features representing values are scaled between 0,1 or -1,1

Features representing classes are converted to one hot vectors

Outputs a one hot vector selecting a class

Classification is unsupervised learning and might not always classify in a useful way

Different runs with data that's been shuffled between runs can classify the data differently

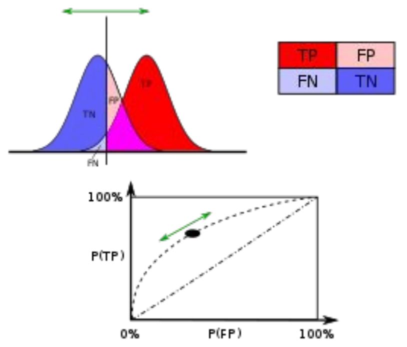
K-Means Clustering

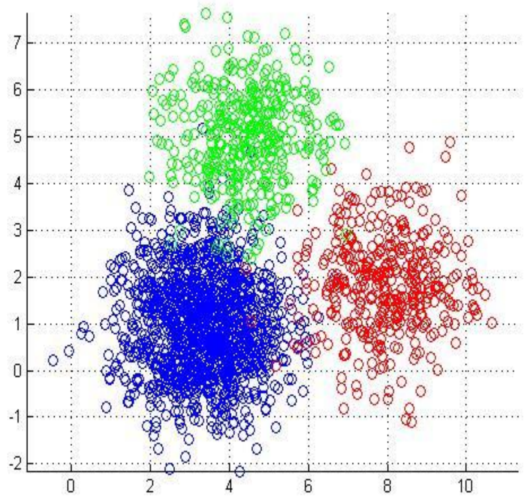
Pick n points at random (centroids), must give algorithm the number of clusters

Put each point in the group of the closest centroid using either Manhattan or Euclidean distance, repeat until stable

- results may vary on sorting order

ROC Receiver operating characteristic ~ 1





http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Classification with missing inputs

Joint Training of Deep Boltzmann Machines for Classification, Goodfellow (Goodfellow 2013b paper)

https://arxiv.org/pdf/1301.3568.pdf

Restricted Deep Boltzmann Machines, Salakhutdinov and Hinton

http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf

Deep Learning 4j has a nice tutorial on Boltzmann machines

https://deeplearning4j.org/restrictedboltzmannmachine

(aka Deep Belief Networks)

Transcription

OCR, Optical Character Recognition

Multi-digit Number Recognition from Street View

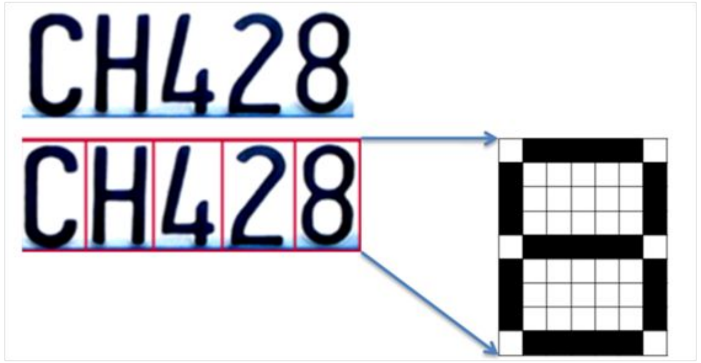
https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdf

Goodfellow talk on the paper

https://www.youtube.com/watch?v=vGPI_JvLoN0

Dataset

http://ufldl.stanford.edu/housenumbers/



Machine Translation

Used for natural language translation

Typically done with a recurrent network (RNN, LSTM, GRU, ...)

One language is the input, the second language is the output.

The network is trained back and forth rather than just forward feeding data and back feeding the errors

https://nlp.stanford.edu/

https://github.com/MicrosoftTranslator/DocumentTranslator

Structured Output

Collobert Home Page

https://scholar.google.com/citations?user=32w7x1cAAAAJ&hl=en

Natural Language Processing (Almost) from Scratch (Collobert 2012)

http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf

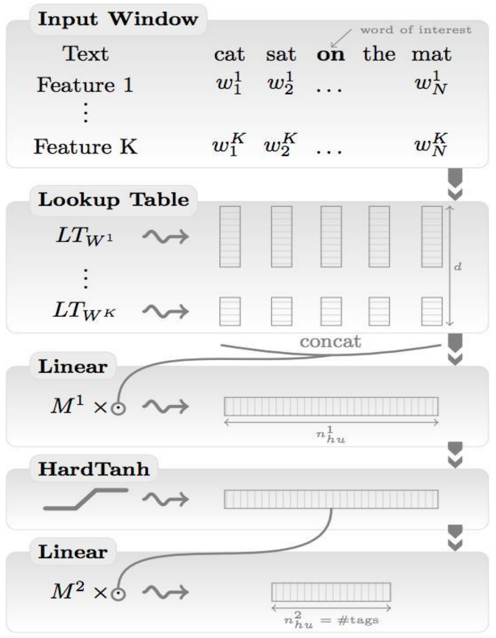


Figure 1: Window approach network.

Anomaly Detection

Make a prediction, points outside the prediction area are anomalies. This is a high value research area -- many problems still to be solved (equipment time to failure, stock market crashes, earthquake prediction......)

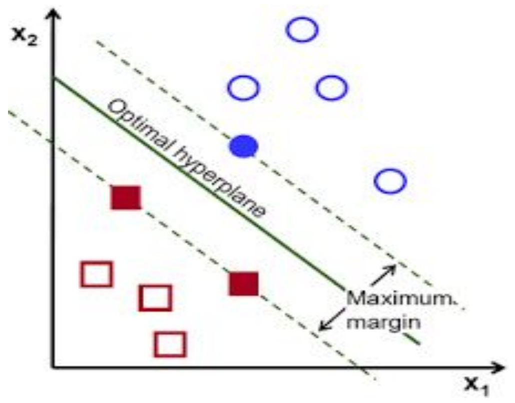
RAD - Outlier Detection on Big Data

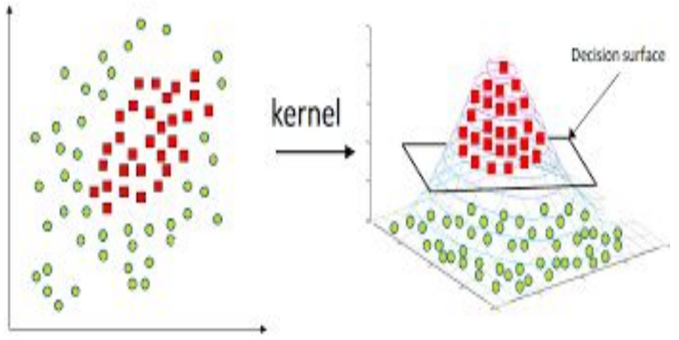
https://medium.com/netflix-techblog/rad-outlier-detection-on-big-data-d6b0494371cc

Bayesian Anomaly detection

http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA610860

Support Vector Machines





O(n^2), can only separate 2 classes per SVM

http://scikit-learn.org/stable/modules/svm.html

https://docs.opencv.org/2.4.13.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

Decision Tree

Use entropy to make splits to give largest information gain

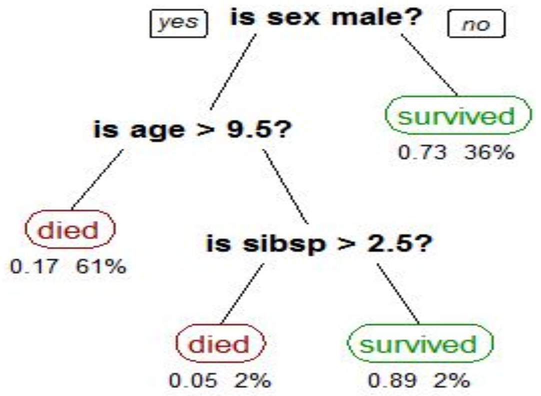
http://scikit-learn.org/stable/modules/tree.html

XGBoost

http://xgboost.readthedocs.io/en/latest/model.html

Using decision trees to see what a neural network is doing

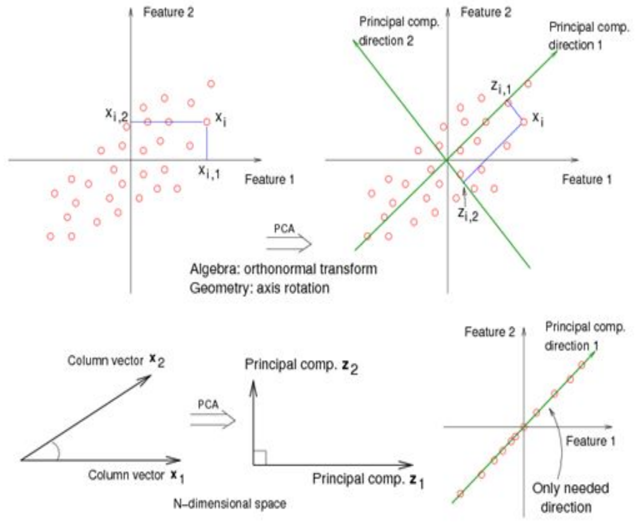
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.6011&rep=rep1&type=pdf



Principal Components Analysis

Used to compress data to reduce features by combining them in a way that maximizes variance

Use this when data is highly correlated to create independent feature representations


Feature 2
$x_{i,2}$
x_i
$x_{i,1}$
Feature 1
PCA
Principal comp. direction 2
Feature 2
Principal comp. direction 1
$z_{i,1}$
x_i
Feature 1
$z_{i,2}$
Algebra: orthonormal transform
Geometry: axis rotation
Column vector $\mathbf{x}_2$
Column vector $\mathbf{x}_1$
PCA
Principal comp. $\mathbf{z}_2$
Principal comp. $\mathbf{z}_1$
N-dimensional space
Feature 2
Principal comp. direction 1
Feature 1
Only needed direction

https://onlinecourses.science.psu.edu/stat857/node/35

http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Manifold Learning

Find a useful way to represent the data in less dimensions

Example: represent road as 2D in a 3D world

It's like PCA but attempts to retain more information

t-SNE Used often for visualization

Doesn't always work, information can be lost or distorted

http://scikit-learn.org/stable/modules/manifold.html

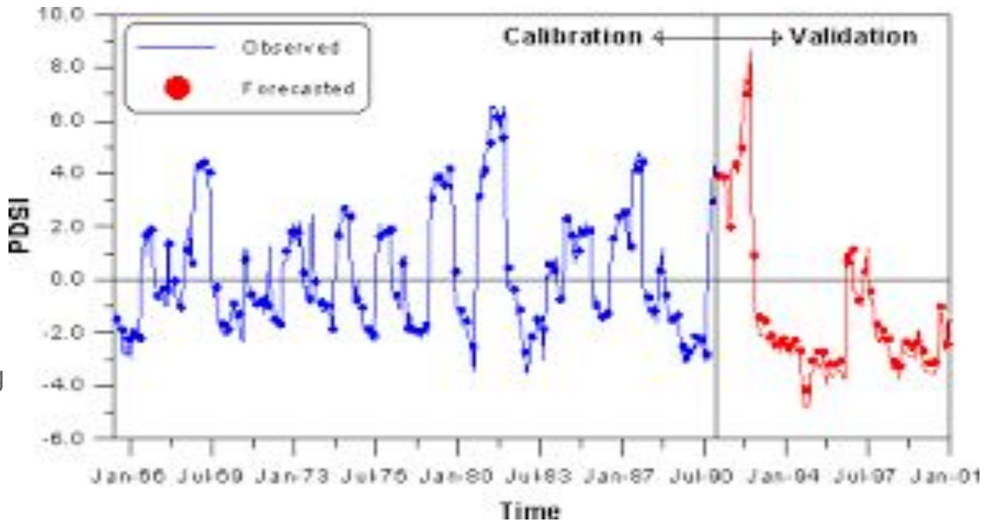
Time Series Predictions

Used to predict future values

- Must be stationary (rotate down to x axis)
- Generally use a log scale
- Validation data must be future data

Facebook's Prophet is a great open source tool for doing

https://facebook.github.io/prophet/



Synthesis and Sampling

Used in the game industry to create landscapes (fractal), characters (AI), etc

https://www.gamasutra.com/topic/game-developer

Microsoft Speech recognition and synthesis

https://github.com/Microsoft/Windows-universal-samples/tree/master/Samples/SpeechRecognitionAndSynthesis

https://www.microsoft.com/en-us/research/blog/microsoft-researchers-achieve-new-conversational-speech-recognition-milestone/

https://arxiv.org/pdf/1708.06073.pdf

No Free Lunch Theorem

Says that if you average over all the machine learning algorithms each will have the same performance on the same data set.

The only way one strategy can outperform another is by adjusting the network to the problem.

- convolutional networks for vision
- recursive networks for time series
- deep vs wide layers

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

Building a Machine Learning Algorithm

Cleanup data

Multiply data by weights, iterative training for non-linear data

Minimize difference between cost function and actual data

(* no free lunch theorem)

New old things being tried to adjust weights

NEAT, NeuroEvolution of Augmenting Topologies http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf

Genetic Algorithms https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html

Reinforcement Learning http://www.incompleteideas.net/book/bookdraft2018jan1.pdf

Videos

Bay Area Deep Learning School 2016

https://www.youtube.com/watch?v=eyovmAtoUx0 and https://www.youtube.com/watch?v=9dXiAecyJrY

Deep Learning Summer School Montreal

https://www.youtube.com/playlist?list=PL5bqIc6XopCbb-FvnHmD1neVlQKwGzQyR

http://videolectures.net/deeplearning2017_montreal/

Neural Networks for Machine Learning (Hinton Lectures 78 videos)

https://www.youtube.com/playlist?list=PLoRl3Ht4JOcdU872GhiYWf6jwrk_SNhz9

Blogs and things

ArXiv Papers https://arxiv.org/

Code for Arvix papers: http://www.gitxiv.com/

Uber ML Blog: https://eng.uber.com/

Deep Mind https://deepmind.com/

Netflix ML Blog: https://medium.com/@NetflixTechBlog

Aylien NLP Blog and Data: http://blog.aylien.com/research/

Open AI: https://openai.com/systems Test environments for RL learning

Kaggle: https://blog.kaggle.com/ Data, ML Blog, forums, examples, contests

Tensorflow

Tensorflow https://www.tensorflow.org/

Tensorflow for poets: https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/#0

https://www.youtube.com/watch?v=cSKfRcEDGUs

Github: https://github.com/tensorflow/tensorflow

TensorFlow Models https://github.com/tensorflow/models

Manning MEAP Books https://www.manning.com/meap-catalog (*these assume you know ML and don't know TF)

Numerical computing

Numerical Recipes in C/C++, Fortran 77/90

http://numerical.recipes/oldverswitcher.html

Introduction to Statistical Learning

http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf

Elements of Statistical Learning

https://web.stanford.edu/~hastie/Papers/ESLII.pdf

Deep Learning

Neural Networks and Deep Learning http://neuralnetworksanddeeplearning.com/

Must Know Tricks in Deep Neural Networks http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html

Neural Network Zoo http://www.asimovinstitute.org/neural-network-zoo/

Neural Network Playground http://playground.tensorflow.org

WildML http://www.wildml.com/

Software

Anaconda https://www.anaconda.com/

Microsoft CNTK https://github.com/Microsoft/CNTK

Deep Learning for Java https://deeplearning4j.org/index.html

Deep Learn JS https://deeplearnjs.org/

Keras https://keras.io/#keras-the-python-deep-learning-library

XGBoost - Extreme Gradient Boosting https://github.com/dmlc/xgboost

Data: https://archive.ics.uci.edu/ml/index.php