Numerical Computation and Machine Learning Basics

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

Numerical computing resources:

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

http://numerical.recipes/oldverswitcher.html

Accelerated Numerical Analysis Tools with GPUs

https://developer.nvidia.com/how-to-numerical-analysis

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/

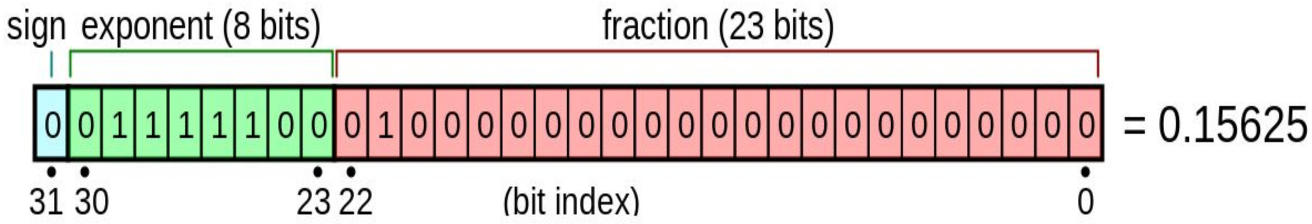
Most ML libraries will handle NaN, -/+ inf, div by zero for you

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

Poor Conditioning

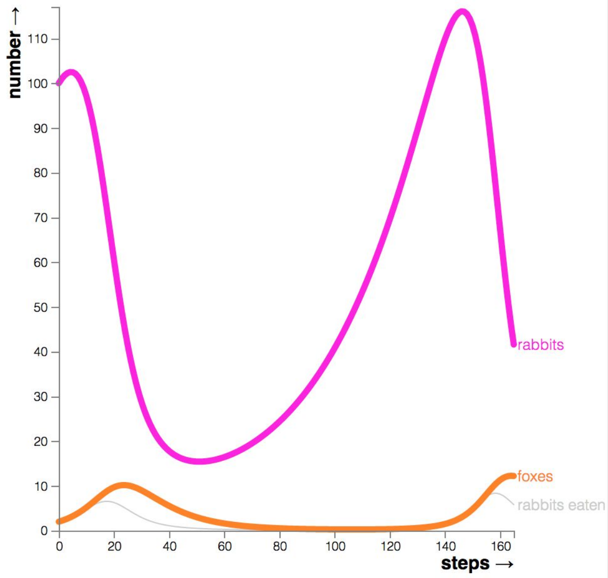
Conditioning is how quickly a function changes

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 ...

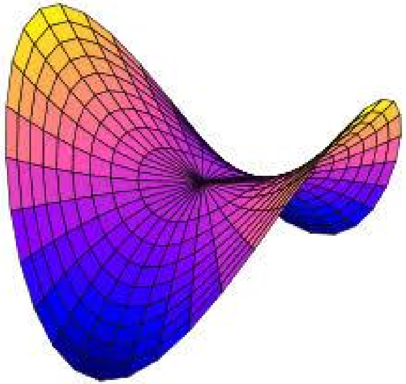


Gradient Based Optimization

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

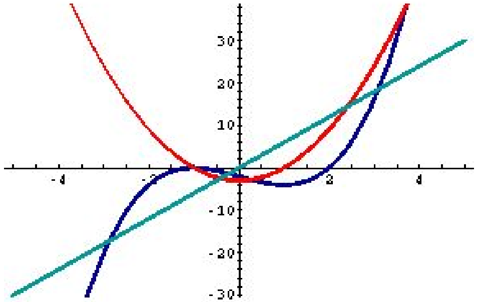
Measures the tangent line on a slope which gives you the direction of greatest increase. In neural networks we use the negative of it to get steepest descent.

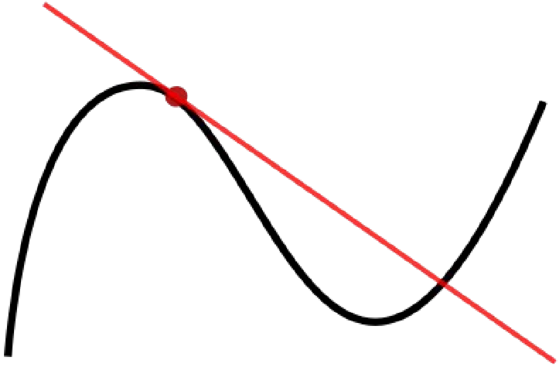
The 2nd derivative can tell you if you are at a max or min

f" < 0 max

f" == 0 no information

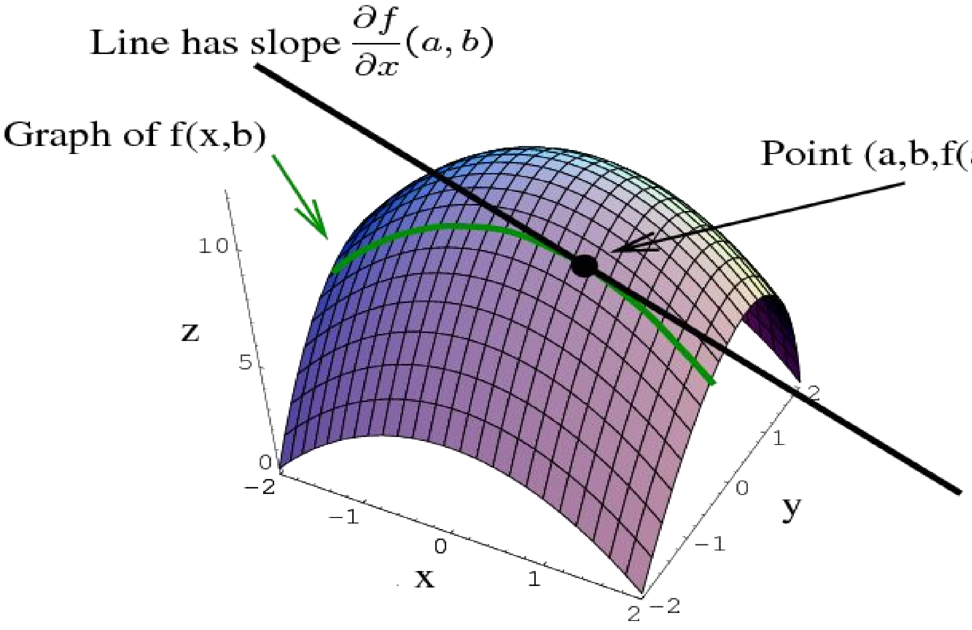
f" > 0 minimum





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

negative -> maxes negative curvature is down,

positive -> mins curvature is up

zero -> flat space or a saddle point

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

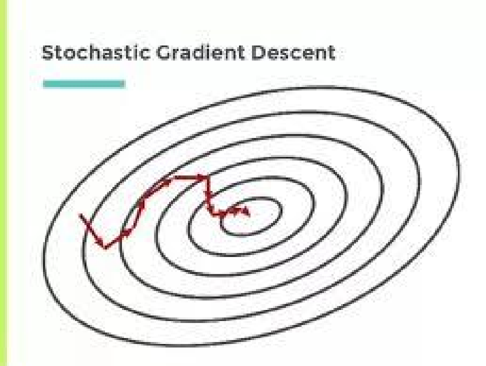
Stochastic Gradient Descent

Repeat until convergence

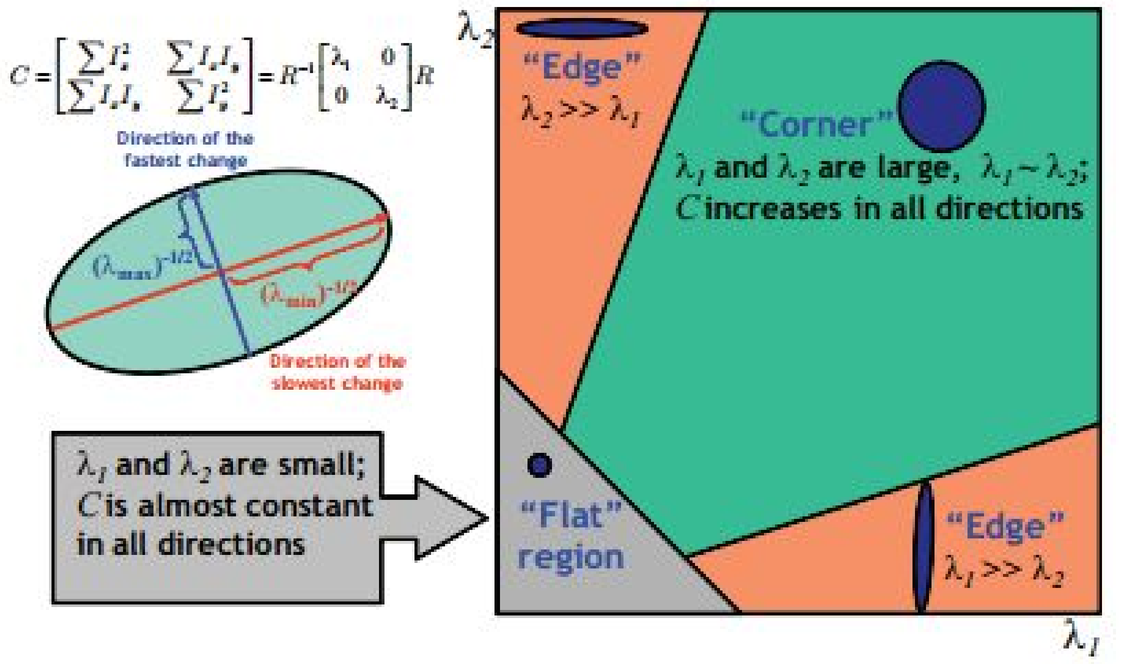
{

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

}



Eigenvalues of Hessian



Newton's Method

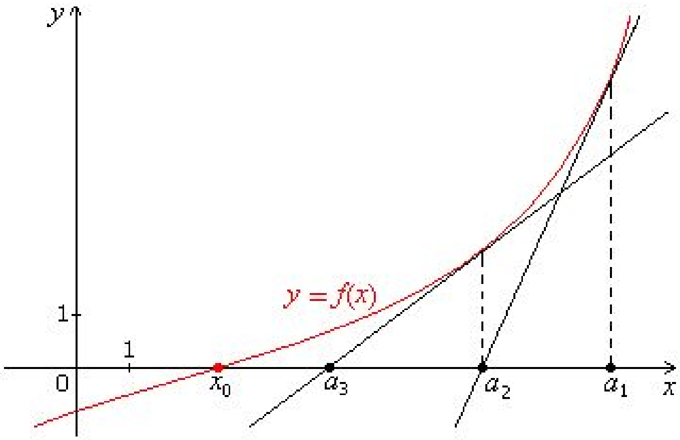
Approximate f(x) = 0

Keep guessing, split the difference between guesses until f(x) = 0

Sometimes fails spectacularly, extremely dependent on step size

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

Karush-Kuhn-Tucker Approach

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

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

Lagrangian:

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

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

and

$$\mathbf{g} \leq \mathbf{0}, \quad \mathbf{h} = \mathbf{0} \qquad \text{(feasibility)}$$

$$\boldsymbol{\lambda} \neq \mathbf{0}, \quad \boldsymbol{\mu} \geq \mathbf{0}, \quad \mu_i \; g_i = \mathbf{0} \qquad \text{(complementarity)}$$

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

Linear Least Squares

Solve: f(x) = ½ ||Ax - b||^2

Gradient: df(x) = A.T(Ax-b)

while ||A.TAx - A.Tb|| > acceptable_error:

    x = x - learning_rate(A.TAx - A.Tb)

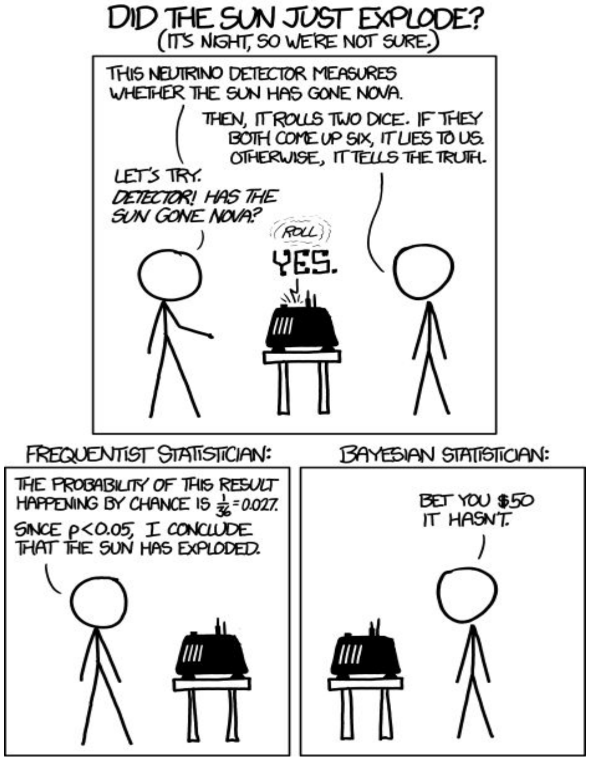
* do not use Newton's Method

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	$dL/dz = x.Tx - 1$

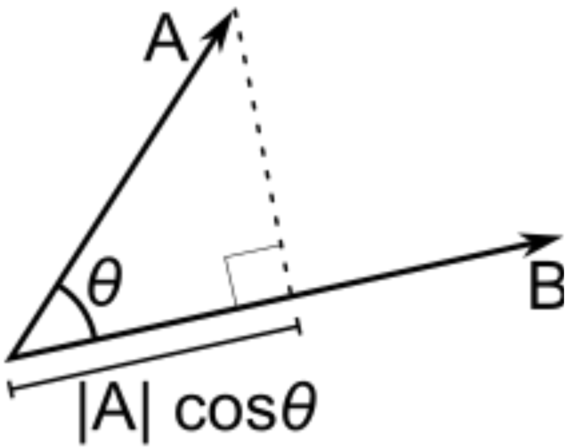
Bayesian vs Frequentist

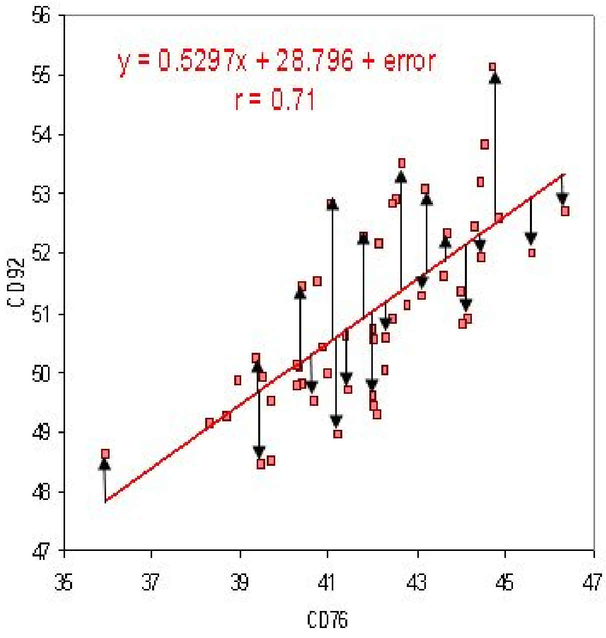
Bayesian	**Frequentist**
Prior beliefs	Repeatable random sample
Fixed parameters	Fixed data
mu = mx + b	y = mx + b
y = N(mu, std)	
N - normal distribution	
std - standard deviation	
mu - mean	



Linear Regression

prediction = dot(w.T, x)





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

Linear Least Squares Regression

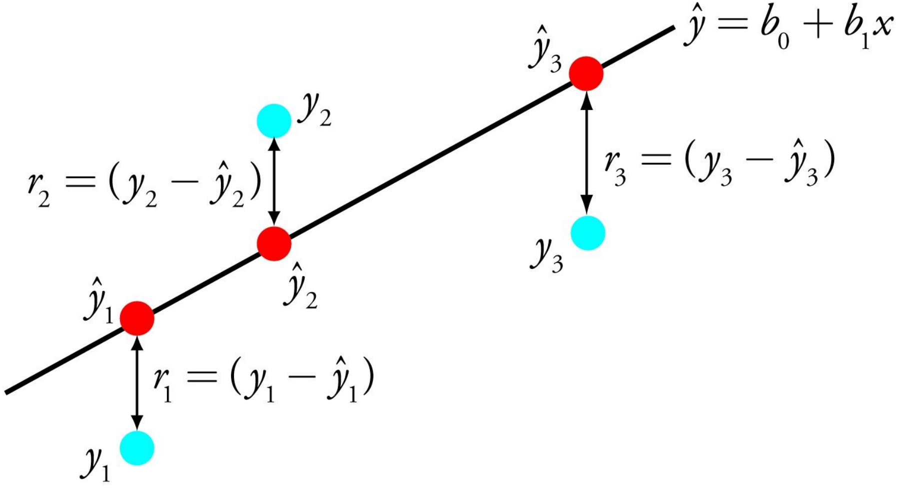
Solve for unknown f(x)

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

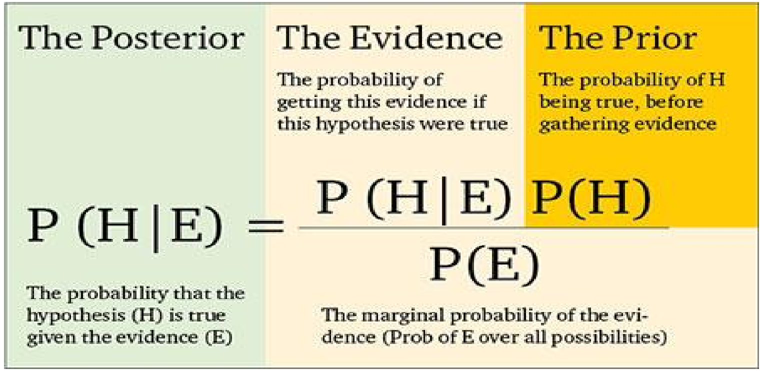
Use derivative to find direction to change weights

dJ/dw = 2 * XTXw -2 * XTy

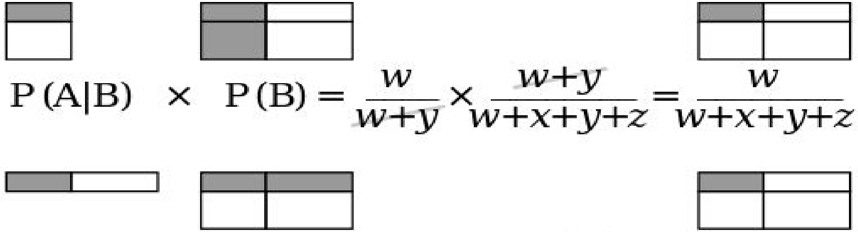
w = w - learning_rate * 2 * (XTXw - XTy)



Bayesian Statistics



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

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

T: goals

P: error

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

? Car learned to follow the curb

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

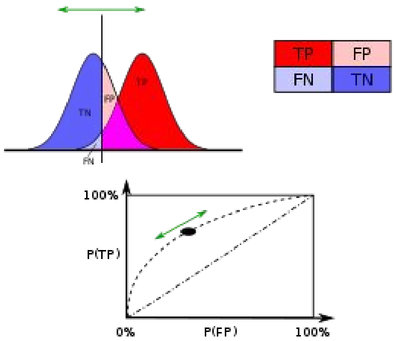
Different runs with data that's been shuffled between runs can classify the data differently

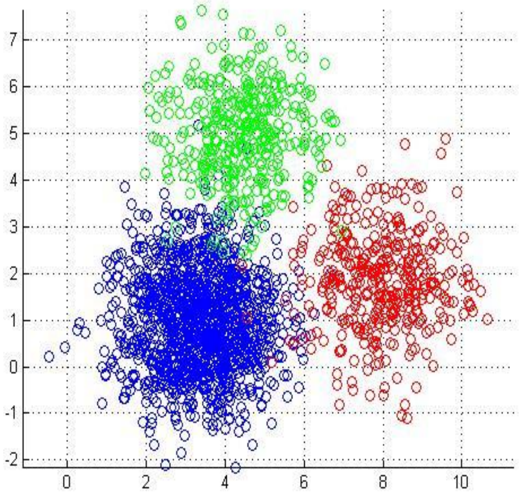
K-Means Clustering

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

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

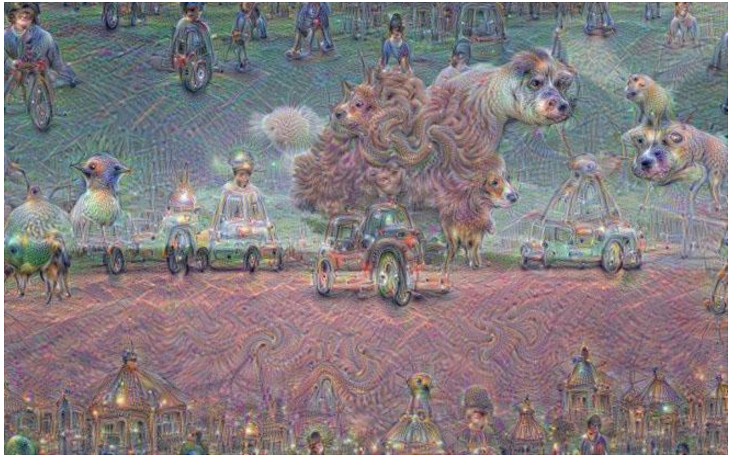
Restricted Deep Boltzmann Machines, Salakhutdinov and Hinton

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

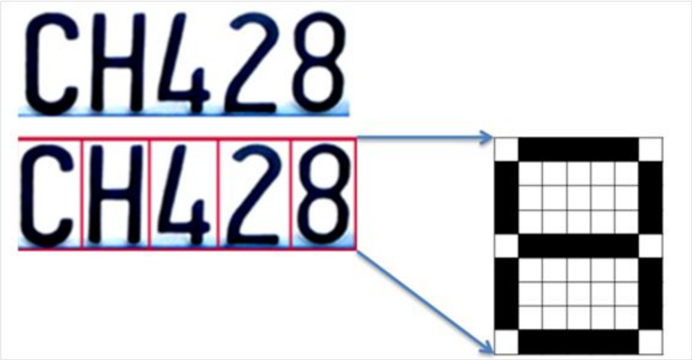
https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42241.pdf

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

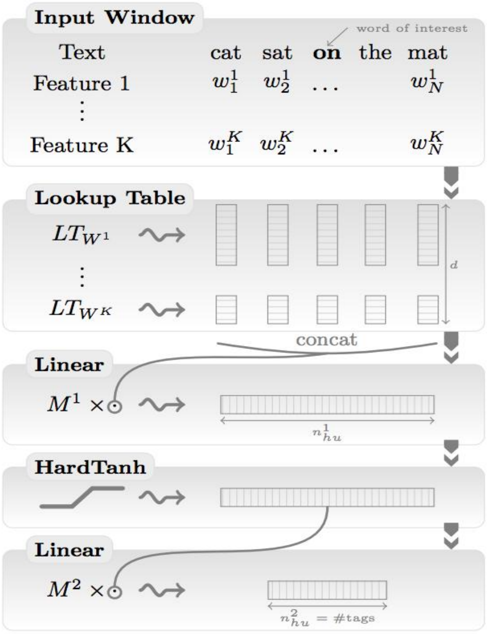


Figure 1: Window approach network.

Anomaly Detection

Make a prediction, points outside the prediction area are anomalies

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

Synthesis and Sampling

Mostly used in the game industry to create landscapes (fractal), characters (AI), etc   Gamasutra

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

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.

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

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)%:

    run training data in batches

    run validation every 100th epoch or so:

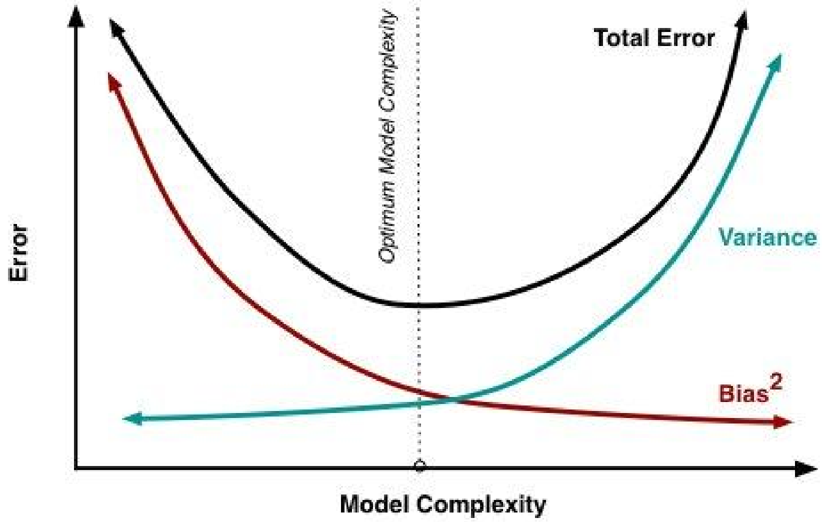
Then test on hold out data after training is complete

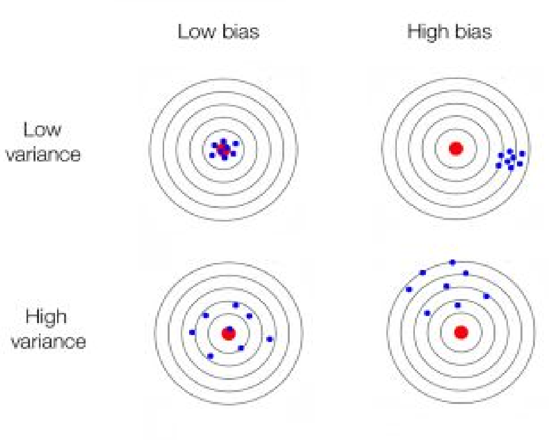
Training, testing, validation data must be statistically similar

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)

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

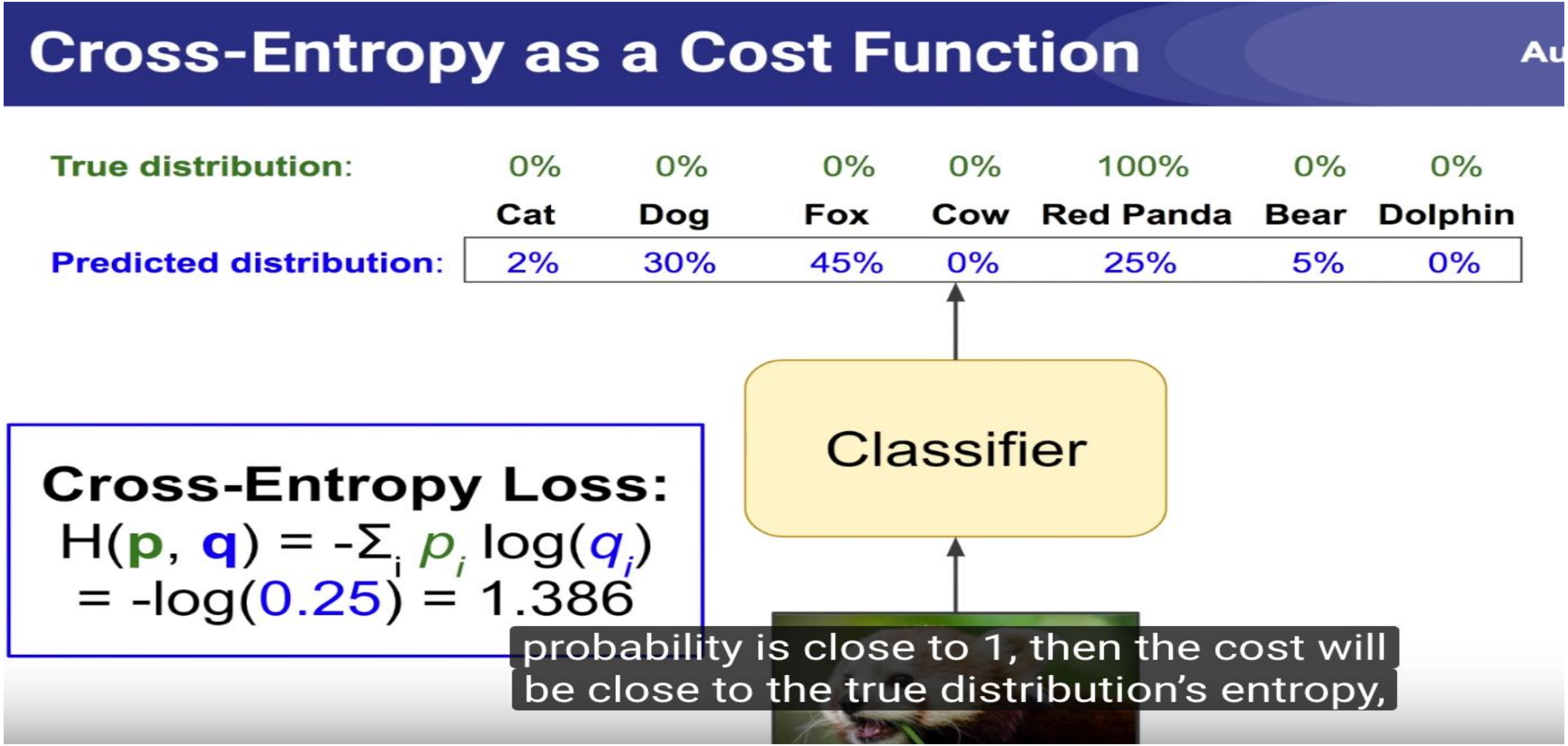
A short intro to entropy, cross-entropy and KL divergence

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

A Mathematical Theory of Communication, Shannon

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

Cross entropy as cost function



KL Divergence

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

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

The principle of maximum likelihood says that given the training data, we should use as our model the distribution f(w) that gives the greatest possible probability to the training data

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

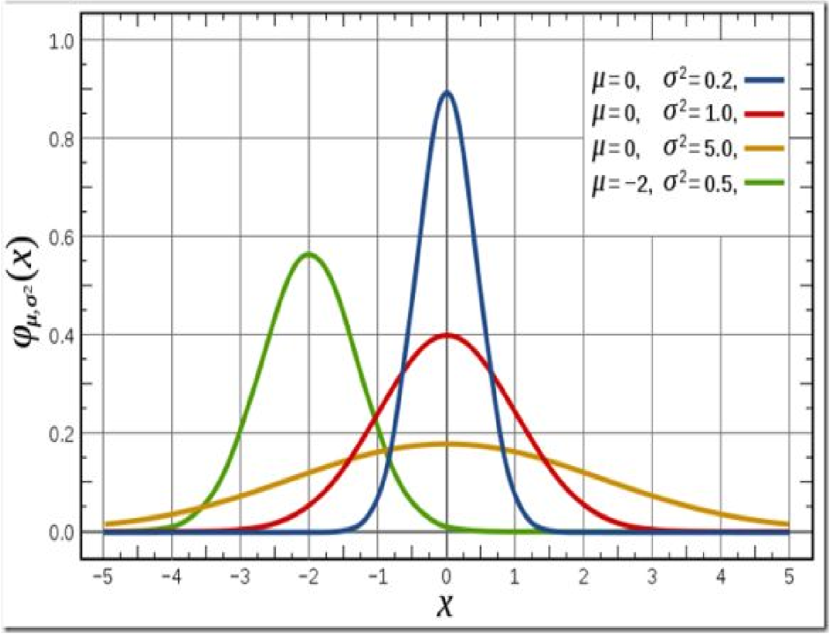
http://cseweb.ucsd.edu/~elkan/250B/logreg.pdf

Maximum Log Likelihood

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

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

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



Conditional Log Likelihood

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

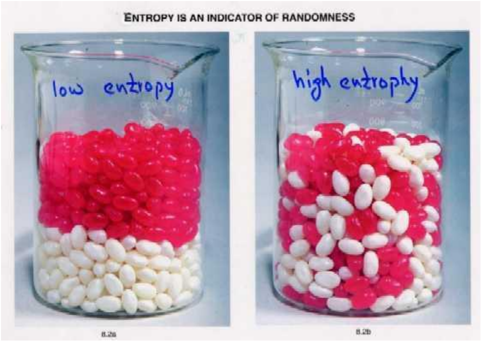
http://www.econ.ucla.edu/workingpapers/wp239.pdf

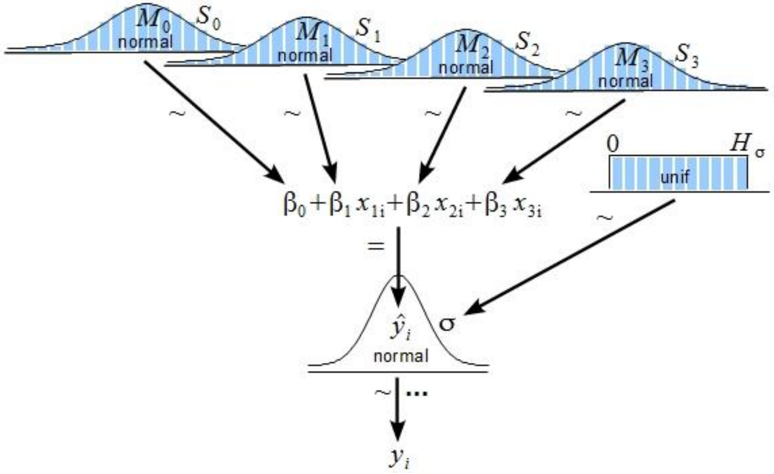
Bayesian Linear Regression

- 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

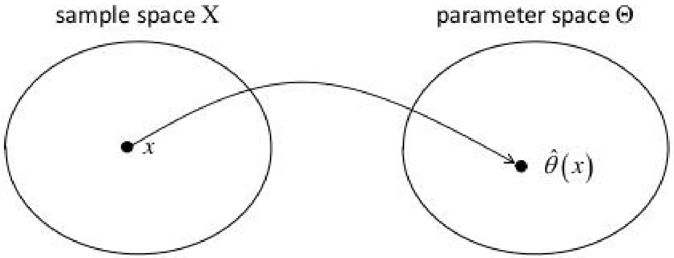




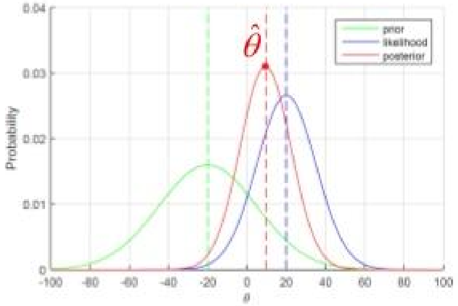
https://stats.stackexchange.com/questions/252577/bayes-regression-how-is-it-done-in-comparison-to-standard-regression

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 Guassian 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/

Probabilistic Supervised Learning

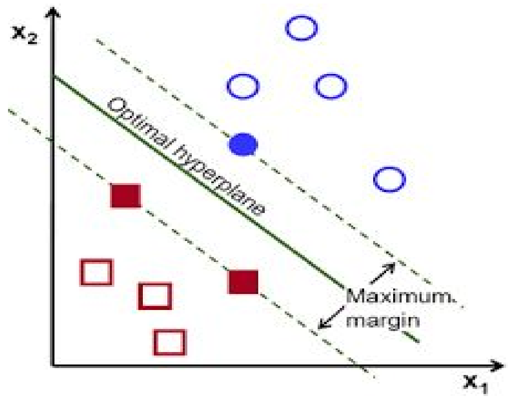
Normal (Gaussian Distribution)

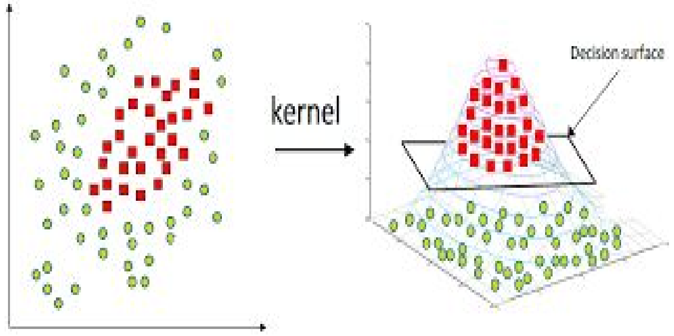
Probability of y given x

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 non-linear, can appromimate 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)

Support Vector Machines





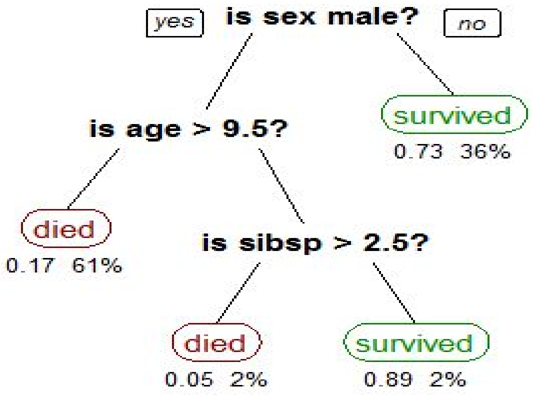
O(n^2), can only separate 2 classes per SVM

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

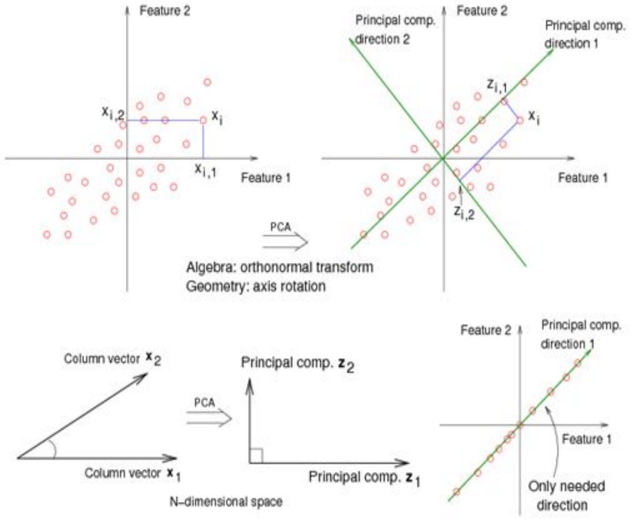
Using decision trees to see what a neural network is doing

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



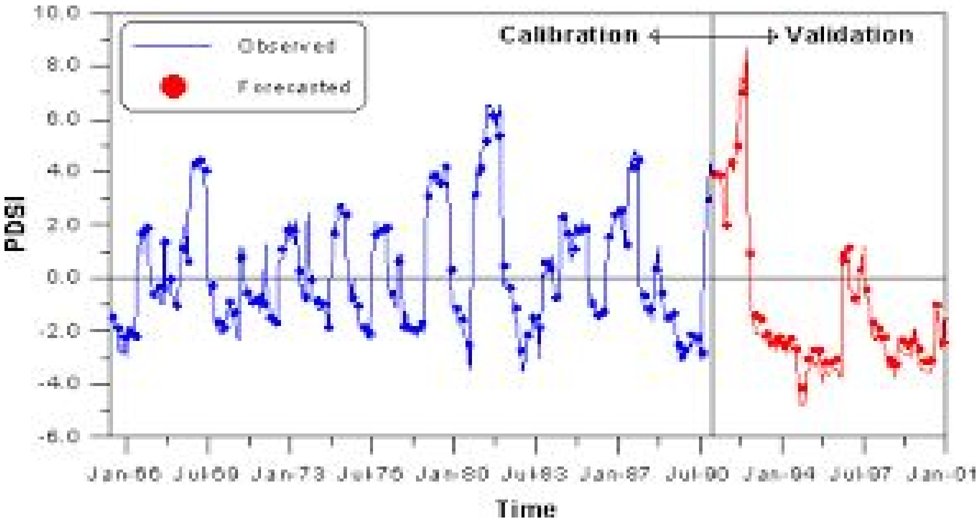
https://onlinecourses.science.psu.edu/stat857/node/35

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

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 things being tried (or re-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

Gradient Boosted Trees http://xgboost.readthedocs.io/en/latest/model.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

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

J(w) = Mean Standard Error + 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

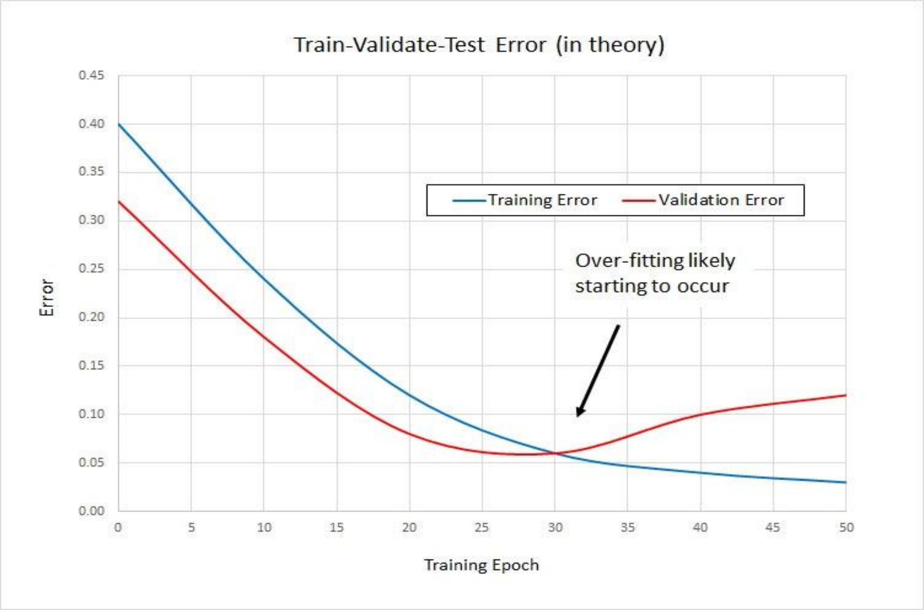
Rotational invariance is the main reason it is used instead of L1

L2 = complexity_constant * sum(W)^2

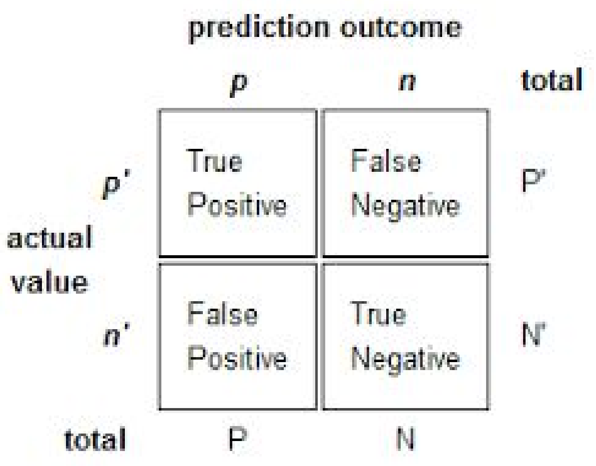
- L1 and L2 have the effect of choosing a different prior in Bayesian solutions
- Elastic net uses both L1 and L2

Capacity, Overfitting, Underfitting

Early stopping



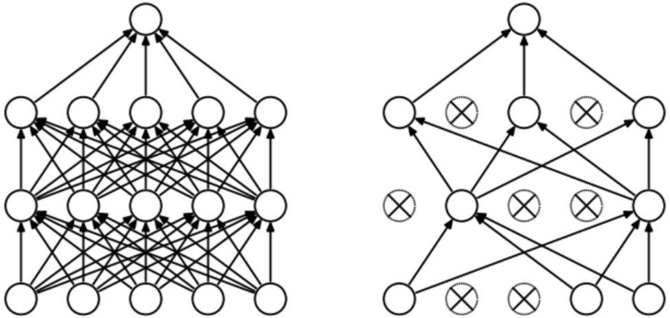
Confusion Matrix - very useful for debugging



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.



(a) Standard Neural Net (b) After applying dropout.

Dropout: A Simple Way to Prevent NN from Overfitting

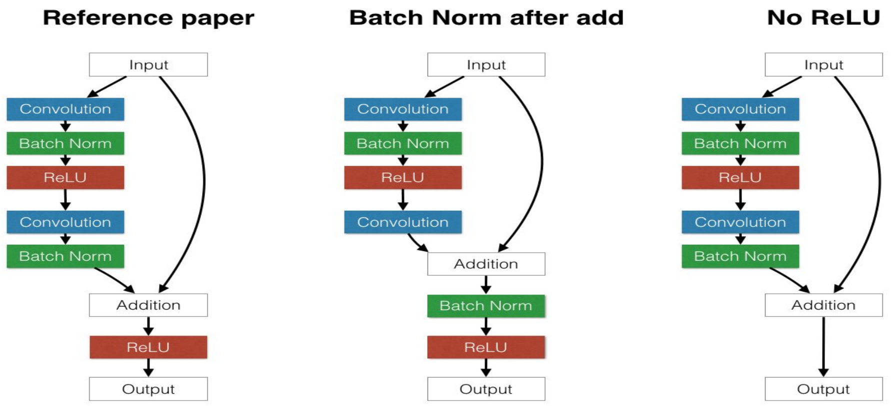
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

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 re-fed to later layers



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

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

Resources

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

Must Know Tricks in Deep Neural Networks

http://lamda.nju.edu.cn/weixs/project/CNNTricks/CNNTricks.html

more resources...

University of Chicago is also doing a deep learning course on this book

Slides and links to recommended papers: http://ttic.uchicago.edu/~shubhendu/Pages/CMSC35246.html

ArXiv Papers https://arxiv.org/ Code for Arvix papers: http://www.gitxiv.com/

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

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

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

Google Research Blog: https://research.googleblog.com/

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