Neural Network Basics
(short version)

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

(Linda MacPhee-Cobb)

Main types of networks

Feed forward: map dot product of inputs to outputs

Convolutional: use random noise convoluted with input to detect edges

Recurrent: stack series data shifting one step, add memory

AutoEncoder: squeeze data (compress) <-> decompress

Reinforcement: adjust rewards over time, keeps an array of error data instead of one error

Alpha Go: min-max tree with a feed forward network tacked on end

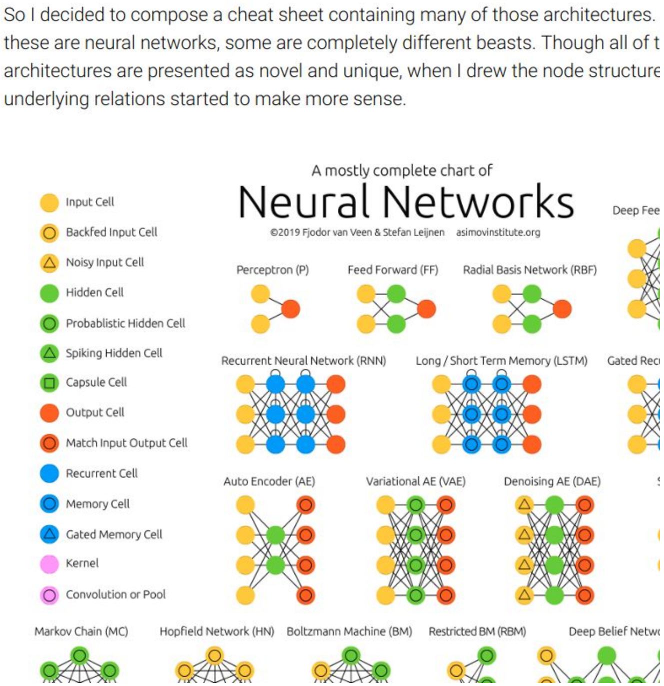
No Free Lunch Theorem

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.

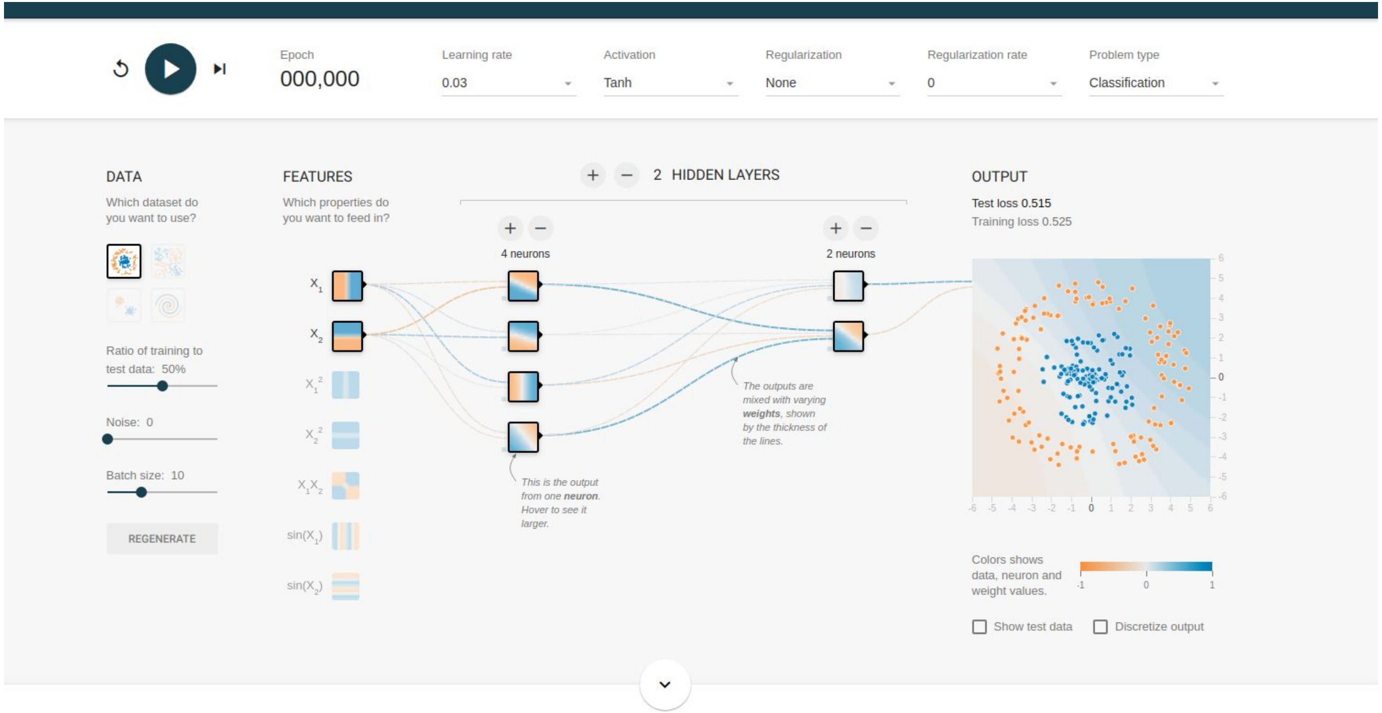
- convolutional networks for vision
- recursive networks for time series
- deep vs wide layers



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

Tensorflow playground

http://playground.tensorflow.org



Building a Machine Learning Algorithm

Cleanup data

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

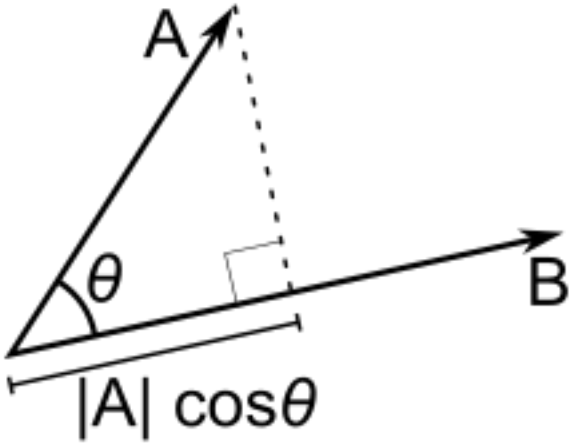
Minimize difference between cost function and actual data

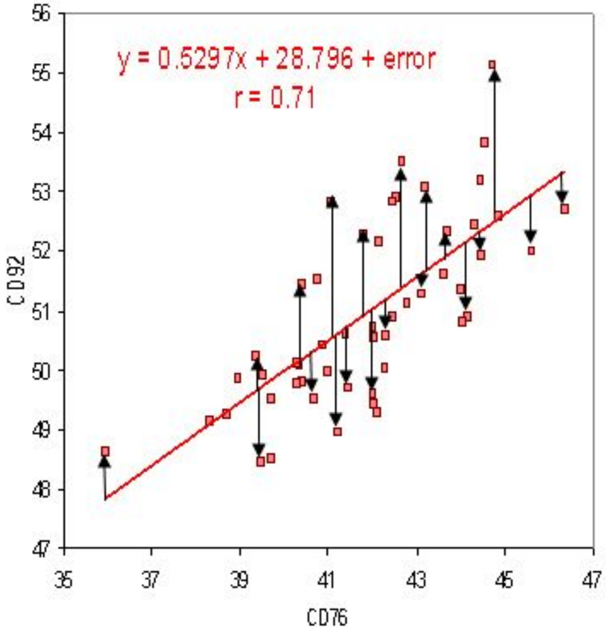
Forward Pass Feedforward Networks

prediction = activation_function (dot(w.T, x))

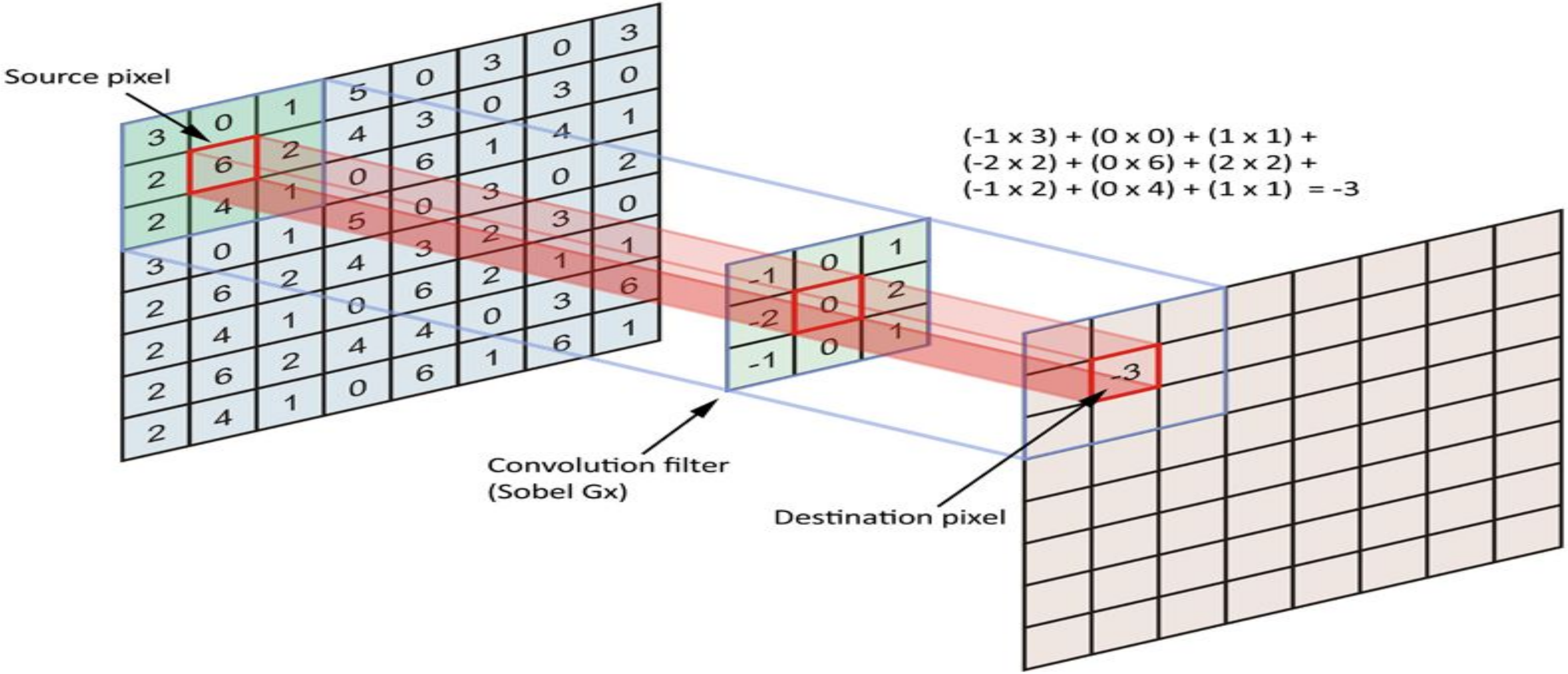
MSE = 1/n_samples * Sum(predicted - actual)^2

Solve for unknown f(x)





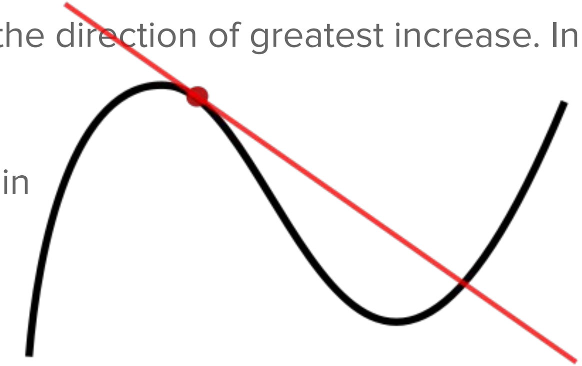
Forward Pass Convolutional Network



Backward Pass

Uses derivative of activation function to adjust the weights

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.

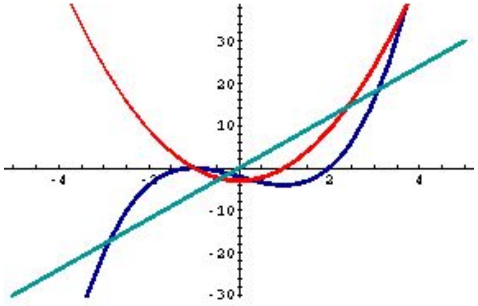


The 2nd derivative can tell you if you are at a max or min

f" < 0 max

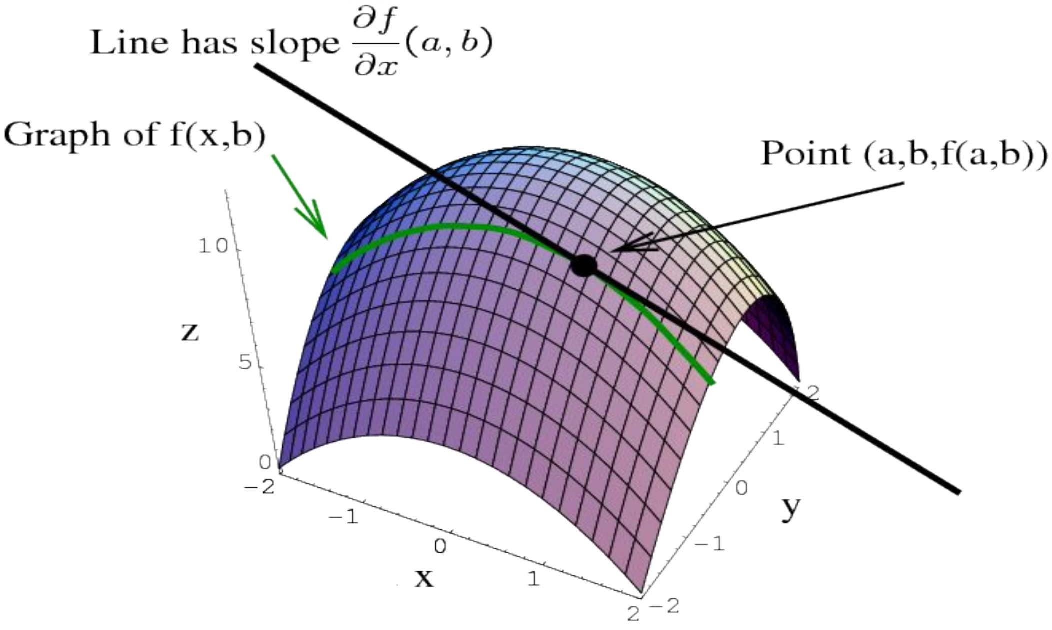
f" == 0 no information

f" > 0 minimum



Partial Derivative

Derivatives calculated on vectors (gradient)

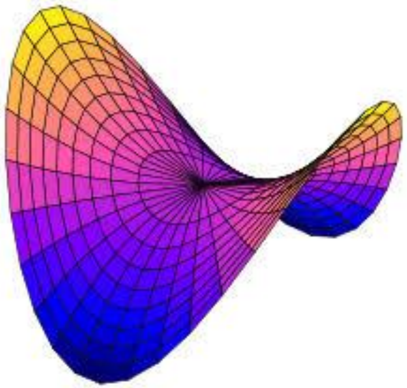


Gradient Based Optimization

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



Stochastic Gradient Descent, deep nets change everything

Local mins become saddle points, then vanish completely

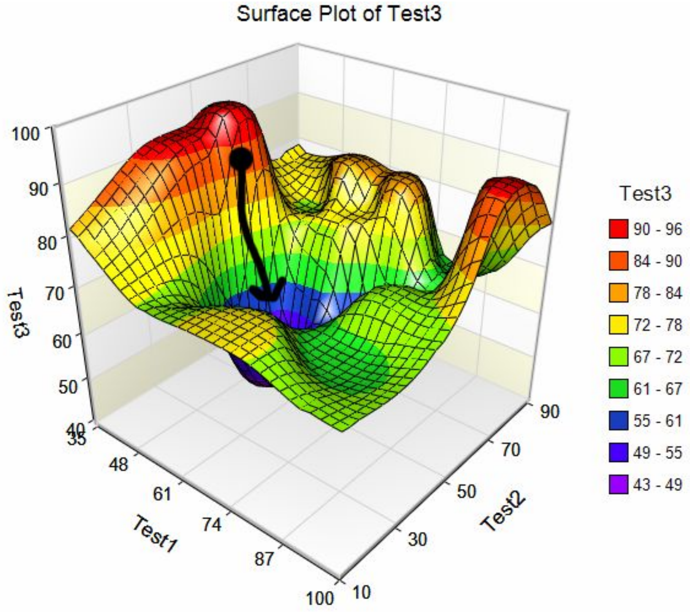
Possible multiple correct paths through the network

Repeat until convergence

{

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

}



Data ...


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Independent Identically Distributed

Garbage in, garbage out

Independent --- features cannot be correlated

- one hot encoding (M 1, F 0 should be M_F 0, 1) ... leave one out

Identically Distributed and Balanced

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

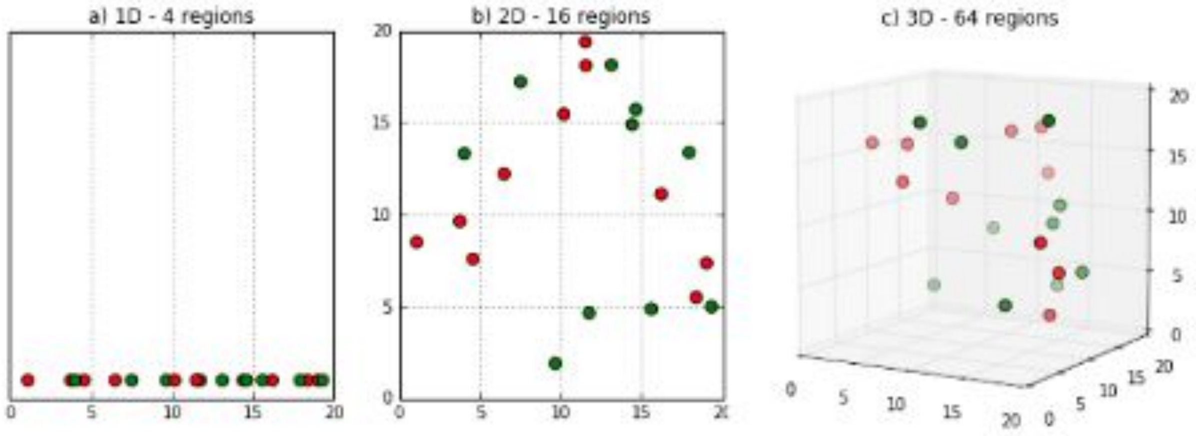
https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.corr.html

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

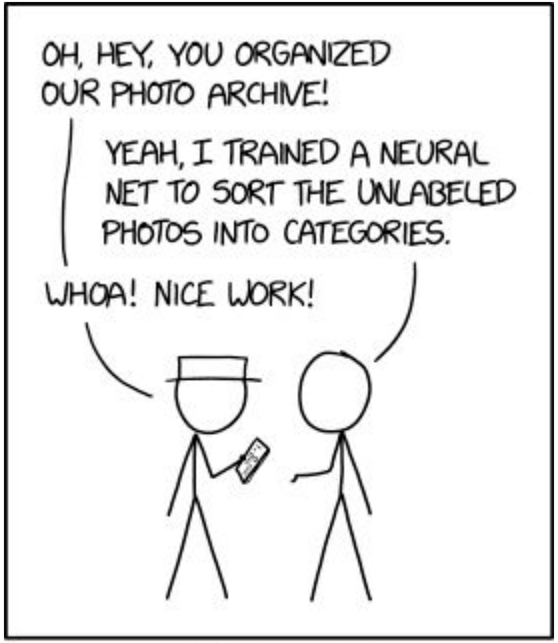
Curse of dimensionality

Linear algebra - unknowns can't outnumber features, samples

- current thinking is there might be multiple correct ways through the deep networks



Details...



ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND, YOU CAN TECHNICALLY SAY YOU TRAINED A NEURAL NET TO DO IT.

Goals (Cost functions)

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.

If it's not converging try a different error function

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 car had learned to follow the curb, not the road

- it's not always learning what you think it is learning

The Performance Measure

~ 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

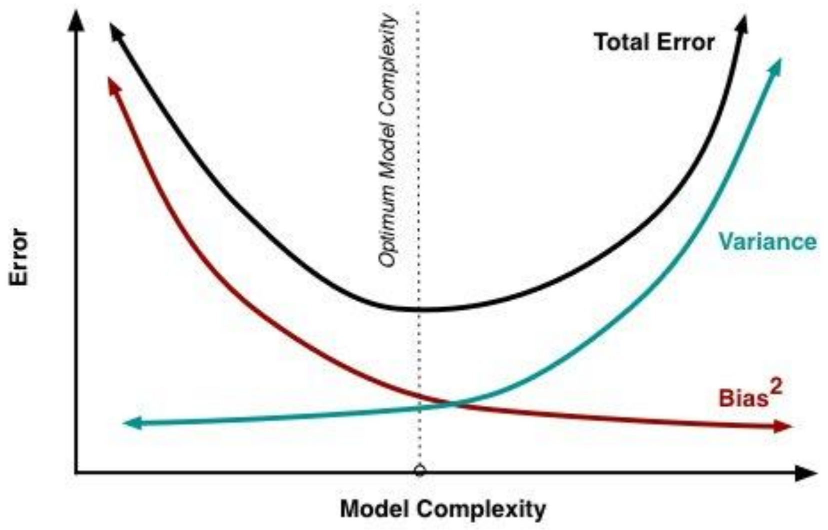
run validation every 100th epoch or so:

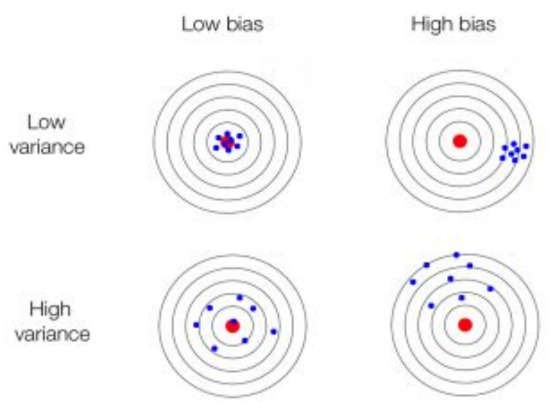
Then test on hold out data after training is complete (* never let subcontractors, contest entrants see hold out data)

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

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

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

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

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

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

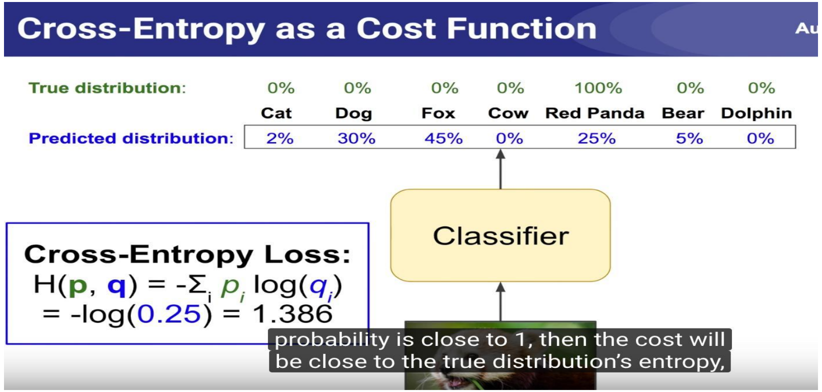
The principle of maximum likelihood says that given the training data, we should use as our model the distribution f(w) that gives the greatest possible probability to the training data

- not a distance measure (you can reverse your Ps and Qs but keep them consistent)

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 = Entropy + KL Divergence

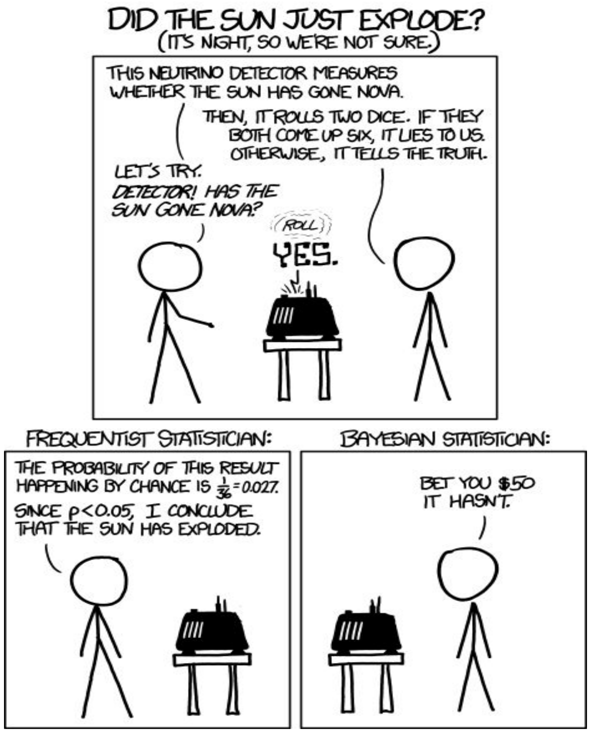
Entropy = minimum bits needed to transmit actual information

KL Divergence = difference between predicted and actual information

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

Bayes vs Frequentist

Statistical religious wars



Bayesian vs Frequentist

Bayesian

Prior beliefs

Fixed parameters

mu = mx + b

y = N(mu, std)

N - normal distribution

std - standard deviation

mu - mean

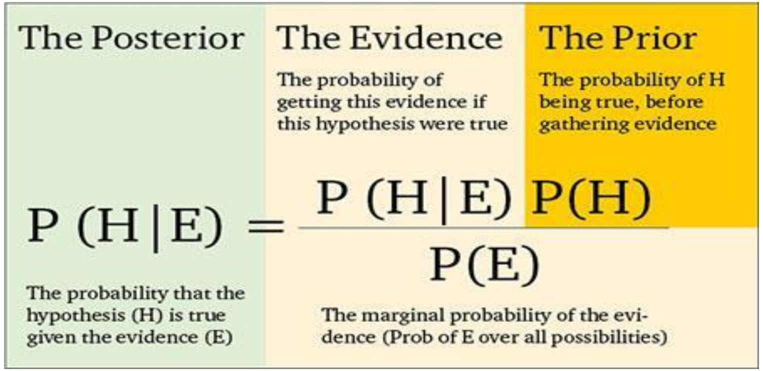
Frequentist

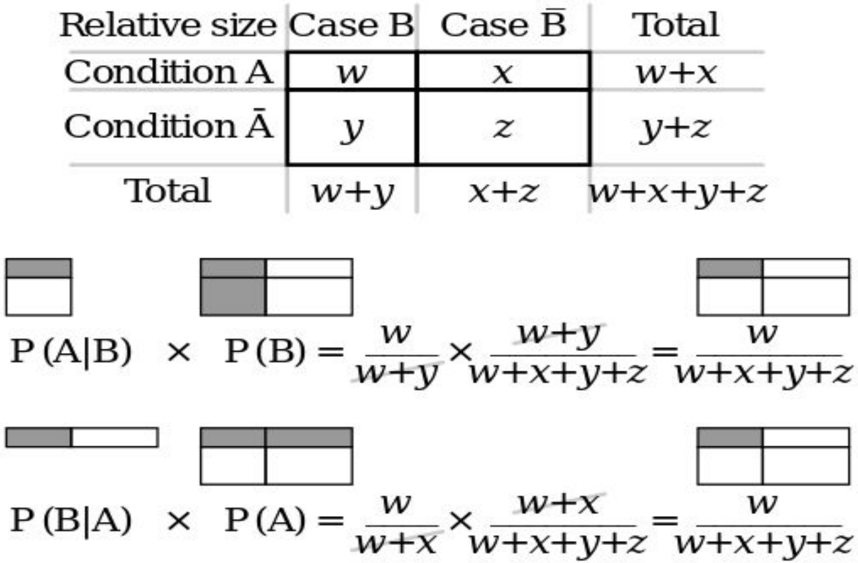
Repeatable random sample

Fixed data

y = mx + b

Bayesian Statistics





Intuitive understanding of Bayes

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

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

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

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

MLE (Frequentist) vs MAP (Bayesian)

Both compute a single value, not a distribution

MLE - fit a Gaussian to the data

MAP - uses posterior distribution

Frequentist is a specific case of Bayesian

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

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

http://papers.nips.cc/paper/3-supervised-learning-of-probability-distributions-by-neural-networks.pdf

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

Useful things

Cross validation

Split the dataset into 3 or more sections:

Train on one or more sections
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 (city blocks distance)
- L2 regularization (as the crow flies distance)
- 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

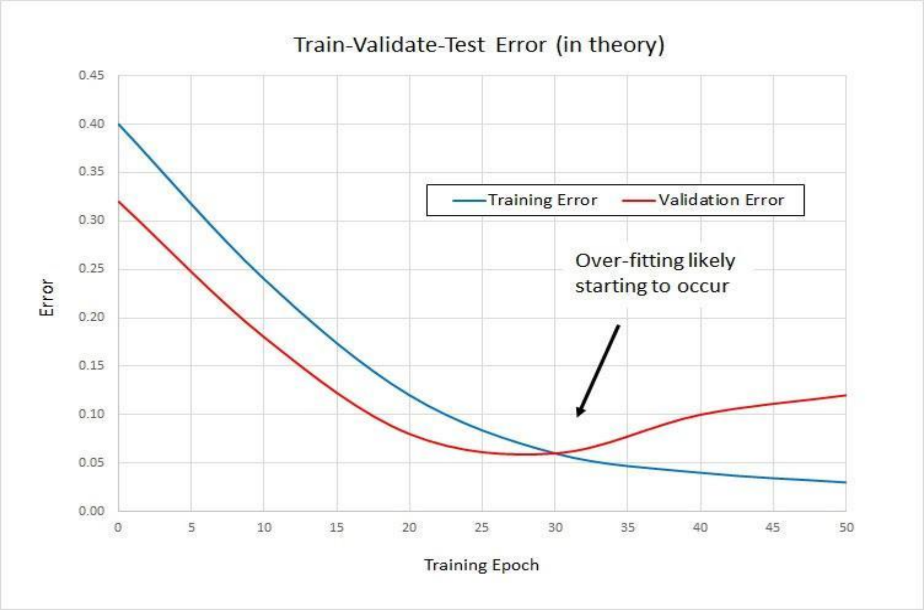
Rotational invariance is the main reason it is used instead of L1

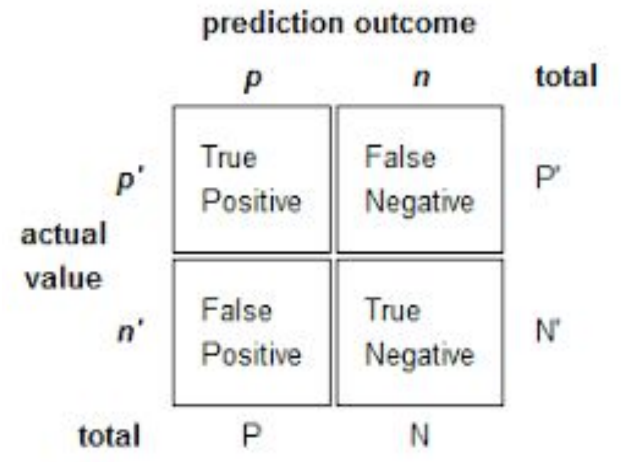
L2 = complexity_constant * sum(W)^2

- 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





Dropout

Averages predictions over all weights

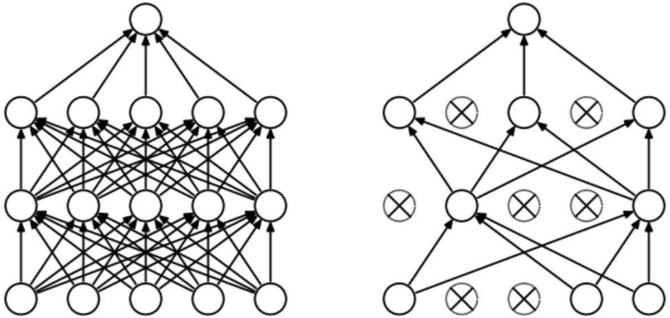
Randomly exclude a percent of hidden nodes from the neural network, change and randomly exclude a different group on next batch during training.

How to:

Create a matrix the size of the weights matrix

Randomly fill with ones and zeros

Multiply weights by the matrix



(a) Standard Neural Net

(b) After applying dropout.

Dropout: A Simple Way to Prevent NN from Overfitting

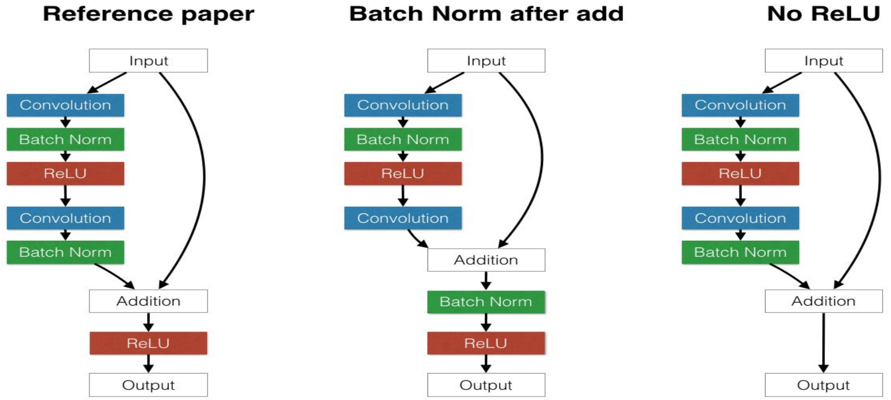
https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

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 fed to later layers



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

Probabilistic Supervised Learning

Normal (Gaussian Distribution)

Probability of y given x

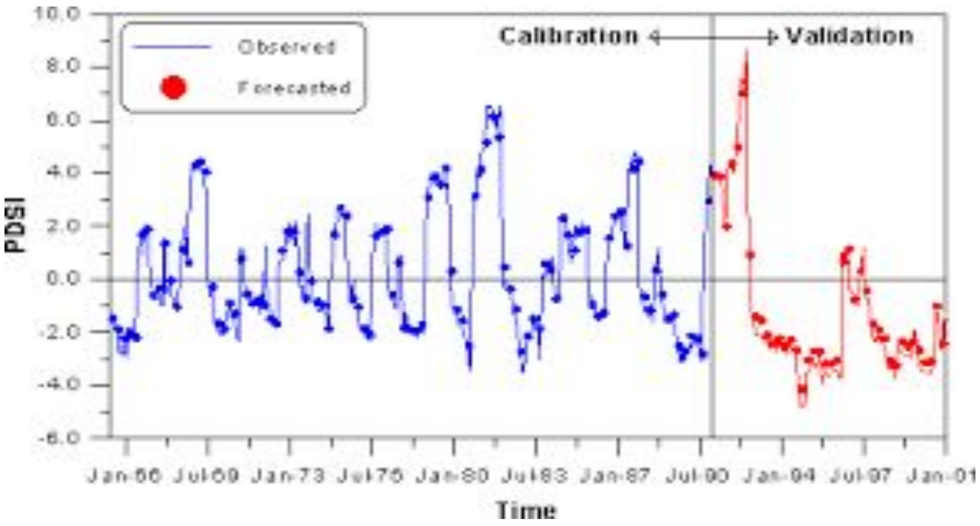
The activation functions are used to create non-linearity

- sigmoid, oldest, may cause vanishing gradients
- tanh, steeper than sigmoid, still can cause vanishing gradients but usually gives better results than sigmoid
- 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 to 1, used in last layer to give multi-class probability)

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 an open source tool for doing forecasts with large

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

More information...

Everything old is new again

New old things being tried with better hardware, more data

Bayes algorithm, 1700s invented by a monk studying chance

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 (BFSkinner) http://www.incompleteideas.net/book/bookdraft2018jan1.pdf

AlphaGo is a modern remake of Blondie24 http://www.davidfogel.com/

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/

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

Otoro: http://blog.otoro.net/ Misc cool AI

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 Books https://www.manning.com/meap-catalog

Numerical computing

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

http://numerical.recipes/oldverswitcher.html

Introduction to Statistical Learning (high school math)

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

Elements of Statistical Learning   (college level math)

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

The Nature of Statistical Learning Theory (graduate level math)

https://www.springer.com/us/book/9780387987804

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

WildML http://www.wildml.com/