Numerical Computation

Topics in Chapter 4 of Deep Learning

https://github.com/timestocome

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, integers wrap

vanishing gradients [clip gradients, add epsilon]

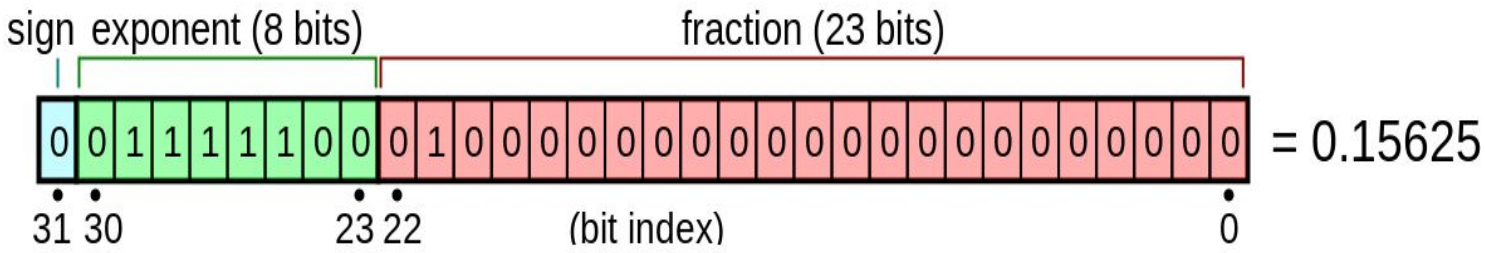
exploding gradients [clip gradients, use regularization]

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

IEEE Standard

- 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

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

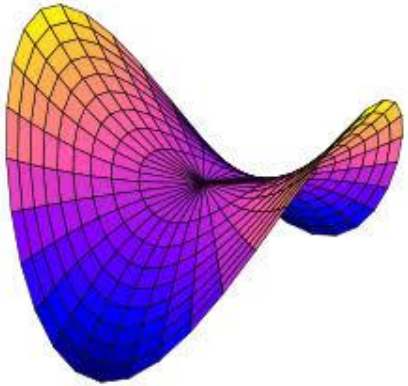
In a small network this is a local minima, in a deep network it's a saddle point

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

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

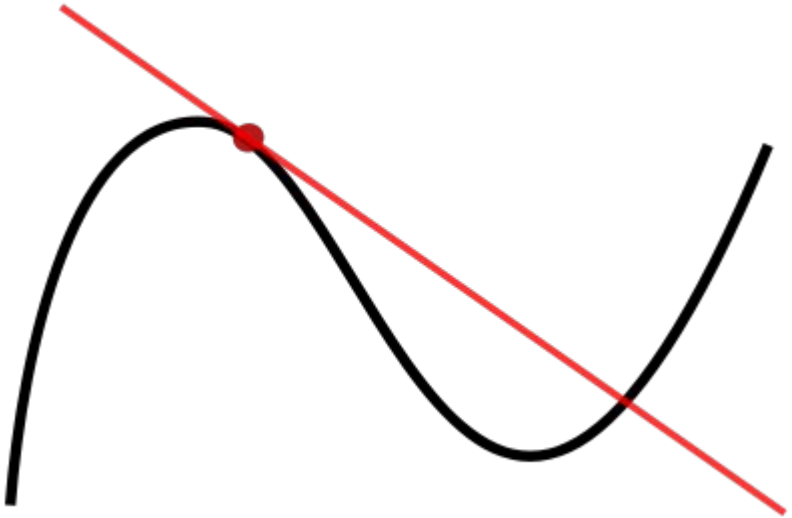
Gradient Based Optimization

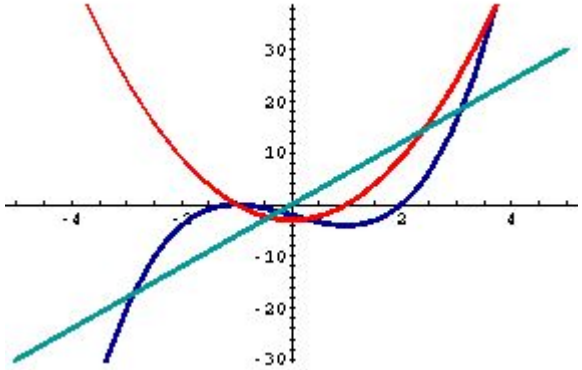
Goal is to minimize or maximize a cost function (loss function, error function)



Derivative

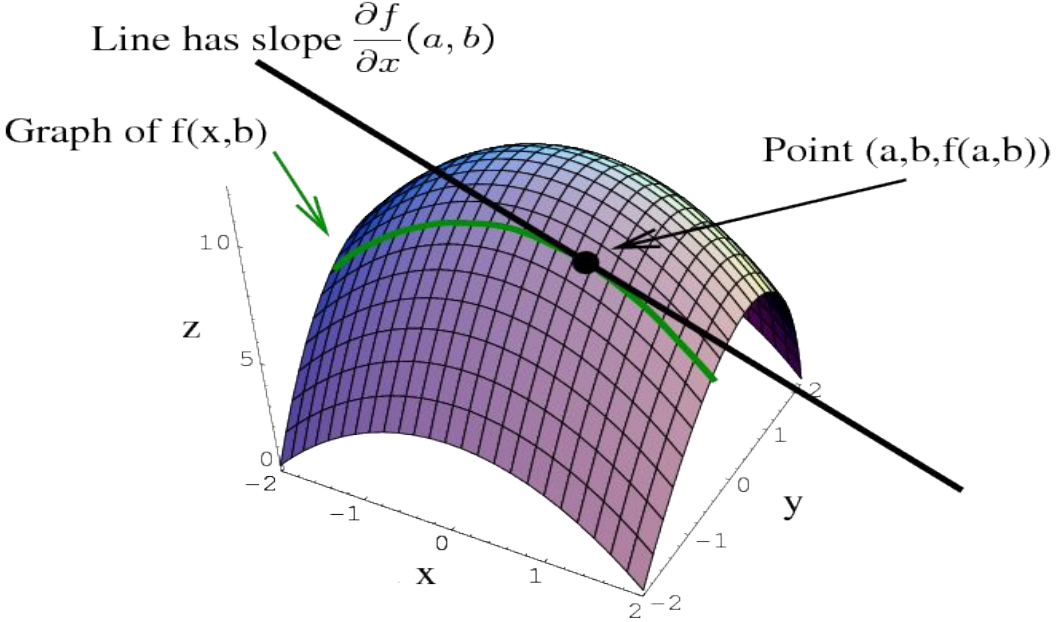
Measures the tangent line on a slope which gives you the direction of greatest increase. In neural networks we use the negative of it to get steepest descent. The 2nd derivative can tell you if you are at a max or min





Partial Derivative

Derivatives calculated on vectors (gradient)



Jacobian Matrix

Used to take derivatives of vectors (arrays), aka gradient

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

Hessian Matrix

.... is the 2nd derivative of the vector, the gradient of the gradient

used to find maxes (negative),

mins (positive), saddle points

(mixed signs)

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

Constrained Optimization

Used to force RNNs, Swarms, SVM to a global minimum solution of a convex problem

Maximize f(x,y) subject to g(x,y)

.... possibly h(x,y), j(x,y).....

Lagrangian

Example: Solve for both Revenue and Cost

R(x,y) = lambda * C(x,y) # 3 unknowns (lambda, x, y), need 3 equations

Take gradient and set equal to zero

L(x, y, lambda) = R(x,y) - lambda * C(x,y)

L(x, y, lambda) = 0

Lagrange Multiplier and Constrained Optimization

https://datastoriesweb.wordpress.com/2017/06/25/lagrange-multiplier-and-constrained-optimization/

Karush-Kuhn-Tucker Approach

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

- 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:

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

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

and

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

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

Machine Learning Basics

Chapter 5 Deep Learning

Resources

ICLR

https://www.youtube.com/channel/UCqxFGrNL5nX10lS62bswp9w

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

The Task

This is what you want your neural network to solve.

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

Classification is unsupervised learning and might not always classify in a useful format. Re-evaluate your features if it's not returning useful classes

Classification with missing inputs

Joint Training of Deep Boltzmann Machines for Classification, Goodfellow

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

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)

Frequentist Linear Regression

Transcription

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

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

Robust PCA

http://statweb.stanford.edu/~candes/papers/RobustPCA.pdf

Bayesian Anomaly detection

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

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 90%: (pick a number

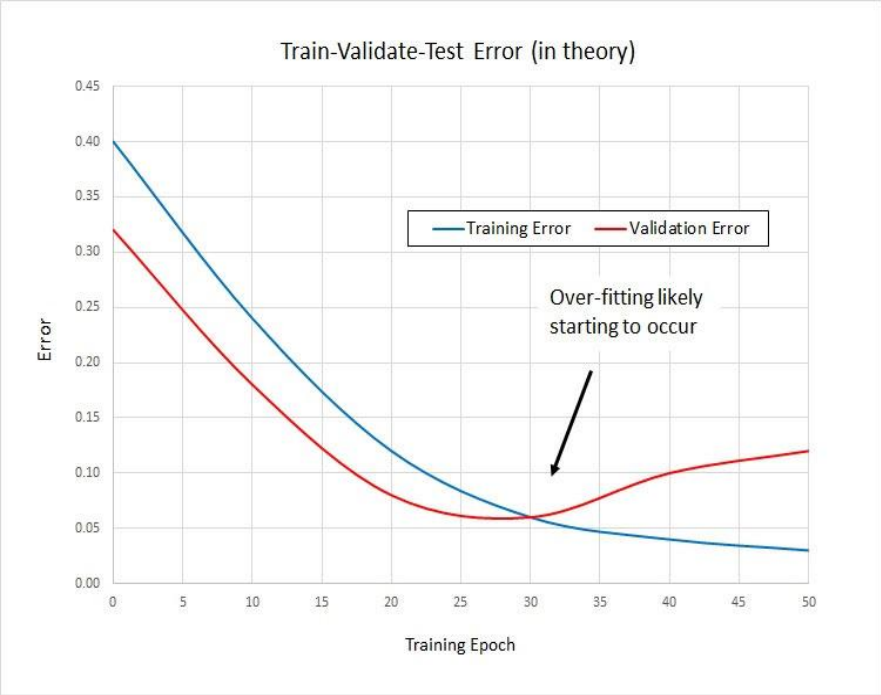
    run training data in batches

    run validation every 100th epoch or so:

Then test on hold out data.

Training, testing, validation data must be statistically similar

Capacity, Overfitting, Underfitting



Confusion Matrix - very useful for debugging

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

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

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

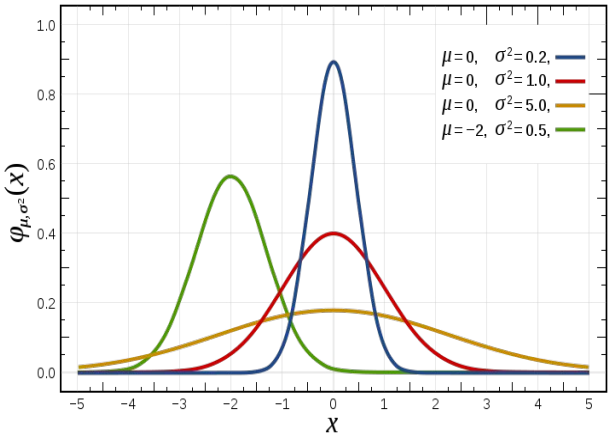
A recent development is that randomly picking values for a grid search is more effective than stepping through all values.

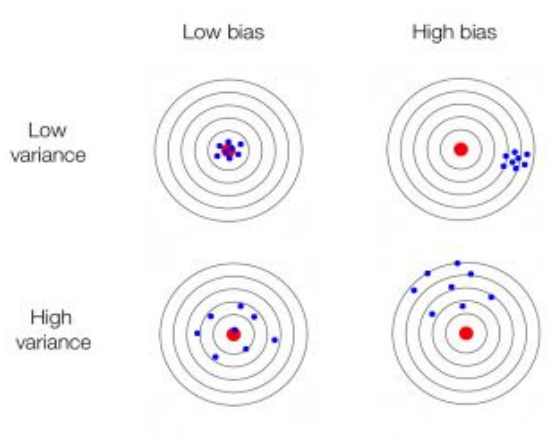
http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Bias vs Variance

High bias happens during under fitting, high variance during overfitting

Standard error is the square root of variance





Bias is hitting the 12 on the dart board every time, variance is hitting a wide area on the board

Maximum Likelihood Estimation 1

A Mathematical Theory of Communication, Shannon

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

Entropy = how many bits are needed to encode information

$$\text{Entropy} = -\sum_{i} P_i \log_2 P_i$$

information theory uses Log2, not e, not 10

Maximum Likelihood Estimation 2

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

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

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

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

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

Maximum Likelihood 3

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

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

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

Conditional Log Likelihood

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

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

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

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

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

Independent Identically Distributed

Garbage in, garbage out

Independent - features cannot be correlated

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

Identically Distributed

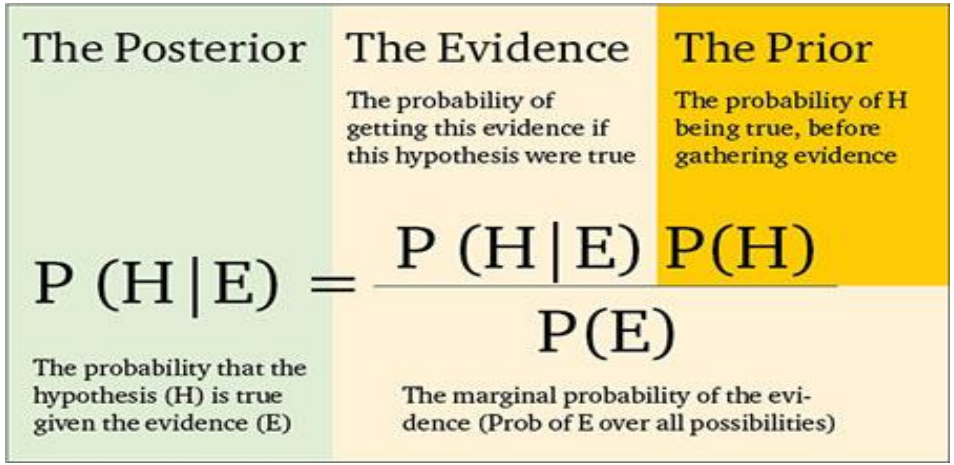
- training, validation, test data
- outputs (if 90% of the output is true, the network will always guess true and you'll have 90% accuracy - check the confusion matrix)

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

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

Bayesian Linear Regression

Maximum a Posteriori (MAP) Estimation

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

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

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

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

Maximum Likelihood and Maximum Posteriori Estimation

$$\theta_{MAP} = \arg\max_{\theta} \sum_{i} \log P(x_i|\theta)P(\theta)$$

$$= \arg\max_{\theta} \sum_{i} \log P(x_i|\theta)\, const$$

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

$$= \theta_{MLE}$$

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

Probabilistic Supervised Learning

Normal (Gaussian Distribution)

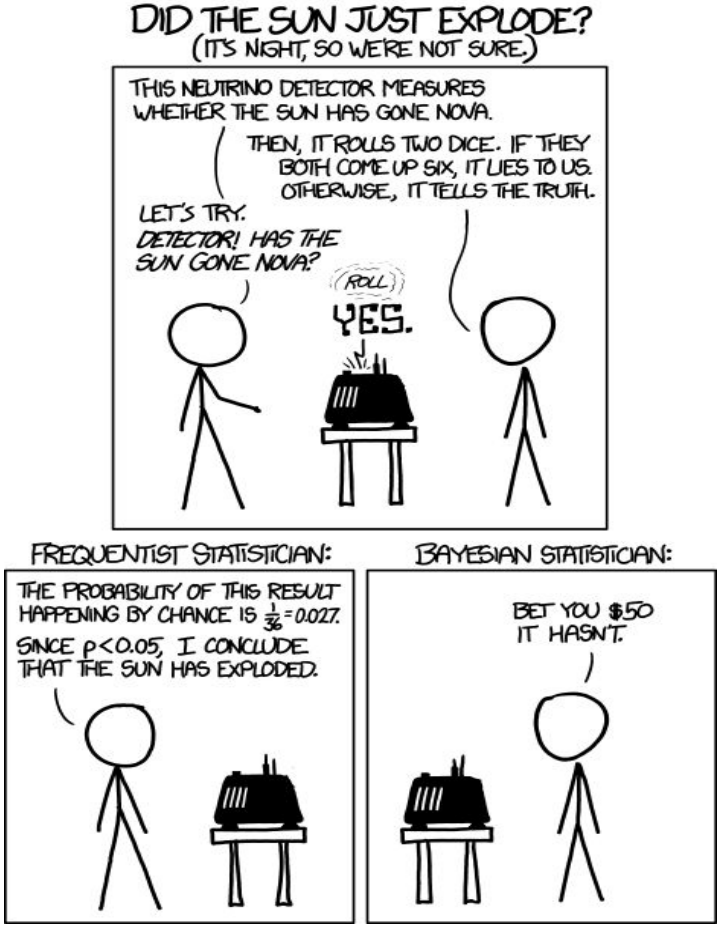
Probability of y given x

The activation functions are used to create non-linearity

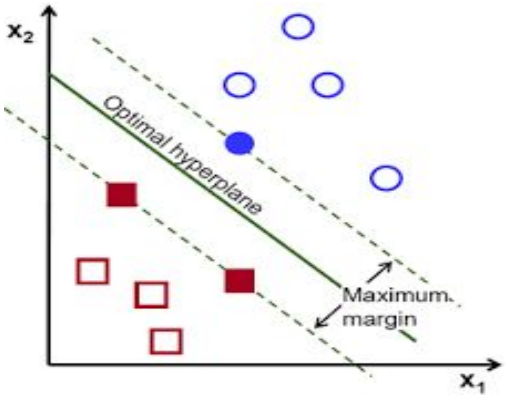
- sigmoid
- tanh
- relu
- softmax (used in last layer to scale values so they sum up to 1)

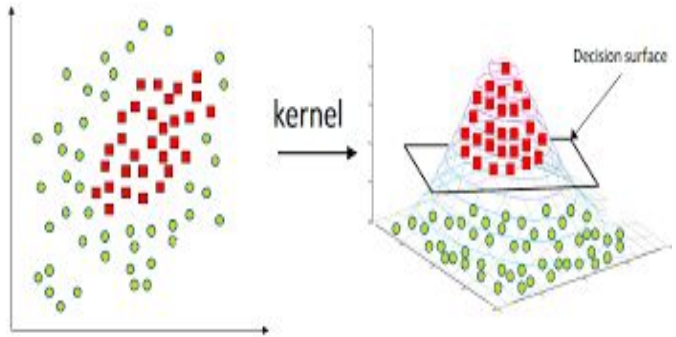
Bayesian vs Frequentist

Bayesian	Frequentist
Prior beliefs	Repeatable random sample
Fixed parameters	Fixed data



Support Vector Machines



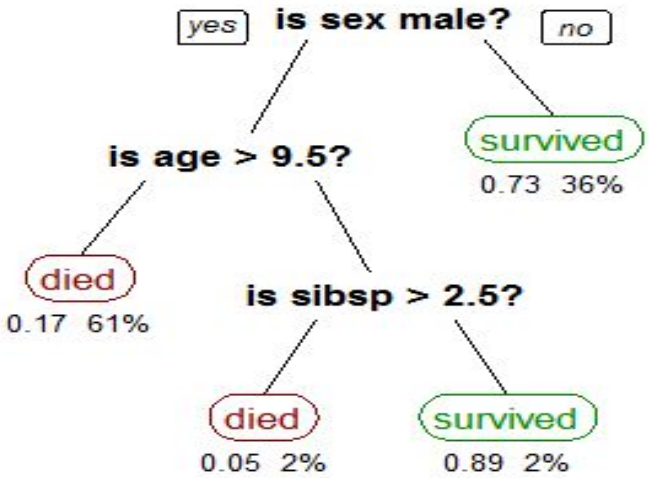


http://scikit-learn.org/stable/modules/svm.html

Intro to SVM

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

Decision Tree



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 the features are not easily to separate: images, lots of slightly correlated features all of which have a strong correlation to output

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

Facebook's Prophet is a great open source tool for doing forecasts with large amounts of data

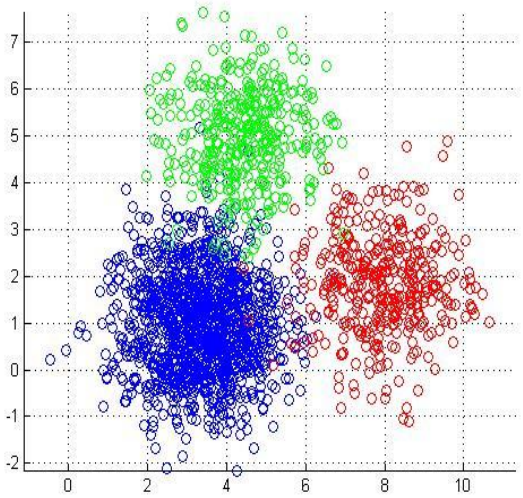
https://facebook.github.io/prophet/

K-Means Clustering

plot points

pick n points at random (centroids)

put each point in the group of the closest centroid



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

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

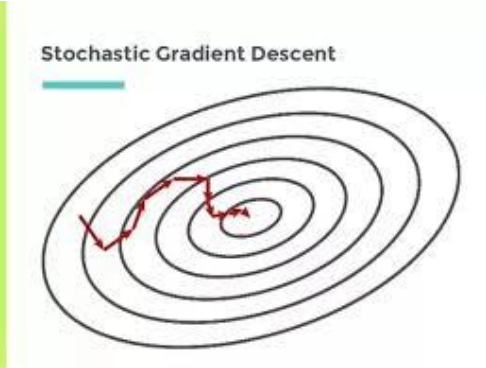
Stochastic Gradient Descent

Repeat until convergence

{

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

}



Building a Machine Learning Algorithm

Cleaned data

Multiply it 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)

NEAT, NeuroEvolution of Augmenting Topologies

Genetic Algorithms

Reinforcement Learning

Curse of Dimensionality

This is the same issue you've seen in linear algebra.

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.

Regularization

J(w) = Mean Standard Error(train) + (small constant) * wTw

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

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

L1 = c1 * sum(abs(W))

L2 Regularization

Least squares: Minimize the square of the difference

Limits weight size without driving any to zero

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

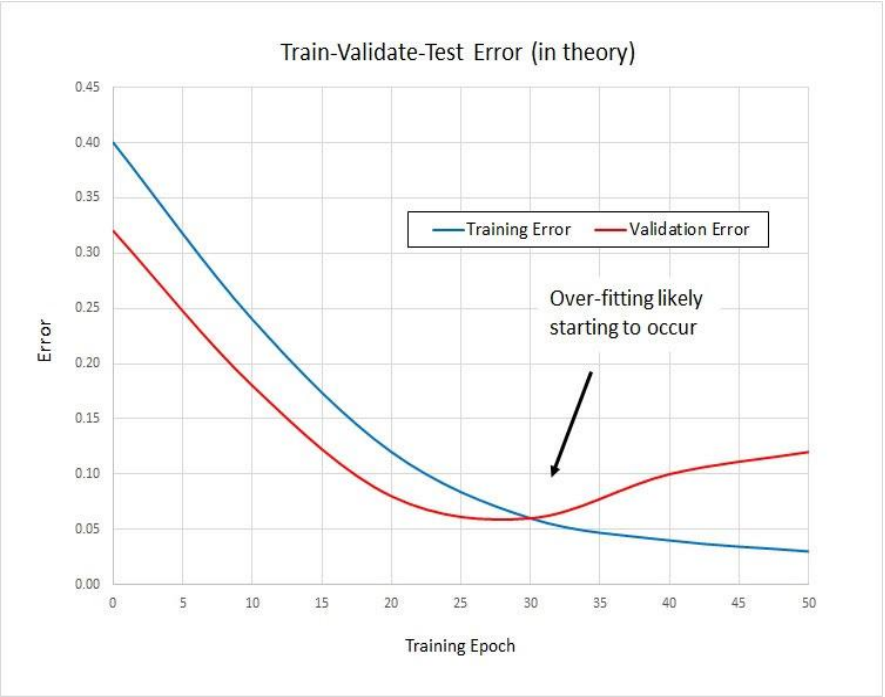
L2 = c2 * sum(W)^2

Early Stopping

Plot results during training and

stop training when training and validation

errors converge



Dropout

Averages predictions over all weights

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

Dropout: A Simple Way to Prevent NN from Overfitting

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

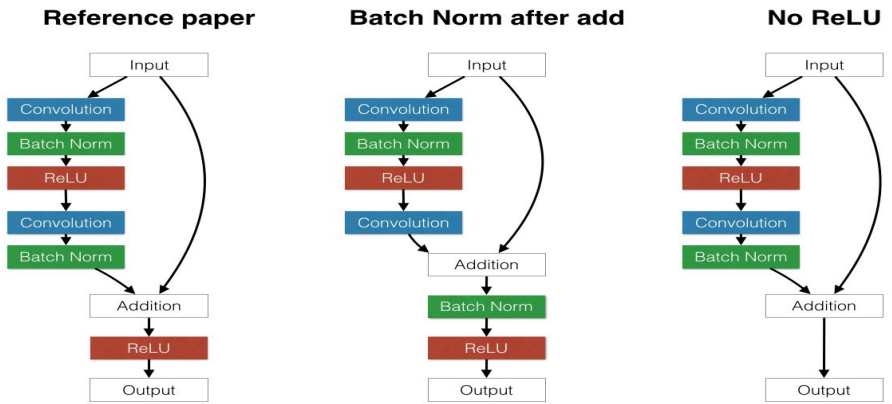
Batch Normalization

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



Manifold Learning

Find a useful way to represent the data in less dimensions

Example: represent road as 1D 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