Irving Fisher Committee on
Central Bank Statistics

BANK FOR INTERNATIONAL SETTLEMENTS

IFC – Bank Indonesia International Workshop and Seminar on *"Big Data for Central Bank Policies / Building Pathways for Policy Making with Big Data"*

Bali, Indonesia, 23-26 July 2018

Machine learning: classification and clustering[1]

Sanjiv R. Das,
Santa Clara University

[1] This presentation was prepared for the meeting. The views expressed are those of the author and do not necessarily reflect the views of the BIS, the IFC or the central banks and other institutions represented at the meeting.

Machine Learning: Classification and Clustering

Sanjiv R. Das
Santa Clara University
http://srdas.github.io

Bank of Indonesia
IFC Workshop on "Big Data for Central Bank Policies"
July 2018

Outline, Slides, Code

Machine learning ⊆ artificial intelligence

ARTIFICIAL INTELLIGENCE

Design an intelligent agent that perceives its environment and makes decisions to maximize chances of achieving its goal.
Subfields: vision, robotics, machine learning, natural language processing, planning, …

MACHINE LEARNING

Gives "computers the ability to learn without being explicitly programmed" (Arthur Samuel, 1959)

SUPERVISED LEARNING

Classification, regression

UNSUPERVISED LEARNING

Clustering, dimensionality reduction, recommendation

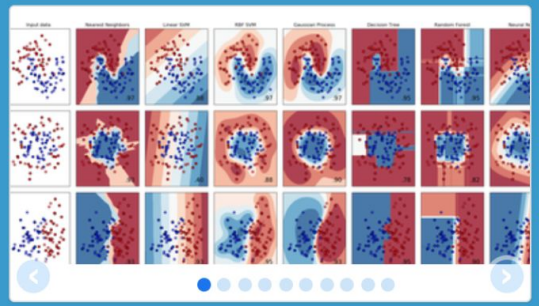
REINFORCEMENT LEARNING

Reward maximization

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

http://scikit-learn.org/stable/

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, … — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, … — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, … — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

Supervised and Unsupervised Learning

1. Machine Learning
2. Linear Models
3. Logistic Regression
4. Discriminant Analysis
5. Bayes Classifier
6. Support Vector Machines
7. Nearest Neighbors (kNN)
8. Decision Trees
9. Random Forest

1. Clustering
   a. K-means
   b. Hierarchical
2. Dimension Reduction
   a. PCA & Factor Analysis
3. Neural Networks and Deep Learning

Ensemble Methods

1. Bagging
2. Stacking
3. Boosting

Small Business Association Loans Dataset

```
#Import the SBA Loans dataset

sba = pd.read_csv("data/SBA.csv")
print(sba.columns)
print(sba.shape)
sba.head()
```

```
Index(['LoanID', 'GrossApproval', 'SBAGuaranteedApproval', 'subpgmdesc',
       'ApprovalFiscalYear', 'InitialInterestRate', 'TermInMonths',
       'ProjectState', 'BusinessType', 'LoanStatus', 'RevolverStatus',
       'JobsSupported'],
      dtype='object')
(527700, 12)
```

The dependent variable is the guaranteed amount as a percentage of the gross loan approved.

```
#Create the dependent variable
Y = sba.SBAGuaranteedApproval.astype("float")/sba.GrossApproval.astype("float")
X = sba[['GrossApproval', 'ApprovalFiscalYear', 'InitialInterestRate', 'TermInMonths',
         'RevolverStatus','JobsSupported']]
```

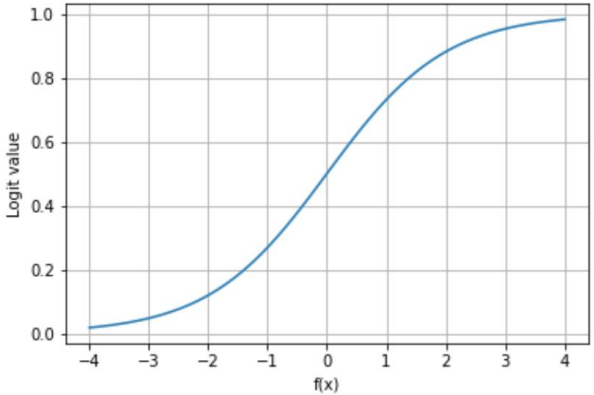
Program code:
http://srdas.github.io/Presentations/ClassClust/Linear_Regression.slides.html#/

Logistic Regression

Limited Dependent Variables

- The dependent variable may be discrete, and could be binomial or multinomial. That is, the dependent variable is limited. In such cases, we need a different approach.
- Discrete dependent variables are a special case of limited dependent variables. The Logit model we look at here is a discrete dependent variable model. Such models are also often called qualitative response (QR) models.

```
#Sigmoid Function
def logit(fx):
    return exp(fx)/(1+exp(fx))

fx = linspace(-4,4,100)
y = logit(fx)
plot(fx,y)
xlabel('f(x)')
ylabel('Logit value')
grid()
```



The Logistic Function

$$y = \frac{1}{1 + e^{-f(x_1, x_2, \ldots, x_n)}} \in (0, 1)$$

where

$$f(x_1, x_2, \ldots, x_n) = a_0 + a_1 x_1 + \ldots + a_n x_n \in (-\infty, +\infty)$$

Program code: http://srdas.github.io/Presentations/ClassClust/LogisticRegression.slides.html#/

Odds Ratio

What are odds ratios? An odds ratio (OR) is the ratio of probability of success to the probability of failure. If the probability of success is p, then

$$OR = \frac{p}{1-p}; \qquad p = \frac{OR}{1+OR}$$

Odds Ratio Coefficients

- In a linear regression, it is easy to see how the dependent variable changes when any right hand side variable changes. Not so with nonlinear models. A little bit of pencil pushing is required (add some calculus too).
- The coefficient of an independent variable in a logit regression tell us by how much the log odds of the dependent variable change with a one unit change in the independent variable. If you want the odds ratio, then simply take the exponentiation of the log odds.

http://srdas.github.io/Presentations/ClassClust/LogisticRegression.slides.html#/12

```
#Example

p = 0.3
OR = p/(1-p)
print('OR old =', OR)

beta = 2
OR_new = OR * exp(beta)
print('OR new =', OR_new)

p_new = OR_new/(1+OR_new)
print('p new =', p_new)
```

```
OR old = 0.4285714285714286
OR new = 3.1667383281131363
p new = 0.7600041276283266
```

Classification Metrics

1. Accuracy: the number of correctly predicted class values.
2. ROC and AUC: The Receiver-Operating Characteristic (ROC) curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) for different levels of the cut-off posterior probability. This is an essential trade-off in all classification systems.
3. TPR = sensitivity or recall = TP/(TP+FN)
4. FPR = (1 − specificity) = FP/(FP+TN)

1. Precision = $\frac{TP}{TP+FP}$
2. Recall = $\frac{TP}{TP+FN}$
3. F1 score = $\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$

(F1 is the harmonic mean of precision and recall.)

Confusion matrix

```
print(metrics.confusion_matrix(y_test, predicted))
print(metrics.classification_report(y_test, predicted))
```

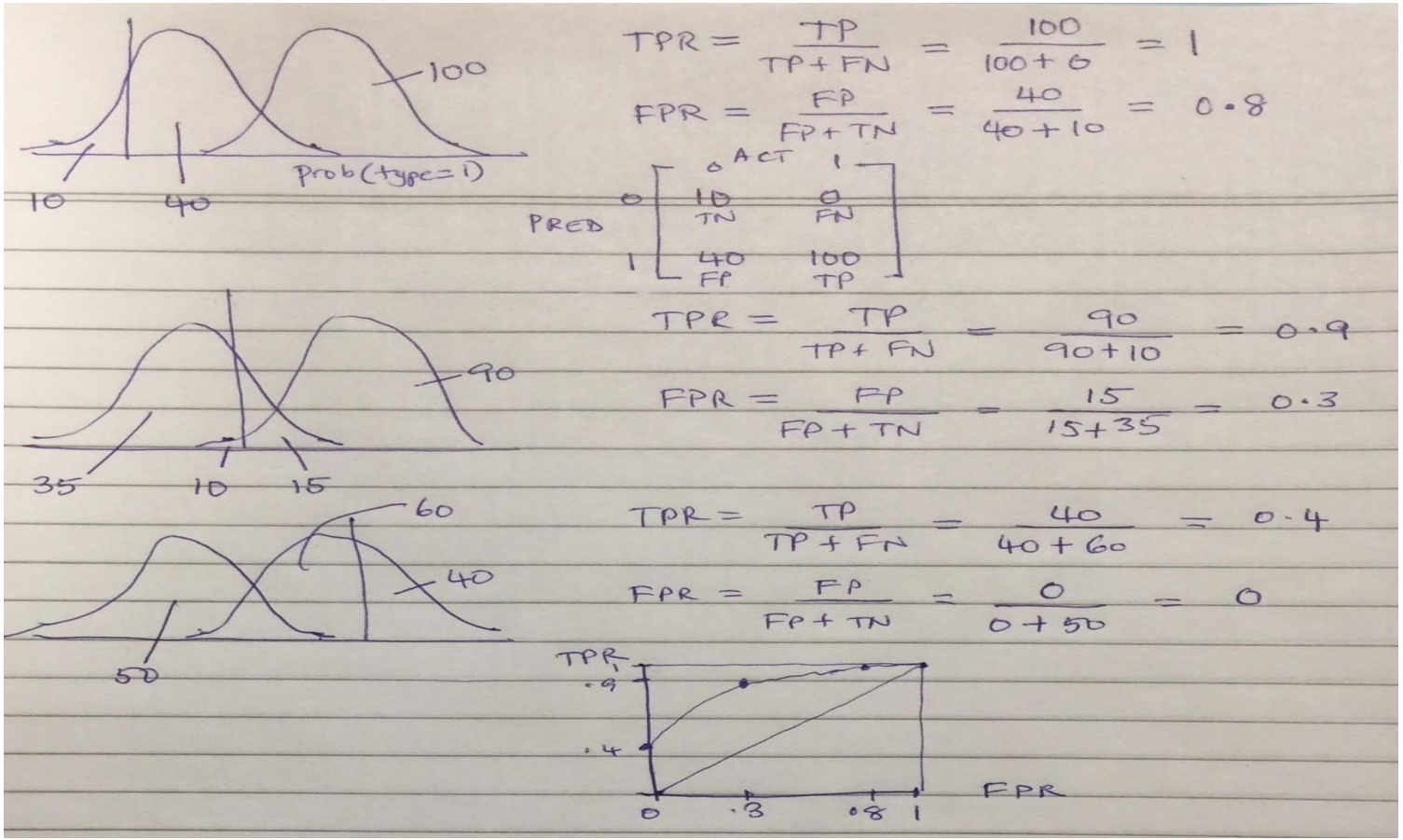
```
[[43753  3435]
 [ 8168 11739]]
             precision    recall  f1-score   support

          0       0.84      0.93      0.88     47188
          1       0.77      0.59      0.67     19907

avg / total       0.82      0.83      0.82     67095
```

http://srdas.github.io/Presentations/ClassClust/LogisticRegression.slides.html#/17

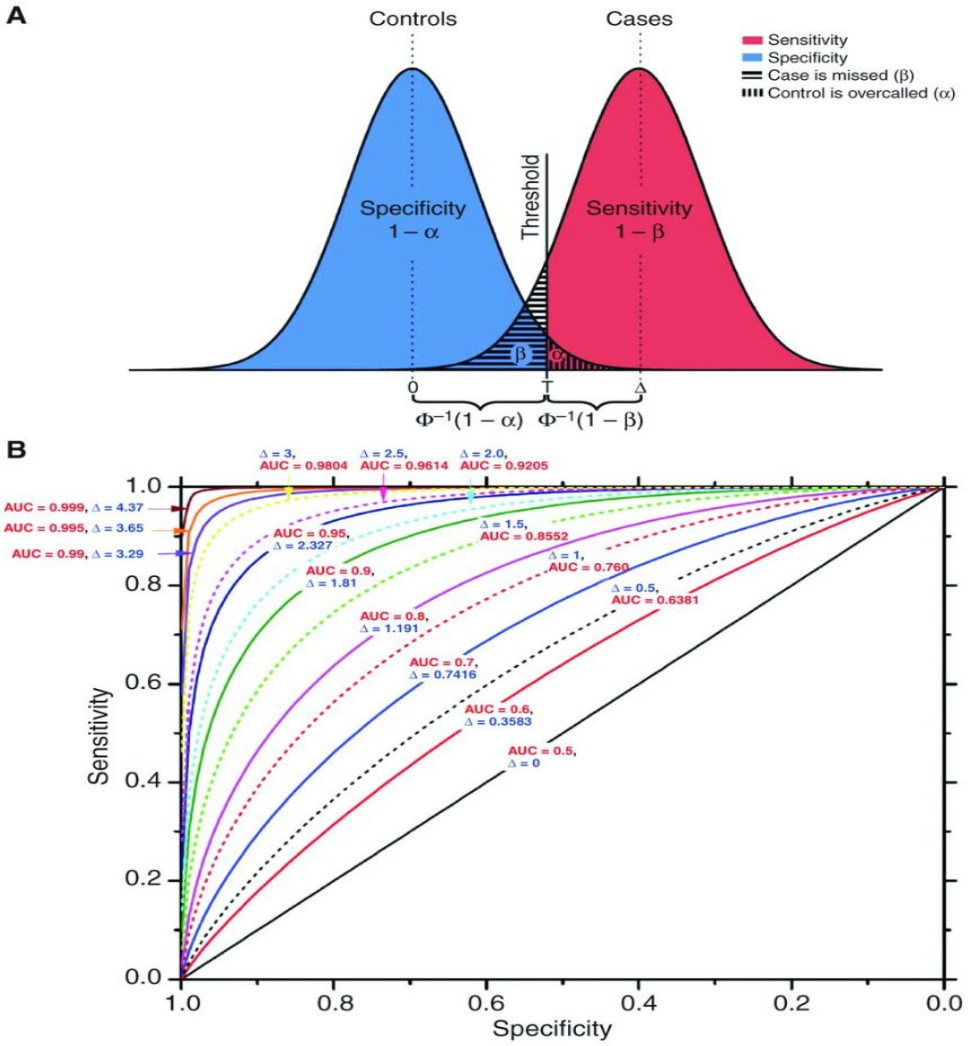
		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	F_1 score $= \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

ROC Curve


100
Prob(type=1)
10
40
90
35
10
15
60
40
50
TPR = TP / (TP + FN) = 100 / (100 + 0) = 1
FPR = FP / (FP + TN) = 40 / (40 + 10) = 0.8
ACT
PRED
0
1
10 TN
0 FN
40 FP
100 TP
TPR = TP / (TP + FN) = 90 / (90 + 10) = 0.9
FPR = FP / (FP + TN) = 15 / (15 + 35) = 0.3
TPR = TP / (TP + FN) = 40 / (40 + 60) = 0.4
FPR = FP / (FP + TN) = 0 / (0 + 50) = 0
TPR
.9
.4
0
.3
.8
1
FPR

ROC and AUC


A
Controls
Cases
Sensitivity
Specificity
Case is missed (β)
Control is overcalled (α)
Threshold
Specificity
1 − α
Sensitivity
1 − β
β
α
0
T
Δ
Φ⁻¹(1 − α)
Φ⁻¹(1 − β)
B
Δ = 3,
AUC = 0.9804
Δ = 2.5,
AUC = 0.9614
Δ = 2.0,
AUC = 0.9205
AUC = 0.999, Δ = 4.37
AUC = 0.995, Δ = 3.65
AUC = 0.99, Δ = 3.29
AUC = 0.95,
Δ = 2.327
Δ = 1.5,
AUC = 0.8552
AUC = 0.9,
Δ = 1.81
Δ = 1,
AUC = 0.760
Δ = 0.5,
AUC = 0.6381
AUC = 0.8,
Δ = 1.191
AUC = 0.7,
Δ = 0.7416
AUC = 0.6,
Δ = 0.3583
AUC = 0.5,
Δ = 0
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
Specificity
1.0
0.8
0.6
0.4
0.2
0.0

Multinomial Logit

The probability of each class $(0, 1, \ldots, k)$ for $(k + 1)$ classes is as follows:

$$Pr[y = j] = \frac{e^{a_j^\top x}}{\sum_{i=1}^{k} e^{a_i^\top x}}$$

and

$$Pr[y = 0] = \frac{1}{\sum_{i=1}^{k} e^{a_i^\top x}}$$

Note that $\sum_{i=1}^{k} Pr[y = i] = 1$.

http://srdas.github.io/Presentations/ClassClust/LogisticRegression.slides.html#/28

Discriminant Analysis

Kaggle credit card fraud dataset

Quick Class counts

```
data[["Class","V1"]].groupby(["Class"]).count()
```

	V1
Class	
0	284315
1	492

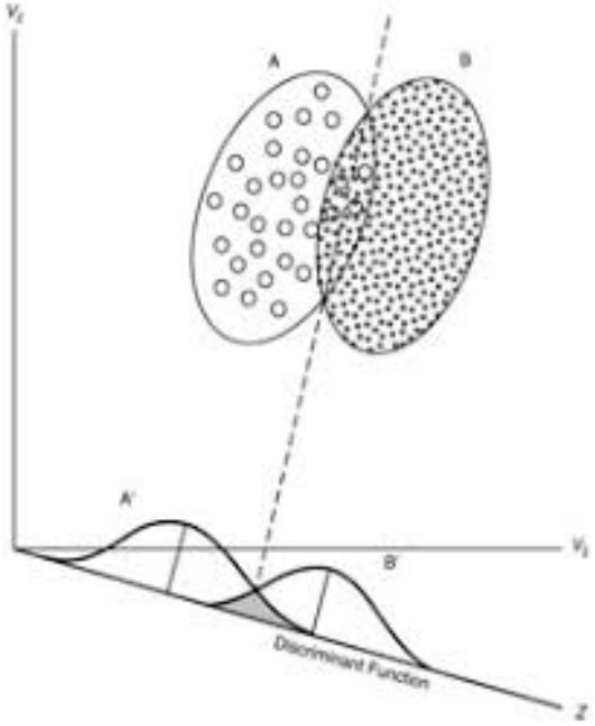
Mean Amount in Each class

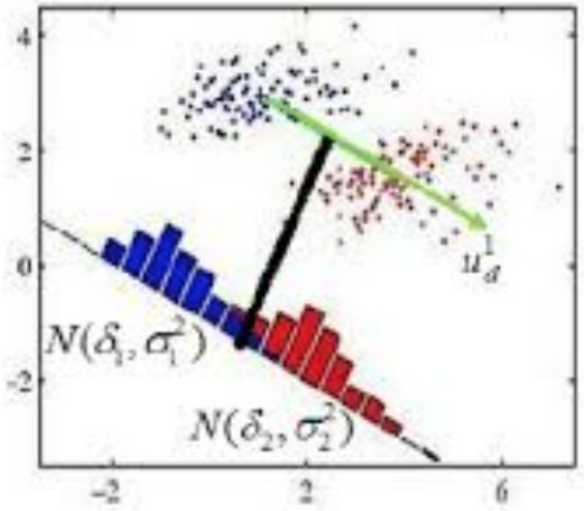
```
data[["Class","Amount"]].groupby(["Class"]).mean()
```

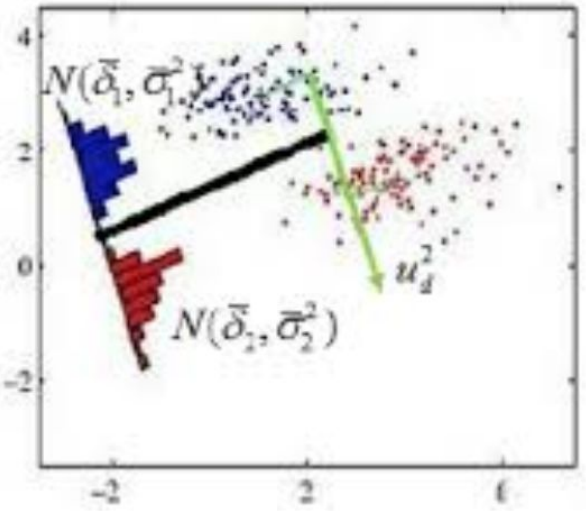
	Amount
Class	
0	88.291022
1	122.211321

http://srdas.github.io/Presentations/ClassClust/Discriminant_Analysis.slides.html#/

Linear Discriminant Analysis







http://srdas.github.io/MLBook/DiscriminantFactorAnalysis.html#discriminant-analysis

NCAA Dataset

http://srdas.github.io/Presentations/ClassClust/Discriminant_Analysis.slides.html#/13

```
ncaa = pd.read_table("data/ncaa.txt")
yy = append(list(ones(32)), list(zeros(32)))
ncaa["y"] = yy
ncaa.head()
```

	No NAME	GMS	PTS	REB	AST	TO	A/T	STL	BLK	PF	
0	1. NorthCarolina	6	84.2	41.5	17.8	12.8	1.39	6.7	3.8	16.7	0.5
1	2. Illinois	6	74.5	34.0	19.0	10.2	1.87	8.0	1.7	16.5	0.4
2	3. Louisville	5	77.4	35.4	13.6	11.0	1.24	5.4	4.2	16.6	0.4
3	4. MichiganState	5	80.8	37.8	13.0	12.6	1.03	8.4	2.4	19.8	0.4
4	5. Arizona	4	79.8	35.0	15.8	14.5	1.09	6.0	6.5	13.3	0.5

Feature Set

```
#CREATE FEATURES
y = ncaa['y']
X = ncaa.iloc[:,2:13]
X.head()
```

	PTS	REB	AST	TO	A/T	STL	BLK	PF	FG	FT	3P
0	84.2	41.5	17.8	12.8	1.39	6.7	3.8	16.7	0.514	0.664	0.417
1	74.5	34.0	19.0	10.2	1.87	8.0	1.7	16.5	0.457	0.753	0.361
2	77.4	35.4	13.6	11.0	1.24	5.4	4.2	16.6	0.479	0.702	0.376
3	80.8	37.8	13.0	12.6	1.03	8.4	2.4	19.8	0.445	0.783	0.329
4	79.8	35.0	15.8	14.5	1.09	6.0	6.5	13.3	0.542	0.759	0.397

Naive Bayes Classifier

Classification based on the class with the highest posterior probability:

$$Pr[C_j|x_1, \ldots, x_n] = \frac{Pr[x_1, \ldots, x_n|C_j] \cdot Pr[C_j]}{\sum_i Pr[x_1, \ldots, x_n|C_i] \cdot Pr[C_i]}$$

and

$$Pr[x_1, \ldots, x_n|C_j] = f[x_1|C_j] \cdot f[x_2|C_j] \cdots f[x_n|C_j]$$

where the last equation encapsulates "naivety", i.e., $x_1, \ldots, x_n$ are independent and Gaussian with density function $f(x) \sim N(\mu_x, \sigma_x^2)$, computed from the raw data.

http://srdas.github.io/Presentations/ClassClust/Naive_Bayes.slides.html#/

Support Vector Machines

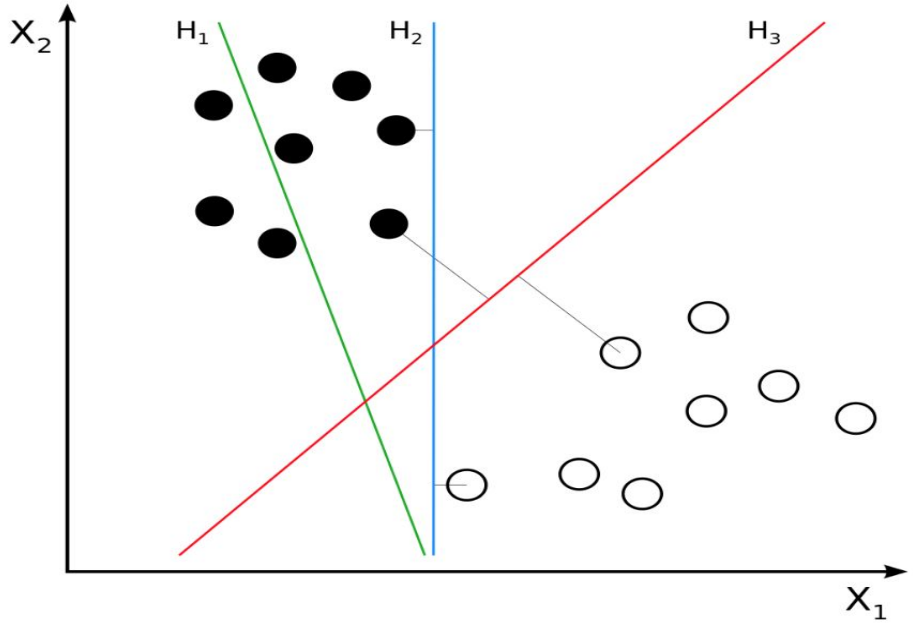
What is a SVM?

The goal of the SVM is to map a set of entities with inputs $X = \{x_1, x_2, \ldots, x_n\}$ of dimension n, i.e., $X \in R^n$, into a set of categories $Y = \{y_1, y_2, \ldots, y_m\}$ of dimension m, such that the n-dimensional X-space is divided using hyperplanes, which result in the maximal separation between classes Y. A hyperplane is the set of points $\mathbf{x}$ satisfying the equation

$$\mathbf{w} \cdot \mathbf{x} = b$$

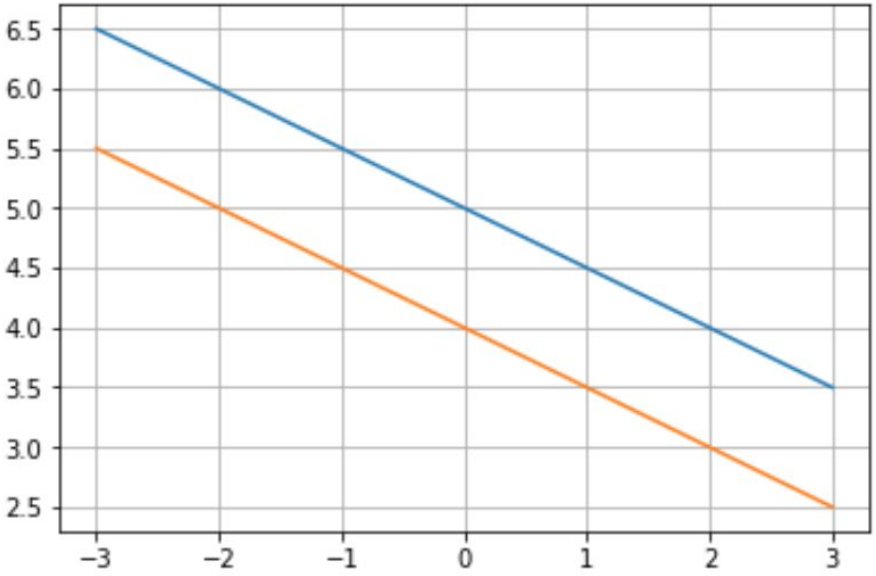
where b is a scalar constant, and $\mathbf{w} \in R^n$ is the normal vector to the hyperplane, i.e., the vector at right angles to the plane. The distance between this hyperplane and $\mathbf{w} \cdot \mathbf{x} = 0$ is given by $b/||\mathbf{w}||$, where $||\mathbf{w}||$ is the norm of vector $\mathbf{w}$.

http://srdas.github.io/Presentations/ClassClust/SVM.slides.html#/



H_3 is the best separating hyperplane.

- Suppose we have two categories of data, i.e., $y = \{y_1, y_2\}$.
- Assume that all points in category y_1 lie above a hyperplane $\mathbf{w} \cdot \mathbf{x} = b_1$, and all points in category y_2 lie below a hyperplane $\mathbf{w} \cdot \mathbf{x} = b_2$.
- Then the distance between the two hyperplanes is $\frac{|b_1 - b_2|}{||\mathbf{w}||}$.




```
#Example of hyperplane geometry
w1 = 1; w2 = 2
b1 = 10
#Plot hyperplane in x1, x2 space
x1 = linspace(-3,3,100)
x2 = (b1-w1*x1)/w2
plot(x1,x2)
#Create hyperplane 2
b2 = 8
x2 = (b2-w1*x1)/w2
plot(x1,x2)
grid()
#Compute distance to hyperplane 2
print('Distance between two hyperplanes =',abs(b1-b2)/sqrt(w1**2+w2**2))
```

```
Distance between two hyperplanes = 0.89442719099999159
```

Regularization

- Of course, there may be no linear hyperplane that perfectly separates the two groups. Hence, L2 regularization.

$$\min_{b_1, b_2, \mathbf{w}, \{\eta_i\}} \frac{1}{2}||\mathbf{w}||^2 + C_1 \sum_{i=1}^{n} \eta_i + C_2 \sum_{i=1}^{n} \eta_i$$

- where C_1, C_2 are the costs for slippage in groups 1 and 2, respectively.
- Often implementations assume $C_1 = C_2$.
- The values η_i are positive for observations that are not perfectly separated, i.e., lead to slippage.

K Nearest Neighbors

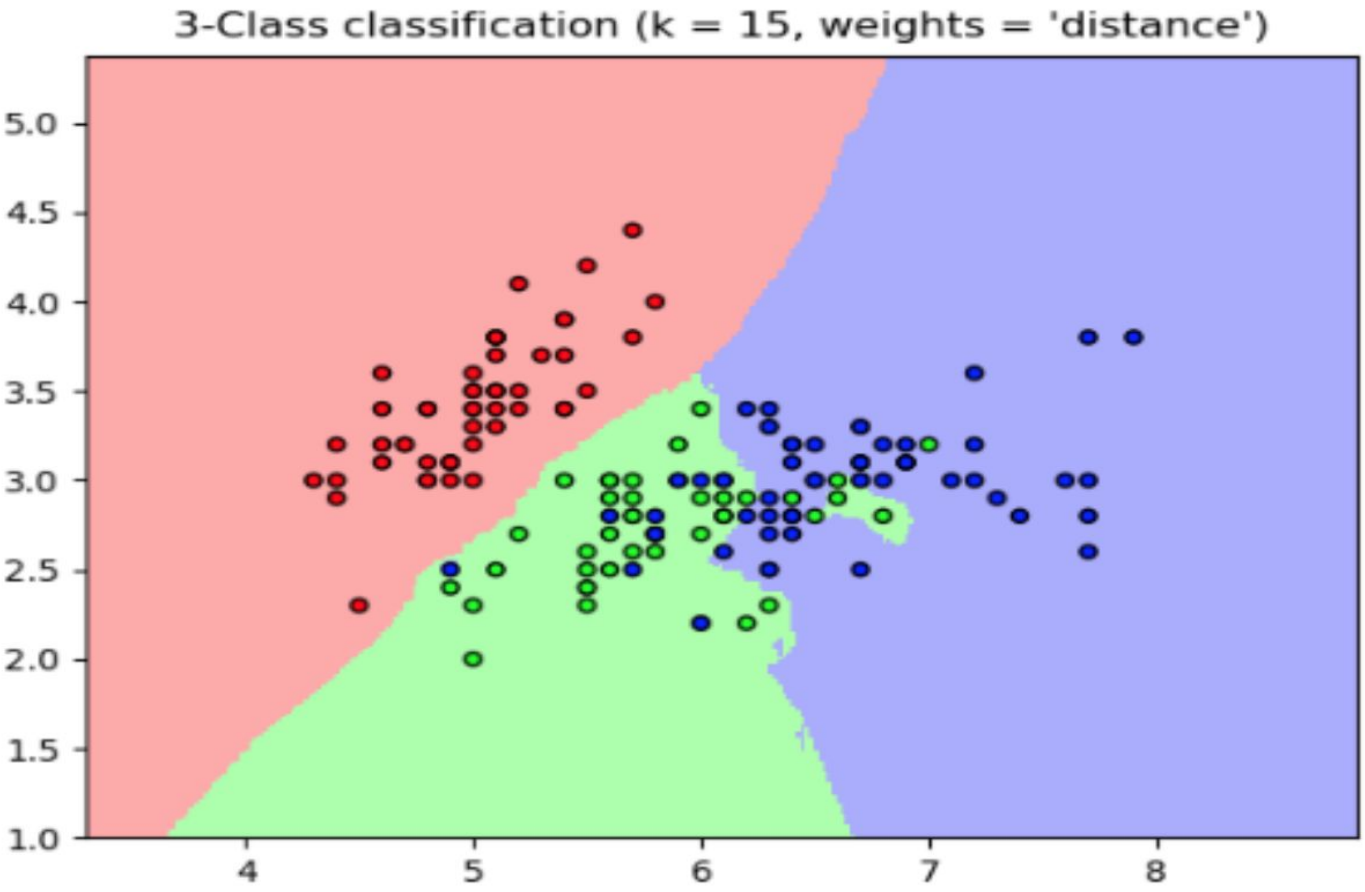
- This is one of the simplest algorithms for classification and grouping.
- Simply define a distance metric over a set of observations, each with M characteristics, i.e., $x_1, x_2, \ldots, x_M$.
- Compute the pairwise distance between each pair of observations, using any of the standard metrics. For example, Euclidian distance between data x and y:

$$d = \sqrt{\sum_{i=1}^{M} (x_i - y_i)^2}$$

- Next, fix k, the number of nearest neighbors in the population to be considered.
- Finally, assign the category based on which one has the majority of nearest neighbors to the case we are trying to classify.

http://srdas.github.io/Presentations/ClassClust/kNN.slides.html#/

Classification



Decision Trees

- A natural outcome of recursive partitioning of the data.
- CART, which stands for classification analysis and regression trees.
- Prediction trees are of two types: (a) Classification trees, where the leaves of the trees are different categories of discrete outcomes. and (b) Regression trees, where the leaves are continuous outcomes.
- We may think of the former as a generalized form of limited dependent variables, and the latter as a generalized form of regression analysis.

http://srdas.github.io/Presentations/ClassClust/Decision_Trees.slides.html#/

Recursive Partitioning

- Bifurcate the data into two categories such that the additional information from categorization results in better **information** than before the binary split.
- Raw frequency p of how many people made donations, i.e., and number between 0 and 1. The **information** of the predicted likelihood p is inversely related to the sum of squared errors (SSE) between this value p and the $x_i = 0, 1$ values of the observations.

$$SSE_1 = \sum_{i=1}^{n} (x_i - p)^2$$

- Second bifurcation:

$$SSE_2 = \sum_{i,Income<K} (x_i - p_L)^2 + \sum_{i,Income \geq K} (x_i - p_R)^2$$

- By choosing K correctly, our recursive partitioning algorithm will maximize the gain, i.e., $\delta = (SSE_1 - SSE_2)$. We stop branching further when at a given tree level δ is less than a pre-specified threshold.

C4.5 Classifier

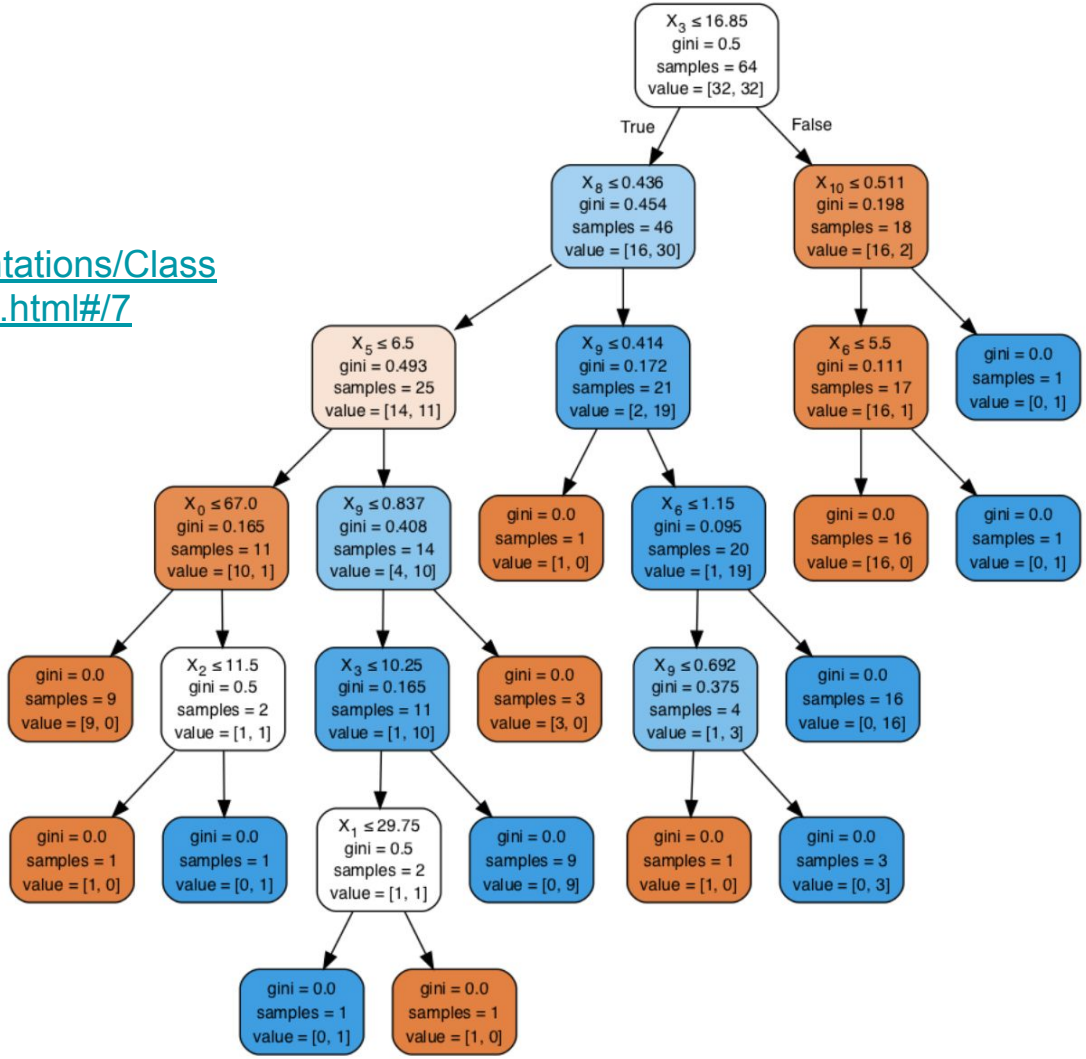
Recursive partitioning as in the previous case, but instead of minimizing the sum of squared errors between the sample data x and the true value p at each level, here the goal is to minimize entropy. This improves the information gain. Natural entropy (H) of the data x is defined as

$$H = -\sum_{x} f(x) \cdot ln\, f(x)$$

where $f(x)$ is the probability density of x. This is intuitive because after the optimal split in recursing down the tree, the distribution of x becomes narrower, lowering entropy. This measure is also often known as "differential entropy."

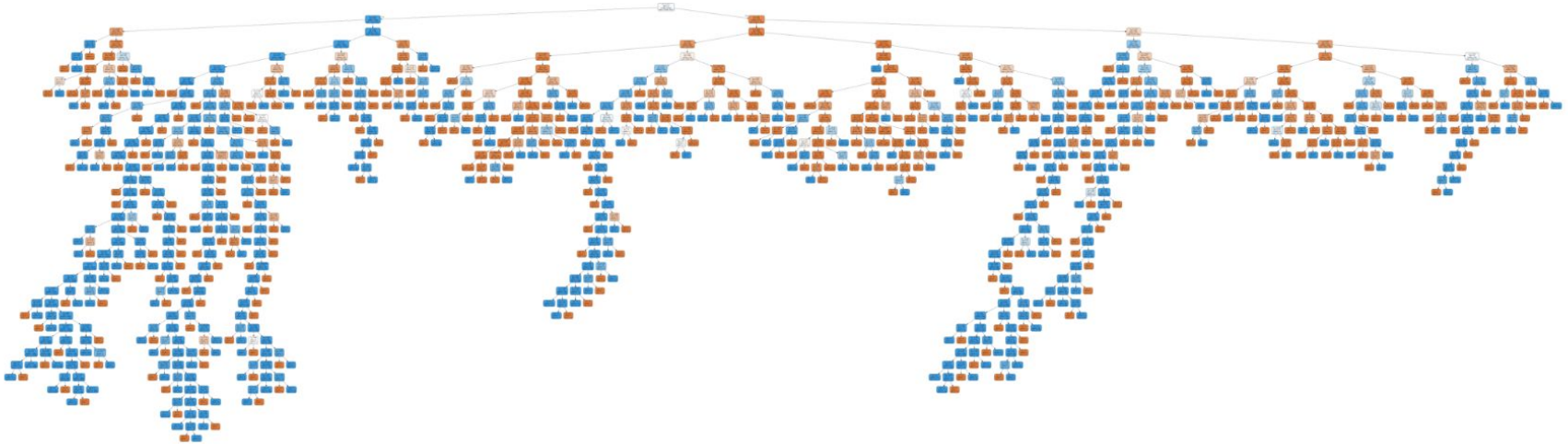
NCAA Tree

http://srdas.github.io/Presentations/ClassClust/Decision_Trees.slides.html#/7



Credit Card Dataset

```
Image(graph.create_png())
```



http://srdas.github.io/Presentations/ClassClust/Decision_Trees.slides.html#/15

Random Forest Classifier

```
%pylab inline
import pandas as pd
from sklearn.model_selection import train_test_split
from imblearn.combine import SMOTEENN
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_curve,auc
from sklearn.metrics import confusion_matrix
```

```
Populating the interactive namespace from numpy and matplotlib
```

http://srdas.github.io/Presentations/ClassClust/RandomForest_CC_Fraud.slides.html#/

Rebalance sample with under- or over-sampling

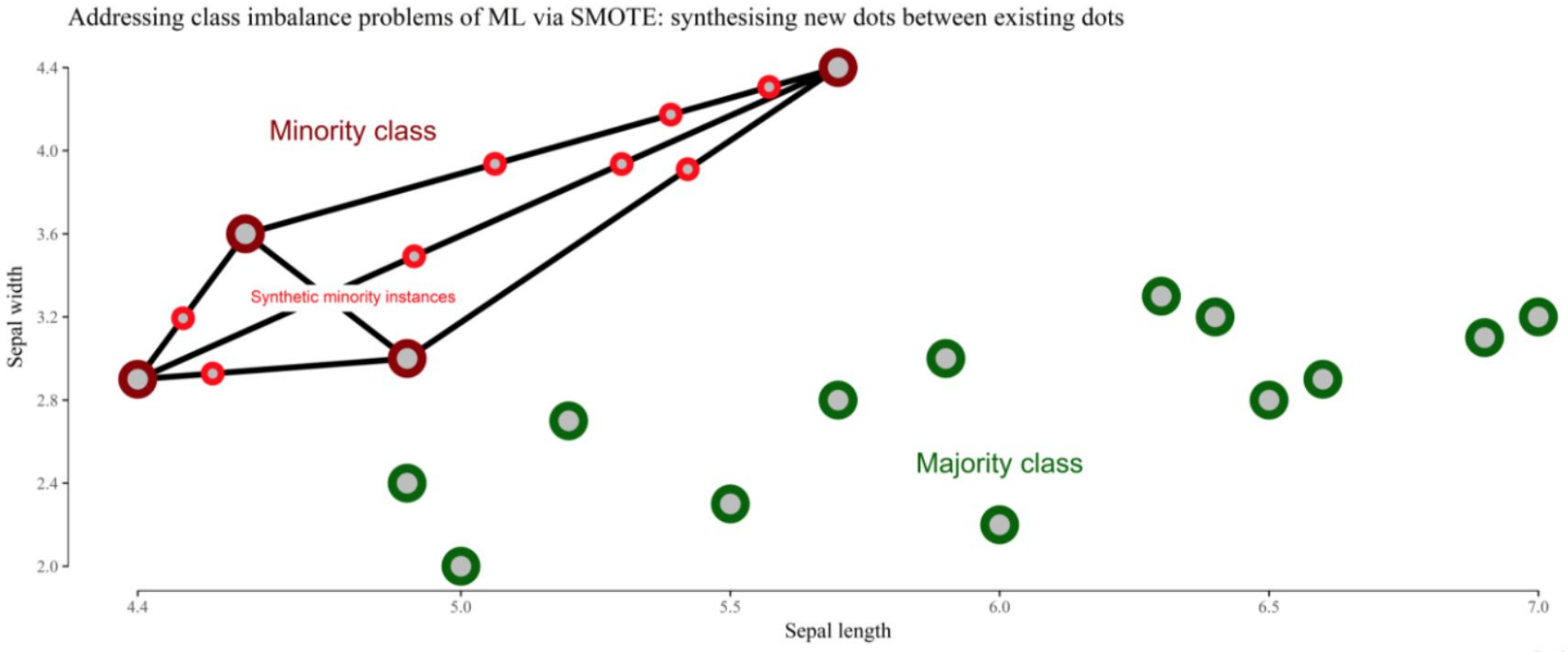
While a Random Forest classifier is generally considered imbalance-agnostic, in this case the severity of the imbalance resuts in overfitting to the majority class.

The Synthetic Minority Over-sampling Technique (SMOTE) is one of the most well-known methods to cope with it and to balance the different number of examples of each class.

The basic idea is to oversample the minority class, while trying to get the most variegated samples from the majority class.

http://srdas.github.io/Presentations/ClassClust/RandomForest_CC_Fraud.slides.html#/7

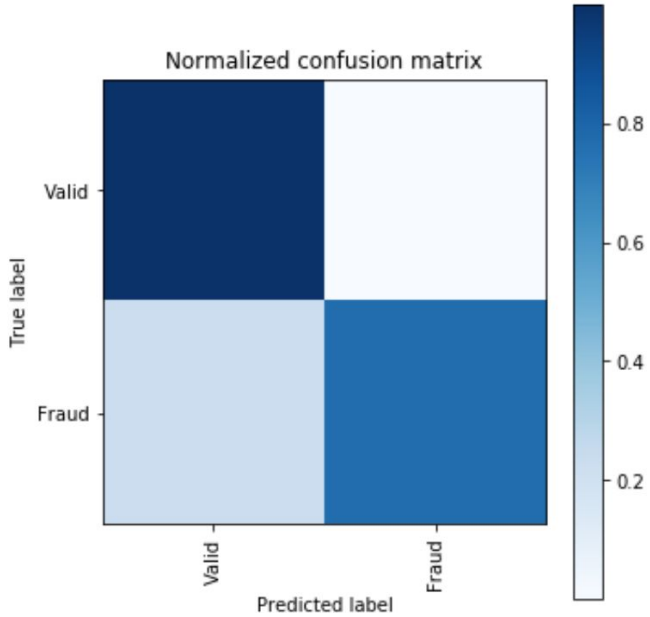
SMOTE



http://rikunert.com/SMOTE_explained

Confusion Matrix

```
cm = confusion_matrix(y_test, y_test_hat)
cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(5,5))
plot_confusion_matrix(cm_normalized, title='Normalized confusion matrix')
```



http://srdas.github.io/Presentations/ClassClust/RandomForest_CC_Fraud.slides.html#/21

Dimension Reduction

A Matrix Reduction

Suppose we reduce the $k = 11$ dimensional feature space X to reduced factor space R with $k = 3$. We translate with a matrix L.

$$R = X \cdot L$$

where F is (64×3), X is (64×11), and L is (11×3).

Where does matrix L come from?

- From Principal Components Analysis.
- Based on an Eigenvalue Decomposition of the covariance matrix of the features, i.e., $C = Cov(X)$, which is size (11×11).
- Decomposition is based on solving the following equation:

$$\lambda l = C \cdot l$$

- There are 11 solutions l and the first 3 will form the matrix L.

http://srdas.github.io/Presentations/ClassClust/Dimension_Reduction.slides.html#/

Principal Components Analysis (PCA)

```
#REDUCED DATA
from sklearn import decomposition
pca = decomposition.PCA(n_components=3)
pca.fit(Xs)
R = pca.transform(Xs)
print(R.shape)
```

```
(64, 3)
```

http://srdas.github.io/Presentations/ClassClust/Dimension_Reduction.slides.html#/7

```
#LOADINGS MATRIX L
L = pca.components_.T
print(L.shape)
print(X.columns)
L
```

```
(11, 3)
Index(['PTS  ', 'REB  ', 'AST  ', 'TO  ', 'A/T  ', 'STL  ', 'BLK  ', 'PF  ',
       'FG  ', 'FT  ', '3P'],
      dtype='object')

array([[-0.43884425,  0.02285078, -0.18631231],
       [ 0.1867903 , -0.43294301, -0.21835274],
       [-0.47238137,  0.04962787, -0.18252437],
       [ 0.17651088, -0.02325077, -0.68627945],
       [-0.45266018,  0.08602947,  0.37681287],
       [ 0.03888779,  0.57289362, -0.29086594],
       [ 0.02703794,  0.08087251, -0.16285993],
       [ 0.05607815,  0.51652087, -0.12761847],
       [-0.44993945, -0.18657986, -0.21900567],
       [ 0.03599791,  0.40620447,  0.17479897],
       [-0.32279124, -0.01667957, -0.25572689]])
```

```
#CHECK THAT DECOMPOSITION IS CORRECT
sum(R - Xs.dot(L))
```

```
0   -1.893624e-14
1    3.152340e-14
2    1.458902e-15
dtype: float64
```

Treasury Rates Dataset

```
rates = pd.read_table("data/tryrates.txt")
print(rates.shape)
rates.head()
```

```
(367, 9)
```

	DATE	FYGM3	FYGM6	FYGT1	FYGT2	FYGT3	FYGT5	FYGT7	FYGT10
0	Jun-76	5.41	5.77	6.52	7.06	7.31	7.61	7.75	7.86
1	Jul-76	5.23	5.53	6.20	6.85	7.12	7.49	7.70	7.83
2	Aug-76	5.14	5.40	6.00	6.63	6.86	7.31	7.58	7.77
3	Sep-76	5.08	5.30	5.84	6.42	6.66	7.13	7.41	7.59
4	Oct-76	4.92	5.06	5.50	5.98	6.24	6.75	7.16	7.41

http://srdas.github.io/Presentations/ClassClust/Dimension_Reduction.slides.html#/11

```
X.corr()
```

	FYGM3	FYGM6	FYGT1	FYGT2	FYGT3	FYGT5	FYGT7	FYGT10
FYGM3	1.000000	0.997537	0.991125	0.975089	0.961225	0.938329	0.922041	0.906564
FYGM6	0.997537	1.000000	0.997350	0.985125	0.972844	0.951266	0.935603	0.920542
FYGT1	0.991125	0.997350	1.000000	0.993696	0.984692	0.966859	0.953130	0.939686
FYGT2	0.975089	0.985125	0.993696	1.000000	0.997767	0.987892	0.978651	0.968093
FYGT3	0.961225	0.972844	0.984692	0.997767	1.000000	0.995622	0.989403	0.981307
FYGT5	0.938329	0.951266	0.966859	0.987892	0.995622	1.000000	0.998435	0.994569
FYGT7	0.922041	0.935603	0.953130	0.978651	0.989403	0.998435	1.000000	0.998493
FYGT10	0.906564	0.920542	0.939686	0.968093	0.981307	0.994569	0.998493	1.000000

```
#PCA
X = rates.drop("DATE",axis=1)
pca = decomposition.PCA(n_components=2)
pca.fit(X)
Y = pca.transform(X)
Y.shape
```

```
(367, 2)
```

Clustering

Overview

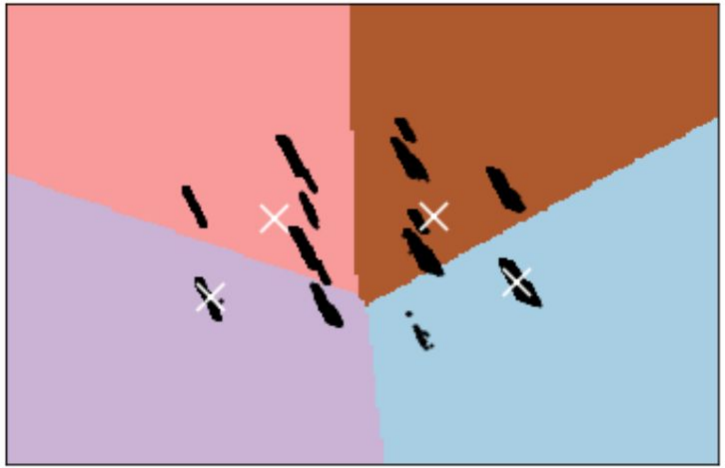
- Grouping individuals, firms, projects, etc.
- Cluster analysis comprises a group of techniques that uses distance metrics to bunch data into categories.
- Two approaches.

1. **Partitioning or Top-down**: In this approach, the entire set of n entities is assumed to be divided into k clusters. Then entities are assigned clusters.
2. **Agglomerative or Hierarchical or Bottom-up**: In this case we begin with all entities in the analysis being given their own cluster, so that we start with n clusters. Then, entities are grouped into clusters based on a given distance metric between each pair of entities. In this way a hierarchy of clusters is built up and the researcher can choose which grouping is preferred.

http://srdas.github.io/Presentations/ClassClust/Clustering.slides.html#/

K-means

1. Form a distance matrix.
2. Initialize cluster centroids with evenly spaced items.
3. Assign each observation to closest cluster (to centroid, closest or farthest mamber).
4. Repeat a few times to re-assign until the scheme stabilizes.



http://srdas.github.io/Presentations/ClassClust/Clustering.slides.html#/5

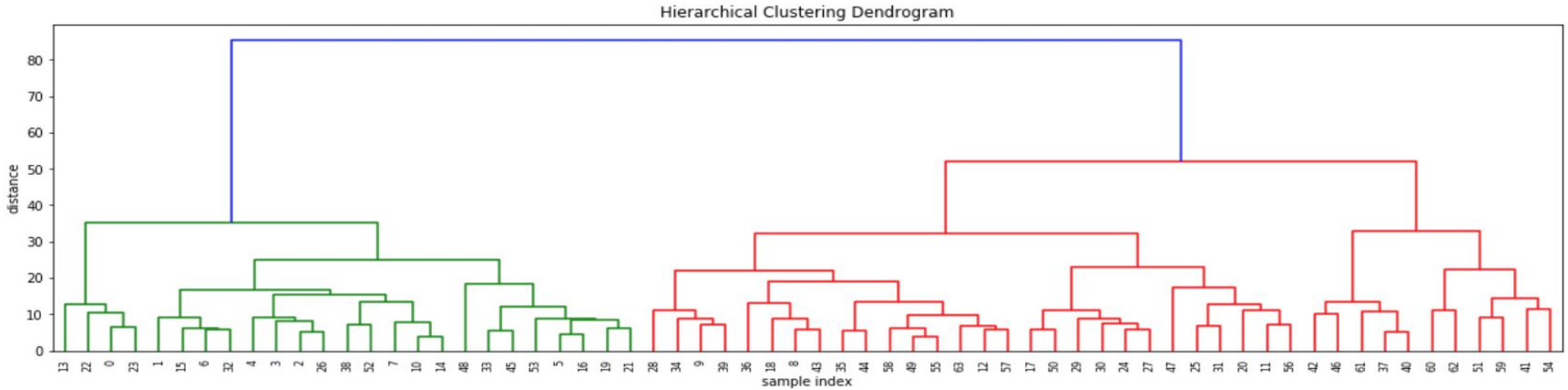
Hierarchical Clustering

1. Get distance matrix for n observations. Each in its own cluster.
2. Club the two closest observations into a cluster. Now we have $(n-1)$ clusters.
3. Recalculate centroids.
4. Repeat to get hierarchical structure.

http://srdas.github.io/Presentations/ClassClust/Clustering.slides.html#/13

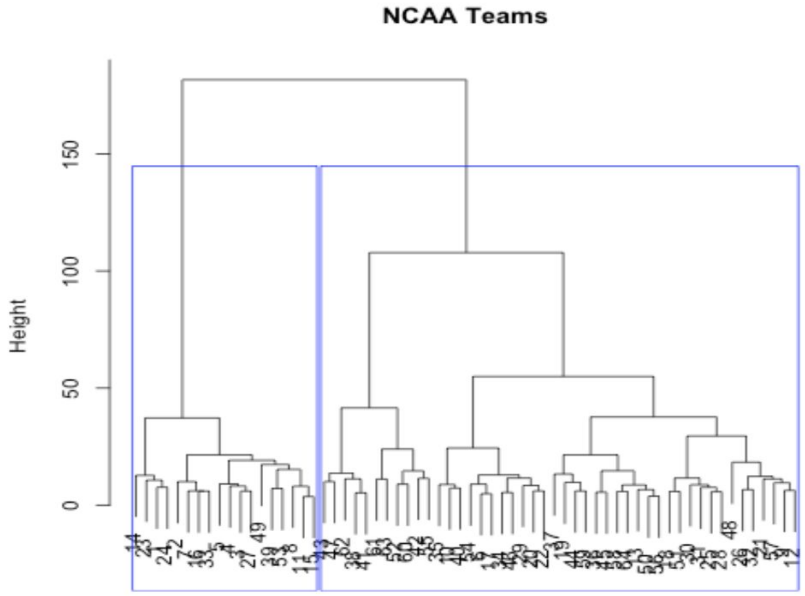
Dendrogram

```
#DENDROGRAM
figure(figsize=(20, 5))
title('Hierarchical Clustering Dendrogram')
xlabel('sample index')
ylabel('distance')
dendrogram(Z,
    leaf_rotation=90.,  # rotates the x axis labels
    leaf_font_size=8.,  # font size for the x axis labels
)
show()
```

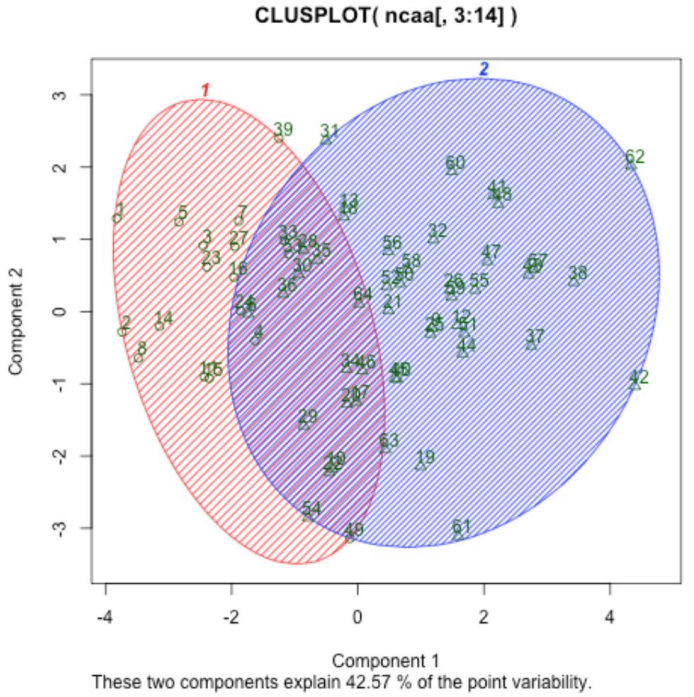


Hierarchical Clustering in R

```
%%R
#CLUSTER
ncaa = read.table("data/ncaa.txt",header=TRUE)
d = dist(ncaa[,3:14], method="euclidian")
fit = hclust(d, method="ward.D")
plot(fit,main="NCAA Teams")
groups = cutree(fit, k=2)
rect.hclust(fit, k=2, border="blue")
```




```
%%R
#CLUSTER PLOT
library(cluster)
clusplot(ncaa[,3:14],groups,color=TRUE,shade=TRUE,
```



http://srdas.github.io/Presentations/ClassClust/Clustering.slides.html#/19

Deep Learning with Neural Networks

Deep Learning is Pattern Recognition

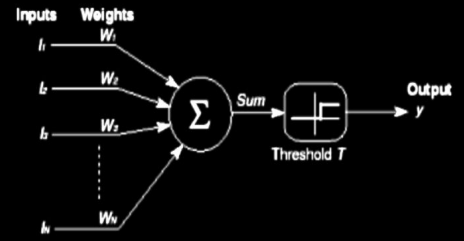


Figure 1.1: McCullough Pitts neuron

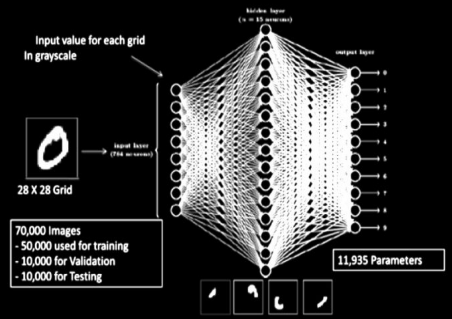


Figure 2.2: The NN used for Solving the MNIST Digit Recognition Problem

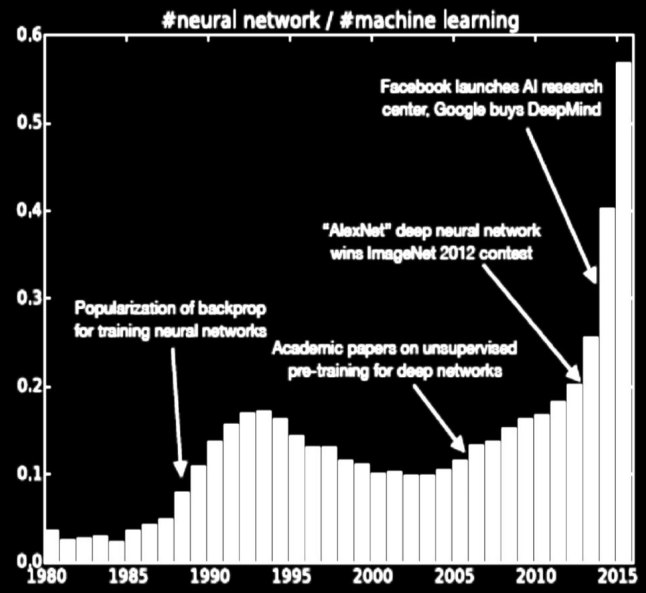
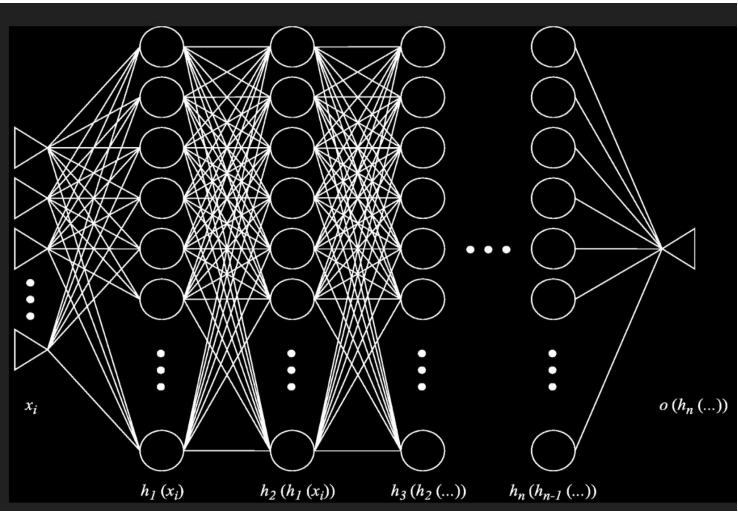


Figure 1.2: Number of NN research reports

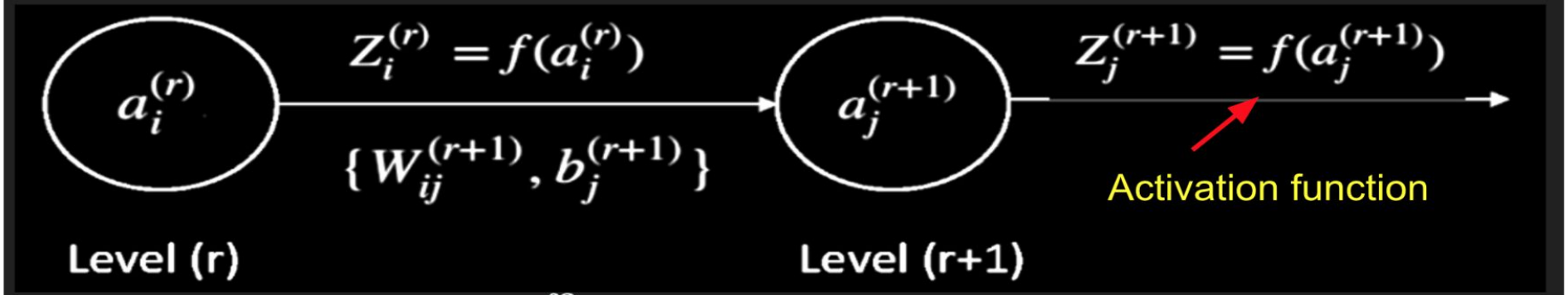
Reference Book:
http://srdas.github.io/DLBook



http://srdas.github.io/Presentations/ClassClust/DeepLearning_Introduction_Short.slides.html#/

Zooming In

Subset of the Net

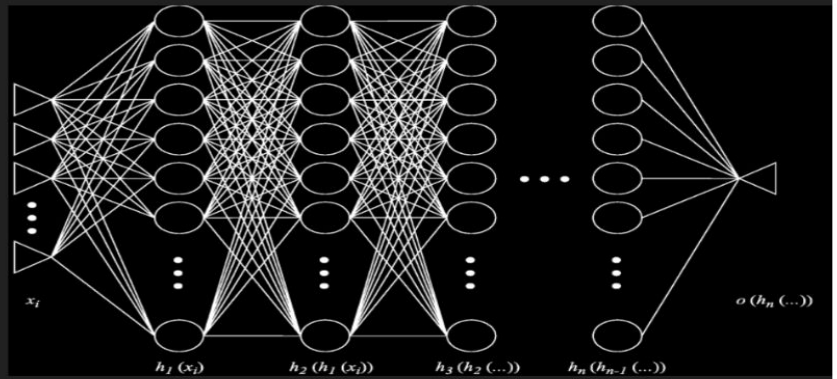


$$a_j = b_j + \sum_{I=1}^{n} w_{ij} z_i$$

Net input

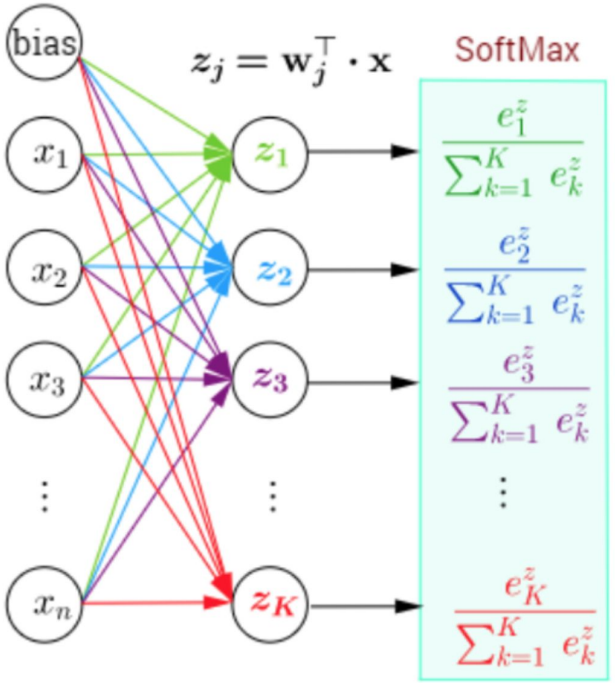
$$\sigma(a_j) = \frac{1}{1 + \exp(-a_j)}$$

Sigmoid function



Activation Functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	



Loss Function

Fitting the DLN is an exercise where the best weights $\{W, b\} = \{W_{ij}^{(r+1)}, b_j^{(r+1)}\}, \forall r$ for all layers are determined to minimize a loss function generally denoted as

$$\min_{W,b} \sum_{m=1}^{M} L_m[h(X^{(m)}), T^{(m)}]$$

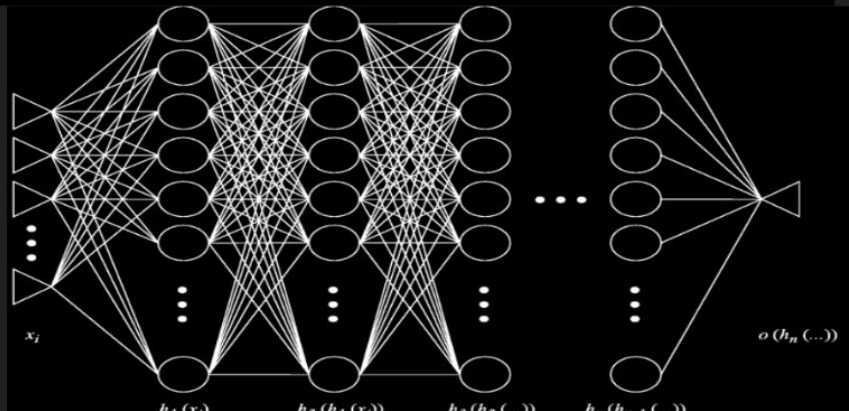
where M is the number of training observations (rows in the data set), $T^{(m)}$ is the true value of the output, and $h(X^{(m)})$ is the model output from the DLN. The loss function L_m quantifies the difference between the model output and the true output.

Cross Entropy

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)]$$

Quadratic Loss

$$C = \frac{(y - a)^2}{2}$$



Gradient Descent

Fitting the DLN requires getting the weights $\{W, b\}$ that minimize L_m. These are done using gradient descent, i.e.,

$$W_{ij}^{(r+1)} \leftarrow W_{ij}^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial W_{ij}^{(r+1)}}$$

$$b_j^{(r+1)} \leftarrow b_j^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial b_j^{(r+1)}}$$

Here η is the learning rate parameter. We iterate on these functions until the gradients become zero, and the weights discontinue changing with each update, also known as an **epoch**.

Total Gradient

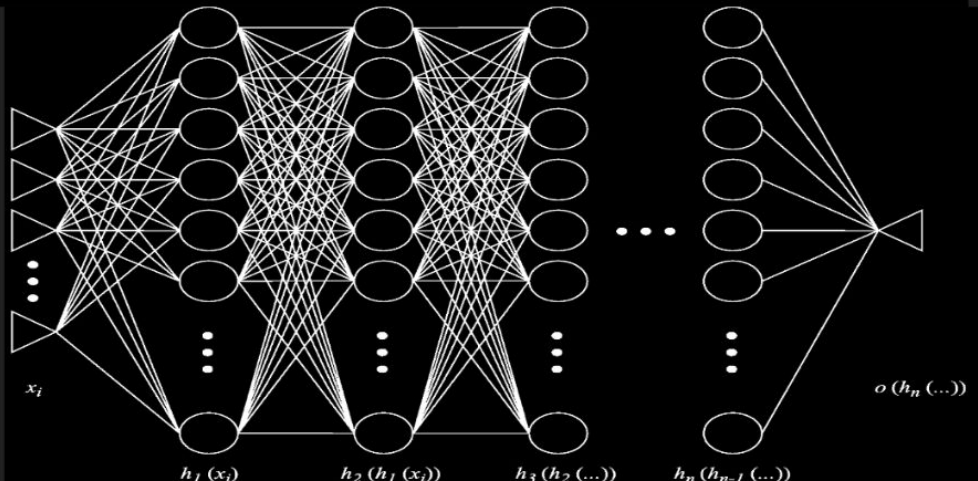
$$w_{ij}^{(r)} \leftarrow w_{ij}^{(r)} - \frac{\eta}{M} \sum_{m=1}^{M} \frac{\partial L_m(w, b; X(m), T(m))}{\partial w_{ij}^{(r)}}$$

$$b_i^{(r)} \leftarrow b_i^{(r)} - \frac{\eta}{M} \sum_{m=1}^{M} \frac{\partial L_m(w, b; X(m), T(m))}{\partial b_i^{(r)}}$$

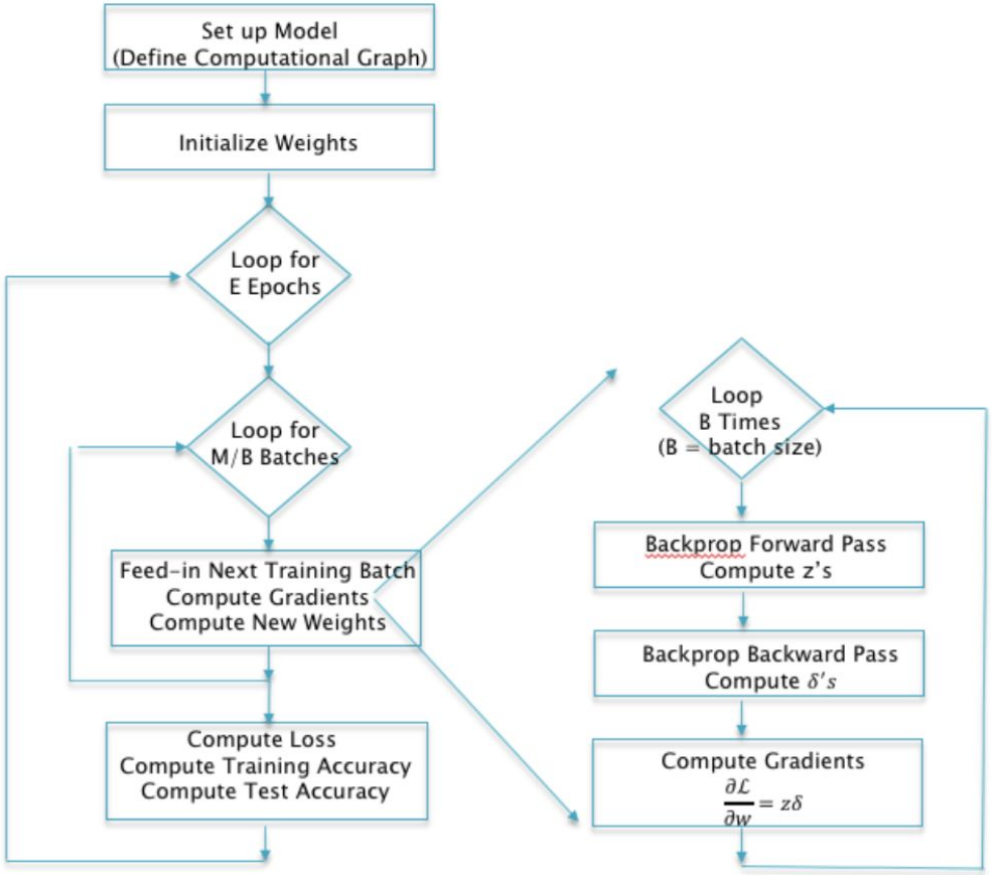
Stochastic Batch Gradient

$$w_{ij}^{(r)} \leftarrow w_{ij}^{(r)} - \frac{\eta}{s} \sum_{m=1}^{s} \frac{\partial L_m(w, b; X(m), T(m))}{\partial w_{ij}^{(r)}}$$

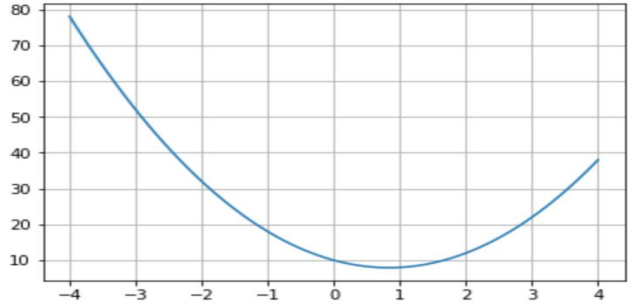
$$b_i^{(r)} \leftarrow b_i^{(r)} - \frac{\eta}{s} \sum_{m=1}^{s} \frac{\partial L_m(w, b; X(m), T(m))}{\partial b_i^{(r)}}$$



Batch Gradient Descent



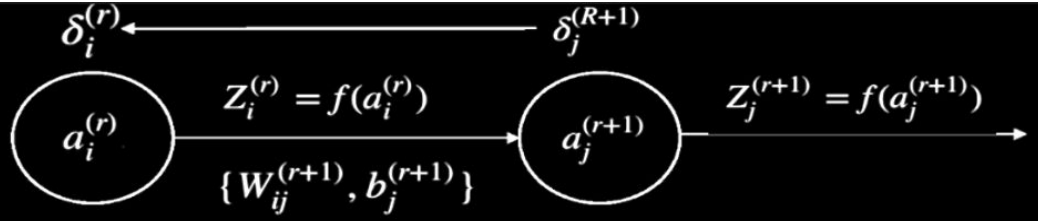

```
def f(x):
    return 3*x**2 -5*x + 10

x = linspace(-4,4,100)
plot(x,f(x))
grid()
```




```
dx = 0.001
eta = 0.05  #learning rate
x = -3
for j in range(20):
    df_dx = (f(x+dx)-f(x))/dx
    x = x - eta*df_dx
    print(x,f(x))
```

```
-1.8501500000001698 29.519915067502733
-1.0452550000002532 18.503949045077853
-0.4818285000003115 13.105618610239208
-0.08742995000019249 10.460081738472072
0.18864903499989083 9.163520200219716
0.3819043244999847 8.528031116715445
0.5171830271499616 8.216519714966186
0.6118781190049631 8.06379390252634
0.6781646833034642 7.988898596522943
0.7245652783124417 7.95215813604575
0.7570456948186948 7.934126078037087
0.7797819863730542 7.925269906950447
0.7956973904611502 7.920916059254301
0.8068381733227685 7.918772647178622
0.8146367213259147 7.917715356568335
0.8200957049281836 7.917192371084045
0.8239169934497053 7.916932669037079
0.8265918954147882 7.9168030076222955
0.8284643267903373 7.916737788340814
0.8297750287532573 7.916704651261121
```

Magic of Backpropagation



$$\min_{W,b} \sum_{m=1}^{M} L_m[h(X^{(m)}), T^{(m)}]$$

Loss Function

$$a_j^{(r+1)} = \sum_{i=1}^{n_r} W_{ij}^{(r+1)} Z_i^r + b_j^{(r+1)} = \sum_{i=1}^{n_r} W_{ij}^{(r+1)} f(a_i^{(r)}) + b_j^{(r+1)} \Rightarrow \frac{\partial a_j^{(r+1)}}{\partial a_i^{(r)}} = W_{ij}^{(r+1)} \cdot f'(a_i^{(r)})$$

Gradients

$$\frac{\partial L_m}{\partial W_{ij}^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial W_{ij}^{(r+1)}} = \delta_j^{(r+1)} \cdot Z_i^{(r)}$$

$$\frac{\partial L_m}{\partial b_j^{(r+1)}} = \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial b_j^{(r+1)}} = \delta_j^{(r+1)} \cdot 1 = \delta_j^{(r+1)}$$

$$\begin{aligned}
\delta_i^{(r)} &= \frac{\partial L_m}{\partial a_i^{(r)}} \\
&= \sum_{j=1}^{n_{r+1}} \frac{\partial L_m}{\partial a_j^{(r+1)}} \cdot \frac{\partial a_j^{(r+1)}}{\partial a_i^{(r)}} \\
&= \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)} \cdot f'(a_i^{(r)}) \\
&= f'(a_i^{(r)}) \cdot \sum_{j=1}^{n_{r+1}} \delta_j^{(r+1)} \cdot W_{ij}^{(r+1)}
\end{aligned}$$

Deltas

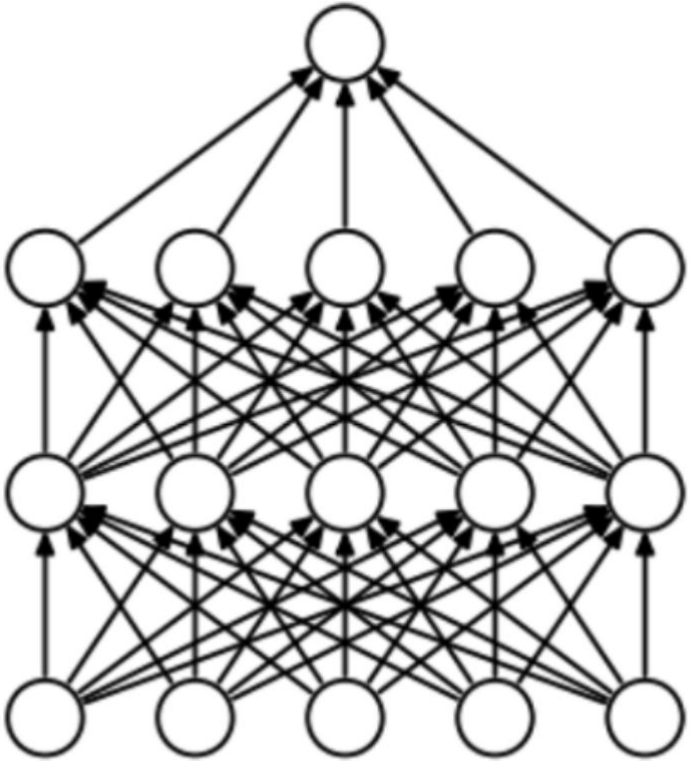
$$\delta_j^{(R+1)} = \frac{\partial L_m}{\partial a_j^{(R+1)}} = h'(a_j^{(R+1)})$$

Gradient Descent

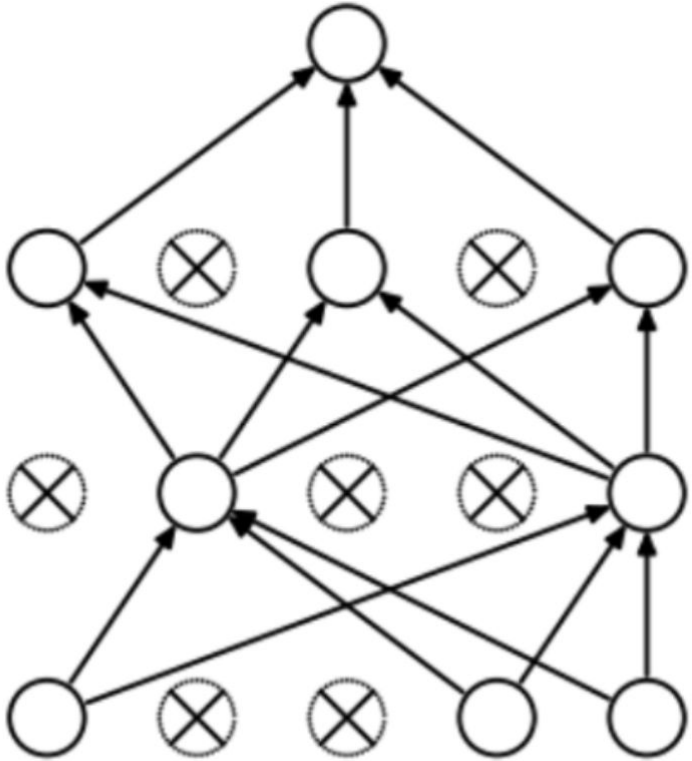
$$W_{ij}^{(r+1)} \leftarrow W_{ij}^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial W_{ij}^{(r+1)}}$$

$$b_j^{(r+1)} \leftarrow b_j^{(r+1)} - \eta \cdot \frac{\partial L_m}{\partial b_j^{(r+1)}}$$

Dropout Regularization



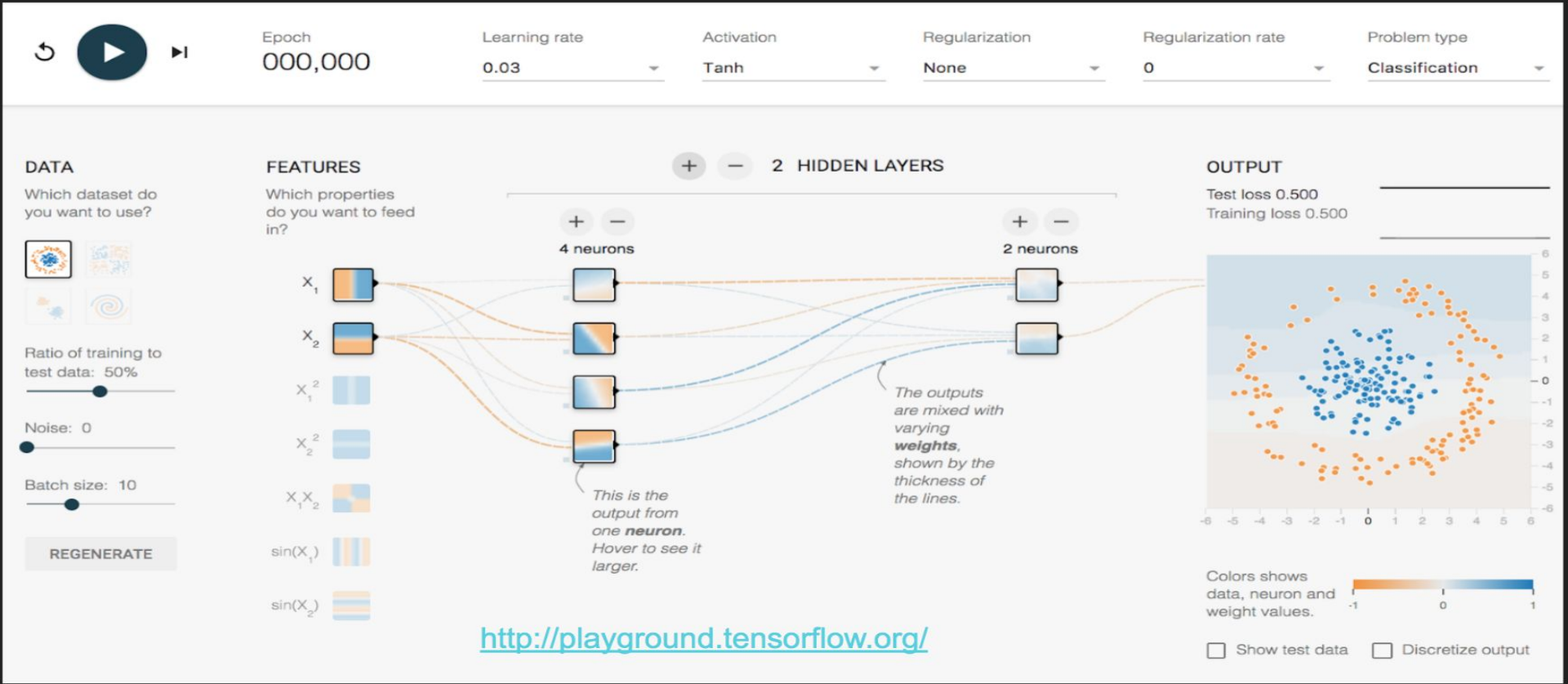
(a) Standard Neural Net



(b) After applying dropout.

http://playground.tensorflow.org/

TensorFlow Playground



Cancer Detection

```
## Read in the data set
data = pd.read_csv("data/BreastCancer.csv")
data.head()
```

	Id	Cl.thickness	Cell.size	Cell.shape	Marg.adhesion	Epith.c.size	Bare.nuclei	Bl.cromatin	Normal.nucleoli	Mitoses	Class
0	1000025	5	1	1	1	2	1	3	1	1	benign
1	1002945	5	4	4	5	7	10	3	2	1	benign
2	1015425	3	1	1	1	2	2	3	1	1	benign
3	1016277	6	8	8	1	3	4	3	7	1	benign
4	1017023	4	1	1	3	2	1	3	1	1	benign

```
## Define the neural net and compile it
from keras.models import Sequential
from keras.layers import Dense, Activation

model = Sequential()
model.add(Dense(32, activation='relu', input_dim=9))
model.add(Dense(32, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

http://srdas.github.io/Presentations/ClassClust/DeepLearning_Introduction_Short.slides.html#/21

Digit Recognition

https://www.cs.toronto.edu/~kriz/cifar.html

http://srdas.github.io/Presentations/ClassClust/DeepLearning_Introduction_Short.slides.html#/28

THE MNIST DATABASE

of handwritten digits

Yann LeCun, Courant Institute, NYU
Corinna Cortes, Google Labs, New York
Christopher J.C. Burges, Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	60,000 x 784						0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Showing 1 to 25 of 60,000 entries

Learning the Black-Scholes-Merton Model

```
from scipy.stats import norm
def BSM(S,K,T,sig,rf,dv,cp):  #cp = {+1.0 (calls), -1.0 (puts)}
    d1 = (math.log(S/K)+(rf-dv+0.5*sig**2)*T)/(sig*math.sqrt(T))
    d2 = d1 - sig*math.sqrt(T)
    return cp*S*math.exp(-dv*T)*norm.cdf(d1*cp) - cp*K*math.exp(-rf*T)*norm.cdf(d2
*cp)

df = pd.read_csv('data/training.csv')
```

C is homogeneous degree one, so

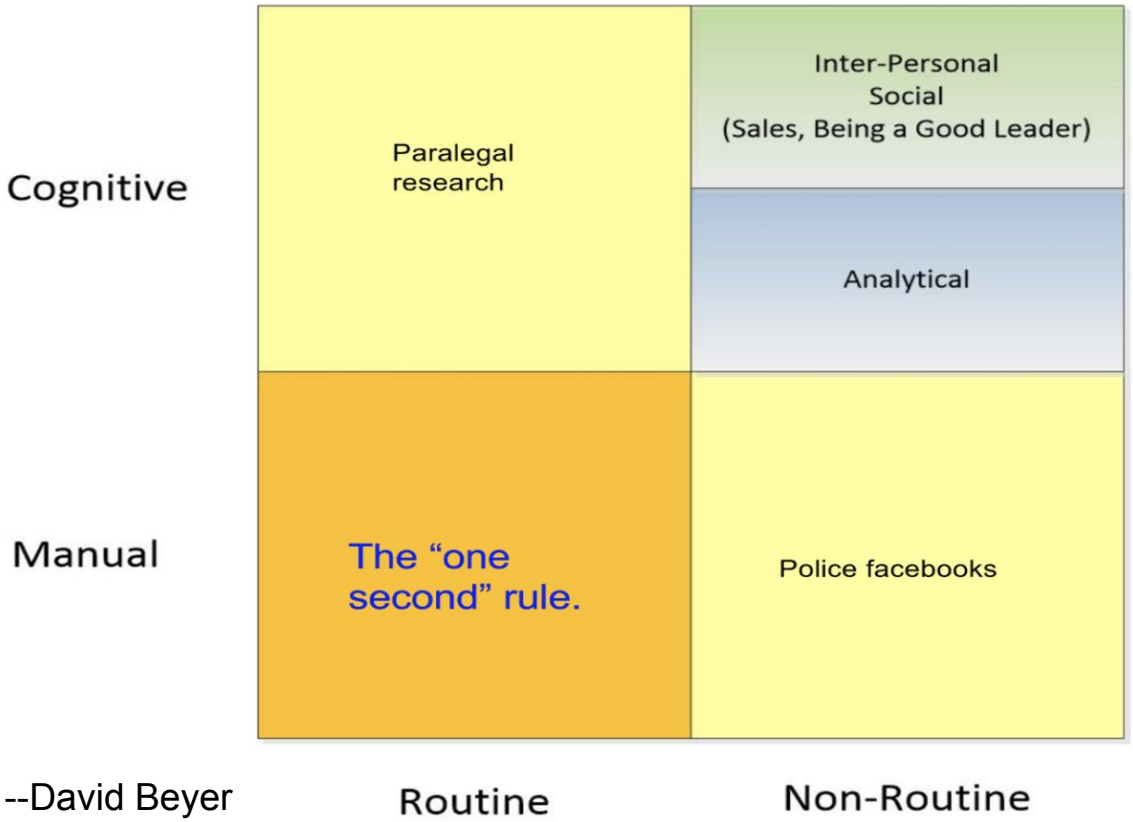
$$aC(S,K) = C(aS,aK)$$

This means we can normalize spot and call prices and remove a variable by dividing by K.

$$\frac{C(S,K)}{K} = C(S/K,1)$$

http://srdas.github.io/Presentations/ClassClust/DeepLearning_Introduction_Short.slides.html#/36

Machine Learning and the Future of Work



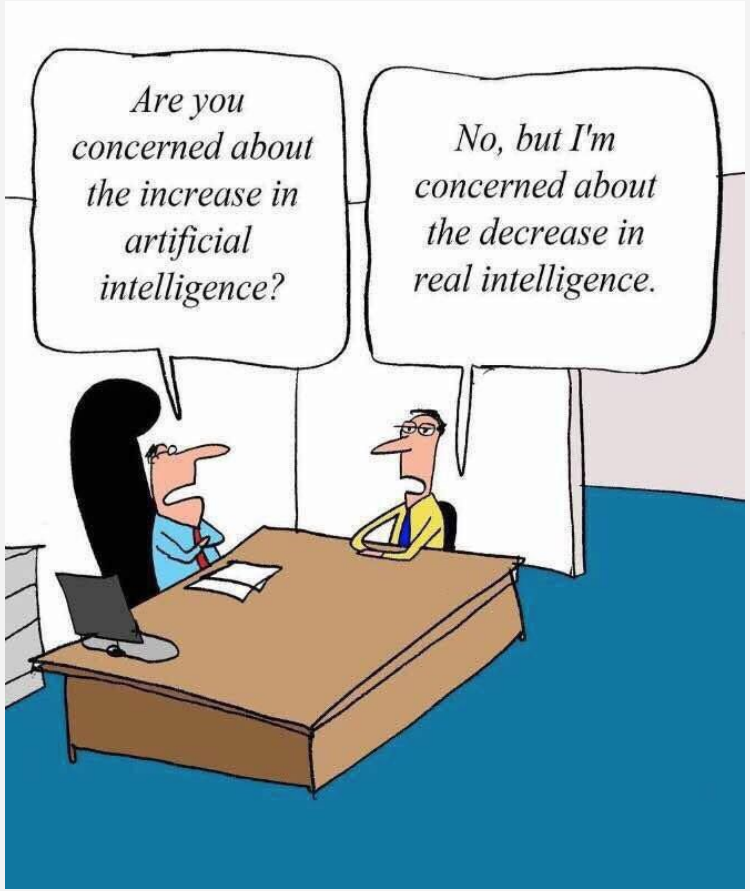
--David Beyer

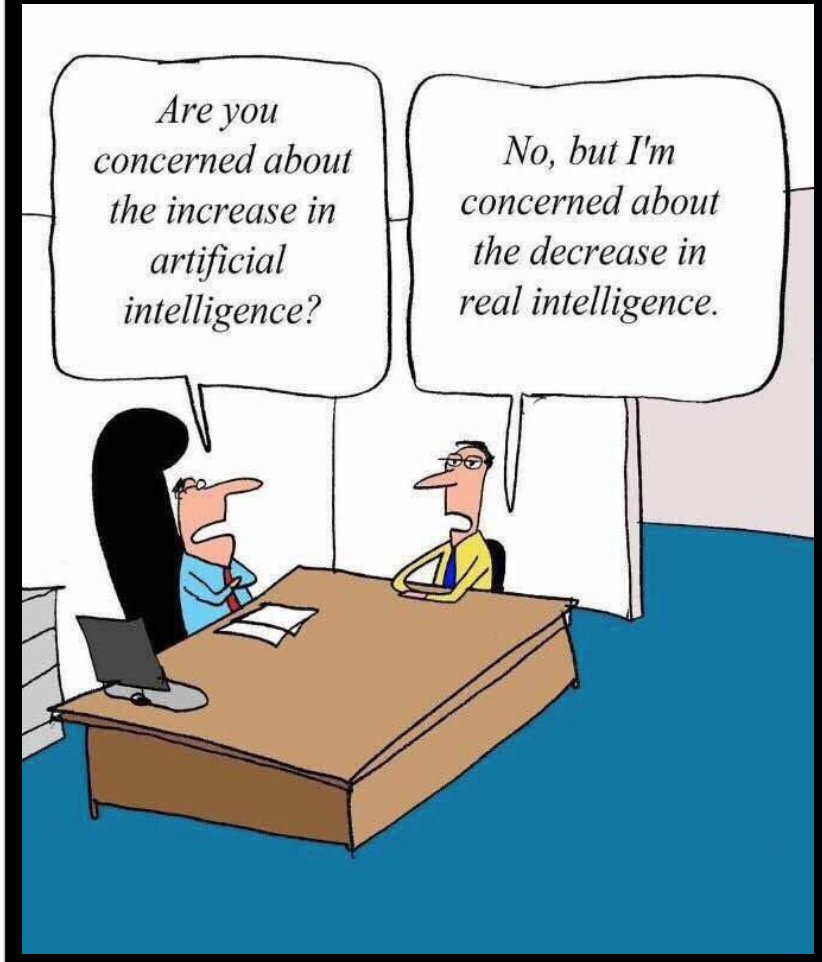
AI researchers salaries go through the roof:
https://www.nytimes.com/2017/10/22/technology/artificial-intelligence-experts-salaries.html?hp&action=click&pgtype=Homepage&clickSource=story-heading&module=second-column-region&region=top-news&WT.nav=top-news

(Roy) Amara's Law: "We tend to overestimate the effect of a technology in the short run and underestimate the effect in the long run."

Arthur C. Clarke's Three Laws:
1. When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.
2. The only way of discovering the limits of the possible is to venture a little way past them into the impossible.
3. Any sufficiently advanced technology is indistinguishable from magic.

Thank you.


Are you concerned about the increase in artificial intelligence?
No, but I'm concerned about the decrease in real intelligence.



http://srdas.github.io/Papers/fintech.pdf

Thank you!!