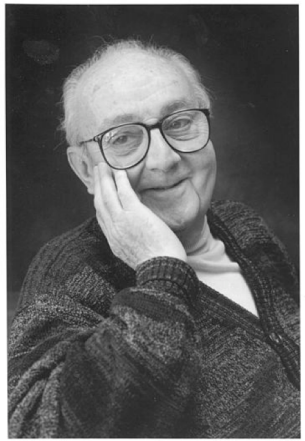
Machine Learning

Overview

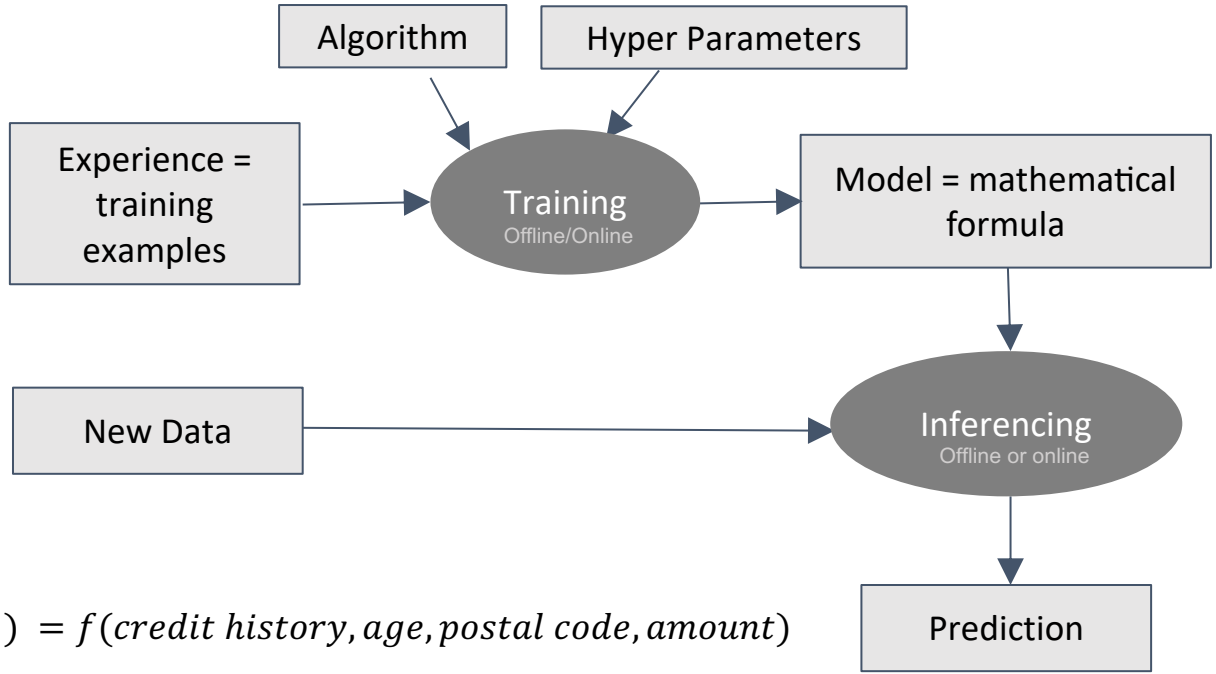
“All models are wrong, some are useful” - George Box.



What is Machine Learning?

The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience

- Tom Mitchell



$$credit\ decision\ (approved?) = f(credit\ history, age, postal\ code, amount)$$

What machine learns?

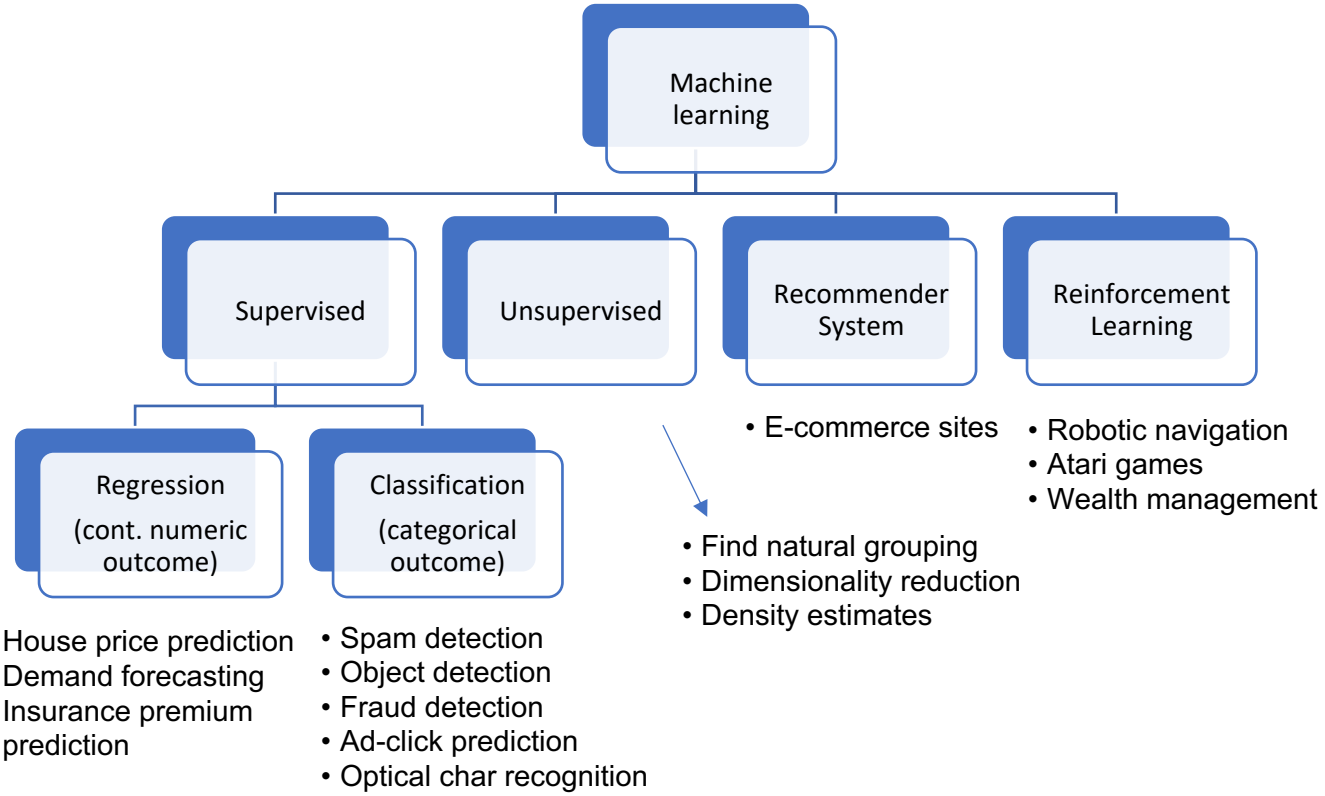
Consider this machine learning problem: Depending on the features like GRE score, GPA score and rank predict whether an applicant will get admission to the university. Below is a sample dataset.

	admit	gre	gpa	rank
1	0	380	3.61	3
2	1	660	3.67	3
3	1	800	4.00	1
4	1	640	3.19	4
5	0	520	2.93	4
6	1	760	3.00	2

$$probability(admit = 1) = \frac{1}{1 + e^{-(-3.45 + 0.002\ gre + 0.78\ gpa - 0.56\ rank)}}$$

weights/coefficients, learned by ML system observing the past data

Types of machine learning



Quiz: Which types of ML problems are these?

1. Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient.
2. Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data
3. Identify the numbers in a handwritten zip code, from a digitized image
4. Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person's blood.
5. Identify the risk factors for prostate cancer, based on clinical and demographic variables

Why machine learning? Faster, cheaper, better

Doing tasks better than human	Complete a task in fraction of time what human would take	Make cost effective solutions	Make predictions
• Predicting transit time • Predict user clicks on digital advertisement • Product recommendation • Self driving vehicle	• Medical diagnosis using medical images • Take decision on loan approvals • Translate text from one language to another • Audio to text • Find cat images in YouTube videos • Group news articles into categories • Identify spams	• Find calories burnt on treadmill • Find weight of the baby in mother's womb	• Forecast market price • Forecast quarterly revenue • Winning probability of a deal

Training-Validation-Test Split

Training Set - build model	Validation/Dev Set - use for model selection	Test Set- test final model to get confidence on the model

Small Data	80%	10%	10%
Big Data (1m+)	98%	1%	1%

Take training, validation and test sets from the same distribution of data, for example,

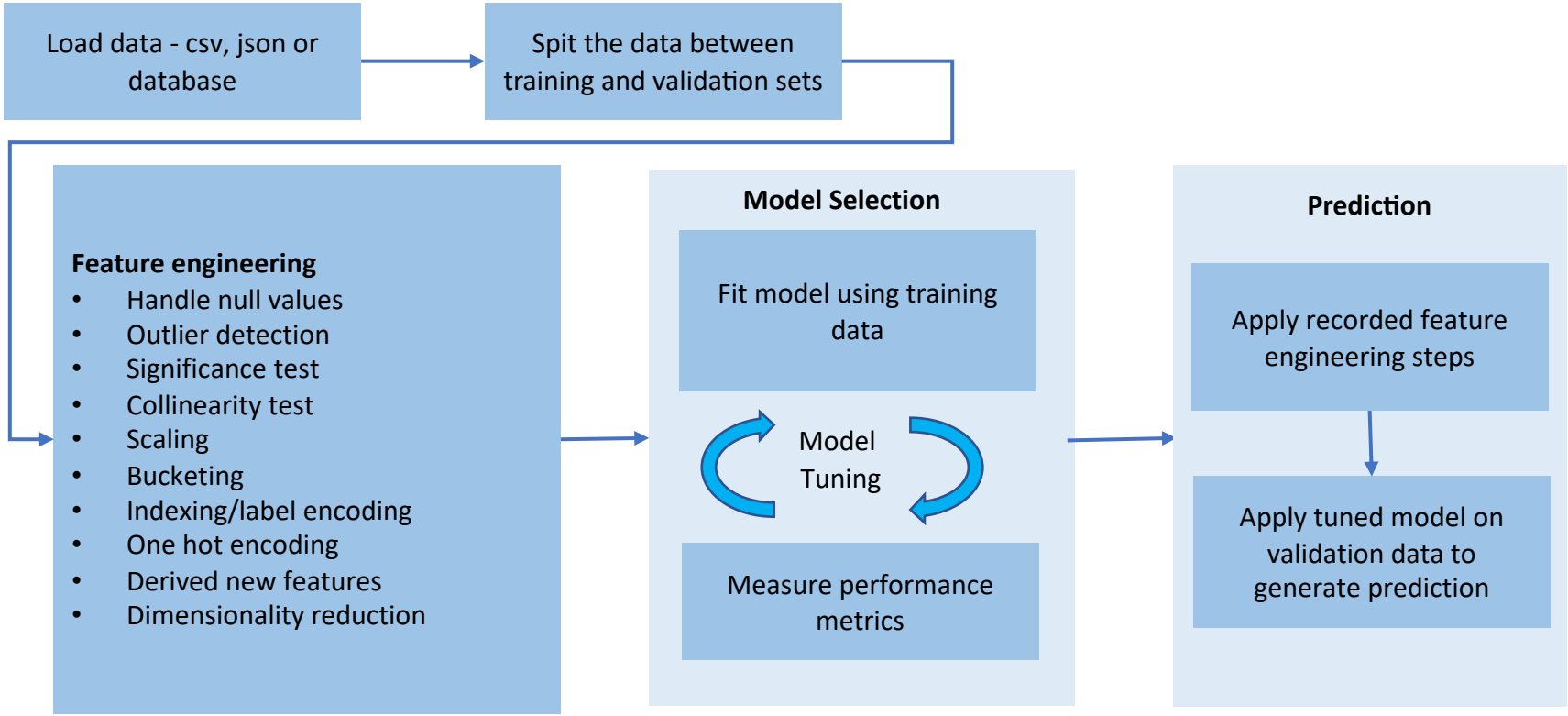
- Building a cat vs dog classifier, to test the model on cat images from Asian countries, include training data from the Asians countries as well
- While building the credit decisioning system, if you want to test the model on low income zip code, include data from low income zip codes

Usecases by industry

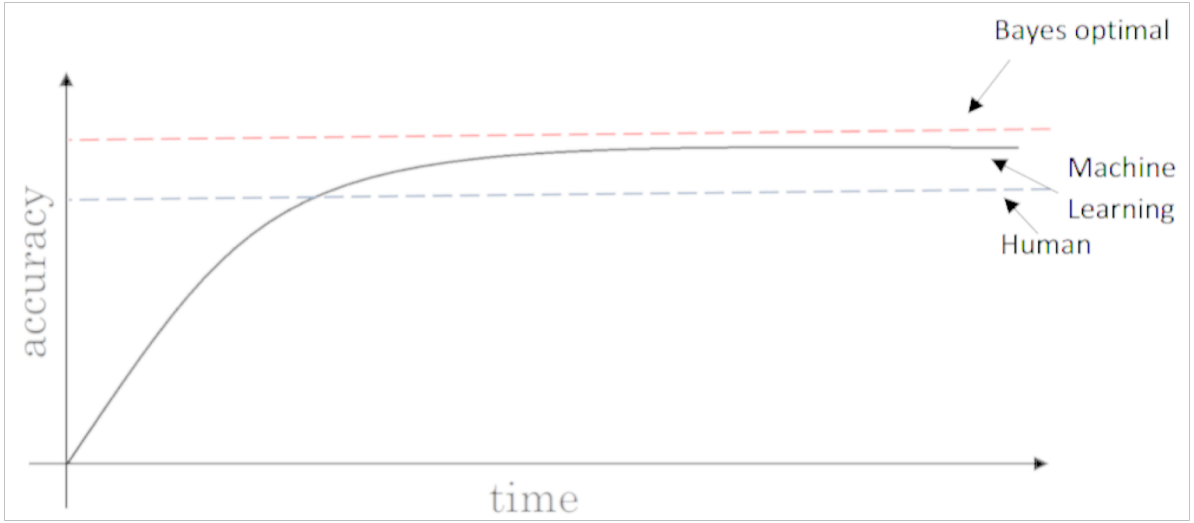
Exercise: pick 5 use cases and fill out the table

Use case	Label	Features/Input	Classification/Regression

Recipe for machine learning



Comparing to human level performance



In statistical classification, Bayes error rate is the lowest possible error rate for any classifier of a random outcome (into, for example, one of two categories) and is analogous to the irreducible error.

Quiz

- What is the role of test data?
- What is the problem with nulls in the dataset?
- Why do we do feature scaling?
- What are the two most common types of feature scaling methods are there?
- Why do we use random seed (random_state) while splitting the data?
- What is one hot encoding?
- Which transformation do we apply on the output variable of regression problems and why?

Parametric vs Non-parametric

Parametric Model: parametric models assume that f takes a specific functional form. Generally such models are more interpretable but less accurate than nonparametric model. Example,

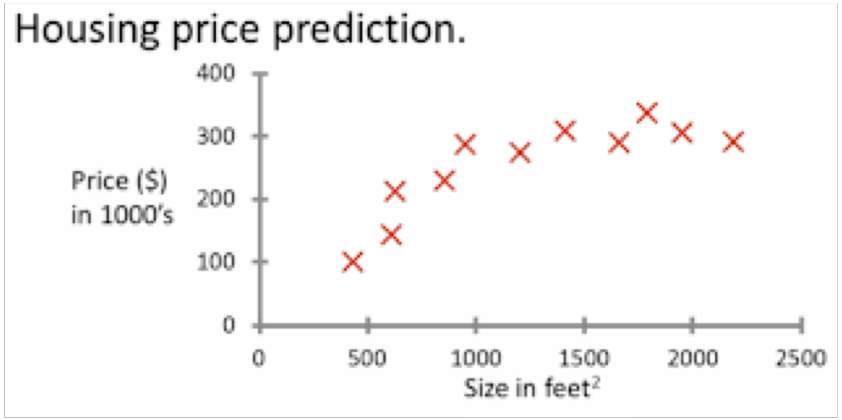
$\hat{y} = f(X) = \ \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ \ldots$

θ_i: thetas/coefficient/weights, x_i: feature, $\hat{y}$ is the estimate

Nonparametric Model: in nonparametric models, f doesn't take a fixed function, but rather the form and complexity of f adapts to the complexity of the data. Examples of a nonparametric model are classification tree, KNN classifiers. Locally weighted linear regression is another example.

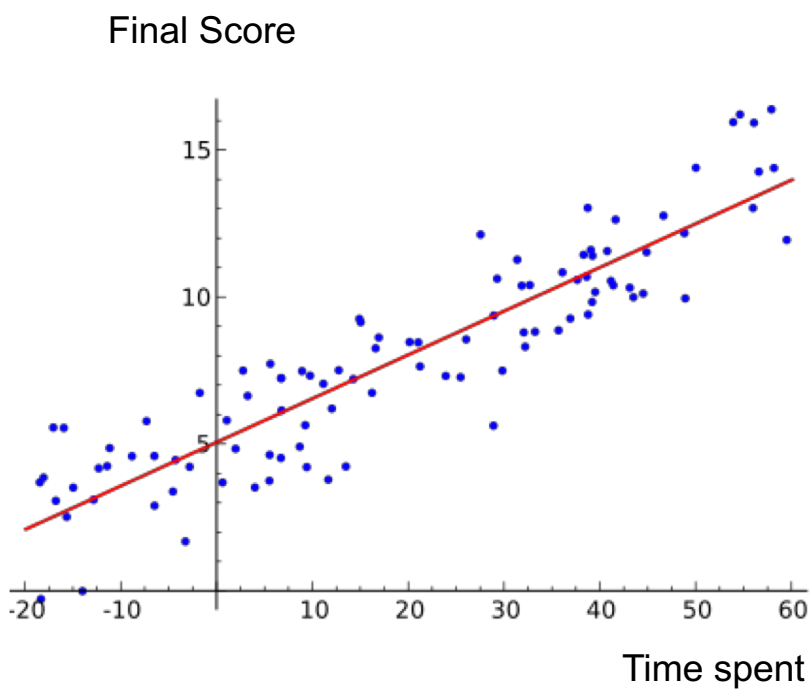
Regression

In regression analysis, we are given a number of predictor (explanatory) variables and a continuous response variable (outcome), and we try to find a relationship between those variables that allows us to predict an outcome



Regression - example

Let's assume that we are interested in predicting the Math SAT scores of students. If there is a relationship between the time spent studying for the test and the final scores, we could use it as training data to learn a model that uses the study time to predict the test scores of future students who are planning to take this test.



Linear Regression

Assume that Y (dependent variable) can be expressed in the following equation in terms of X, where X is predictor matrix

$$\hat{y} = \theta X + \in$$

where, θ can estimated as below based on the observed dataset.

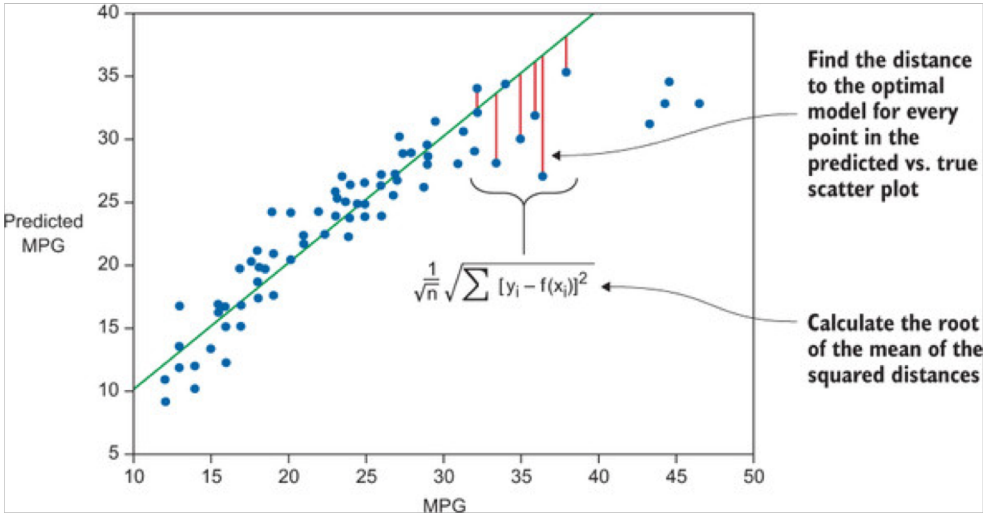
$$\theta = (X^T X)^{-1} X^T y$$

This closed form equation is known as normal equation

$$time\ complexity = O(nm^2 + m^3)$$

Evaluate regression models

An RMSE calculation: in the equation, y_i and x_i are the i^{th} target and feature vector, respectively, and f(x) denotes the application of the model to the feature vector, returning the predicted target value.



Measure performance

One way to compute the performance is to measure mean of squared errors (MSE) of the model on test dataset. Goal will be to minimize this error.

$\text{MSE}_{\text{test}} = \frac{1}{m} \sum_{i=1}^{m} (y_{estimate}^{(i)} - y_{actual}^{(i)})^2$

Other matrices

- Root mean squared error (RMSE)
- Mean Absolute Error (MAE)
- R2

R2- score

$$R2 = 1 - \frac{SSE}{SST} = 1 - \frac{var(error)}{var(outcome)}$$

$$SST = \sum_{i=m}^{m} (\bar{y}_{training} - y^{(i)})^2$$

$$SSE = \sum_{i=m}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

$$R2_{adjusted} = 1 - \frac{(n-1)(1-R^2)}{n - p - 1}$$

n: no of observation

p: degree of freedom (= no of independent features)

- R2 indicates goodness of fit,
- Higher R2 value indicates, better model. Maximum possible value of R2 is 1.0
- Negative value of R2 indicates, the model is not useful
- R2 increases or remains same if new features are added to the model
- Adjusted R2 score penalizes a model more with more number features. This is a technique to regularize the model

Linear regression assumptions

- Linear relationship - the relationship between the independent and dependent variables to be linear
- Multivariate normality - all variables to be multivariate normal
- No or little multicollinearity - multicollinearity occurs when the independent variables are too highly correlated with each other
- No auto-correlation - Autocorrelation occurs when the residuals are not independent from each other. For instance, this typically occurs in stock prices, where the price is not independent from the previous price.
- Homoscedasticity - meaning the residuals are equal across the regression line

Linear Regression

Strengths

- By far the most common approach for modeling numeric data
- Can be adapted to model almost any modeling task
- Provides estimates of both the strength and size of the relationships among features and the outcome

Weaknesses

- Makes strong assumptions about the data
- The model's form must be specified by the user in advance
- Does not handle missing data
- Only works with numeric features, so categorical data requires extra processing
- Requires some knowledge of statistics to understand the model

Quiz

1. What is the type of outcome variable in regression problems – continuous number or categorical?
2. Which of the metrics are independent of scale of the outcome variable – RMSE, MSE, R2?
3. What are the types of input features for regression problems – continuous, categorical or either continuous/categorical?
4. What is the significance of the sign (positive/negative) of the coefficient?
5. What the negative value of R2 indicate?
6. What is the perfect score of R2?
7. What is the perfect score of RMSE?

Why normal equations does not work in practice

- A matrix can be non-invertible either because there collinearity exists or number of observations is lower than the number of features
- Computing inverse of a matrix is expensive
  - time complexity of calculating the inverse of a $n_x n$ matrix which is O(n^3), n = dimension
  - n is low (n < 1000 or n < 10000) normal equation is a viable option

Case for Gradient Descent

Gradient overcome the limitations and offers some advantages as below

- Time complexity is $O(mn)\ per\ iteration$ vs $O(nm^2 + m^3)$ of closed form equation
- Regularization
- Boosting/bagging
- Can be extended to solve problems like Neural Networks

Gradient Descent

- While in some cases it is possible to find the global minimum of the cost function analytically (when it exists), in the great majority of cases we will have to use an optimization algorithm.
- Optimizers update the set of weights iteratively in a way that decreases the loss over time.
- The most commonly used approach is gradient descent, where we use the loss's gradient with respect to the set of weights

Update rule $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$

$\boldsymbol{\alpha}$ is learning rate, a positive number

Least Mean Square (LMS) algorithm

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost function for the regression problem

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}(h_\theta(x) - y)^2 \\
&= 2\cdot\frac{1}{2}(h_\theta(x) - y)\cdot\frac{\partial}{\partial\theta_j}(h_\theta(x) - y) \\
&= (h_\theta(x) - y)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{n}\theta_i x_i - y\right) \\
&= (h_\theta(x) - y)\, x_j
\end{aligned}$$

Summation is ignored for simplicity

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right)x_j^{(i)}$$

Gradient Descent Algorithm for OLS (ordinary least square)

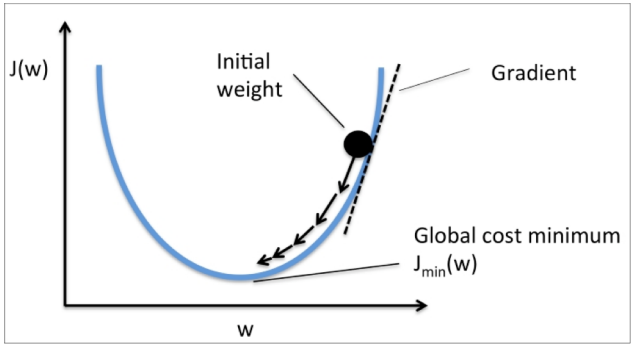
Update Rule:

$$\text{repeat until convergence } \{$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

$$\}$$

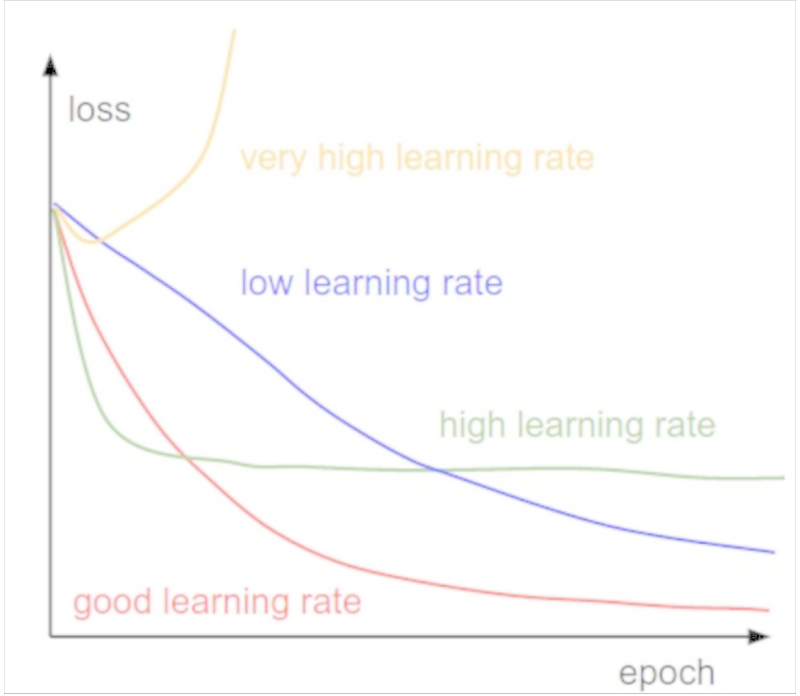


The quadratic cost function always converges for a small α value. Convergence to a local minimum can be guaranteed.

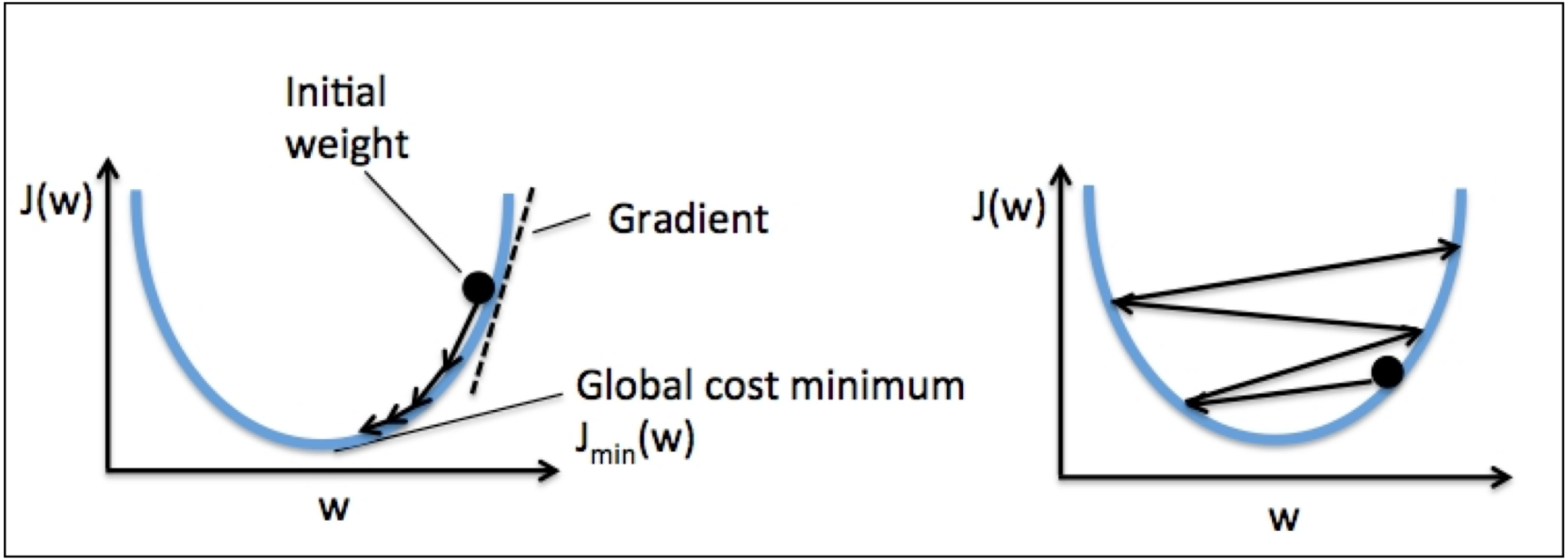
Learning Rate

- Learning rate determines how aggressive each update iteration will be (or in other words, how large the step will be in the direction of the negative gradient).
- We want the decrease in the loss to be fast enough on the one hand, but on the other hand not large enough so that we over-shoot the target and end up in a point with a higher value of the loss function.

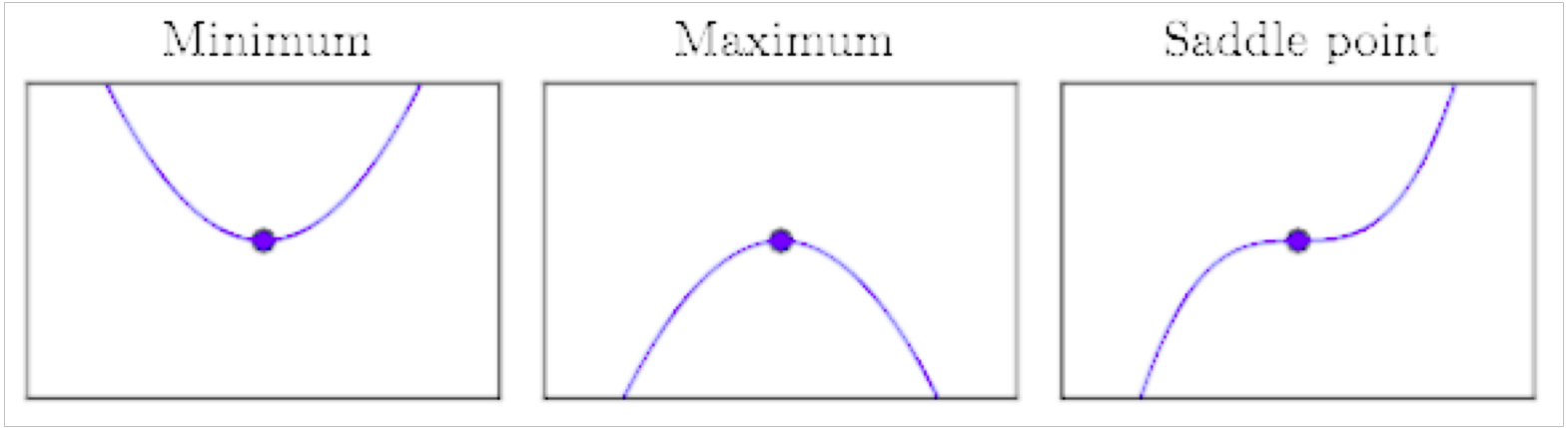
Effect of learning rate on cost decay



Impact of learning rate in search for global minima



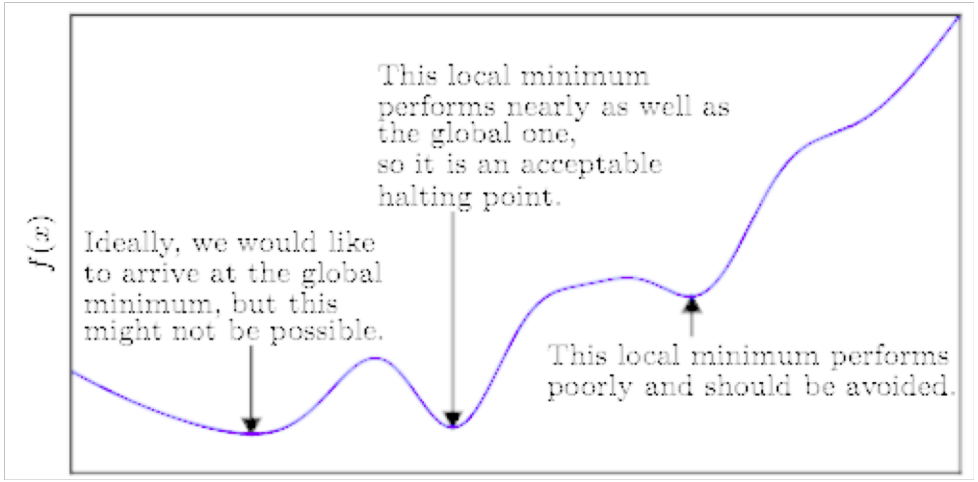
Critical points in gradient descent



A critical point is a point with zero slope. Such a point can either be a local minimum, which is lower than the neighboring points; a local maximum, which is higher than the neighboring points; or a saddle point, which has neighbors that are both higher and lower than the point itself

Global Minimum

Optimization algorithms may fail to find a global
minimum when there are multiple local minima or plateaus present. In the context of
ML, we generally accept such solutions even though they are not truly minimal, so long as they correspond to significantly low values of the cost function



Feature Normalization

Gradient descent algorithm converges faster when the feature normalization is performed on the dataset.

$z_j = \frac{x_j - \bar{x}_j}{s_j}$, j stands for jth feature

Z-score data has zero mean and one standard deviation.

Sampling method for gradient descent

- The gradient of the objective is computed with respect to the model parameters and evaluated using a given set of input samples.
- How many of the samples should we take for this calculation? Intuitively, it makes sense to calculate the gradient for the entire set of samples. This method, however, has some shortcomings, for example, it can be very slow and is intractable when the dataset requires more memory than is available.
- A more popular technique is the “stochastic gradient descent” (SGD), where instead of feeding the entire dataset to the algorithm for the computation of each step, a subset of the data is sampled sequentially.
- The number of samples ranges from one sample at a time to typically a few hundred, but most common sizes are between around 50 to around 500, usually referred to as mini-batches.

Gradient Descent Variants

Batch

- Computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset
- Batch gradient descent performs redundant computations for large datasets, as it re-computes gradients for similar examples before each parameter update.

Stochastic

- Performs a parameter update for *each* training example x_i and label y_i
- SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online.

Mini Batch

- Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n (50-200) training examples
- Reduces the variance of the parameter updates, which can lead to more stable convergence; and
- Can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient.

Challenges with gradient descent

- Choosing a proper learning rate can be difficult. A learning rate that is too small leads to painfully slow convergence, while a learning rate that is too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge.
- Learning rate schedules try to adjust the learning rate during training by e.g. annealing, i.e. reducing the learning rate according to a pre-defined schedule or when the change in objective between epochs falls below a threshold. These schedules and thresholds, however, have to be defined in advance and are thus unable to adapt to a dataset's characteristics
- The same learning rate applies to all parameter updates. If our data is sparse and our features have very different frequencies, we might not want to update all of them to the same extent, but perform a larger update for rarely occurring features
- Minimizing highly non-convex error functions common for neural networks is avoiding getting trapped in their numerous suboptimal local minima. The difficulty arises in fact not from local minima but from saddle points, i.e. points where one dimension slopes up and another slopes down. These saddle points are usually surrounded by a plateau of the same error, which makes it notoriously hard for SGD to escape, as the gradient is close to zero in all dimensions.

Quiz: Define the terms below

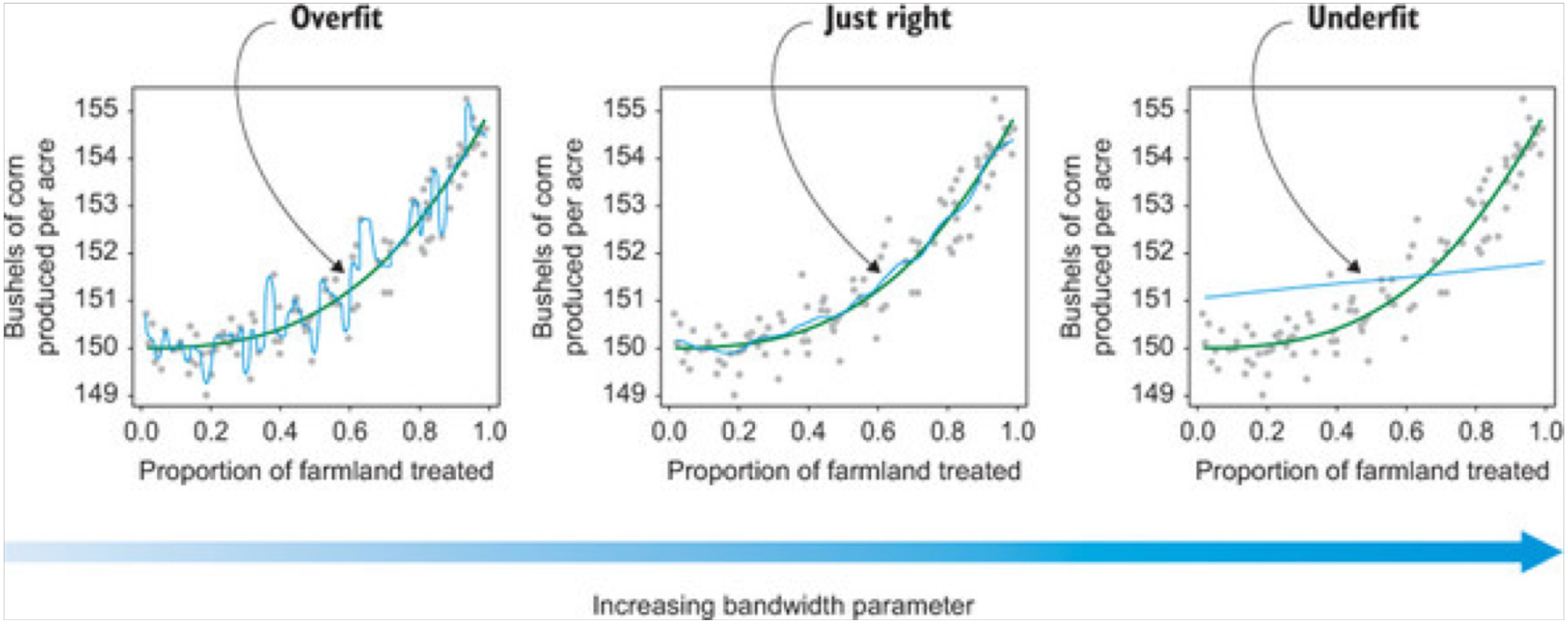
Term	Definition
Weights	
Batch size	
Epoch	
Gradient Descent	
Cost function	
Training	
Stochastic gradient descent	
Learning rate	
Feature engineering	

Quiz

1. What are the typical initial values of the parameters in the gradient descent algorithm?
2. Why MSE is generally used as cost function instead of RMSE?
3. When a gradient descent algorithm terminates?
4. How do we get the value of learning rate that gives the best result?

Regularization

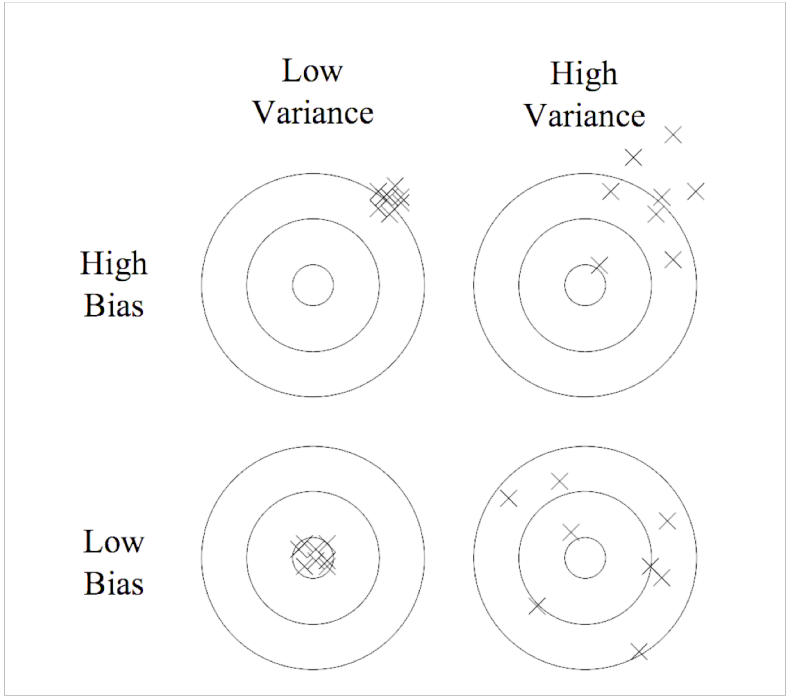
Overfitting, Underfitting and Justright



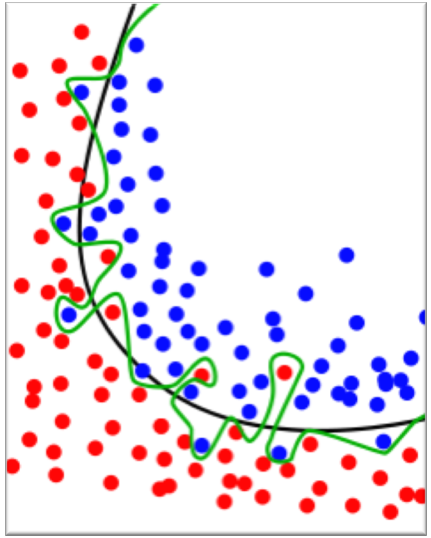
Overfitting, Underfitting and Just Right

- Model performs much better on training than on testing => Overfitting
- Model performs not good both in training and testing => Underfitting
- Model performs almost equally for training and test => Just right

Bias variance in dart throwing



Overfitting



- In overfitting, a statistical model describes random error or noise instead of the underlying relationship.
- Overfitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.
- Over-fitted model has poor predictive performance, as it overreacts to minor fluctuations in the training data.

How to solve overfitting?

Occam's razor - law of parsimony

Among competing hypotheses that explain known observations equally well, we should choose the "simplest" one.

- Less weights of the features
- Fewer features

Regularization - bias variance tradeoff

- Regularization is mostly used to refer to the restriction of an optimization problem by imposing a penalty on the complexity of the solution, in the attempt to prevent overfitting to the given examples.
- Regularization is most often applied by adding implicit information regarding the desired results.
- For example, look at the following improvised loss function that includes a penalty term.

$$L = \sum_{m} (\sum_{i=0}^{d} (\alpha_i x_m^i) - y_m)^2$$

Model without regularization

$$L = \sum_{m} (\sum_{i=0} (\alpha_i x_m^i) - y_m)^2 + \lambda \sum_{i=1} \alpha_i^2$$

Model with L2 regularization

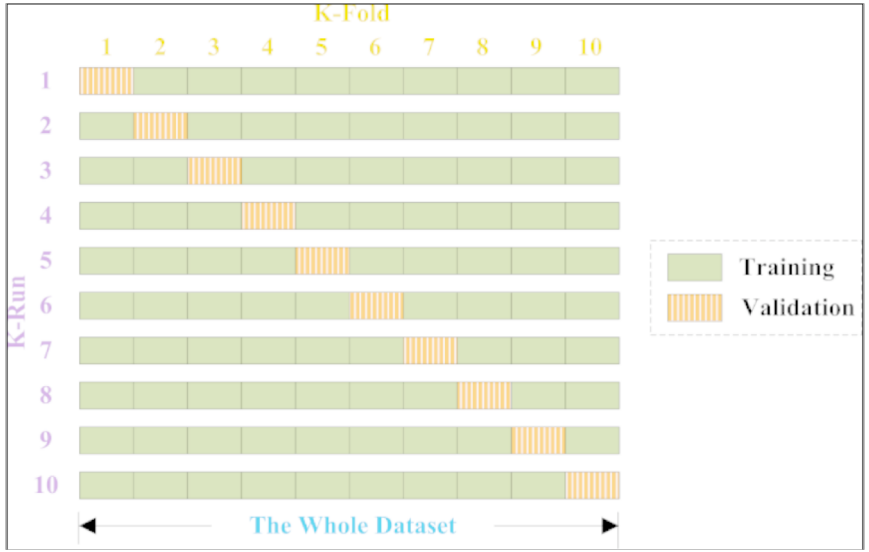
Quiz

1. If the R2 score on training data 0.95 and 0.80 on test. Do we call this phenomenon – overfitting or underfitting?
2. What is objective of regularization – A. control overfitting B. control underfitting C. control both overfitting and underfitting?
3. To overcome underfitting, we need to make the model more complex, such as add more feature or add polynomial terms to the model. Is this true?
4. To overcome overfitting, one possibility is to add more training examples?
5. In LASSO model, which norm is used to penalize the coefficients – L1 or L2?
6. What is the range of the regularization parameters?
7. Which algorithm has natural tendency of selecting the significant features – Lasso, Ridge, Elastic Net?

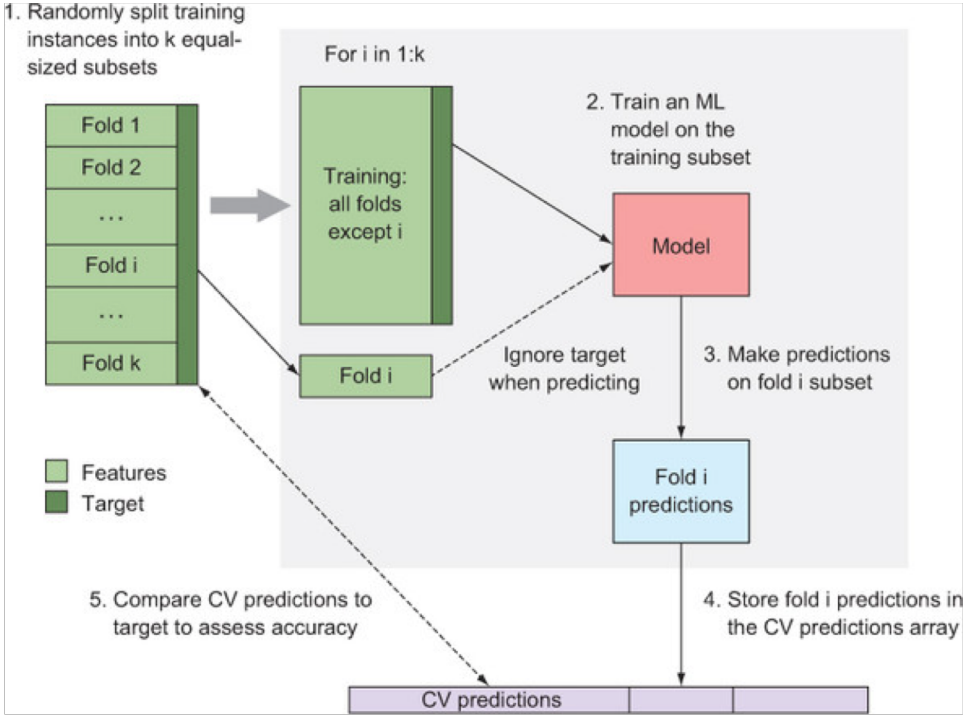
Cross Validation

Cross Validation

CrossValidation basically splits your complete available training data into a number of k folds. This parameter k can be specified. Then, the whole Pipeline is run once for every fold and one machine learning model is trained for each fold. Finally, the different machine learning models obtained are joined. This is done by a voting scheme for classifiers or by averaging for regression.

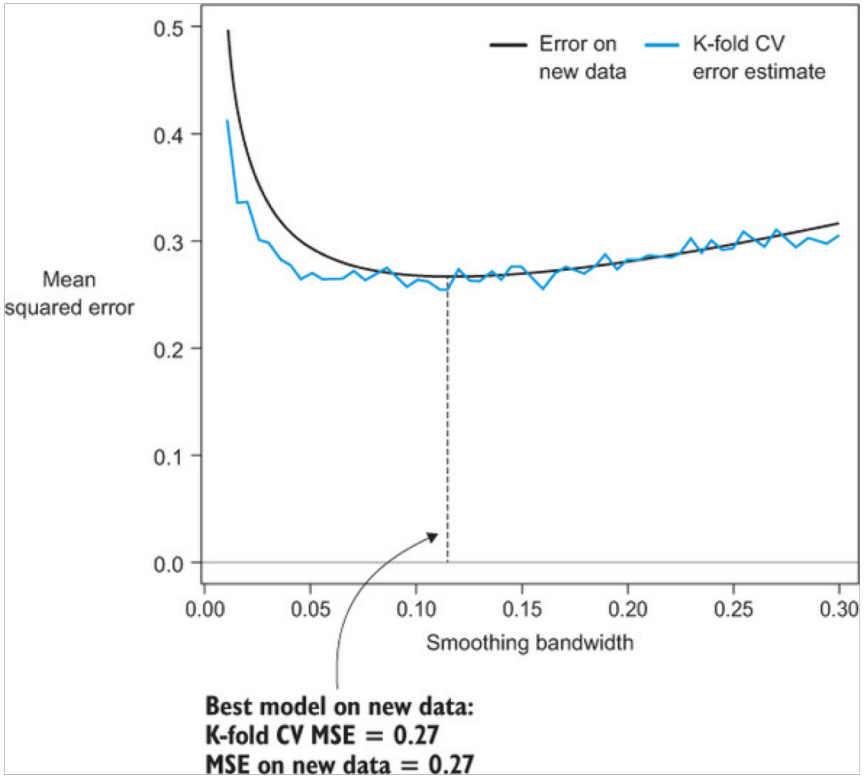


CV Model


1. Randomly split training instances into k equal-sized subsets
Fold 1
Fold 2
...
Fold i
...
Fold k
For i in 1:k
Training: all folds except i
Fold i
2. Train an ML model on the training subset
Model
Ignore target when predicting
3. Make predictions on fold i subset
Fold i predictions
4. Store fold i predictions in the CV predictions array
CV predictions
5. Compare CV predictions to target to assess accuracy
Features
Target

Cross Validation

Comparison of the k-fold cross-validation error MSE to the MSE on new data, using the corn production dataset. The k-fold CV error is a good estimate for how the model will perform on new data, allowing you to use it confidently to forecast the error of the model and to select the best model.



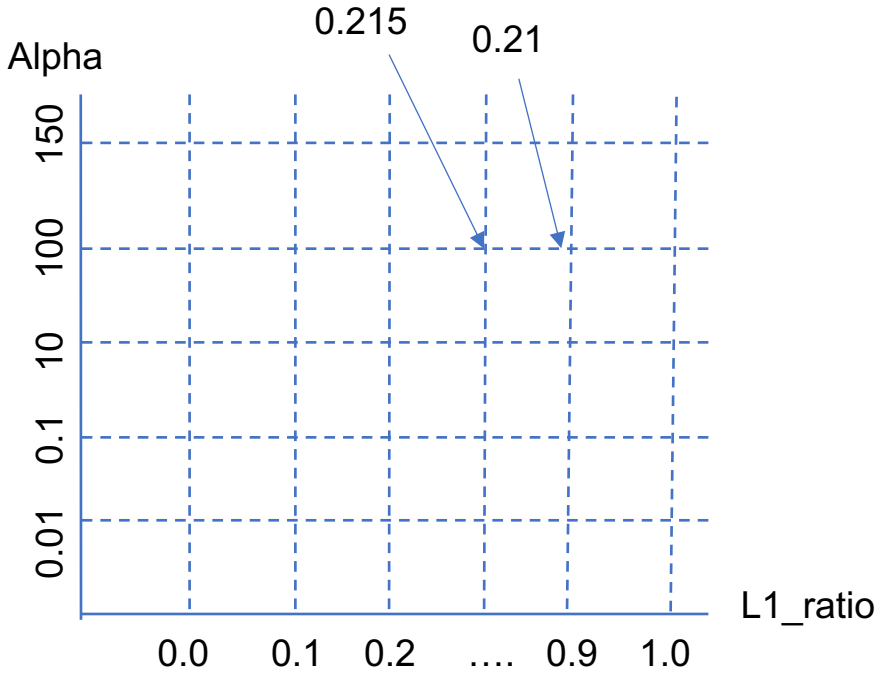
Quiz

1. What is the purpose of cross validation?
2. In k-fold cross validation, do we create the folds by sampling data with replacement or without replacement?
3. What is the downside of using cross validation in big data scale?

Hyper Parameter Tuning

Grid search for best set of hyper parameters

Find score for each combinations of hyper-parameters and find the combination that gives best CV-score. Using CV-score give more confidence on the score.



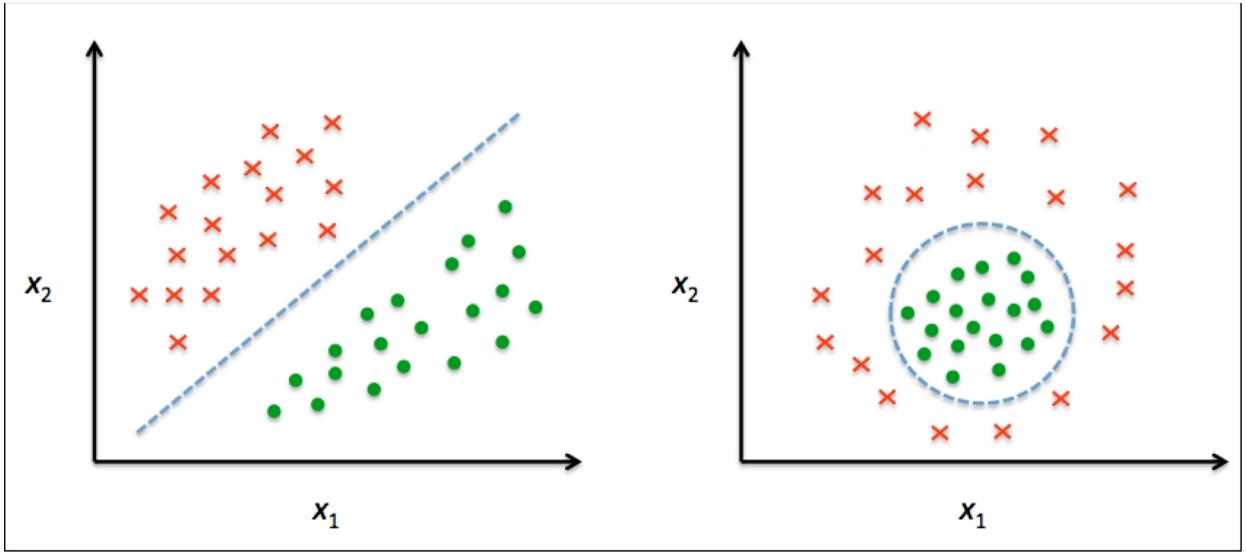
Classification

"As far as the laws of mathematics refer to reality, they are not certain, and as far as they are certain, they do not refer to reality." - Albert Einstein

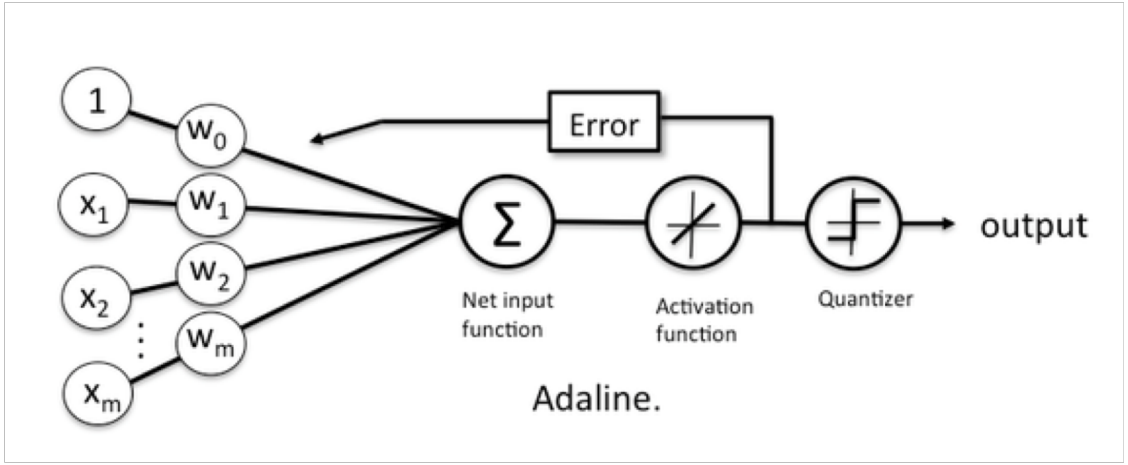
Classification

- Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known.
- An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).

Linear vs non linear classification



Logistic Regression



$$Net\ input, z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots = log\ \frac{P(Y=1)}{1-P(Y=1)}$$

$$Activation\ function\ is\ g(z) = sigmoid(z)$$

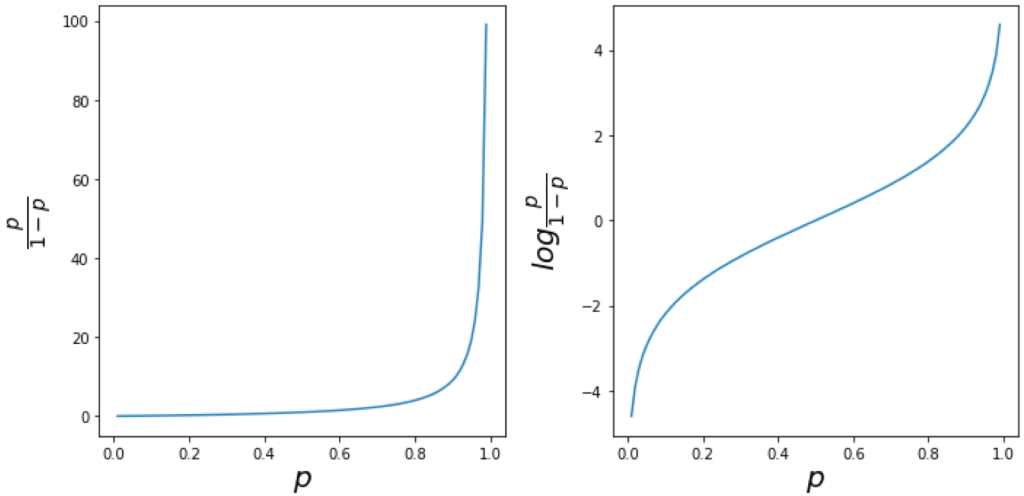
$$Quantizer\ f(x) = \begin{cases} 0\ when\ x < 0.5 \\ 1\ otherwise \end{cases}$$

Odds of positive

Example: there might be an 80% chance of rain today. Odds (more technically the odds of success) is defined as probability of success/probability of failure.

So the odds of a success (80% chance of rain) has an accompanying odds of failure (20% chance it doesn't rain); odds ratio, 0.8/0.2 = 4.

Log odds is the logarithm of the odds. Ln(4) = 1.38629436 ≅ 1.386.



Sigmoid: express P(y=1) in terms of Z

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ \ldots = log \frac{\hat{y}}{1 - \hat{y}}, where, \hat{y} = P(y = 1)$$

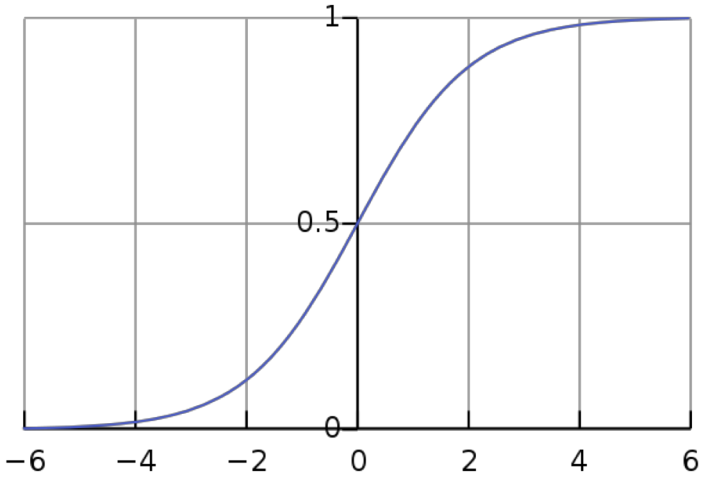
So, we can rearrange the equation to say

$$\hat{y} = \frac{1}{1 + e^{-z}}$$

Sigmoid activation function

$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

The logistic function is useful because it can take any real input, z and whereas the output always takes values between zero and one and hence is interpretable as a probability



Logistic Regression

Probability estimate

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

Likelihood for m independent training examples

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

Maximize the log likelihood

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

Cross entropy

$$-\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Logarithmic Loss

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j})$$

N is the number of observations,
M is the number of class labels,
log is the natural logarithm,
$y_{i,j}$ is 1 if observation *i* is in class *j* and 0 otherwise
$p_{i,j}$ is the predicted probability that observation i is in class j.

Log-loss measures the performance of a classification model where the prediction input is a probability value between 0 and 1. The goal of our machine learning models is to minimize this value.

Logistic Regression - derivative

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial \theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\
&= \left(y - h_\theta(x)\right) x_j
\end{aligned}
$$

$$
\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}
$$

Softmax - multi class classification

The softmax function, or normalized exponential function is a generalization of the logistic function that "squashes" a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values in the range [0, 1] that add up to 1. It is used as class probability in case of multi class classification problems.

$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$

Metrics

Accuracy: (TP+TN) / (P + N)

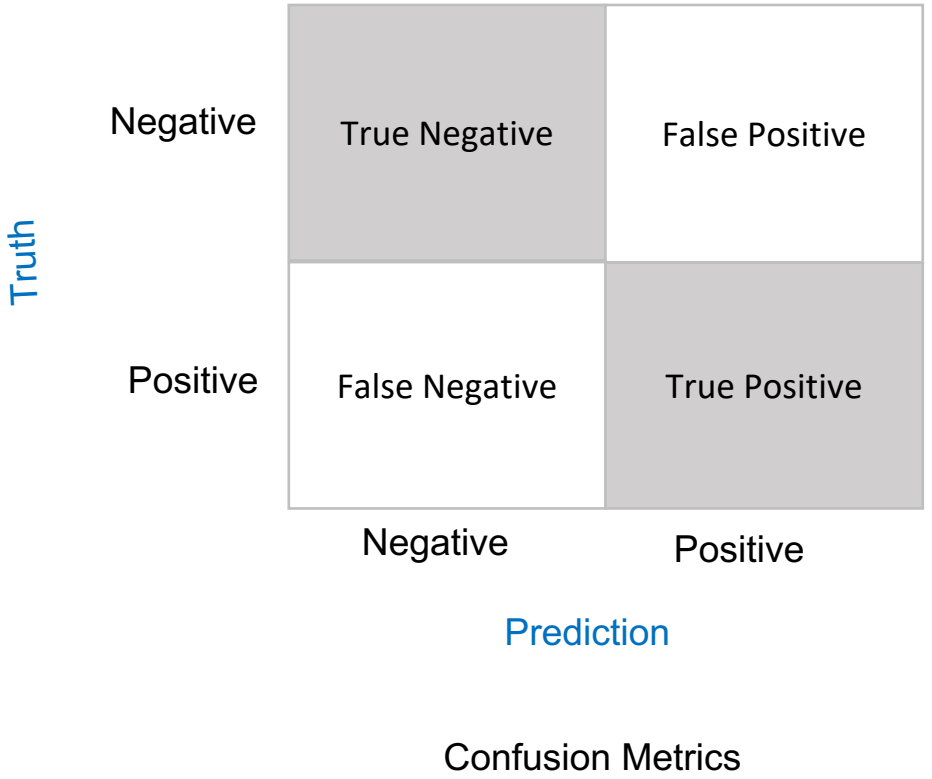
Precision: TP / (TP + FP)

TPR Or Recall Or Sensitivity: TP / P

FPR : FP / N = FP / (FP + TN)

Specificity: TN / N

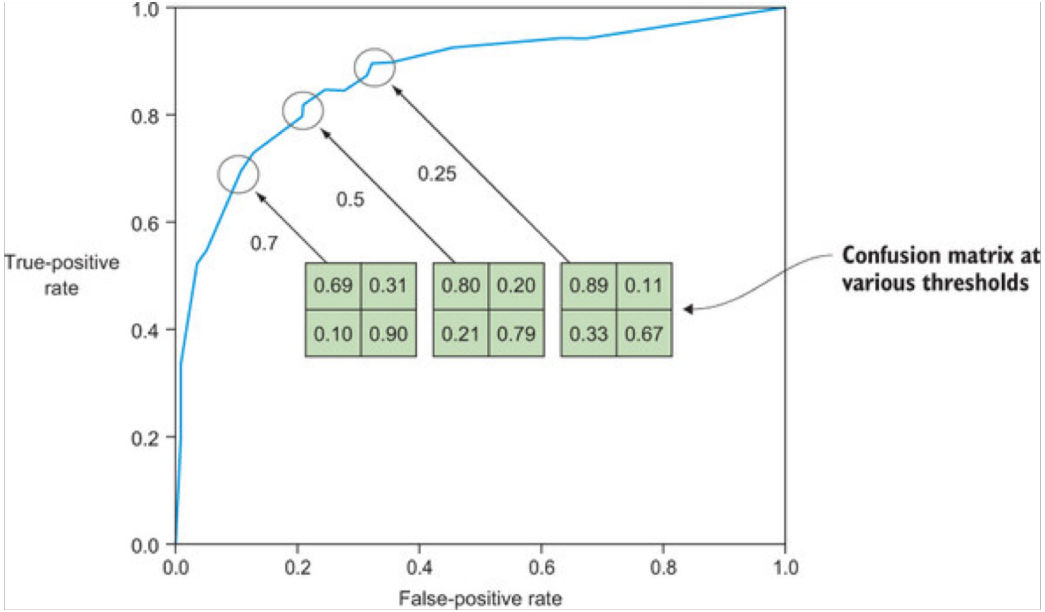
$$F1 = \frac{2 * precision * recall}{precison + recall}$$



Confusion Metrics

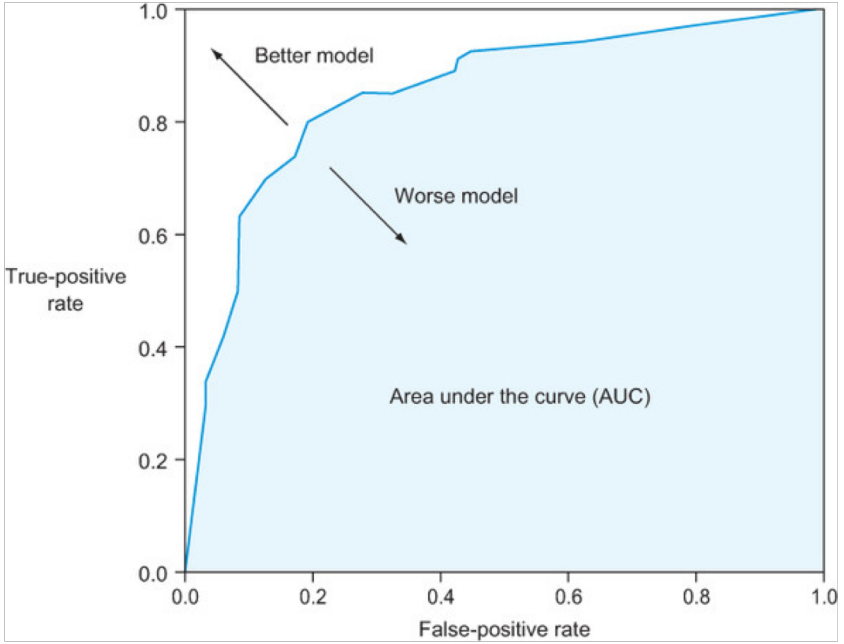
ROCR Curve

The ROC (receiver operating characteristic curve) curve defined by calculating the confusion matrix and ROC metrics at 100 threshold points from 0 to 1. By convention, you plot the false-positive rate on the x-axis and the true-positive rate on the y-axis.



AUC Score

The ROC curve illustrates the overall model performance. You can quantify this by defining the AUC metric: the area under the ROC curve.

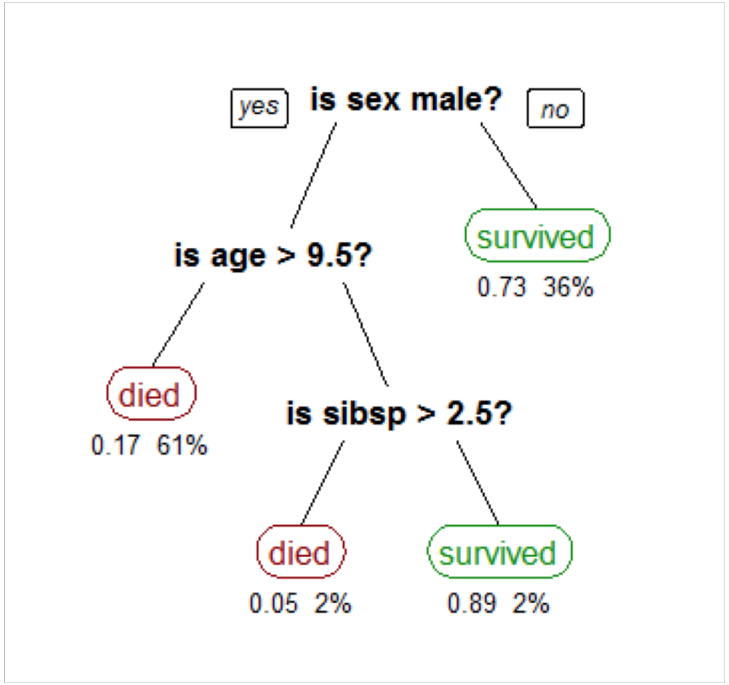


Exercise: define in your own terms

Term	Definition
MSE	
Cross Entropy	
Accuracy	
Precision	
Recall	
ROC Curve	

Decision Tree

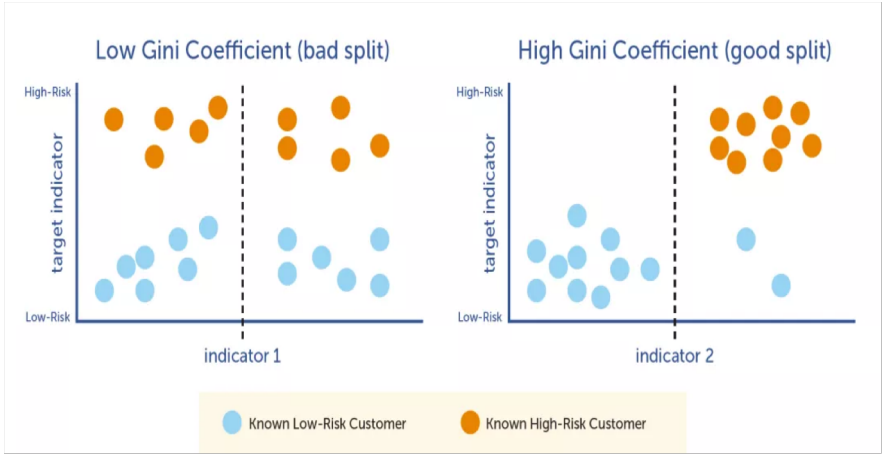
- The decision trees are a branch test-based classifiers
- These classifiers use the knowledge of training data it creates a decision trees that is used to classify test data.
- In the decision tree every branch represents a set of classes and a leaf represent a particular class.
- A decision node identifies a test on a single attribute value with one branch and its subsequent classes represent as class outcomes.



Dataset: Titanic survivors

Decision Tree

- To maximize interpretability these classifiers are expressed as decision trees or rule sets (IF-THEN), forms that are generally easier to understand than neural networks.
- Decision tree based classifiers are easy to use and does not presume any special knowledge of statistics or machine learning.
- Does not require feature engineering



The classifier is built by finding the cut points along the axis such that maximizes the homogeneity of each rectangular region

Gini impurity

- Measures uncertainity in the data
- Algorithm for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items.
- Different algorithms use different metrics for measuring "best".
- Gini impurity metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.
- The Gini coefficient measures the inequality among values of a frequency distribution (for example, levels of income).

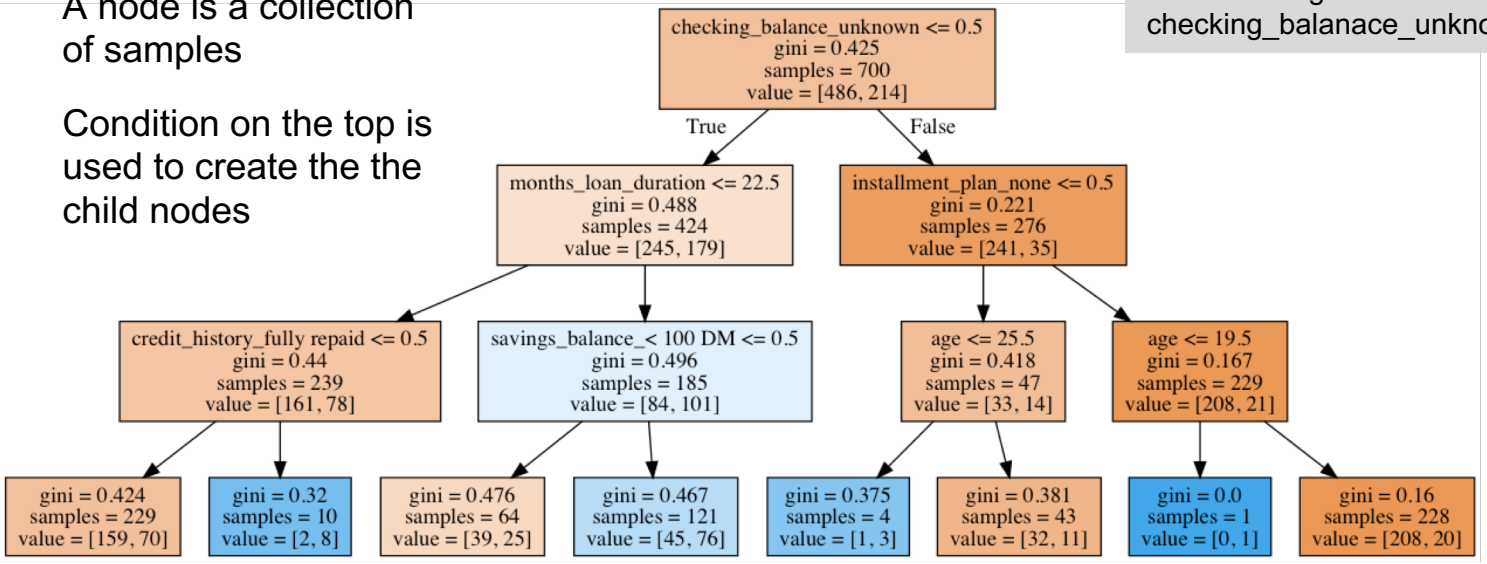
$$I_G(p) = \sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J}(p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2 = \sum_{i \neq k} p_i p_k$$

Decision tree example

samples = 700
frequencies = [0: 486, 1:214]
gini = 0.425
checking_balanace_unknown <= 0.5

A node is a collection of samples

Condition on the top is used to create the the child nodes




```
from sklearn.tree import export_graphviz
export_graphviz(est, out_file = "tree.dot", feature_names = X.columns, filled=True)
!dot -Tpng tree.dot -o tree.png
```

Display the tree on windows machine

Install graphviz using MSI on windows machine
https://graphviz.gitlab.io/_pages/Download/Download_windows.html

Open command prompt and run

```
C:/> set PATH=%PATH%;"C:\Program Files (x86)\Graphviz2.38\bin\"
C:/> dot -Tpng tree.dot -o tree.png
```

Feature importance

	feature	importance
9	checking_balance_unknown	0.560980
0	months_loan_duration	0.152312
1	amount	0.094524
39	installment_plan_none	0.062549
8	checking_balance_> 200 DM	0.058001
4	age	0.040380
27	employment_length_1 - 4 yrs	0.031254
42	housing_rent	0.000000

1.you initialize an array feature_importances of all zeros with size n_features.
2.you traverse the tree: for each internal node that splits on feature i you compute the error reduction of that node multiplied by the number of samples that were routed to the node and add this quantity to feature_importances[i].

Potential Use Cases

- Credit scoring models in which the criteria that causes an applicant to be rejected need to be clearly documented and free from bias
- Marketing studies of customer behavior such as satisfaction or churn, which will be shared with management or advertising agencies
- Diagnosis of medical conditions based on laboratory measurements, symptoms, or the rate of disease progression

S/W of decision tree classifiers

Strengths:

- An all-purpose classifier that does well on most problems
- Highly automatic learning process, which can handle numeric or nominal features, as well as missing data
- Excludes unimportant features
- Can be used on both small and large datasets
- Results in a model that can be interpreted without a mathematical background (for relatively small trees)
- More efficient than other complex models

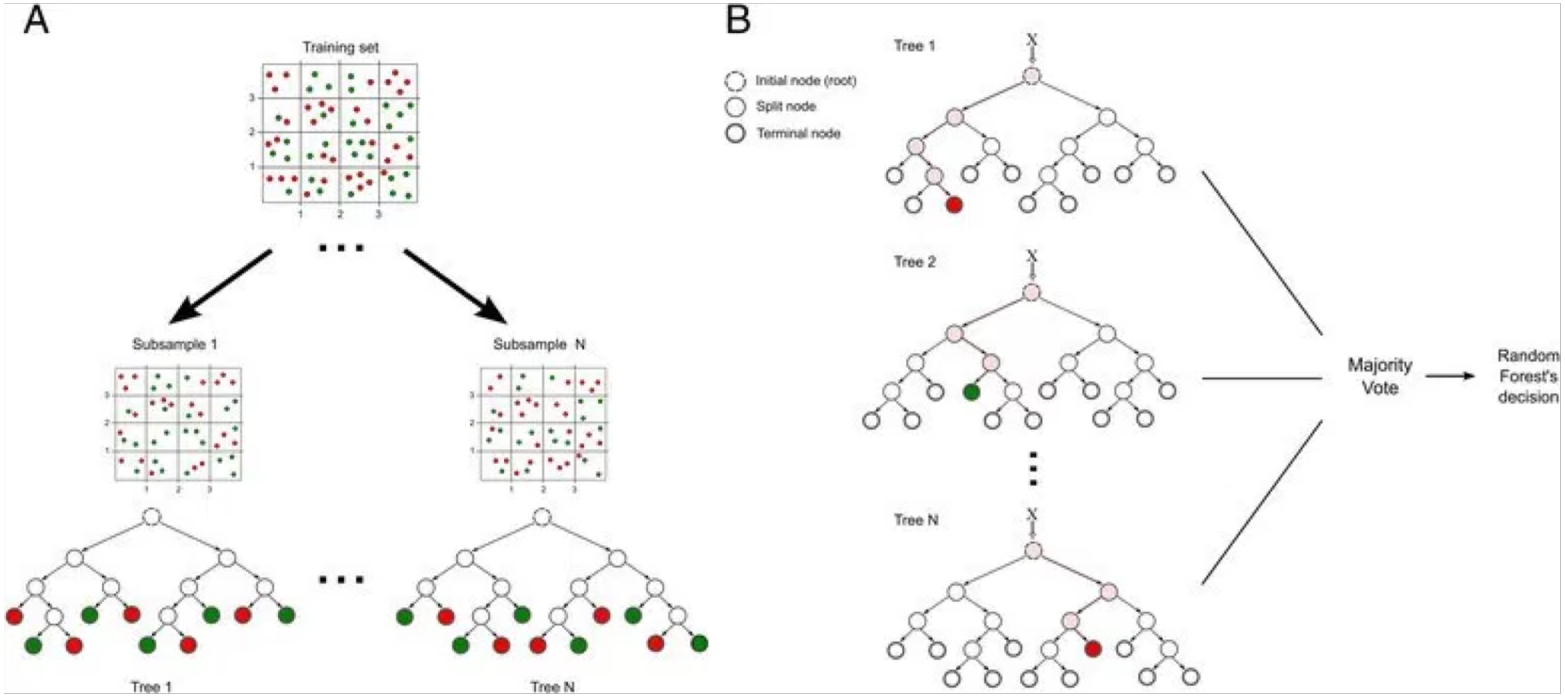
Weaknesses:

- Decision tree models are often biased toward splits on features having a large number of levels
- It is easy to overfit or underfit the model
- Can have trouble modeling some relationships due to reliance on axis-parallel splits
- Small changes in the training data can result in large changes to decision logic
- Large trees can be difficult to interpret and the decisions they make may seem counterintuitive

Random Forest

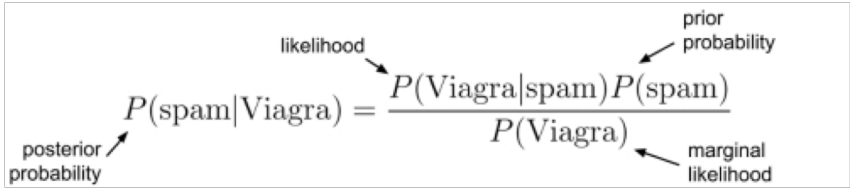
- Random forest classifier used an ensemble of random trees.
- Each of the random trees is generated by using a bootstrap sample data.
- At each node of the tree a subset of feature with highest information gain is selected from a random subset of entire features.
- Thus random forest uses bagging and feature selection to generate the trees.
- Once a forest is generated every tree participates in classification by voting to a class.
- The final classification is based on the majority voting of a particular class.
- It performs better in comparison with single tree classifiers such as CART but it takes longer time to compute.

Random Forest



Naive Bayes Classifier

The Naive Bayes algorithm describes a simple method to apply Bayes' theorem to classification problems. A popular for its in text classification, where it has become the de facto standard. It is named as such because it makes some "naive" assumptions about the data - all of the features in the dataset are equally important and independent. It works only with categorical variables. For numerical continuous data types, a common treatment is to discretize the values based on the percentiles bins.



Likelihood	Viagra: Yes	Viagra: No	Total
spam	4 / 20	16 / 20	20
ham	1 / 80	79 / 80	80
Total	5 / 100	95 / 100	100

$$P(C_L|F_1, ..., F_n) = \frac{1}{Z} p(C_L) \prod_{i=1}^{n} p(F_i|C_L)$$

Strengths and Weaknesses of Naive Bayes

Strengths
- Simple, fast, and very effective
- Does well with noisy and missing data
- Requires relatively few examples for training, but also works well with very large numbers of examples
- Easy to obtain the estimated probability for a prediction

Weaknesses
- Relies on an often-faulty assumption of equally important and independent features
- Not ideal for datasets with many numeric features
- Estimated probabilities are less reliable than the predicted classes

Naive Bayes Classifier

- Naive Bayes classifier is a statistical method based on Bayes theorem.
- It calculates the probability of each training data for each class.
- The class of test data assigns by using the inverse probability.
- It assumes the entire variables are independent, so only mean and variance are required to predict the class.
- The main advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the mean and variances that are used to predict the class.

Formula:

$$P(C \mid x_i) = \frac{P(x_i \mid C)P(C)}{P(x_i)}$$

*Where x_i is some GE level or CN and C is class (i.e., subtype***

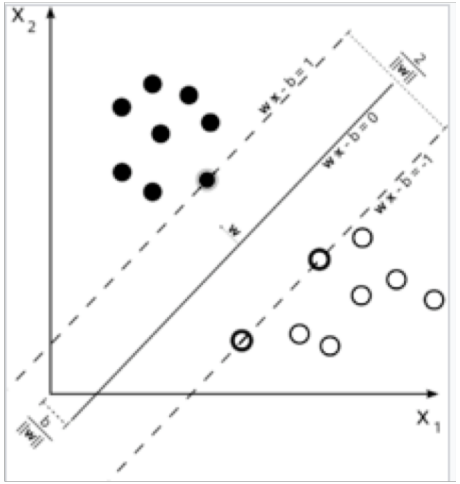
Expansion:

$$P(C \mid x_1, x_2, \ldots, x_{8569}) = \frac{P(x_1 \mid C)P(x_2 \mid C)\cdots P(x_{8569} \mid C)P(C)}{P(x_1)P(x_2)\cdots P(x_{8569})}$$

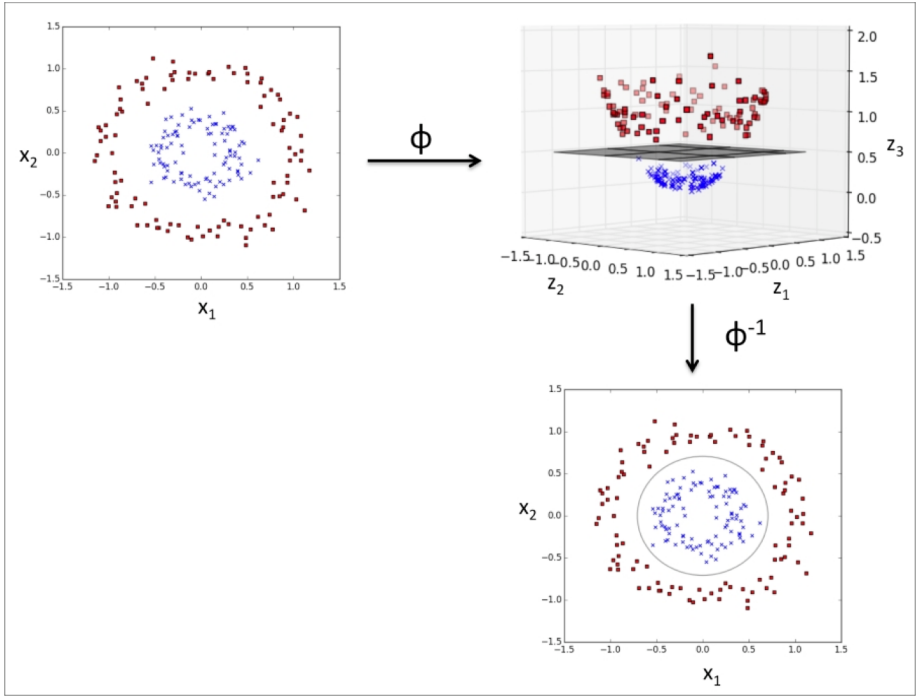
- Key assumption: all features are independent

Support Vector Machine (SVM)

- SVM is based on the statistical learning theory.
- The SVM is capable of resolving linear and non-linear classification problems.
- The principal idea of classification by support vector is to separate examples with a linear decision surface and maximize the margin of separation between the classes to be classified.
- SVM works by mapping data with a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.



SVM



Gaussian Transformation

Single Variable Gaussian

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

Multiple Variable Gaussian

$$P(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

μ is the mean vector and Σ is the covariance matrix

$$\Sigma = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])\,(\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\mathrm{T}}\right]$$

Support Vector Machine (SVM) … contd

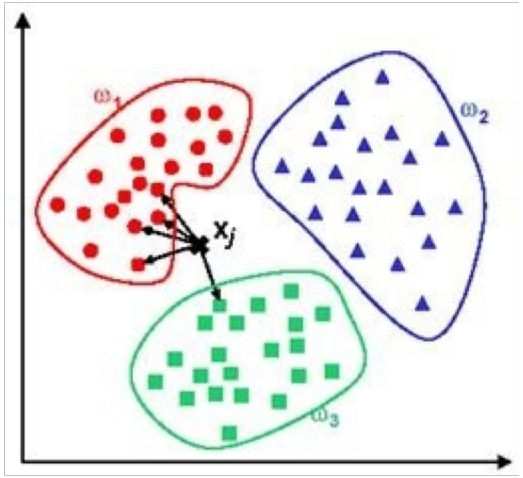
- A separator between the categories is found, and then the data are transformed in such a way that the separator could be drawn as a hyperplane.
- Following this, characteristics of new data can be used to predict the group to which a new record should belong.
- After the transformation, the boundary between the two categories can be defined by a hyperplane.
- The mathematical function used for the transformation is known as the kernel function.

Support Vector Machine (SVM) ... contd

- SVM supports the linear, polynomial, radial basis function (RBF) and sigmoid kernel types.
- When there is a straightforward linear separation then linear function is used otherwise we used polynomial, radial basis function (RBF) and sigmoid kernel function.
- Besides the separating line between the categories, a SVM also finds marginal lines that define the space between the two categories.
- The data points that lie on the margins are known as the support vectors.

K-Nearest Neighbor (K-NN)

- The k-Nearest Neighbors algorithm is a non-parametric method used for classification and regression.
- The input consists of the k closest training examples in the feature space.
- The output depends on whether k-NN is used for classification or regression.
- In k-NN classification, the output is a class membership.



K-Nearest Neighbor (K-NN)

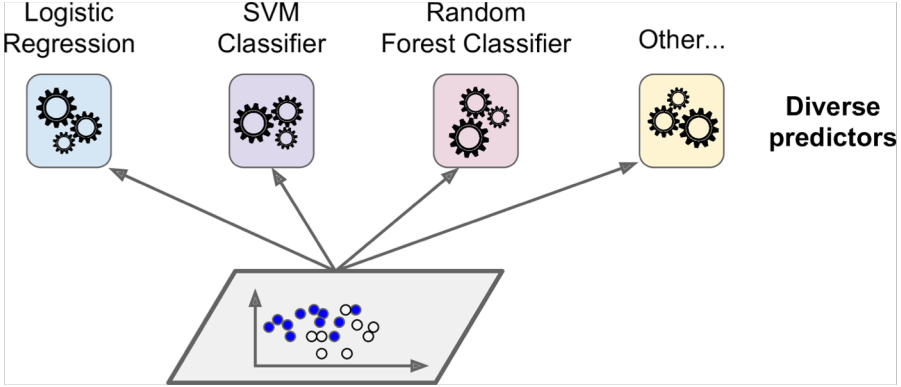
- The k-NN classifiers are based on finding the k nearest neighbor and taking a majority vote among the classes of these k neighbors to assign a class for the given query.
- The k-NN is a type of instance based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification.
- The k-NN is more efficient for large datasets and robustness when processing noisy data but high computation cost reduces its speed.
- To apply KNN, you must normalize (Z-score) the data.

Ensemble Method

The whole of science is nothing more than a refinement of everyday thinking
- Albert Einstein

Ensemble technique in general

- ”Wisdom of crowd”
- Aggregate the predictions of a group of predictors (such as classifiers or regressors)
- Ensemble methods work best when the predictors are as independent from one another as possible. One way to get diverse classifiers is to train them using very different algorithms.



Code example

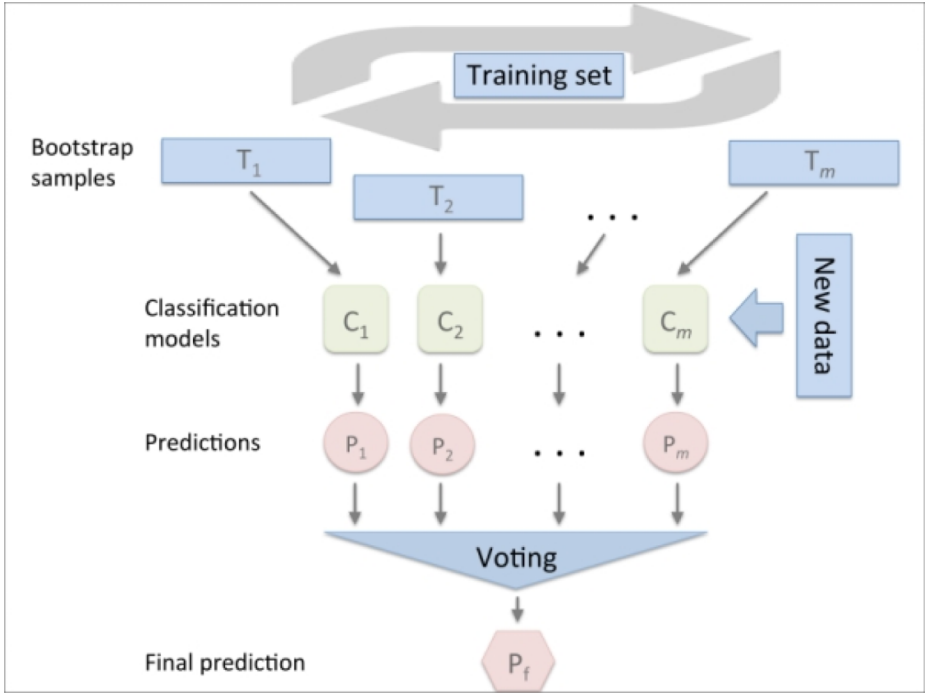
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model
import LogisticRegression
from sklearn.svm import SVC

log_clf = LogisticRegression()
rnd_clf = RandomForestClassifier()
svm_clf = SVC()
voting_clf = VotingClassifier(voting='hard', estimators=[
              ('lr', log_clf),
              ('rf', rnd_clf),
              ('svc', svm_clf)])
voting_clf.fit(X_train, y_train)
```

Random subspace method

- Let the number of training objects be N and the number of features in the training data be D.
- Choose L to be the number of individual classifiers in the ensemble.
- For each individual classifier C, choose dc (dc < D) to be the number of input variables for C. It is common to have only one value of dc for all the individual classifiers.
- For each individual classifier C, create a training set by choosing dc features from D without replacement and train the classifier.
- For classifying a new object, combine the outputs of the L individual classifiers by majority voting or by combining the posterior probabilities.

Bagging

- One way to get a diverse set of classifiers is to use very different training algorithms, as in ensemble technique. Another approach is to use the same training algorithm for every predictor, but to train them on different random subsets of the training set.
- Each cycle through the process results in one classifier.
- After the construction of several classifiers, taking a vote of the predictions of each classifier performs the final prediction



Bagging and Pasting

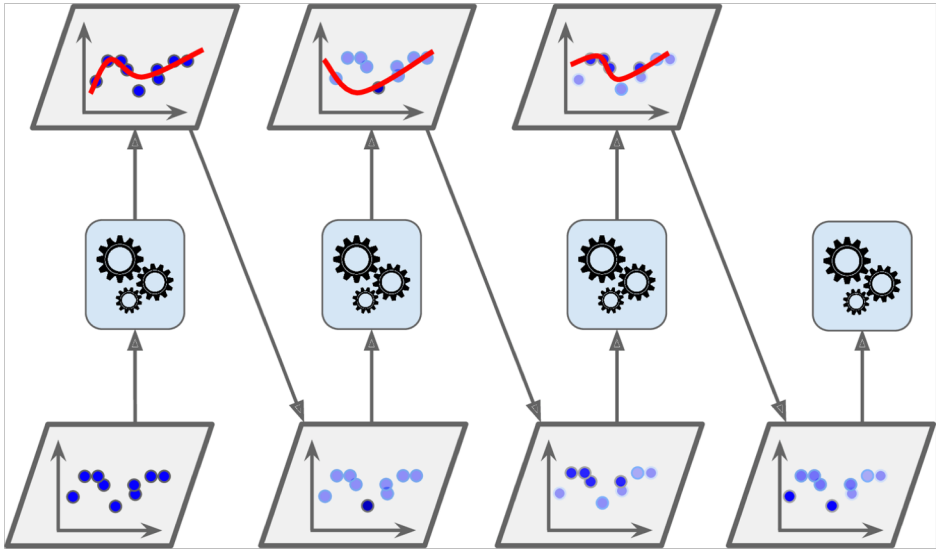
- Bagging is ensemble classifiers.
- In bagging ‘n’ random instances are selected using a uniform distribution (with replacement) from a training dataset of size ‘n’.
- The learning process starts using these ‘n’ randomly selected instances and this process can be repeated several times.
- Since the selection is with replacement, usually the selected instances will contain some duplicates and some omissions as compared to the original training dataset.
- When sampling is performed with replacement, this method is called **bagging** (short for bootstrap aggregating).
- When sampling is performed without replacement, it is called **pasting**.

Boosting

Boosting refers to any Ensemble method that can combine several weak learners into a strong learner. The general idea of most boosting methods is to train predictors sequentially, each trying to correct its predecessor.

AdaBoosting

One way for a new predictor to correct its predecessor is to pay a bit more attention to the training instances that the predecessor underfitted. This results in new predictors focusing more and more on the hard cases.



Gradient Boosting

Gradient Boosting works like AdaBoost by sequentially adding predictors to an ensemble, each one correcting its predecessor. However, instead of tweaking the instance weights at every iteration like AdaBoost does, this method tries to fit the new predictor to the residual errors made by the previous predictor.

Gradient Boosting - by example

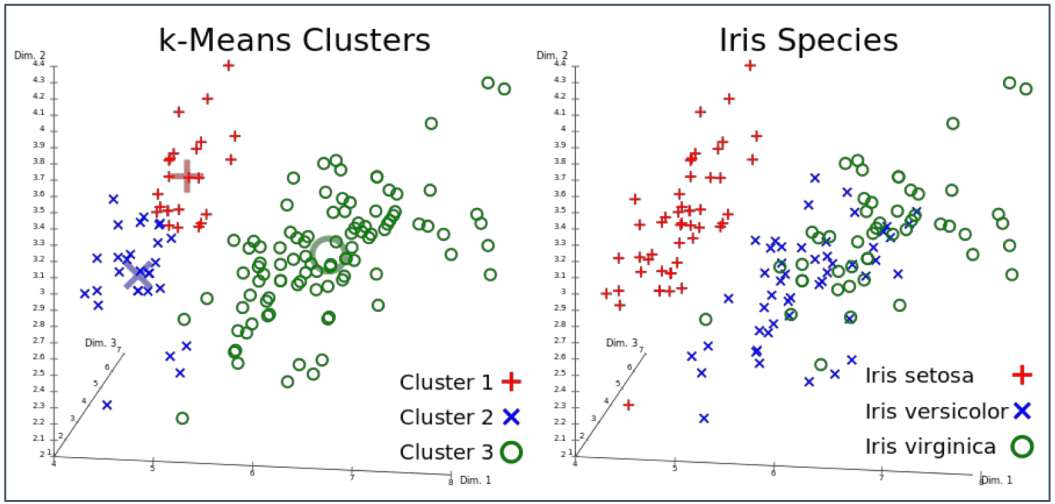
```
from sklearn.tree import DecisionTreeRegressor

tree1 = DecisionTreeRegressor(max_depth=2)
tree1.fit(X, y)

y2 = y - tree1.predict(X)
tree2 = DecisionTreeRegressor(max_depth=2)
tree2.fit(X, y2)

y3 = y2 - tree2.predict(X)
tree3 = DecisionTreeRegressor(max_depth=2)
tree3.fit(X, y3)

y_pred = sum(tree.predict(X_new) for tree in (tree1, tree2,
tree3))
```

Clustering

The more extensive a man's knowledge of what has been done, the greater will be his power of knowing what to do. - Benjamin Disraeli

K-Means Clustering

k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster



Common use cases of clustering

- Behavioural segmentation:
  - Segment by purchase history
  - Segment by activities on application, website, or platform
  - Define personas based on interests
  - Create profiles based on activity monitoring
- Sorting sensor measurements:
  - Detect activity types in motion sensors
  - Group images
  - Separate audio
  - Identify groups in health monitoring
- Inventory categorization:
  - Group inventory by sales activity
  - Group inventory by manufacturing metrics
- Detecting bots or anomalies:
  - Separate valid activity groups from bots
  - Group valid activity to clean up outlier detection

Measure clustering

Metrics	Description
Inertia	The within-cluster sum of squares criterion, can be recognized as a measure of how internally coherent clusters are
Silhouette Coefficient	Measures how well defined the clusters are.
Homogeneity, Completeness	Given the knowledge of the ground truth class assignments of the samples, it is possible to define some intuitive metric using conditional entropy analysis.

Dimensionality Reduction

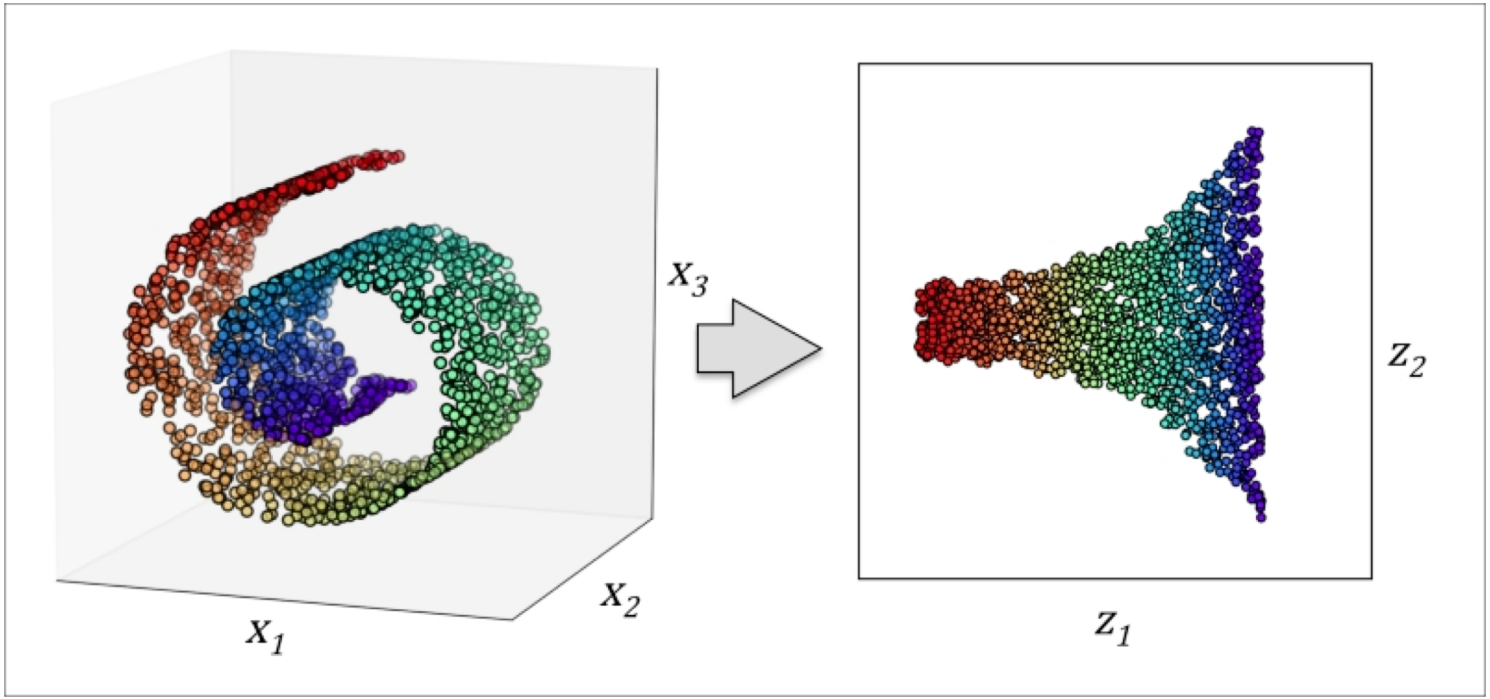
Curse of Dimensionality for KNN

- The curse of dimensionality describes the phenomenon where the feature space becomes sparse for an increasing number of dimensions of a fixed size training set.
- KNN is very susceptible to overfitting due to curse of dimensionality.
- While regularization is applicable for linear/logistic regression, it is not applicable for decision trees or KNN.
- We can use dimensionality reduction techniques to help avoid curse of dimensionality such as feature selections, PCA, SVD etc.

Imagine a 3 dimensional cube. What will be the fraction of the total area covered by cube if we shrink each dimension by 1%. What happens if the dimensions are 10, 100, or 1000.

0.99 ^ 3 = 0.97
0.99 ^ 10 = 0.90
0.99 ^ 100 = 0.37
0.99 ^ 1000 = 0.000043

Dimensionality Reduction

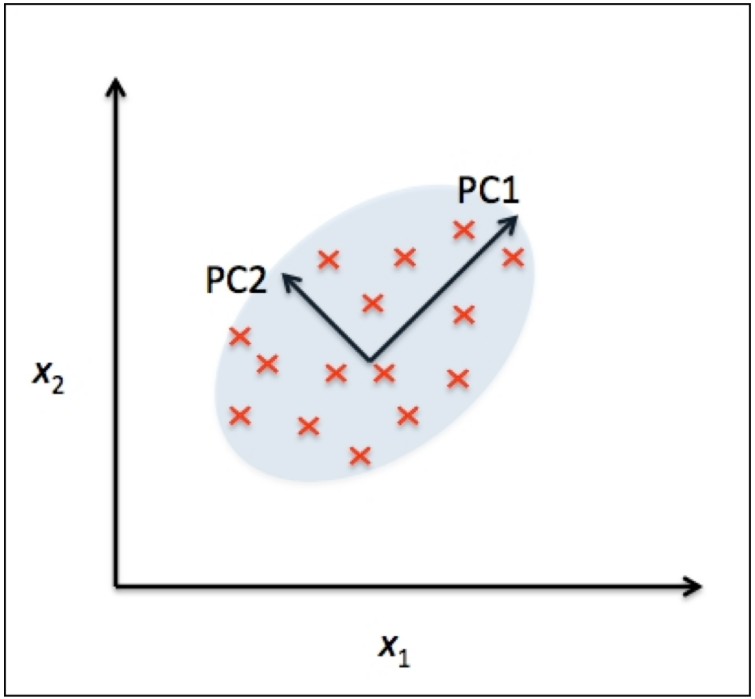


Feature Selection

- It is a process of selecting a best subset of features. Goals of feature selection are
  - reduce computational time and space by reducing the size of the problem
  - improve performance of models by removing noisy and irrelevant features and reduce likelihood of overfitting
- Lasso, Decision Tree, Random Forest models as byproduct gives out information to carry out feature selection
- Other common techniques are forward elimination and backward elimination process that looks at p-value and gradually removes features that is least significant

Principle Component Analysis

- PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions that the original one.
- The orthogonal axes (principal components) of the new subspace can be interpreted as the directions of maximum variance given the constraint that the new feature axes are orthogonal to each other.

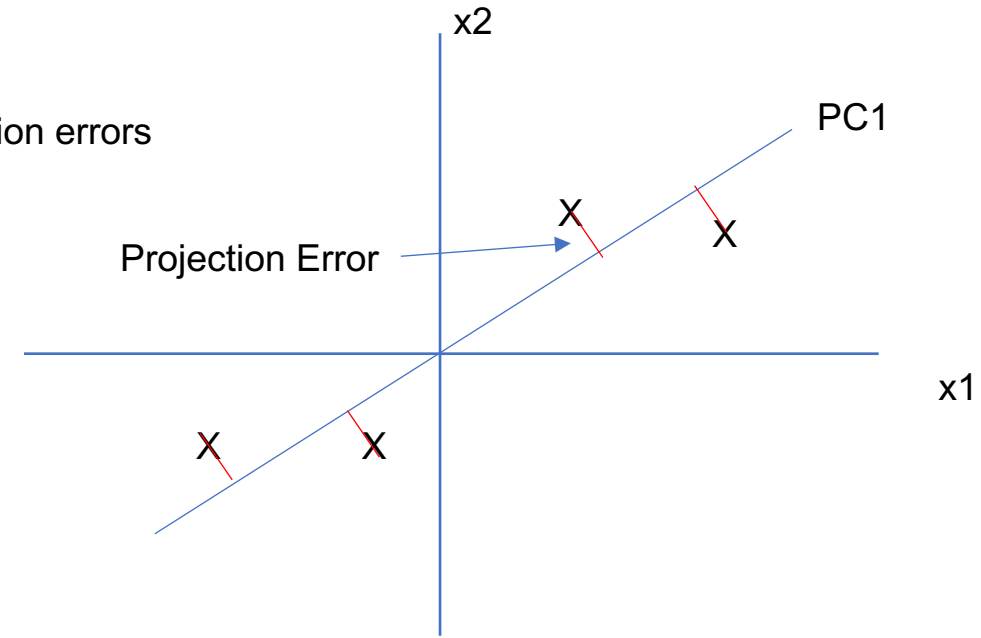


Principle Component Analysis

- Principal component analysis (PCA) is an unsupervised linear transformation technique that is widely used across different fields, most prominently for dimensionality reduction.
- Other popular applications of PCA include exploratory data analyses and de-noising of signals in stock market trading, and the analysis of genome data and gene expression levels in the field of bioinformatics.
- PCA helps us to identify patterns in data based on the correlation between features.
- Before you apply PCA, always consider mean normalization and even feature scaling (Z-scoring)

Problem formulation

Reduce sum of projection errors



Select p-Components

Average Squared Projection Error = $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)} - x_{approx}^{(i)}\right\|^2$

Total variation in data = $\frac{1}{m}\sum_{i=1}^{m}\left\|x^{(i)}\right\|^2$

Choose K so that, $\frac{average\ squared\ projection\ error}{Total\ variation} \leq 0.01$

Singular Value Decomposition

Any $n \times m$ matrix $\mathbf{A}$ can be written as

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T,$$

where

$$\begin{aligned} \mathbf{U} &= \text{eigenvectors of } \mathbf{A}\mathbf{A}^T && n \times n \\ \mathbf{D} &= \sqrt{\text{diag}(\text{eig}(\mathbf{A}\mathbf{A}^T))} && n \times m \\ \mathbf{V} &= \text{eigenvectors of } \mathbf{A}^T\mathbf{A} && m \times m \end{aligned}$$

The matrices U and V are both defined to be orthogonal matrices. That means, $UU^T = U^TU = I$. The matrix D is defined to be a diagonal matrix. The elements along the diagonal of D are known as the singular values of the matrix A - positive real numbers. The columns of U are known as the left-singular vectors. The columns of V are known as as the right-singular vectors.

Covariance matrix

$$\underbrace{cov(X)}_{\mathbb{R}^{mxm}} = \frac{1}{n} \underbrace{\big(X - E(X)\big)^T}_{\mathbb{R}^{mxn}} \underbrace{(X - E(X))}_{\mathbb{R}^{nxm}} \quad)$$

X: matrix where each column represents a feature.

Note: numpy.cov functions treats columns as records as rows an features

```
np.random.seed(1230)
X = np.random.ranf((5, 3))
X
```

```
array([[0.27702631, 0.36855193, 0.64431478],
       [0.78019793, 0.50860458, 0.52375554],
       [0.84079088, 0.36703687, 0.67039217],
       [0.83824478, 0.68695113, 0.10454645],
       [0.43739591, 0.312447  , 0.25789323]])
```

```
n = X.shape[0]
X0 = X - np.mean(X, axis = 0)
X0.T.dot(X0)/n
```

```
array([[ 0.05438659,  0.01918617, -0.00915189],
       [ 0.01918617,  0.01840192, -0.01705645],
       [-0.00915189, -0.01705645,  0.04950637]])
```

```
np.cov(X, ddof=0, rowvar=False)
```

```
array([[ 0.05438659,  0.01918617, -0.00915189],
       [ 0.01918617,  0.01840192, -0.01705645],
       [-0.00915189, -0.01705645,  0.04950637]])
```

PCA (…continue)

To reduce the dimension of A, we use k as the desired dimension, such that k ≤ n and take a subset of matrix U by taking first k columns of U, let's call it U_{reduce}.

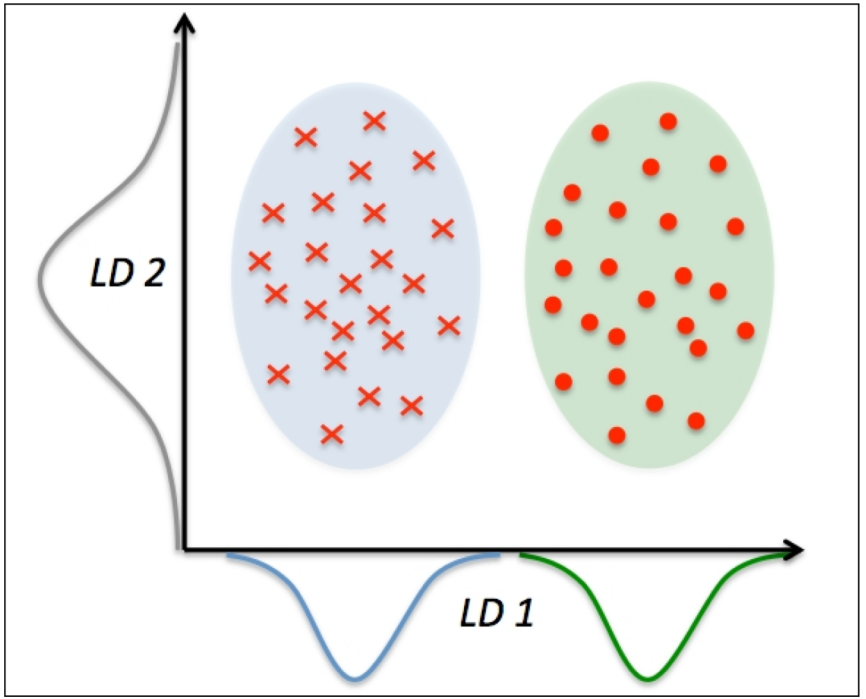
$$A_{pca} = U_{reduce}^{T} A$$

Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) can be used as a technique for feature extraction to increase the computational efficiency and reduce the degree of over-fitting due to the curse of dimensionality in non-regularized models.
- The general concept behind LDA is very similar to PCA, whereas PCA attempts to find the orthogonal component axes of maximum variance in a dataset; the goal in LDA is to find the feature subspace that optimizes class separability.
- Both LDA and PCA are linear transformation techniques that can be used to reduce the number of dimensions in a dataset; the former is an unsupervised algorithm, whereas the latter is supervised.

LDA

A linear discriminant, as shown on the *x*-axis (LD 1), would separate the two normally distributed classes well. Although the exemplary linear discriminant shown on the *y*-axis (LD 2) captures a lot of the variance in the dataset, it would fail as a good linear discriminant since it does not capture any of the class-discriminatory information.



Market Basket Analysis

Example of retail transactions

By looking at the sets of purchases, one can infer that there are a couple of typical buying patterns. A person visiting a sick friend or family member tends to buy a get well card and flowers, while visitors to new mothers tend to buy plush toy bears and balloons. Such patterns are notable because they appear frequently enough to catch our interest; we simply apply a bit of logic and subject matter experience to explain the rule.

#	**Purchased items**
1	*{flowers, get well card, soda}*
2	*{plush toy bear, flowers, balloons, candy bar}*
3	*{get well card, candy bar, flowers}*
4	*{plush toy bear, balloons, soda}*
5	*{flowers, get well card, soda}*

Association Rules

- Transactions are specified in terms of itemsets, such as the following transaction that might be found in a typical grocery store:

$$\{\text{bread, peanut butter, jelly}\}$$

- The result of a market basket analysis is a collection of **association rules** that specify patterns found in the relationships among items the itemsets.
- A rule identified from the example transaction might be expressed in the form:

$$\{\text{peanut butter, jelly}\} \rightarrow \{\text{bread}\}$$

Use of association rules

- Searching for interesting and frequently occurring patterns of DNA and protein sequences in cancer data
- Finding patterns of purchases or medical claims that occur in combination with fraudulent credit card or insurance use
- Identifying combinations of behavior that precede customers dropping their cellular phone service or upgrading their cable television package

Metrics for market basket analysis

Strength	Confidence	Lift
• The **support** of an itemset or rule measures how frequently it occurs in the data.	• A rule's **confidence** is a measurement of its predictive power or accuracy.	• The ratio by which by the confidence of a rule exceeds the expected confidence.

Support

- The **support** of an itemset or rule measures how frequently it occurs in the data. For instance the itemset *{get well card, flowers}*, has support of *3 / 5 = 0.6* in the hospital gift shop data.
- Similarly, the support for the association rule *{get well card}* → *{flowers}* is also 0.6.
- The support can be calculated for any itemset or even a single item; for instance, the support for *{candy bar}* is *2 / 5 = 0.4*, since candy bars appear in 40 percent of purchases.
- A function defining support for the itemset *X* can be defined as follows:

$$\text{support}(X) = \frac{\text{count}(X)}{N}$$

Confidence

- A rule's **confidence** is a measurement of its predictive power or accuracy. It is defined as the support of the itemset containing both *X* and *Y* divided by the support of the itemset containing only *X*:

$$\text{confidence}(X \rightarrow Y) = \frac{\text{support}(X, Y)}{\text{support}(X)}$$

- Essentially, the confidence tells us the proportion of transactions where the presence of item or itemset *X* results in the presence of item or itemset *Y*.
- The confidence that *X* leads to *Y* is not the same as the confidence that *Y* leads to *X*.

Confidence Example

- The confidence of *{flowers} → {get well card}* is *0.6 / 0.8 = 0.75*.
- The confidence of *{get well card} → {flowers}* is *0.6 / 0.6 = 1.0*.
- This means that a purchase involving flowers is accompanied by a purchase of a get well card 75 percent of the time, while a purchase of a get well card is associated with flowers 100 percent of the time
- Rules like *{get well card} → {flowers}* are known as **strong rules**, because they have both high support and confidence.
- Ideally, we would like to measure the support and confidence value for each possible itemsets from the retail database, and report back only those rules that meet certain levels of interest.

Lift

- The *lift* of a rule measures how much more likely one item or itemset is purchased relative to its typical rate of purchase, given that you know another item or itemset has been purchased.
- Lift (X -> y) = Confidence(X -> Y) / Support(Y)
- Lift greater than 1 suggests that the presence of the items on the LHS has increased the probability that items on the RHS will on this transactions.
- Lift is below 1 suggests presence of the items on the LHS make the probability that items on the RHS will be part of the transactions lower.
- Lift 1 indicates that items on the left and right are independent.

Strength and Weaknesses of Association Rules

Strengths

- Is capable of working with large amounts of transactional data
- Results in rules that are easy to understand
- Useful for "data mining" and discovering unexpected knowledge in databases

Weaknesses

- Not very helpful for small datasets
- Requires effort to separate the true insight from common sense
- Easy to draw spurious conclusions from random patterns

Apriori algorithm

- The downward closure property of support allows for efficient search and guarantees that for a frequent itemset, all of its subsets are also frequent.

Apriori algorithm

For an infrequent itemset, all of its superset must also be infrequent

Data model of transactions

Collection of receipts and items in each receipt

Transaction	Items
A	X
A	Y
B	X
B	Z
C	Y
C	Z
etc...	etc...

Transaction	Items
A0001	citrus fruit
A0001	margarine
A0001	ready soups
A0001	semi-finished bread
A0002	coffee
A0002	tropical fruit

Binary representation

Transaction	Items
A	X
A	Y
B	X
B	Z
C	Y
C	Z
etc...	etc...

Transactions	X	Y	Z	etc...
A	1	1	0	etc...
B	1	0	1	etc...
C	0	1	1	etc...
etc...	etc...	etc...	etc...	etc...

Binary representation

Transaction	Items
A0001	citrus fruit
A0001	margarine
A0001	ready soups
A0001	semi-finished bread
A0002	coffee
A0002	tropical fruit

Transaction	citrus fruit	margarine	ready soups	semi finished bread	coffee	tropical fruit
A0001	1	1	1	1	0	0
A0002	0	0	0	0	1	1
A0003	0	0	0	0	0	0
A0004	0	0	0	0	0	0

Thresholds

- The market basket analysis requires setting a threshold for detecting patterns in the dataset.
- For example, we specified value of 0.001 (due to high volume of receipts and large product offering) and a confidence level of 0.70
- We set length of rule not to exceed three elements. This ensures that we will have a max 2 items on the LHS and the assessment will produce more meaningful insights at a tertiary glance.

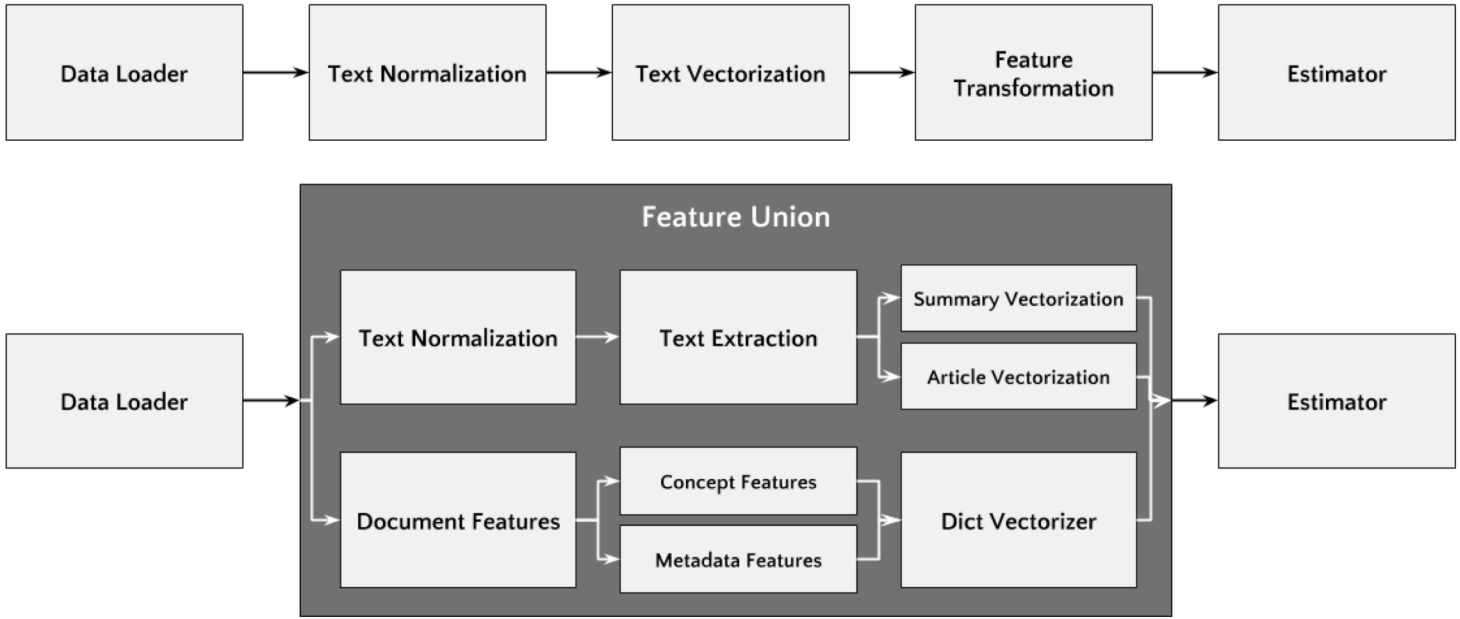
Rules	Support	Confidence	Lift
{liquor, red/blush wine} => {bottled beer}	0.002	0.90	11.24
{cereals, yogurt} => {whole milk}	0.002	0.81	3.17
{butter, jam} => {whole milk}	0.001	0.83	3.26
{chocolate, pickled vegetables} => {whole milk}	0.001	0.86	3.35
{grapes, onions} => {other vegetables}	0.001	0.92	4.74
{hard cheese, oil} => {other vegetables}	0.001	0.92	4.74

Text Analysis

Text Mining

- Its typical tasks are: text categorization, document clustering and organization, and information extraction

Text Analysis



Stemming vs Lemmatization

Word	Stemming	Lemmatization
gardener	garden	gardener
running	run	run
threw	threw	throw
foxes	fox	fox
geese	geese	goose

Algorithm that does not requires language dictionary

Requires language dictionary

Stemmer

- Porter Stemmer
- LancasterStemmer
- SnowballStemmer

TF-IDF

- Term frequency $$\mathrm{tf}(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}}$$
- Inverse Document Frequency $$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$
- tf–idf $$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

D: Corpus
N: total no of documents i.e. size of the corpus D

TF-IDF (Lucene)

$$tf = \sqrt{f_{t,d}}$$

$$idf = 1 + \log\left(\frac{numDocs}{f_{t,d} + 1}\right)$$

$$queryNorm = \frac{1}{\sqrt{\sum w_{t \in q}{}^2}}$$

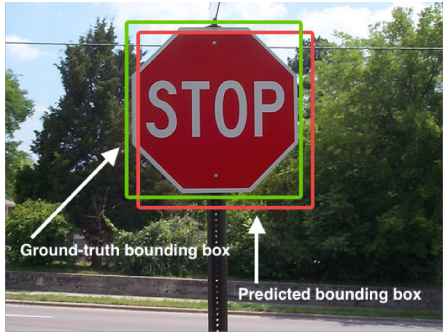
$$and(A, B) = A * B$$

$$or(A, B) = A + B$$

$\max(A, B) = \max(A, B)$ - implemented with Disjunction Max Query

$$score(p, q) = coor(p, d).queryNorm(q) \sum_{t \in q} tf(t \in d).idf(t)^2 \,.t.getBoost.norm(t, d)$$

Similarity Matching



$$\text{Cosine}\left(\underline{x},\underline{y}\right) := \frac{\underline{x}\underline{y}^{T}}{\left\|\underline{x}\right\| \cdot \left\|\underline{y}\right\|}$$

Ranges from 0 to 1. The smaller the angle the greater the similarity between the argument vectors is. Vectors x and y are binary representations respectively for set A and B.

$$Overlap\left(A,B\right) := \frac{\left|A \cap B\right|}{\min\left(\left|A\right|,\left|B\right|\right)}$$

$$Jaccard\left(A,B\right) := \frac{\left|A \cap B\right|}{\left|A \cup B\right|}$$

Commonly used in information retrieval to measure the overlap between two sets. Ranges from 0 to 1

Word2vec - derivative of loss function

$$\frac{\partial J}{\partial v_c} = u_o - \sum_{x=1}^{V} p(x \mid c)\, u_x$$

Hidden Markov Model

- It is a statistical Markov model in which the system being modeled is assumed to be a Markov chain with hidden states.
- An HMM can be presented as the simplest dynamic Bayesian network.
- In a simple Markov models the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.
- In a hidden Markov model, the state is not directly visible, but the output, dependent on the state, is visible.
- Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states.

Hidden Markov Model

- Hidden Markov models are useful in the temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics.
- In hidden Markov model based face recognition and detection method a frontal face including the significant facial regions such as hair, forehead, eyes, nose and mouth occur in a natural order from top to bottom. So, each of these facial regions is assigned to a state in a left-to right one dimensional continuous hidden Markov model

Time Series

AR models: $x_t = \sum_{i=i}^{p} \alpha_i x_{t-i} + z_t$, z_t is the residual error term

ARMA model: $x_t = \sum_{i=i}^{p} \alpha_i x_{t-i} + \sum_{i=1}^{q} \beta_i z_{t-i}$

ARMA (parameterized by p, q) model assumes the data is stationary.

ARIMA: apply ARIMA on non-stationary data with additional parameter, d that indicates the amount of lag to make the data stationary.

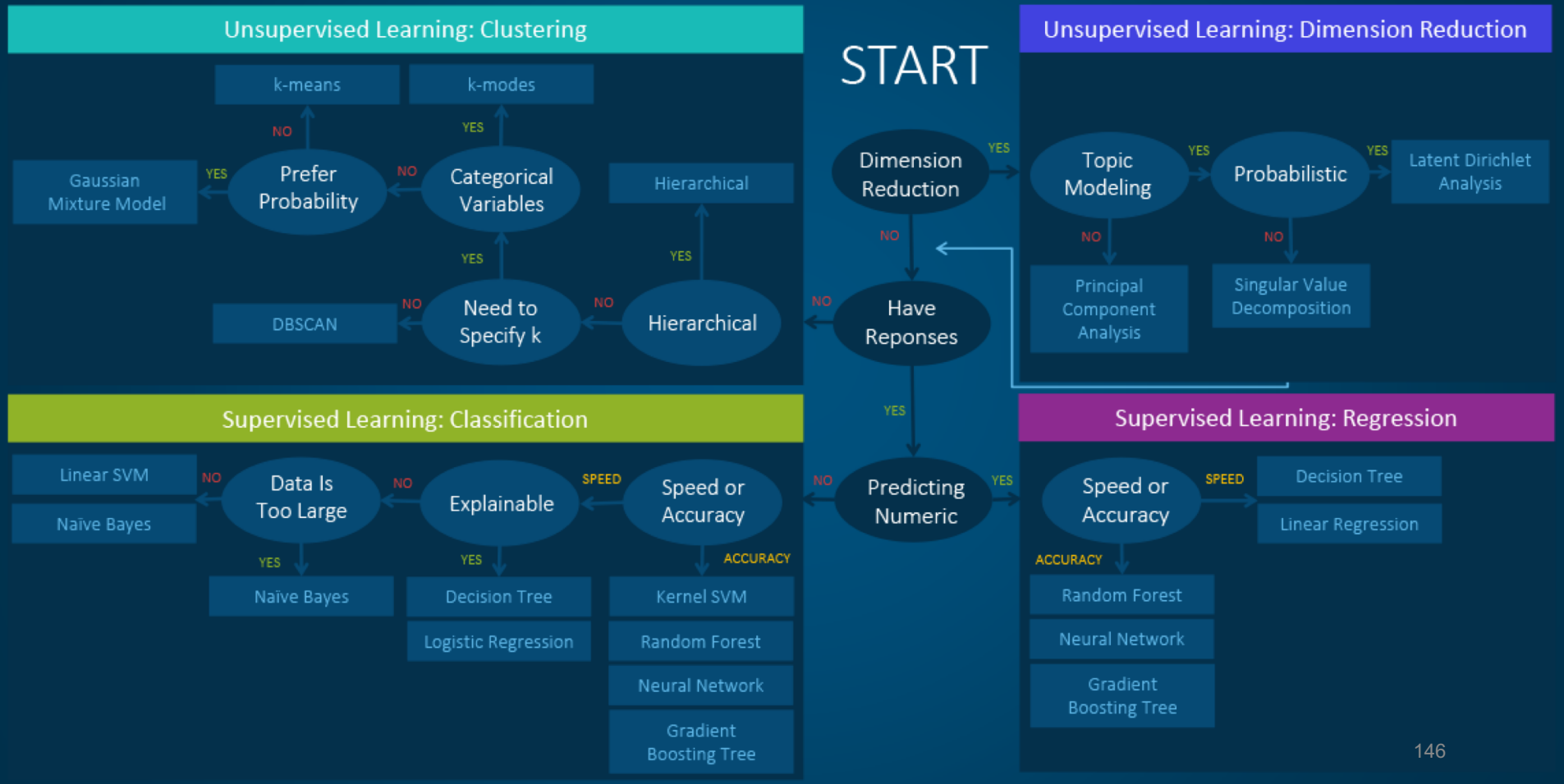
Find the value of p: PACF plot (where the curve cross 0 line first time)
Find the value of q: ACF plot (where the curve cross the upper boundary for first)
Find the value of d: Dickey-Fuller test

Feature Engineering

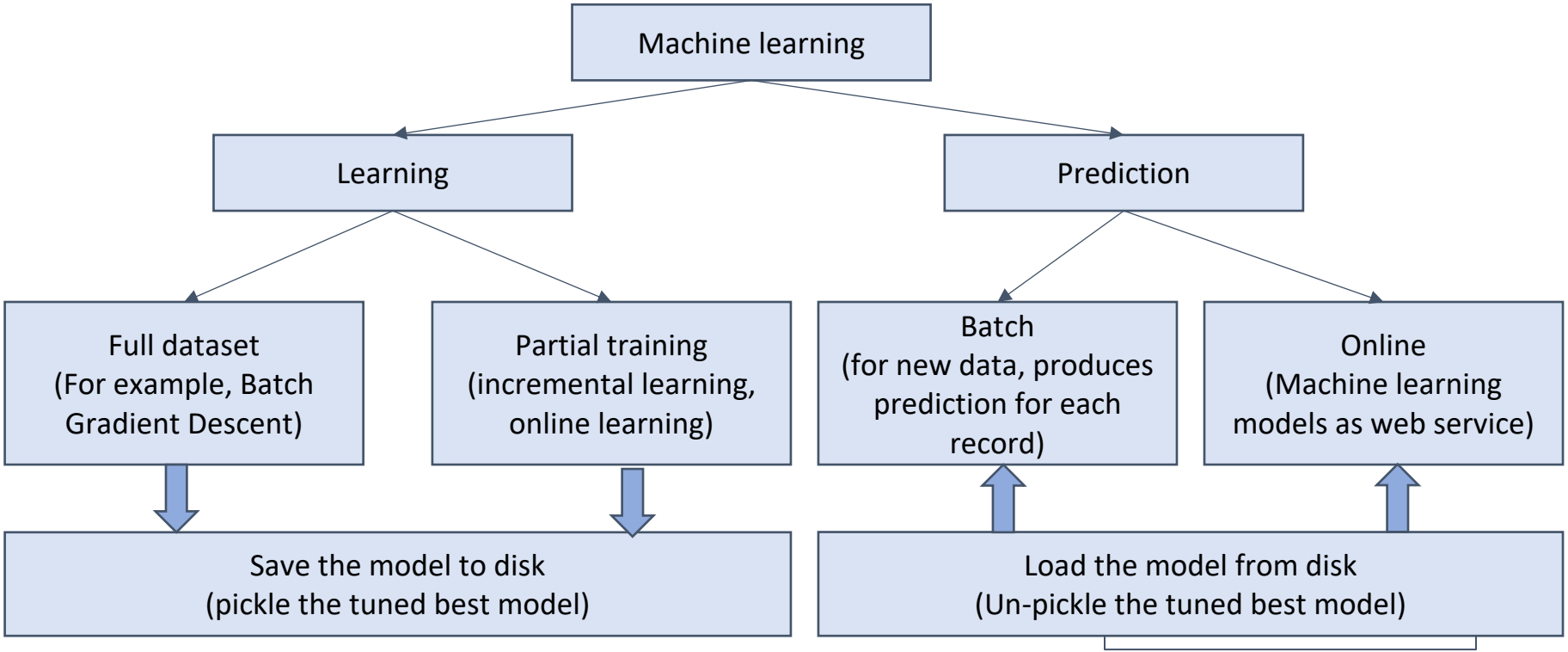
What makes good feature?

- Should be related to the objective
- Should be known at production time
- Has to be numeric with meaningful magnitude
- Has enough examples
- Brings human insight to the problem

Machine Learning Algorithms Cheat Sheet



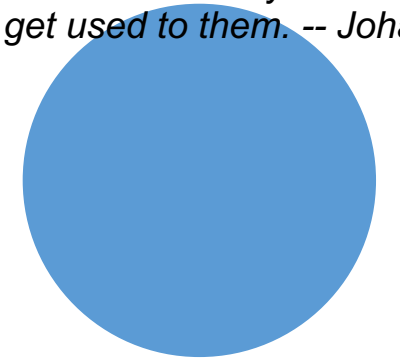
Deployment Strategy

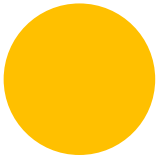


Challenges with machine learning in practice

- Training speed
  - Data volume
  - Number of iterations
  - Wall clock time
  - Cost
  - CPU seconds, RAM seconds, GPU seconds
  - Incremental training, retraining, online
- Prediction time
- Performance metrics
- Max memory requirement
- CPU-seconds
- CPU architecture
- Data volume
- Data formats
- Inference service
  - Batch prediction
  - Online prediction
- Model portability
  - Device portability
  - Language portability
- Formation of training examples
- Confidence on training examples
- Training and serving skew
- Legal and ethical acceptance of the model

In mathematics you don't understand things. You just get used to them. -- Johann von Neumann





Libraries for machine learning in Python

Numpy

- Offers capabilities similar to MATLAB within Python
- N-dimensional homogeneous arrays (ndarray)
- Universal functions (ufunc)- basic math, linear algebra, FFT, PRNGs
- Simple data file I/O - text, raw binary, naïve binary
- Tools for integrating with C/C++/Fortran
- Heavy lifting done by optimized C/Fortran libraries

SciPy

- Large library for scientific algorithms
- Extends NumPy with many tools for science and engineering
- Computationally intensive routines implemented in C and Fortran
- Packages:

Special Functions
Signal Processing
Fourier Transformation
Optimization
Numerical Integration

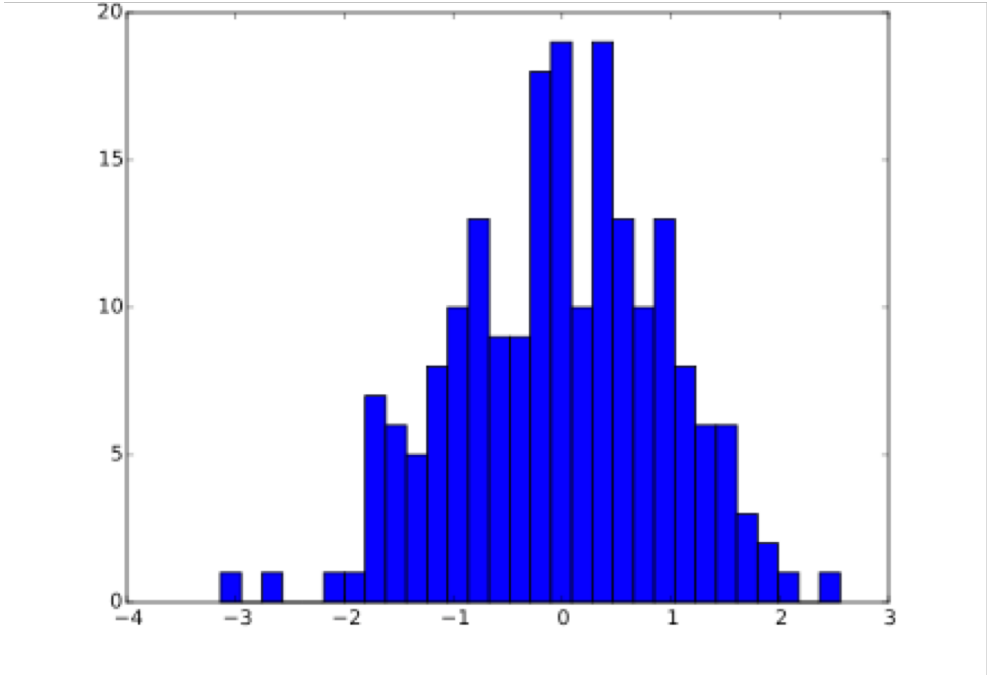
Linear Algebra
Input output
Statistics
Fast Execution
Cluster Algorithms
Space Matrices

Pandas

- In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis.
- It is free software released under the three-clause BSD license.
- First released in 2008
- DataFrame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.

Matplotlib

- Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- Pyplot is a matplotlib module which provides a MATLAB-like interface



Scikit Learn

- Scikit-learn is a free software machine learning library for the Python programming language released under BSD license
- Started as a Google Summer of Code project by David Cournapeau
- First released in 2007

Miscellaneous

Science is about knowing; engineering is about doing. - Henry Petroski

Statistics vs Machine Learning

Statistics

- Introduction to Data
- Probability
- Distribution of Random Variables
- Foundation of Inference
- Inference of numerical data
- Inference for categorical data
- Introduction to linear regression
- Multiple and logistic regression

Machine Learning

- Supervised Learning, Discriminative Algorithms
- Generative Algorithms
- Support Vector Machines
- Learning Theory
- Regularization and Model Selection
- Online Learning and the Perceptron Algorithm
- Unsupervised Learning, k-means clustering.
- Mixture of Gaussians
- The EM Algorithm
- Factor Analysis
- Principal Components Analysis
- Independent Components Analysis
- Reinforcement Learning and Control

Deep Learning

- Linear Algebra
- Probability and Information Theory
- Numerical Computation
- Machine Learning Basic
- Deep Feedforward Networks
- Regularization for Deep Learning
- Optimization for Training Deep Models
- Convolutional Networks
- Sequence Modeling - Recurrent Nets
- Linear Factor Models
- Auto Encoders
- Representation Learning
- Structural Probabilistic Models
- Monte Carlo Methods
- Confronting Partition Function
- Approximate Inference
- Deep Generative Models

Statistics

- Statistics are numerically expressed facts related to some field.
- It presents large quantity of data in a simple and classified form.
- It allows comparison of data, find relationships between variables
- It guides business and government in planning policies and programs.

Limitations

- Analyzes only collective matters not individual events
- Only applicable to quantitative data
- Suffers from selection bias. If there is a bias sampling, results are not reliable

A case for SAS

- Open source projects run a risk of unregulated contributions without necessary quality assurance or extensive prior testing. In heavily regulated environment such as Credit Risk (Basel accord), Insurance (Solvency Accord) and Pharmaceuticals (Food and Drug administration regulator), analytical models are subject to external supervisory review because of their strategic impact to the society.
- SAS offers an end to end solution - framing business problem -> data preprocessing -> analytics model development -> back testing and benchmarking -> stress testing -> regulatory capital calculation.
- Commercial support

What is the position of AI

Artificial intelligence technologies include

Components	Description
Logic programming	using facts and rules to create logical formulas
Bayesian systems	using statistics and probabilities for prediction
Expert systems	using knowledge-based systems for decision making Semantic knowledge bases: understanding language sets, identifying content by its type, meaning, metadata or tagging
Deep learning	using algorithms and artificial neural networks to create models of high-level abstractions. Deep learning comes from the use of multiple layers of networks that are able to improve with exercise (i.e. iteration)

Cognitive Services

Cognitive Services enable your apps, websites, bots to see, hear, speak, understand and interpret customer needs through natural methods of communication.

Vision

Object detection
Object localization
Segmentation
Annotation
Caption

Knowledge

Recommendation
Semantic Search

Language

Translate
Sentiment
Intent

Speech

Text to speech
Speech to text
Voice verification
Speaker recognition

Search

Search by text, voice, image

“If they knew for sure what they have to do, how to do it, and what resources they need for it, enterprises would be able to provision large amounts of resources. But the reality is that you have to experiment. It’s not about how much you can invest, it’s how quickly you can iterate.”