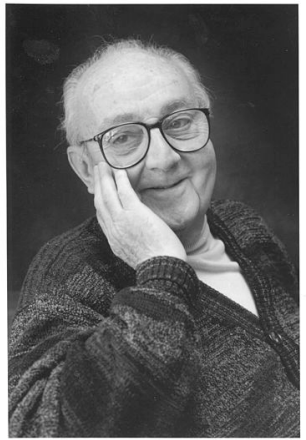
Machine Learning

Overview

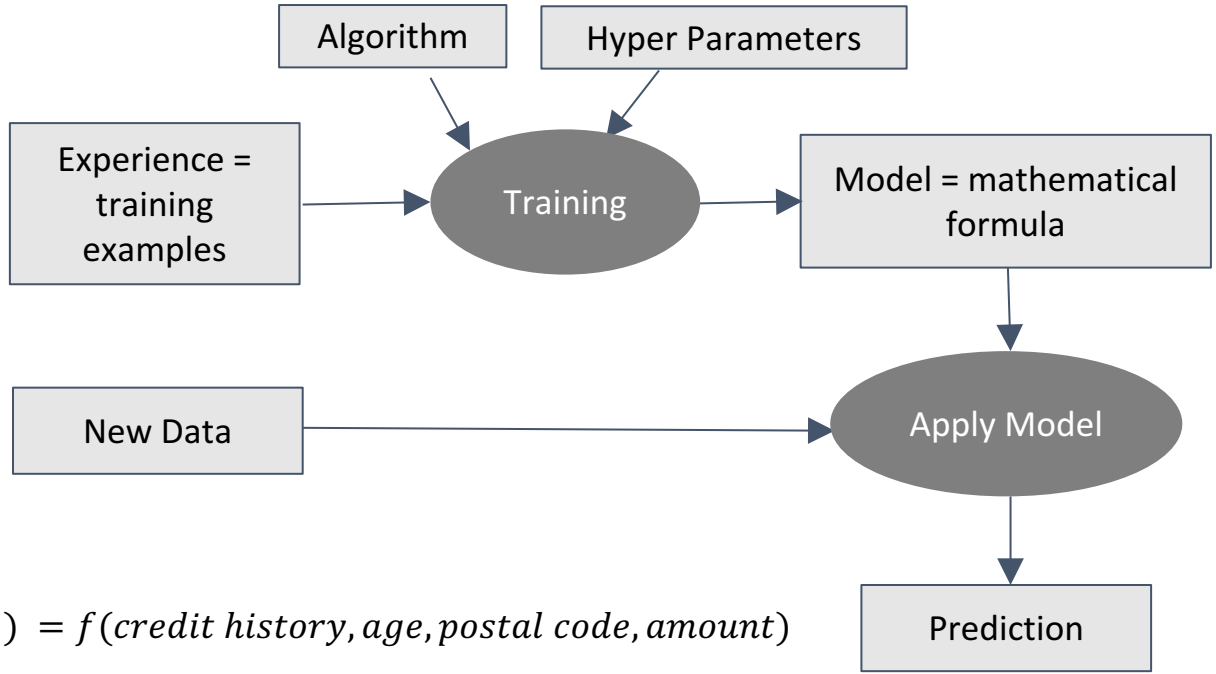
“All models are wrong, some are useful” - George Box.



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?

	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

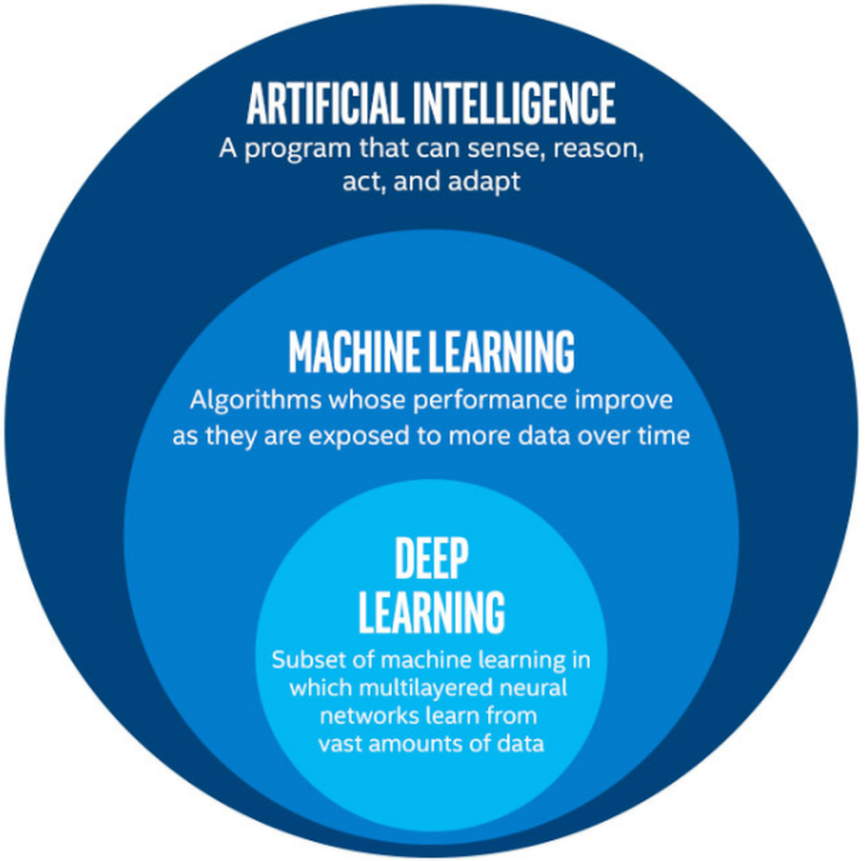
$$probability(admit = 1) = \frac{1}{1 + e^{-(-3.45 + 0.002\ gre + 0.78\ gpa - 0.56\ rank)}}$$

https://stats.idre.ucla.edu/stat/data/binary.csv

Types of machine learning

Supervised learning	Unsupervised learning	Recommender System	Reinforcement learning
• Classification/regression • Spam detection • Image detection • Object detection • Segmentation • Image captioning	• Clustering • Dimensionality reduction • Feature learning • Density estimate	• Item2item collaboration • User2user collaboration • ALS	• Real time decision • Robotic navigation • Game AI

AI, machine learning, deep learning



Why machine learning

Doing tasks better than human	Complete a task in fraction of time what human would take	Make cost effective solutions	Make predictions
• Self driving a car • Product recommendation • Predict user clicks on digital advertisement • Predicting transit time	• Medical diagnosis by looking medical images • Take decision on loan approvals • Translate text from one language to another or from 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

Use cases by industry

Manufacturing

- Predictive maintenance or condition monitoring
- Warranty reserve estimation
- Propensity to buy
- Demand forecasting
- Process optimization
- Telematics

Retail

- Predictive inventory planning
- Recommendation engines
- Upsell and cross-channel marketing
- Market segmentation and targeting
- Customer ROI and lifetime value

Healthcare and Life Sciences

- Alerts and diagnostics from real-time patient data
- Disease identification and risk satisfaction
- Patient triage optimization
- Proactive health management
- Healthcare provider sentiment analysis

Travel and Hospitality

- Aircraft scheduling
- Dynamic pricing
- Social media—consumer feedback and interaction analysis
- Customer complaint resolution
- Traffic patterns and congestion management

Financial Services

- Risk analytics and regulation
- Customer Segmentation
- Cross-selling and upselling
- Sales and marketing campaign management
- Credit worthiness evaluation

Energy, Feedstock and Utilities

- Power usage analytics
- Seismic data processing
- Carbon emissions and trading
- Customer-specific pricing
- Smart grid management
- Energy demand and supply optimization

Quiz: ML use cases

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 the 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

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.

$$f(\mathbf{X}) = \beta_0 + X_1 \times \beta_1 + X_2 \times \beta_2 + \ldots$$

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 equation is known as normal equation

Measure performance

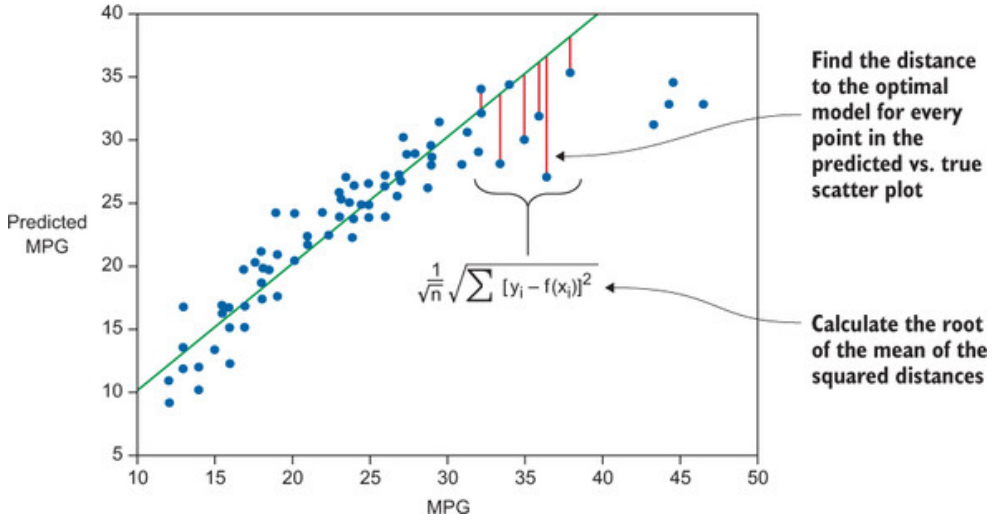
One way to compute the performance is to measure mean of squared errors (MSE) of the model on test dataset. Goal will be to minimize this error.

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

Other matrices
- R2
- Mean Absolute Error (MAE)

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.



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

Case for Gradient Descent

Finding thetas using normal equation is not feasible in practice because

- 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 is low (n < 1000 or n < 10000) normal equation is a viable option

Gradient overcome these limitations and offers some advantages as below

- 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.

LMS Algorithm

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

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

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

Summation is ignored for simplicity

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

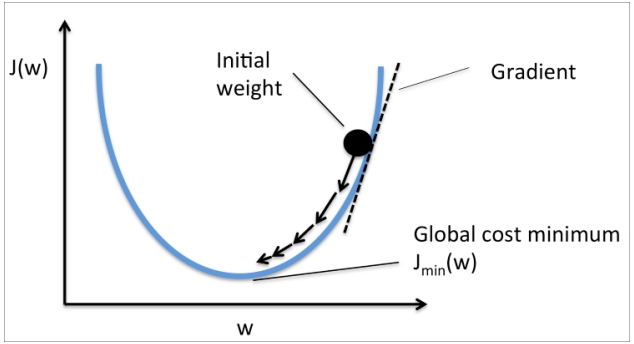
Gradient Descent Algorithm for OLS (ordinary least square)

repeat until convergence {

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

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

}

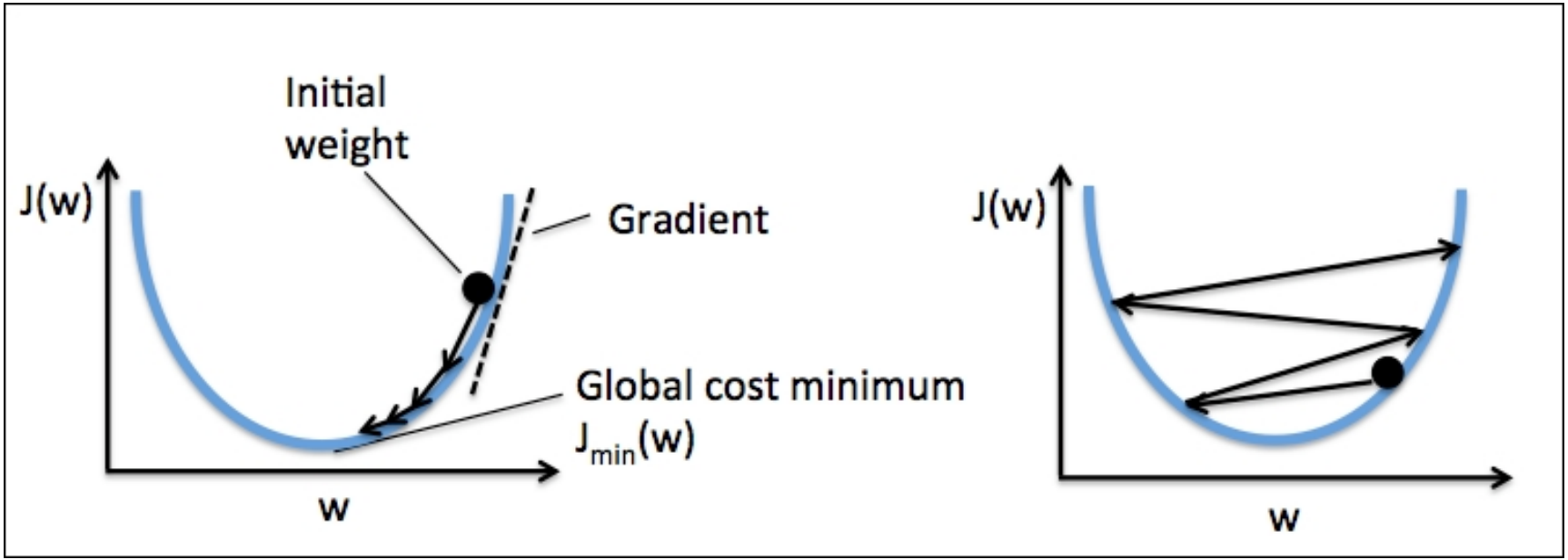


The quadratic cost function always converges for a small α value. Convergence to a local minimum can be guaranteed.

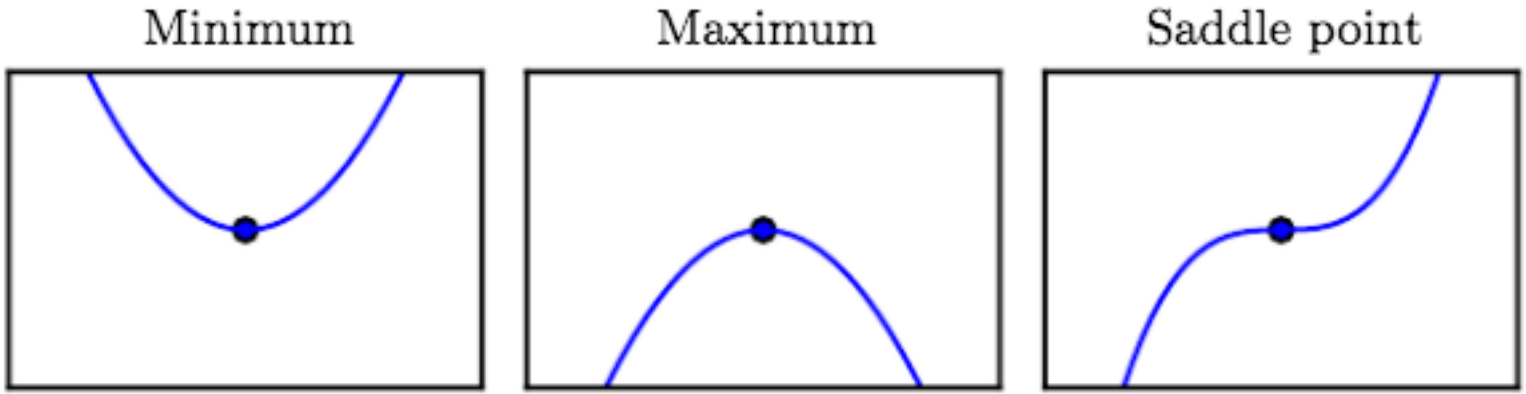
Learning Rate

- 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.

Impact of learning rate in search for global minima



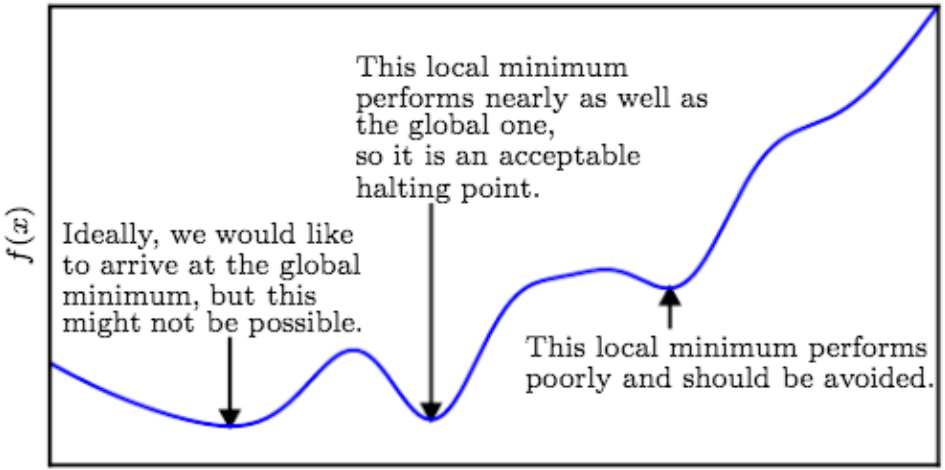
Critical Points in Gradient Descent



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

Approximate minimization. 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



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 recomputes 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.

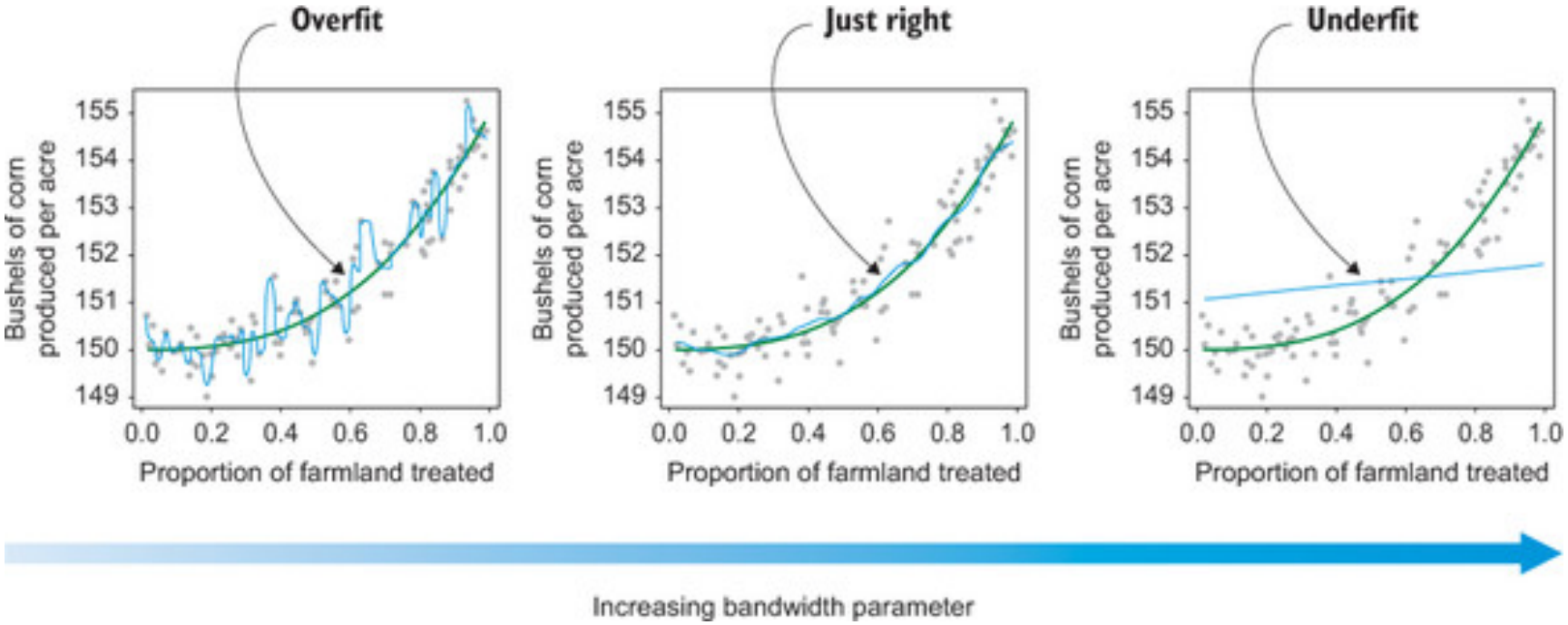
Challenges with Gradient Descent

- 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.

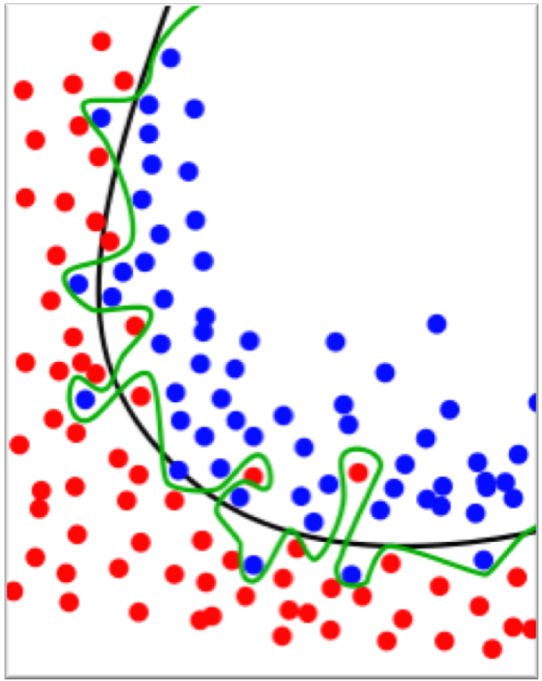
Gradient descent optimization algorithms

- Momentum
- Nesterov accelerated gradient
- Adagrad
- Adadelta
- RMSprop
- Adaptive Moment Estimation (Adam)
- AdaMax
- Nadam

Overfitting, Underfitting and Justright



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.
- Overfitted model has poor predictive performance, as it overreacts to minor fluctuations in the training data.

Occam’s razor - law of parsimony

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

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.

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

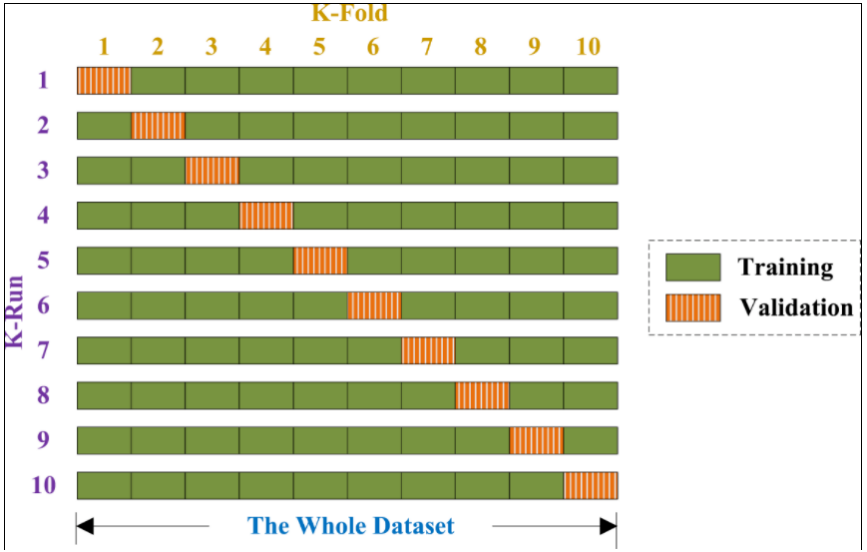
Model without regularization

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

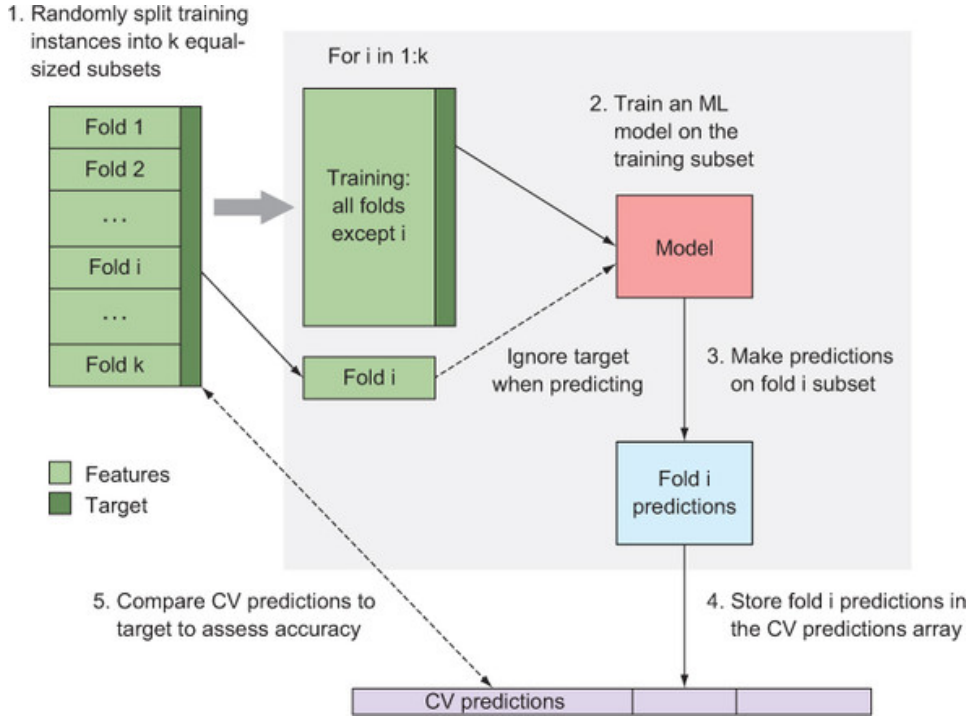
Model with L2 regularization

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.

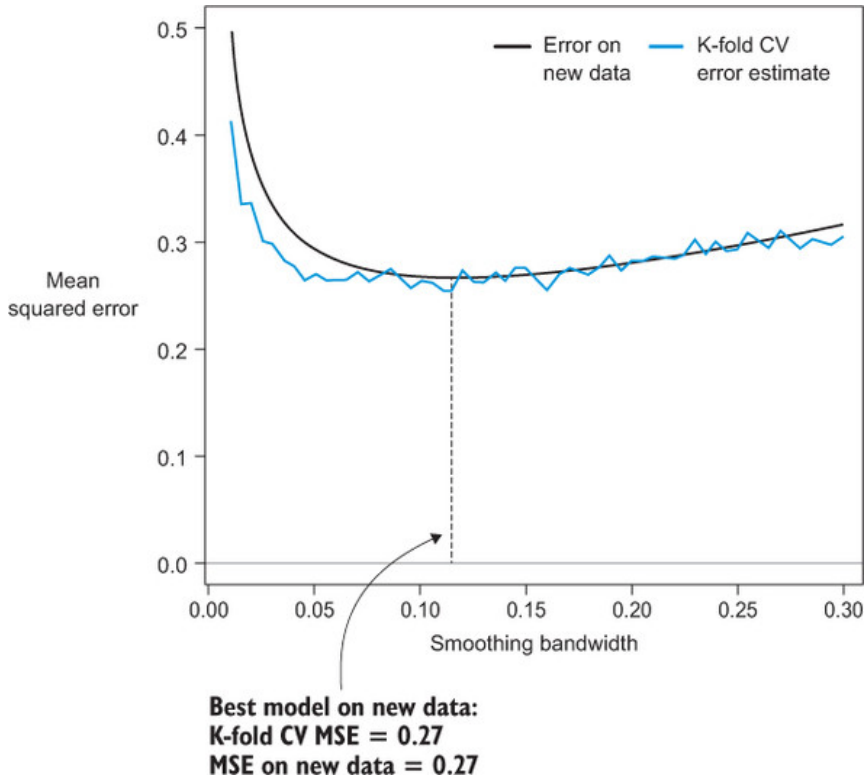


CV Model


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
Features
Target
5. Compare CV predictions to target to assess accuracy
4. Store fold i predictions in the CV predictions array
CV predictions

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.

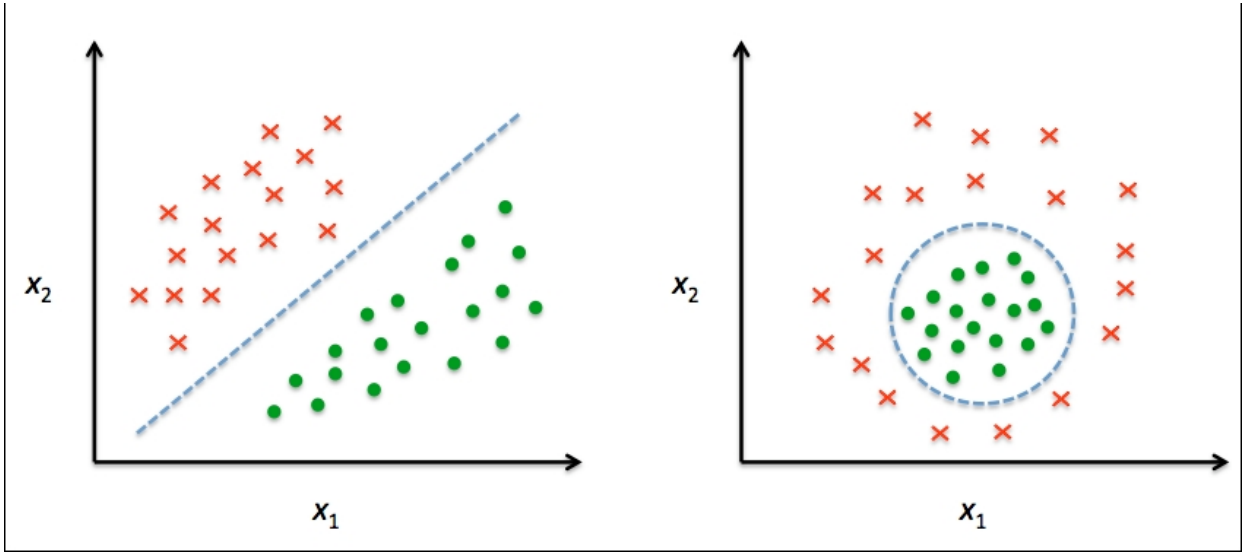


Classification

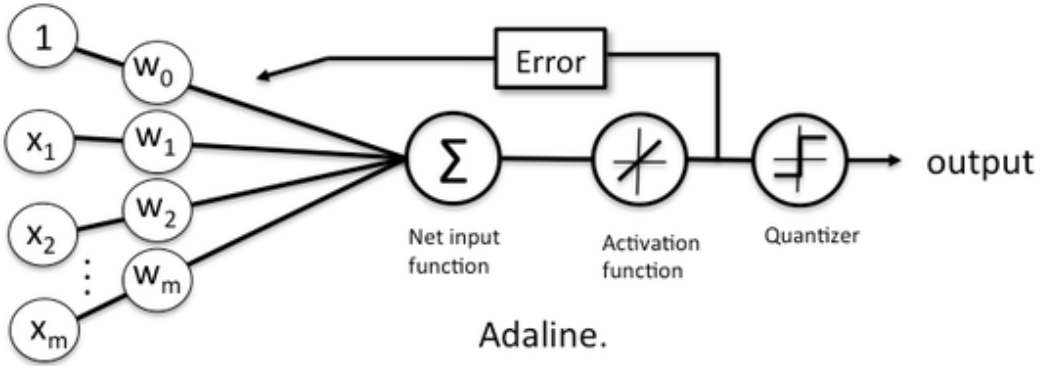
Classification

- 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

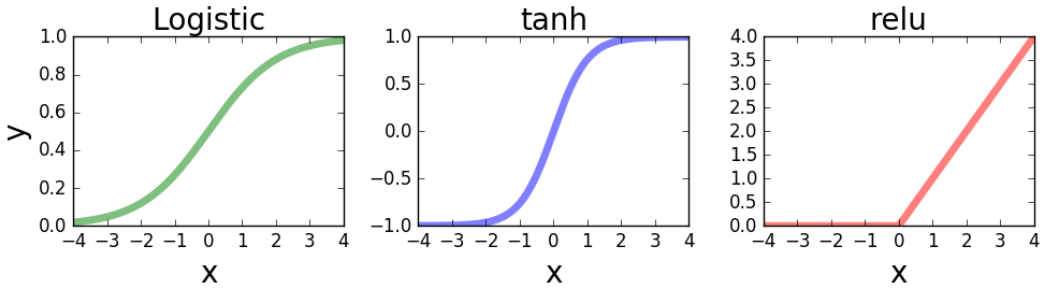


Adaline in classification problems



Adaptive Linear Neuron

Examples of activation function



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

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$$relu(x) = if(x > 0)\ x\ else\ 0$$

rectified linear unit

Logistic Regression

Probability estimate

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

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

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

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.

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

Sampling method for gradient descent

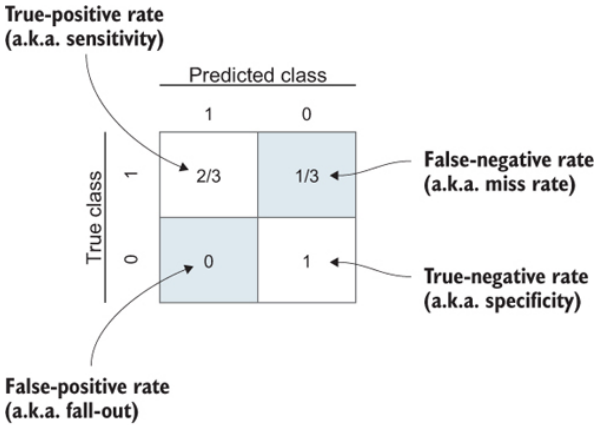
- The gradient of the objective is computed with respect to the model parameters and evaluated using a given set of input samples.
- How many of the samples should we take for this calculation? Intuitively, it makes sense to calculate the gradient for the entire set of samples. This method, however, 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.

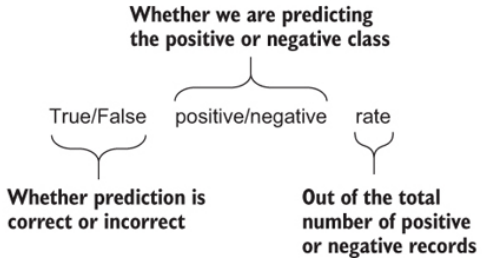
Evaluation of Classification

Accuracy: (TP+TN) / (P + N)
Precision: TP / (TP + FP)
TPR Or Recall Or Sensitivity: TP / P
Specificity: TN / N

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

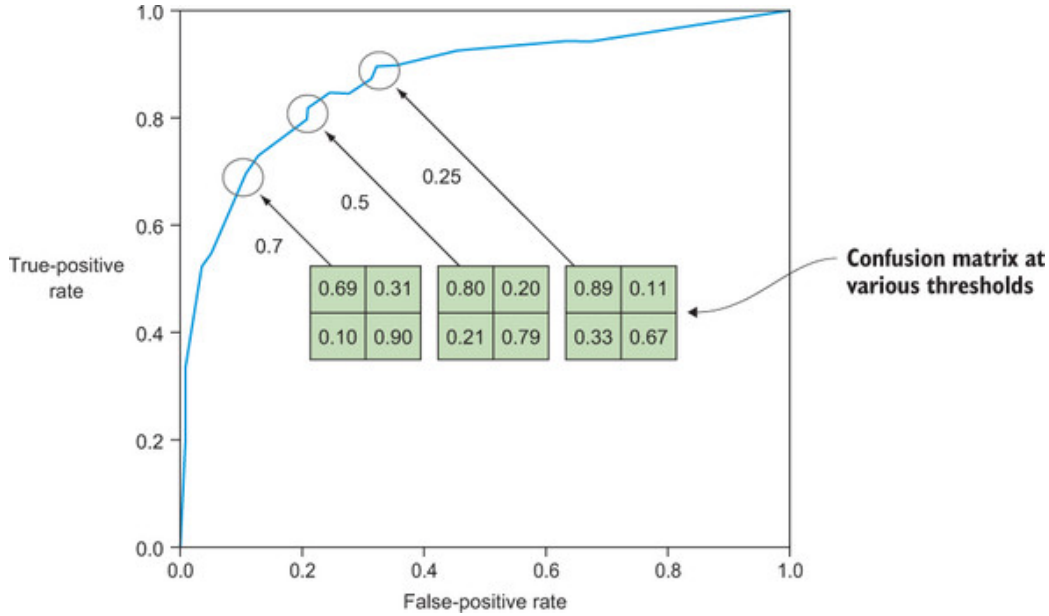
Receiver Operating Characteristics (ROC)





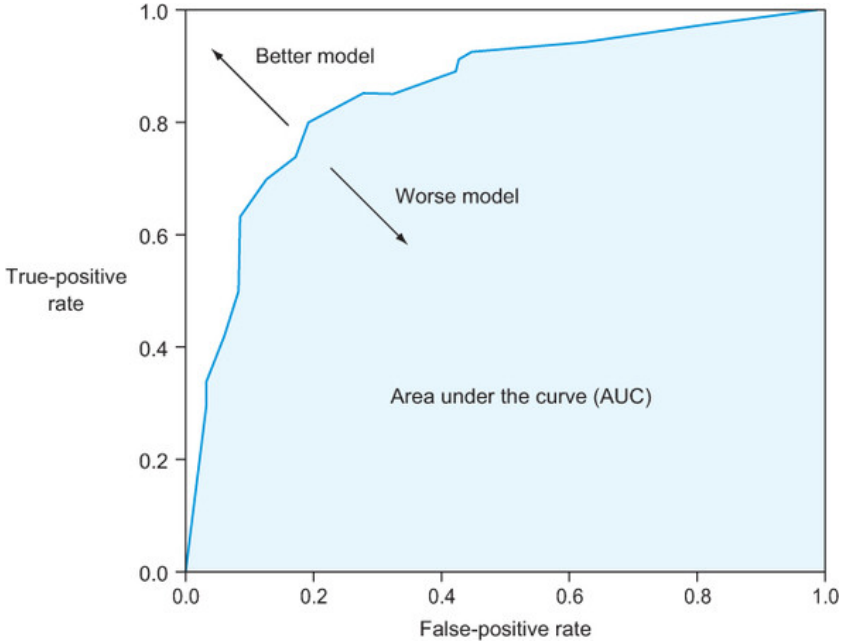
ROCR Curve

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.



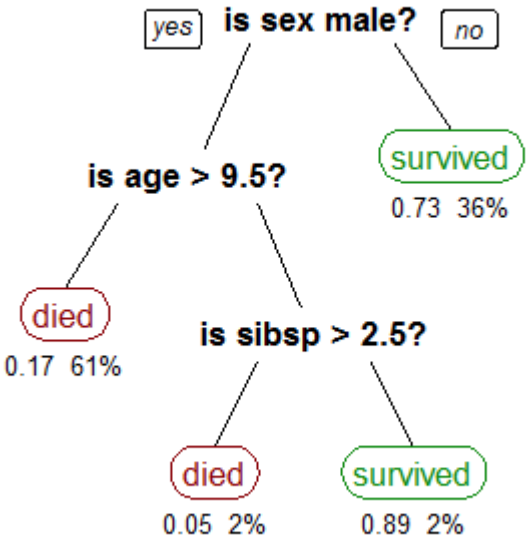
AUC Score

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



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

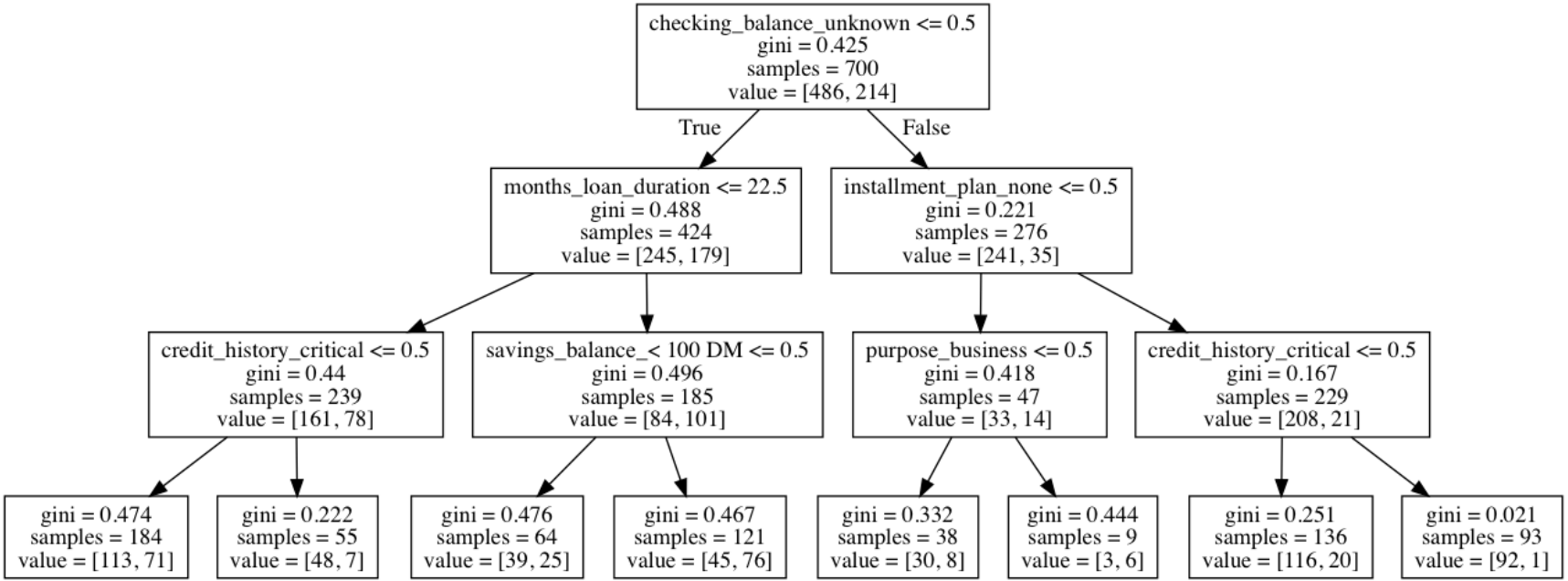
- To maximize interpretability these classifiers are expressed as decision trees or rule sets (IF-THEN), forms that are generally easier to understand than neural networks.
- Decision tree based classifiers are easy to use and does not presume any special knowledge of statistics or machine learning.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

Gini impurity

- 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.

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

Decision tree example



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

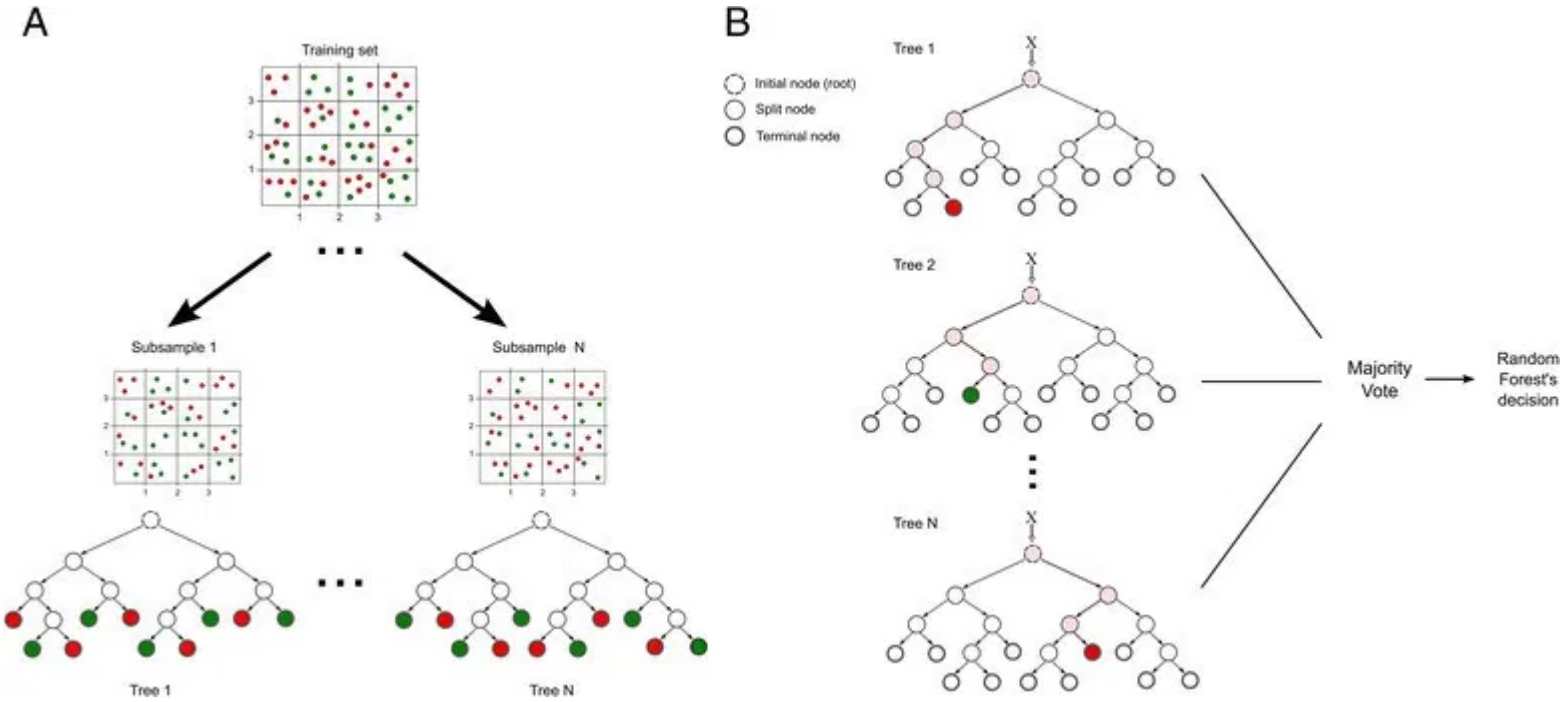
Weaknesses:

- 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



Random Subspace Method

- Random subspace is an ensemble classifier that consists of several classifiers and outputs the class based on the outputs of these individual classifiers.
- Random subspace method has been used for linear classifiers, support vector machines, nearest neighbors and other types of classifiers.
- This method is also applicable to one-class classifiers (... tries to identify objects of a specific class amongst all objects, commonly used in outlier detection, anomaly detection, novelty detection).
- It is an attractive choice for classification problems where the number of features is much larger than the number of training objects.

Random Subspace Method - Algorithm

- 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.

Training Test Split

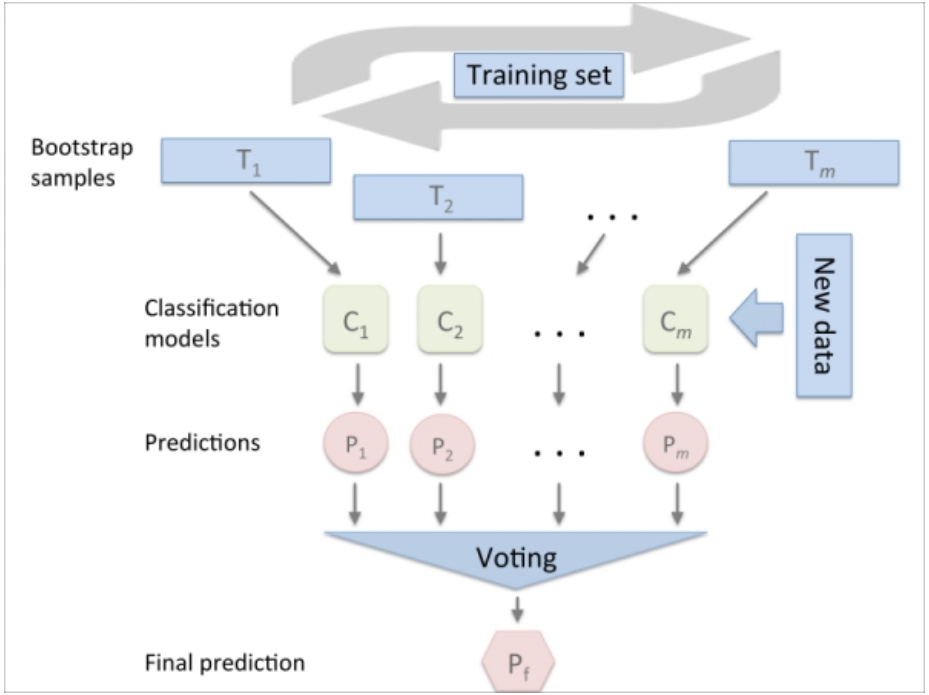
	Training Set - build model	Validation/Dev Set - use for model selection	Test Set- test final model to get confidence on the model
Small Data	80%	10%	10%
Big Data (1m+)	98%	1%	1%

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

Bagging

- 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.



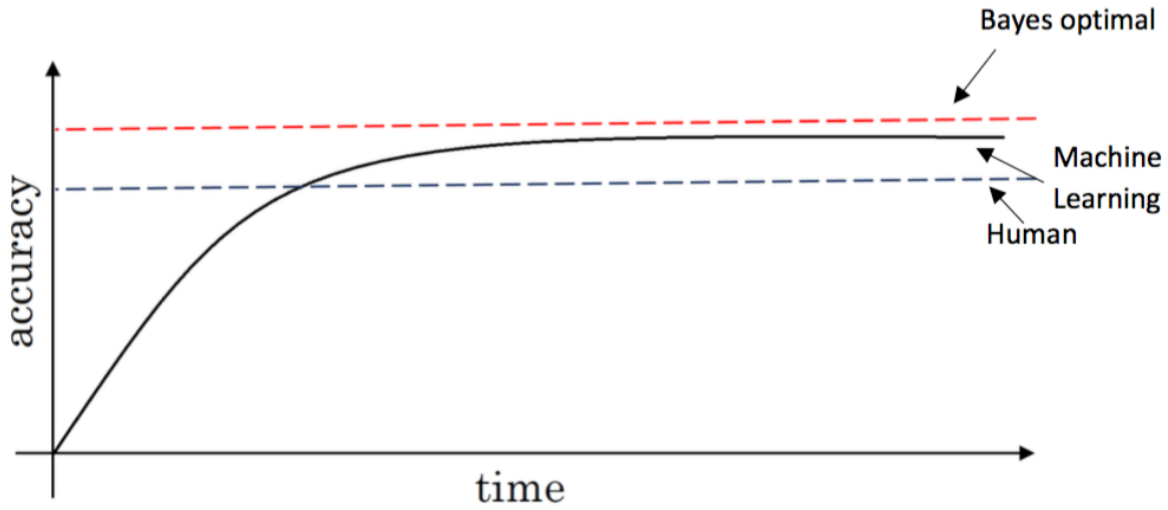
Bagging

- 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.

Boosting

- Boosting is similar to bagging except that one keeps track of the performance of the learning algorithm and forces it to concentrate its efforts on instances that have not been correctly learned.
- Instead of selecting the ‘n’ training instances randomly using a uniform distribution, one chooses the training instances in such a manner as to favors the instances that have not been accurately learned.
- After several cycles the prediction is performed by taking a weighted vote of the predictions of each classifier with the weights being proportional to each classifier’s accuracy on its training set.

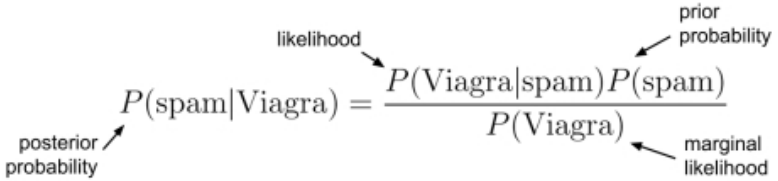
Comparing to human level performance



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.

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.



	Viagra		
Likelihood	**Yes**	**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***

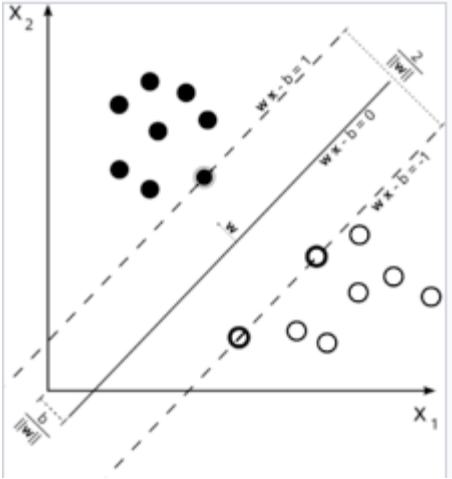
Expansion:

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

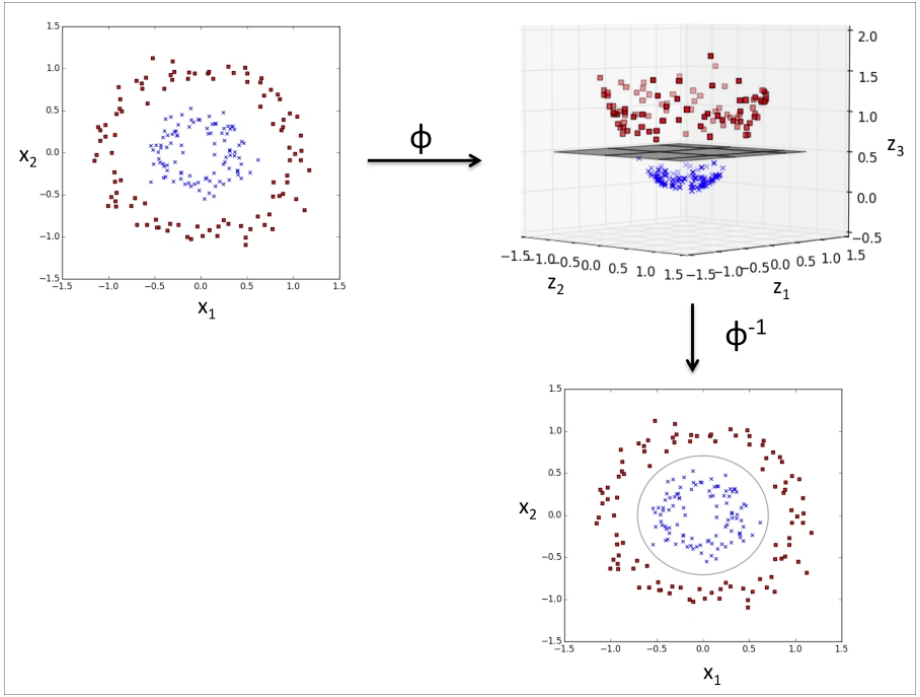
- Key assumption: all features are independent

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

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

Support Vector Machine (SVM) … contd

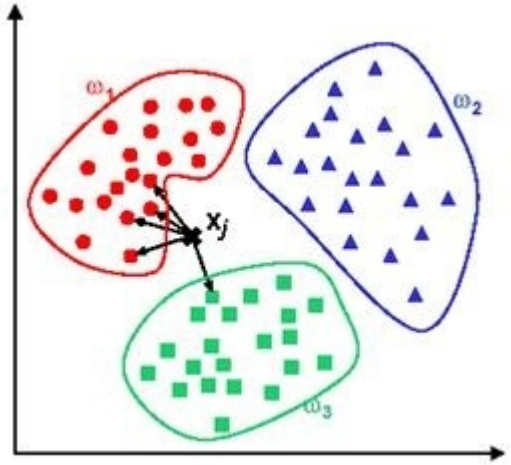
- A separator between the categories is found, and then the data are transformed in such a way that the separator could be drawn as a hyperplane.
- Following this, characteristics of new data can be used to predict the group to which a new record should belong.
- After the transformation, the boundary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

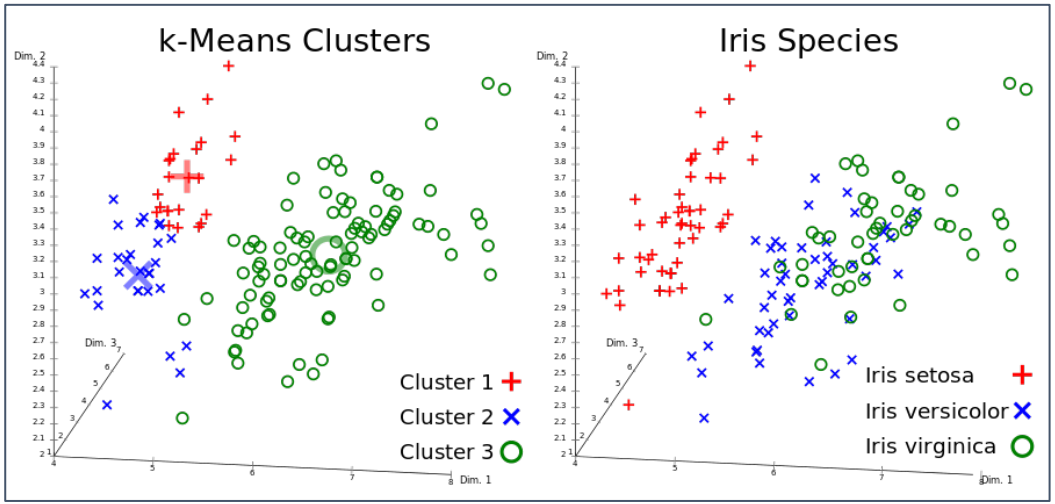


K-Nearest Neighbor (K-NN)

- 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.

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



Dimensionality Reduction

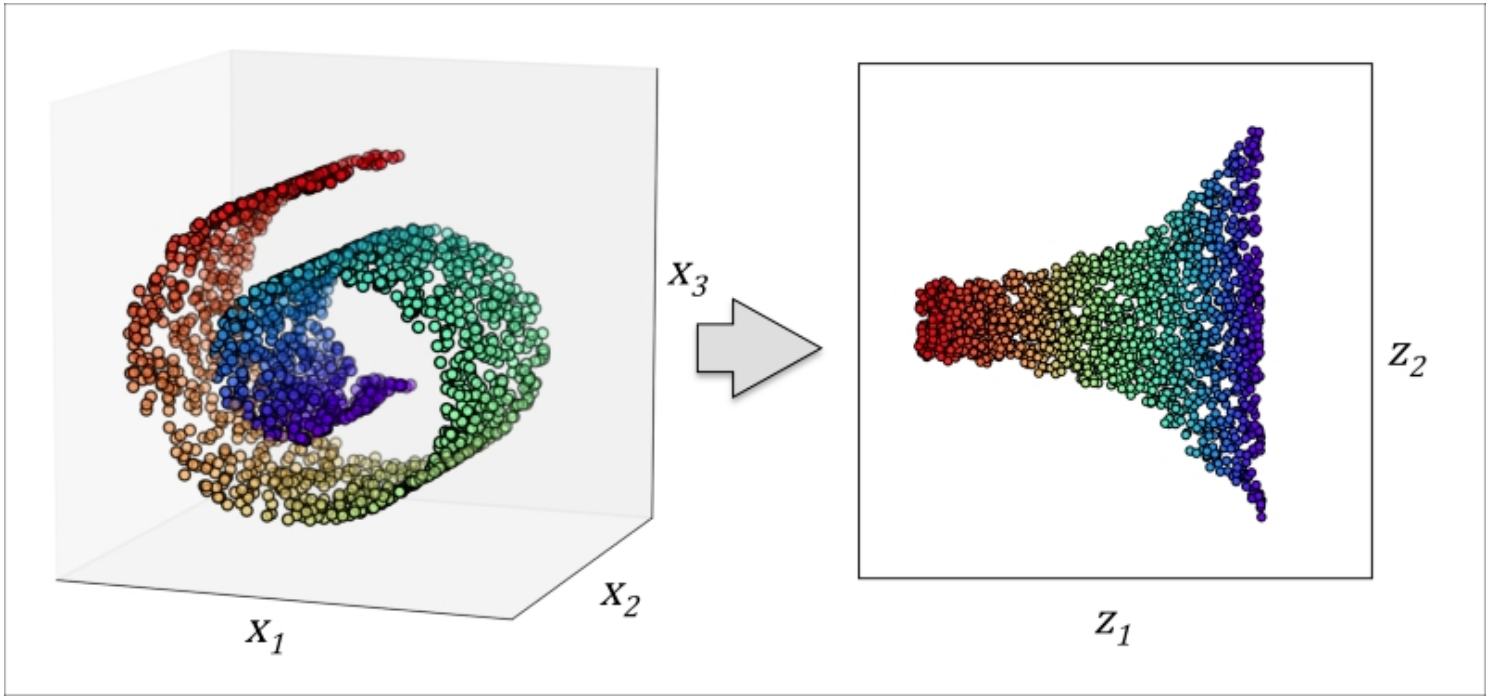
Curse of Dimensionality for KNN

- 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

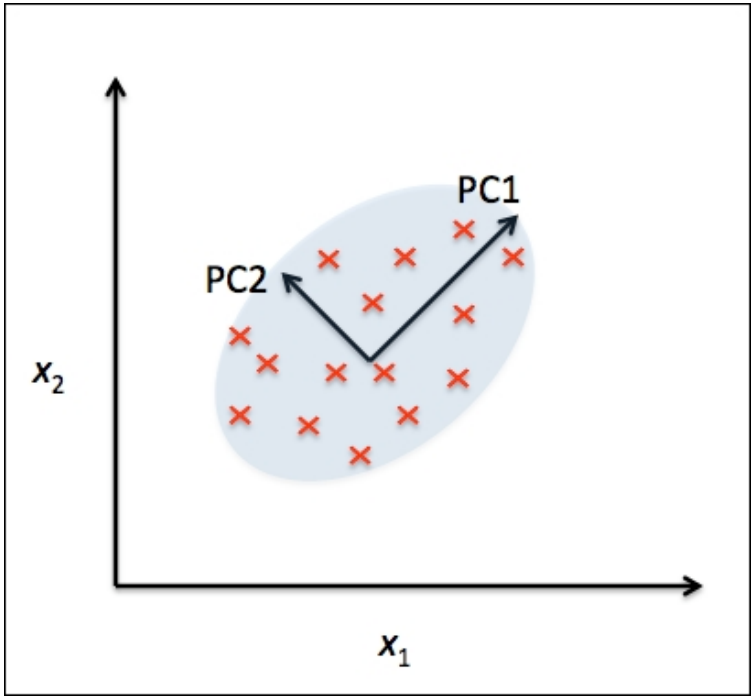
- It is a process of selecting a best subset of features. Goals of feature selection are
  - reduce computational time and space by reducing the size of the problem
  - improve performance of models by removing noisy and irrelevant features and reduce likelihood of overfitting
- Lasso, Decision Tree, 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

- **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.
- In a nutshell, 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

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.



Singular Value Decomposition

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

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

where

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

The matrices U and V are both defined to be orthogonal matrices. That means, $\mathrm{U}\mathrm{U}^{\mathrm{T}} = \mathrm{U}^{\mathrm{T}}\mathrm{U} = \mathrm{I}$. 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.

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

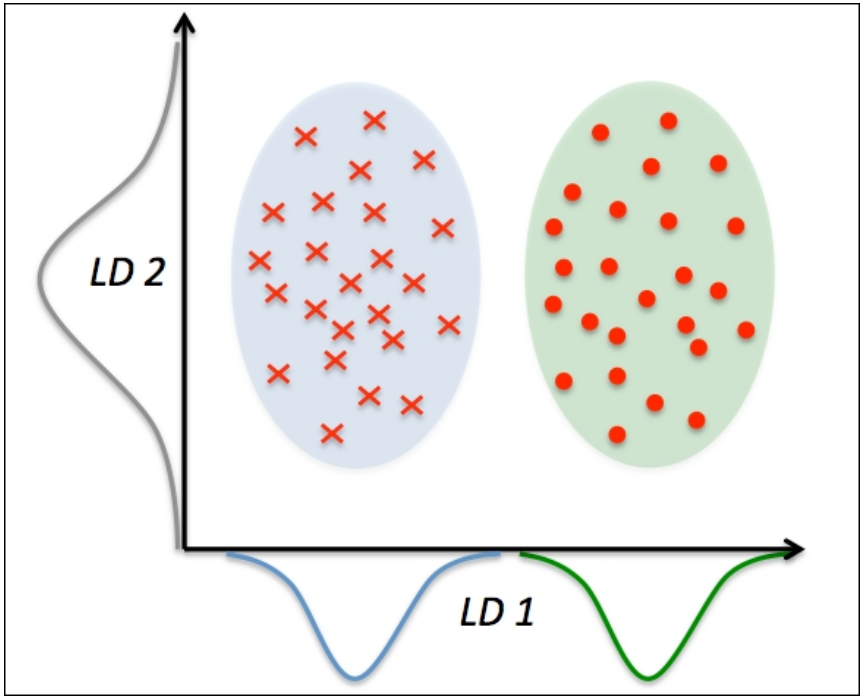
Choose K so that, $\frac{average\ squared\ projection\ error}{Total\ variation} \leq 0.01$

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

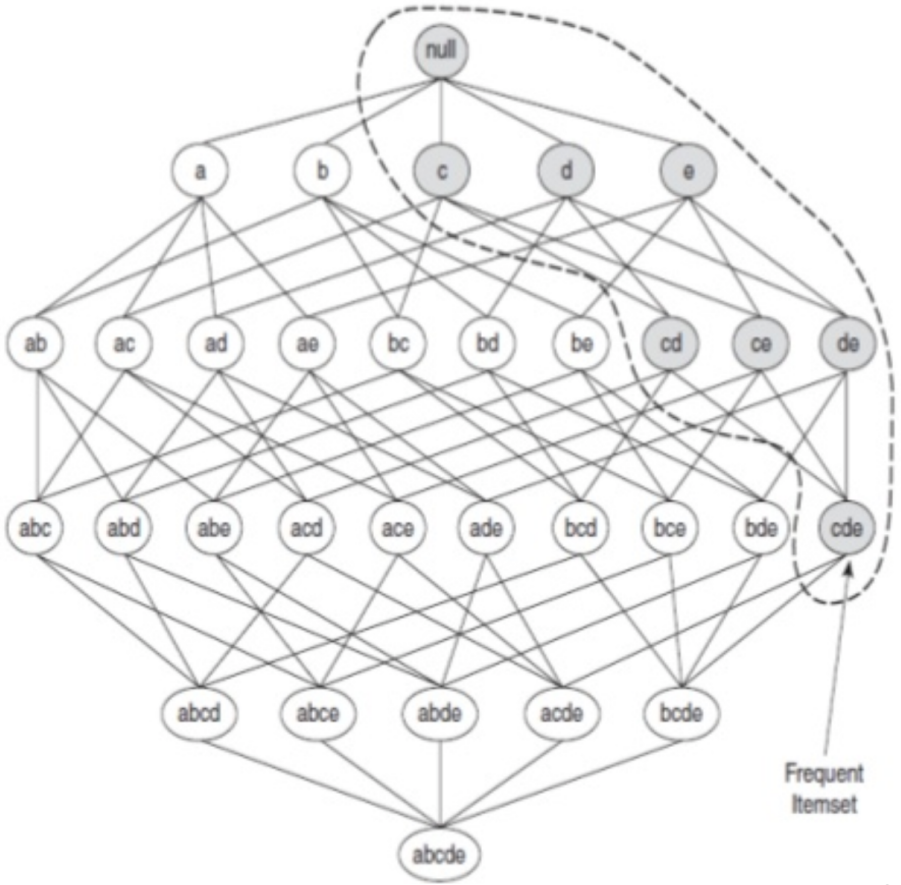
- Is capable of working with large amounts of transactional data
- Results in rules that are easy to understand
- Useful for "data mining" and discovering unexpected knowledge in databases

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

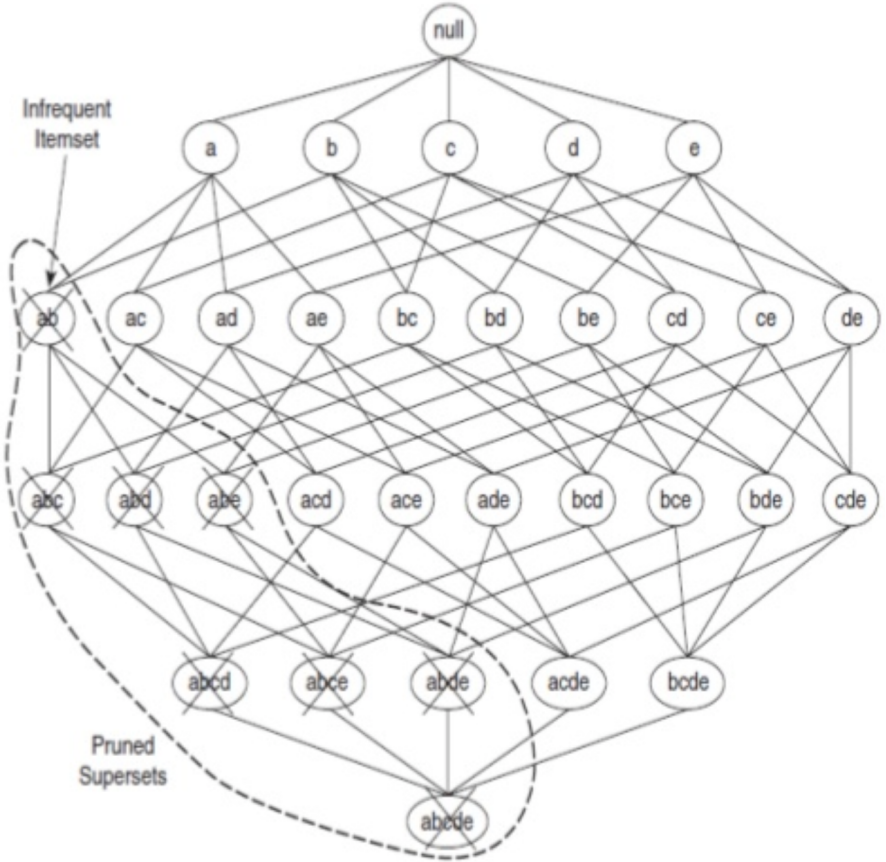
Apriori algorithm

- 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.

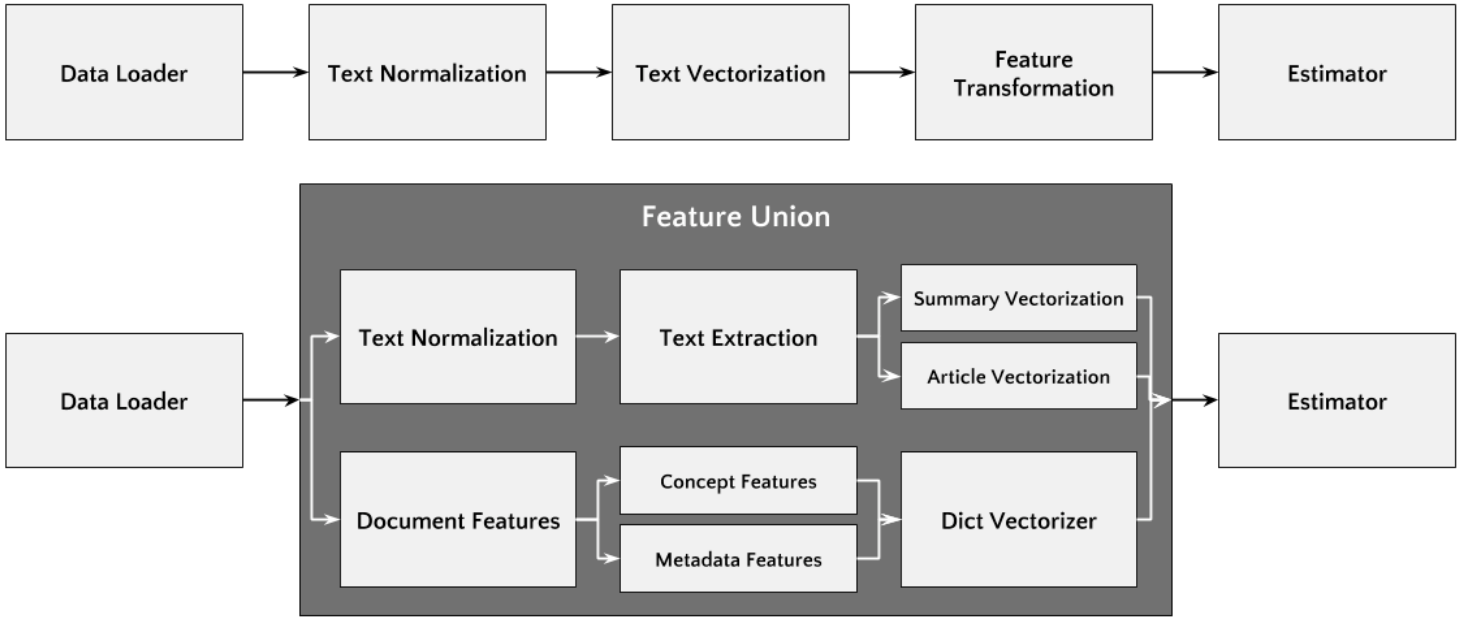
Rules	Support	Confidence	Lift
{liquor, red/blush wine} => {bottled beer}	0.002	0.90	11.24
{cereals, yogurt} => {whole milk}	0.002	0.81	3.17
{butter, jam} => {whole milk}	0.001	0.83	3.26
{chocolate, pickled vegetables} => {whole milk}	0.001	0.86	3.35
{grapes, onions} => {other vegetables}	0.001	0.92	4.74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

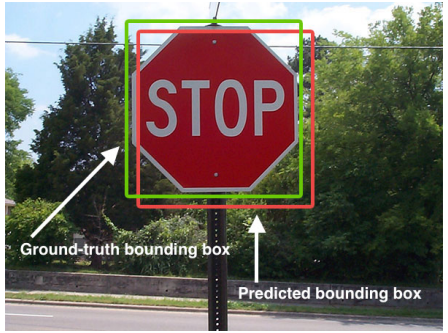
- Porter Stemmer
- LancasterStemmer
- SnowballStemmer

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

Similarity Matching



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

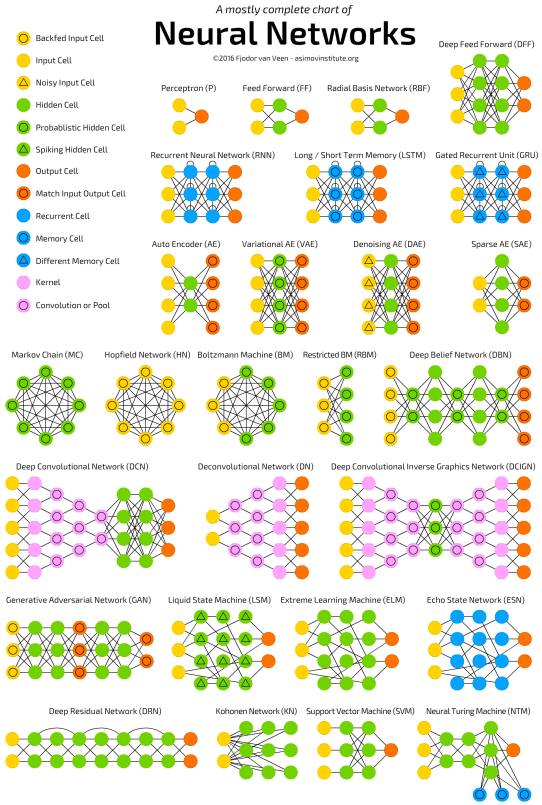
Ranges from 0 to 1. The smaller the angle the greater the similarity between the argument vectors is. Vectors x and y are binary representations respectively for set A and B.

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

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

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

Neural Network

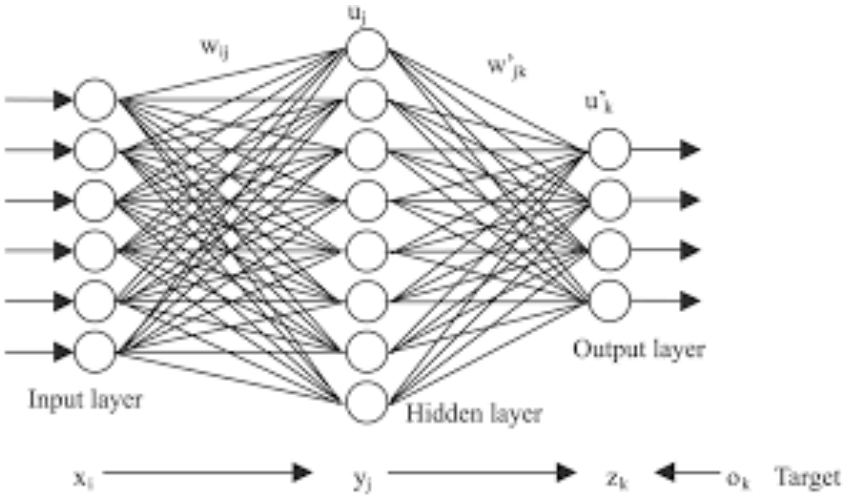


- Basic terms:
  - Deep Learning = Neural Networks
  - Deep Learning is a subset of Machine Learning
- Terms for neural networks:
  - MLP: Multilayer Perceptron
  - DNN: Deep neural networks
  - RNN: Recurrent neural networks
- LSTM: Long Short-Term Memory
- CNN or ConvNet: Convolutional neural networks
- DBN: Deep Belief Networks
- Neural network operations:
- Convolution
- Pooling
- Activation function
- Backpropagation

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

Artificial Neural Network

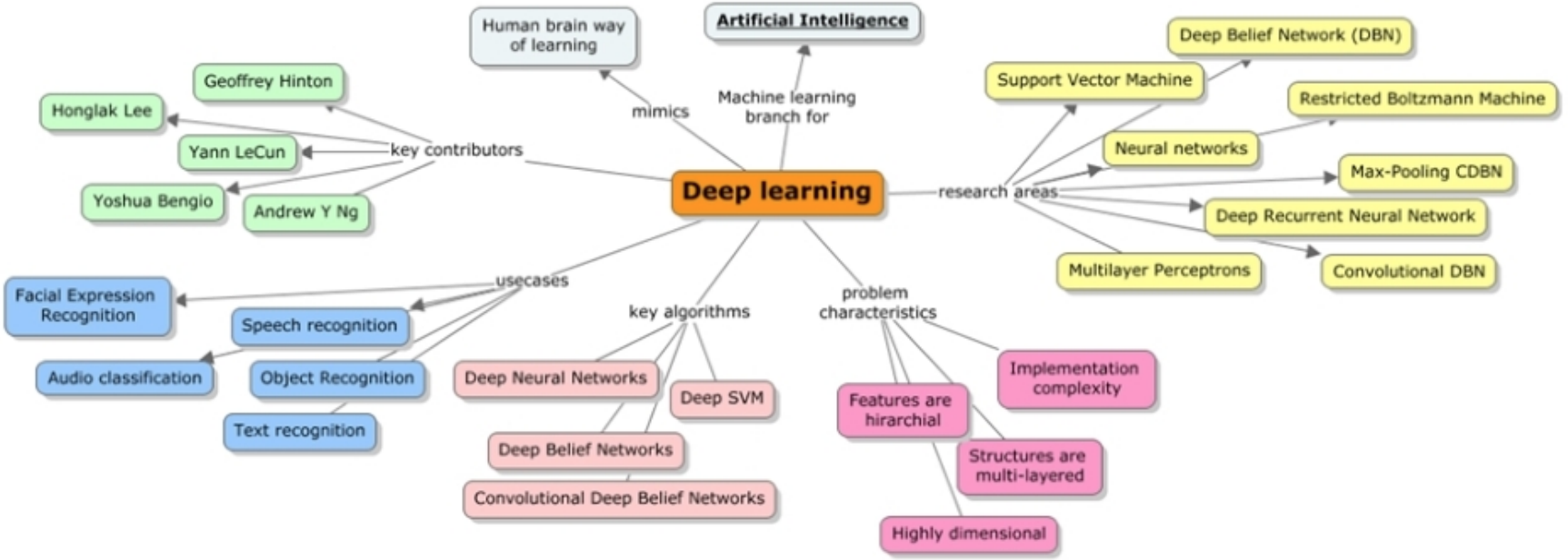
- ANN is inspired by the concept of biological nervous system.
- ANNs are the collection of computing elements (neurons) that may be connected in various ways.
- In ANNs the effect of the synapses is represented by the connection weight, which modulates the input signal.
- The architecture of artificial neural networks is a fully connected, three layered (input layer, hidden layer and output layer) structure of nodes in which information flows from the input layer to the output layer through the hidden layer.



Artificial Neural Network

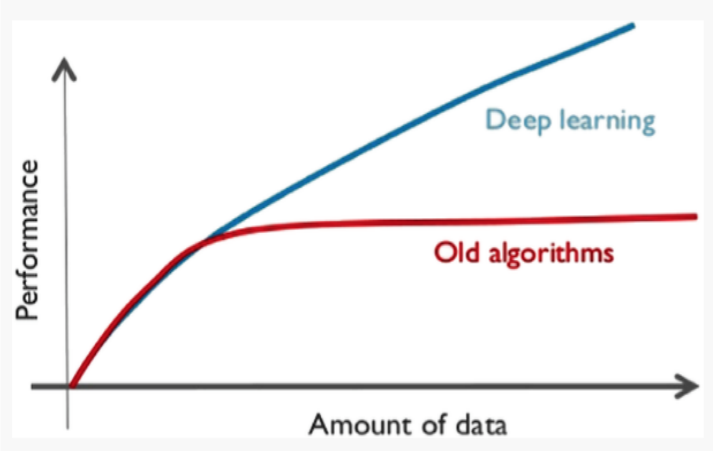
- ANNs are capable of linear and nonlinear classification.
- An ANN learns by adjusting the weights in accordance with the learning algorithms.
- It is capable to process and analyze large complex datasets, containing non-linear relationships.
- There are various types of artificial neural network architecture that are used in protein function prediction such as perceptron, multi-layer perceptron (MLP), radial basis function networks and kohonen self-organizing maps.

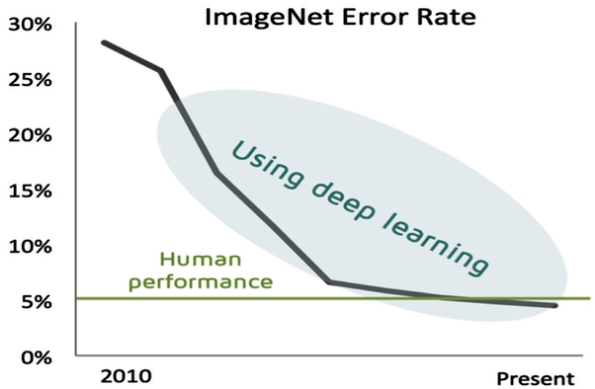
Deep Learning



Deep Learning

- Deep learning network is essentially advanced form of ANN with multiple layers with sophisticated architecture.
- DL is the cornerstone technology behind products for images recognition, video annotation, voice recognition, personal assistant, automated translation and autonomous vehicle





Hidden layers and number of neurons

Number of hidden layers

- 0 - Only capable of representing linear separable functions or decisions.
- 1 - Can approximate any function that contains a continuous mapping from one finite space to another.
- 2 - Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy.

Rule of thumb for the number of neurons in the hidden layers

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- $\frac{(m+n)}{2}$, m = output neurons, n = input neurons
- $\sqrt{m * n}$, m = output neurons, n = input neurons

Choice of activation function

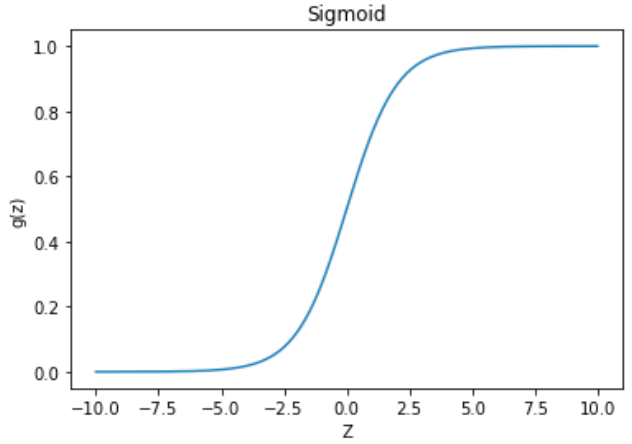
Unlike piecewise linear units, sigmoidal units saturate across most of their domain—they saturate to a high value when z is very positive, saturate to a low value when z is very negative, and are only strongly sensitive to their input when z is near 0. The widespread saturation of sigmoidal units can make gradient-based learning very difficult. For this reason, their use as hidden units in feedforward networks is now discouraged. When a sigmoidal activation function must be used, the hyperbolic tangent activation function typically performs better than the logistic sigmoid.

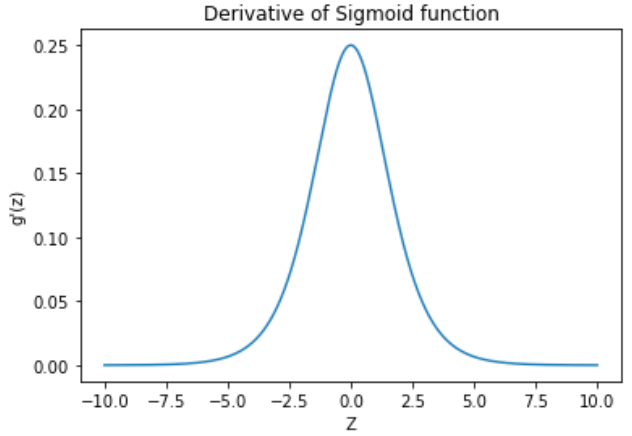
Sigmoid

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

Derivative

$$g'(z) = g(z)\big(1 - g(z)\big)$$
$$= a(1 - a)$$





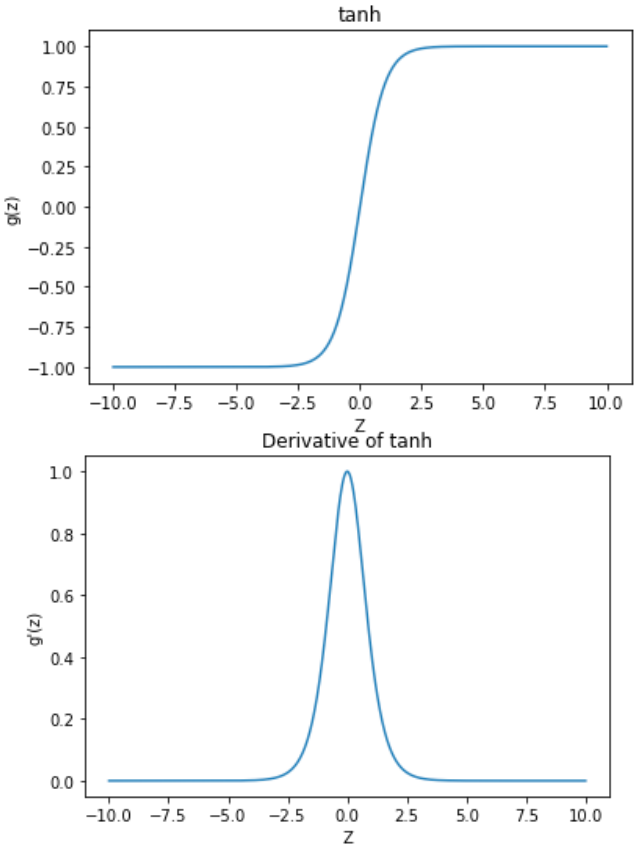
Tanh

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = a$$

Derivative

$$g'(z) = 1 - a^2$$

The tanh activation usually works better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data better for the next layer.

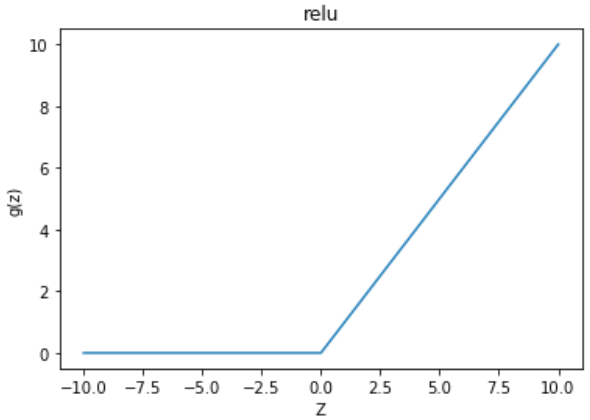


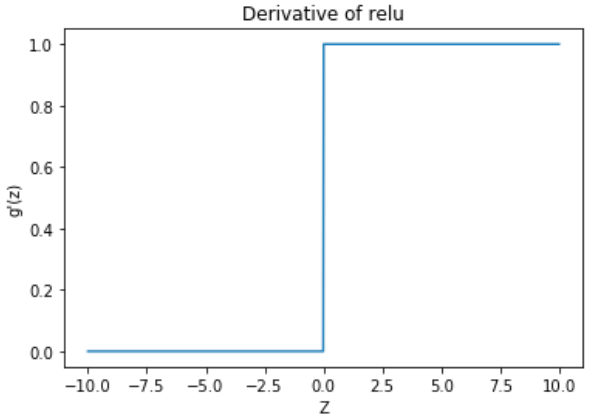
Relu - rectified linear unit

$$g(z) = \max(0, z)$$

Derivative

$$g'(z) = \begin{cases} -0, & z < 0 \\ 1, & z > 0 \\ undefined, & z = 0 \end{cases}$$





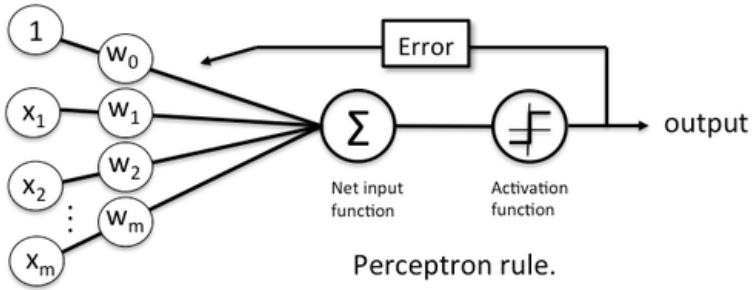
Deep Learning Framework

- Caffe
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- TensorFlow
- Torch

Choosing deep learning frameworks

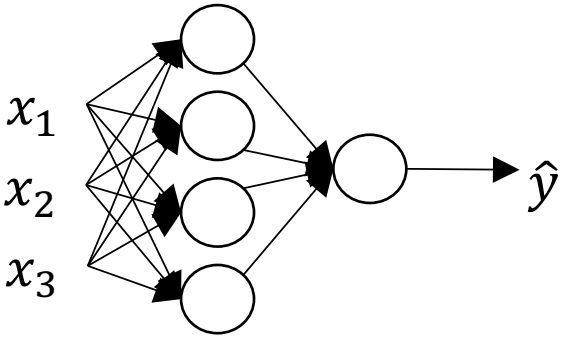
- Ease of programming (development and deployment)
- Running speed
- Truly open (open source with good governance)
- Language of preference
- Field of application - vision, speech, advertisement etc

Vectorized Operation for Logistic Regression



$$Z = W^T X + b$$
$$A = sigmoid(Z)$$
$$dZ = (A - Y)$$
$$dw = \frac{1}{m} X dZ^T$$
$$db = \frac{1}{m} np.sum(dZ)$$
$$W = W - \alpha dW$$
$$b = b - \alpha db$$

Vectorizing across multiple examples



$$z^{[1]}_{(n\_h,m)} = W^{[1]}_{(n\_h,n\_x)} \; x_{(n\_x,m)} + b^{[1]}_{(n\_h,1)}$$

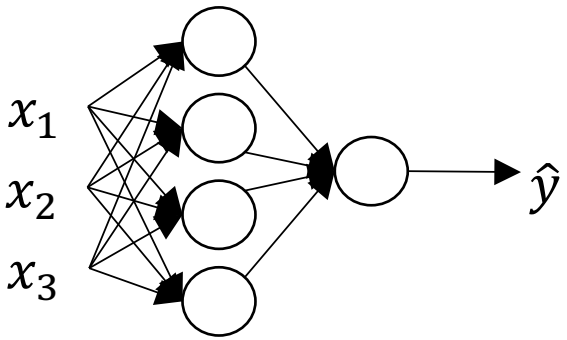
$$a^{[1]}_{(n\_h,m)} = \sigma(z^{[1]}_{(n\_h,m)})$$

$$z^{[2]}_{(n\_y,m)} = W^{[2]}_{(n\_y,n\_h)} \; a^{[1]}_{(n\_h,m)} + b^{[2]}_{(n\_y,1)}$$

$$a^{[2]}_{(n\_y,m)} = \sigma(z^{[2]}_{(n\_y,m)})$$

$$X = \underset{(n\_x,m)}{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}} \qquad A^{[1]} = \underset{(n\_h,m)}{\begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}}$$

Back propagation gradient descent



$$\underset{(n\_y,m)}{dz^{[2]}} = \underset{(n\_y,m)}{a^{[2]}} - \underset{(n\_y,m)}{y}$$

$$\underset{(n\_y,n\_h)}{dW^{[2]}} = \underset{(n\_y,m)}{dz^{[2]}} \underset{(n\_h,m)}{a^{[1]^T}}$$

$$\underset{(n\_y,m)}{db^{[2]}} = \underset{(n\_y,m)}{dz^{[2]}}$$

$$\underset{(n\_h,m)}{dz^{[1]}} = \underset{(n\_y,n\_h)}{W^{[2]T}} \underset{(n\_y,m)}{dz^{[2]}} * \underset{(n\_h,m)}{g^{[1]\prime}(z^{[1]})}$$

$$\underset{(n\_h,n\_x)}{dW^{[1]}} = \underset{(n\_h,m)}{dz^{[1]}} \underset{(n\_x,m)}{x^T}$$

$$\underset{(n\_h,m)}{db^{[1]}} = \underset{(n\_h,m)}{dz^{[1]}}$$

Summary of gradient descent

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]^T}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(\mathrm{z}^{[1]})$$

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$$

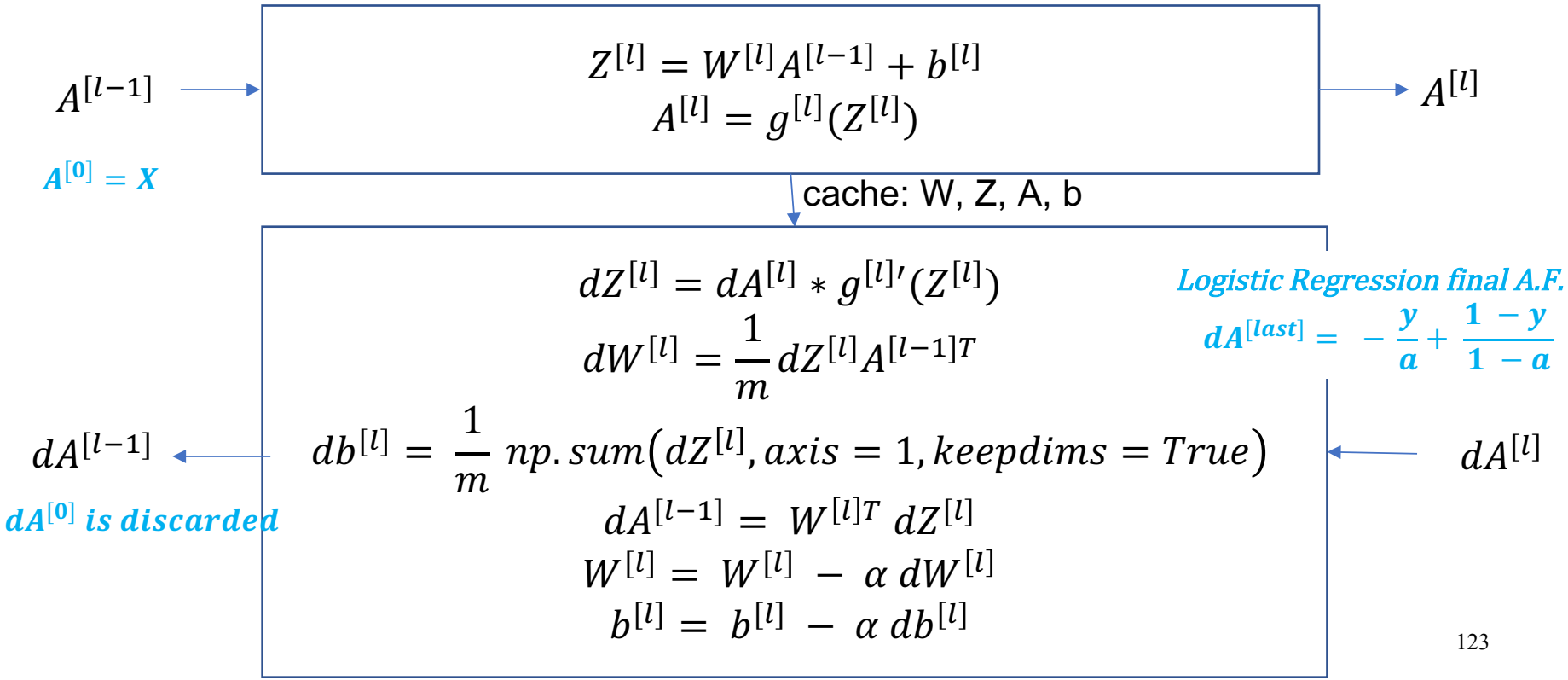
$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$

$$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(\mathrm{Z}^{[1]})$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$

$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

What happens in each layer …



Initialize Weights

- Logistic Regression doesn't have a hidden layer. If you initialize the weights to zeros, the first example x fed in the logistic regression will output zero but the derivatives of the Logistic Regression depend on the input x (because there's no hidden layer) which is not zero. So at the second iteration, the weights values follow x's distribution and are different from each other if x is not a constant vector.
- Neural network has hidden layers. If you decide to initialize the weights and biases to be zero, each neuron in the first hidden layer will perform the same computation. So even after multiple iterations of gradient descent each neuron in the layer will be computing the same thing as other neurons.
- If you initialize the weights to relative large values. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.

Exploding and Vanishing Weights

- If the weights in a network start too small, then the signal shrinks as it passes through each layer until it’s too tiny to be useful.
- If the weights in a network start too large, then the signal grows as it passes through each layer until it’s too massive to be useful.
- It turns out that initialization is surprisingly important. A marked difference can appear with only 3-4 layers in the network.

Xavier and He initialization

- By using Xavier initialization, we make sure that the weights are not too small but not too big to propagate accurately the signals.

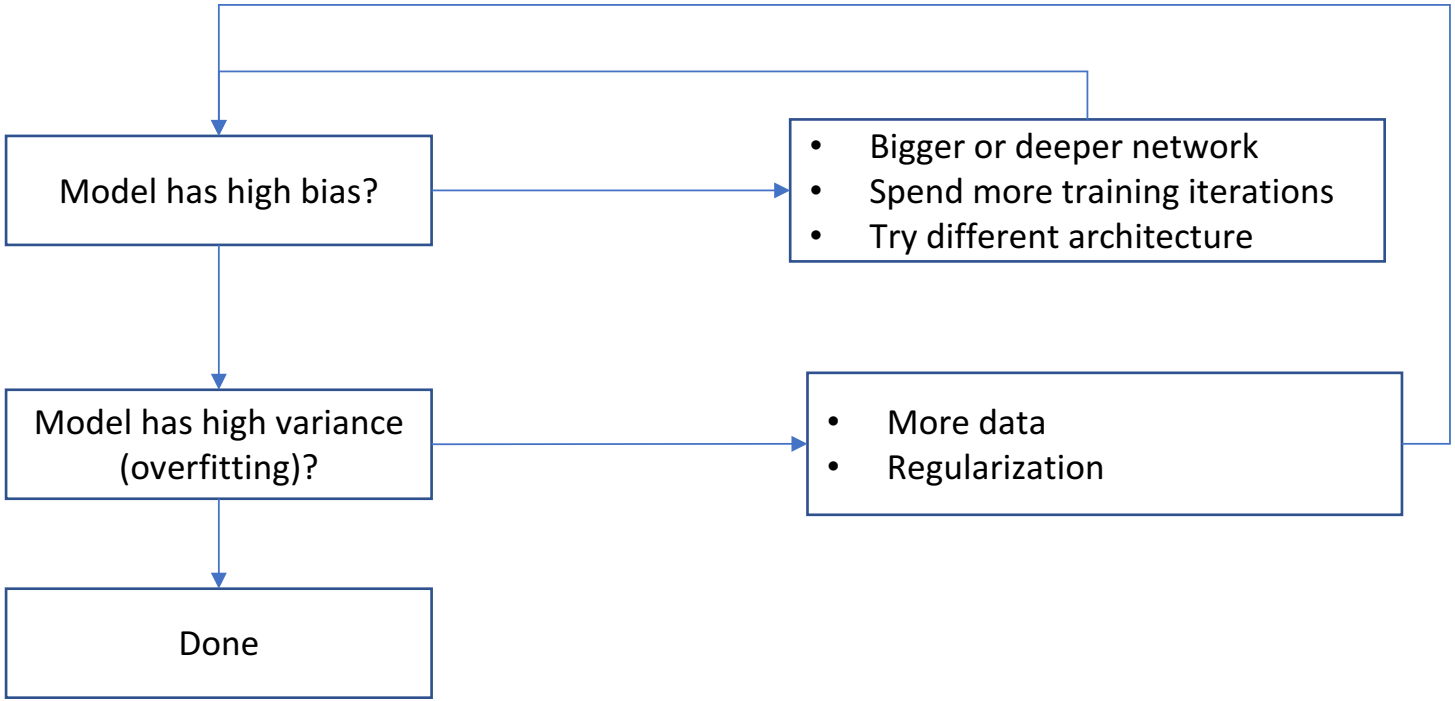
$$var\left(w^{[l]}\right) = \frac{1}{n^{[l-1]}}$$

- He initialization (better choice for ReLU or tanh activation function)

$$var\left(w^{[l]}\right) = \frac{2}{n^{[l-1]}}$$

Regularize Deep Neural Net

Bias Variance Trade Off in Deep Learning



Regularization

Non regularized Cost Function

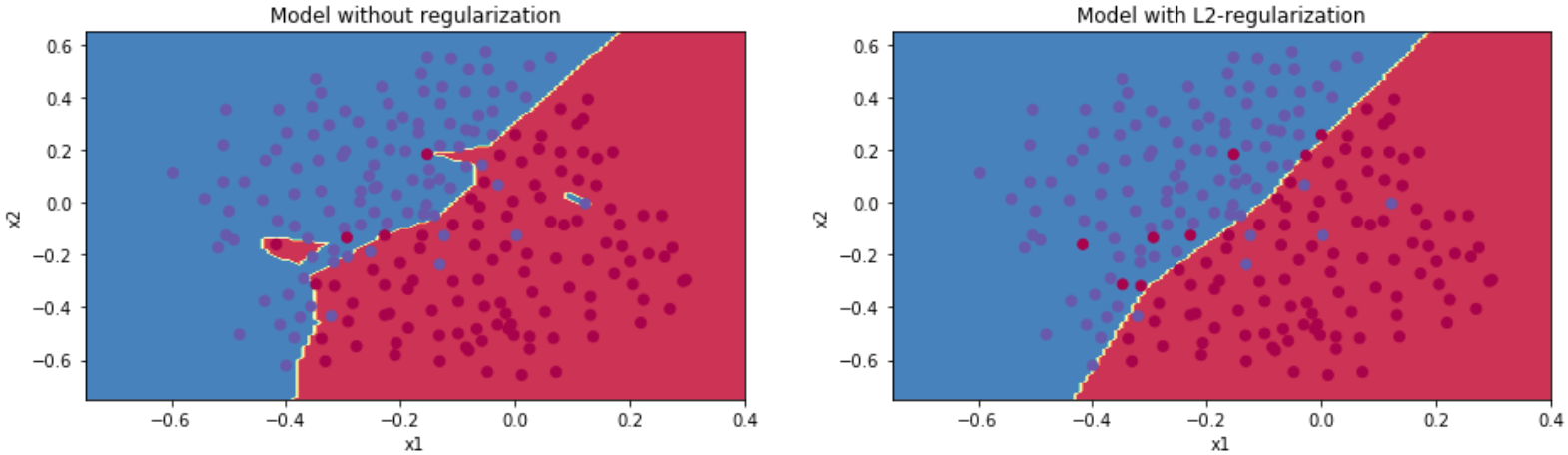
$$\text{Cross entropy } cost, J = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right)$$

L2 Regularized Cost Function

$$J = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) + \frac{\lambda}{2m}\|W\|_F^2$$

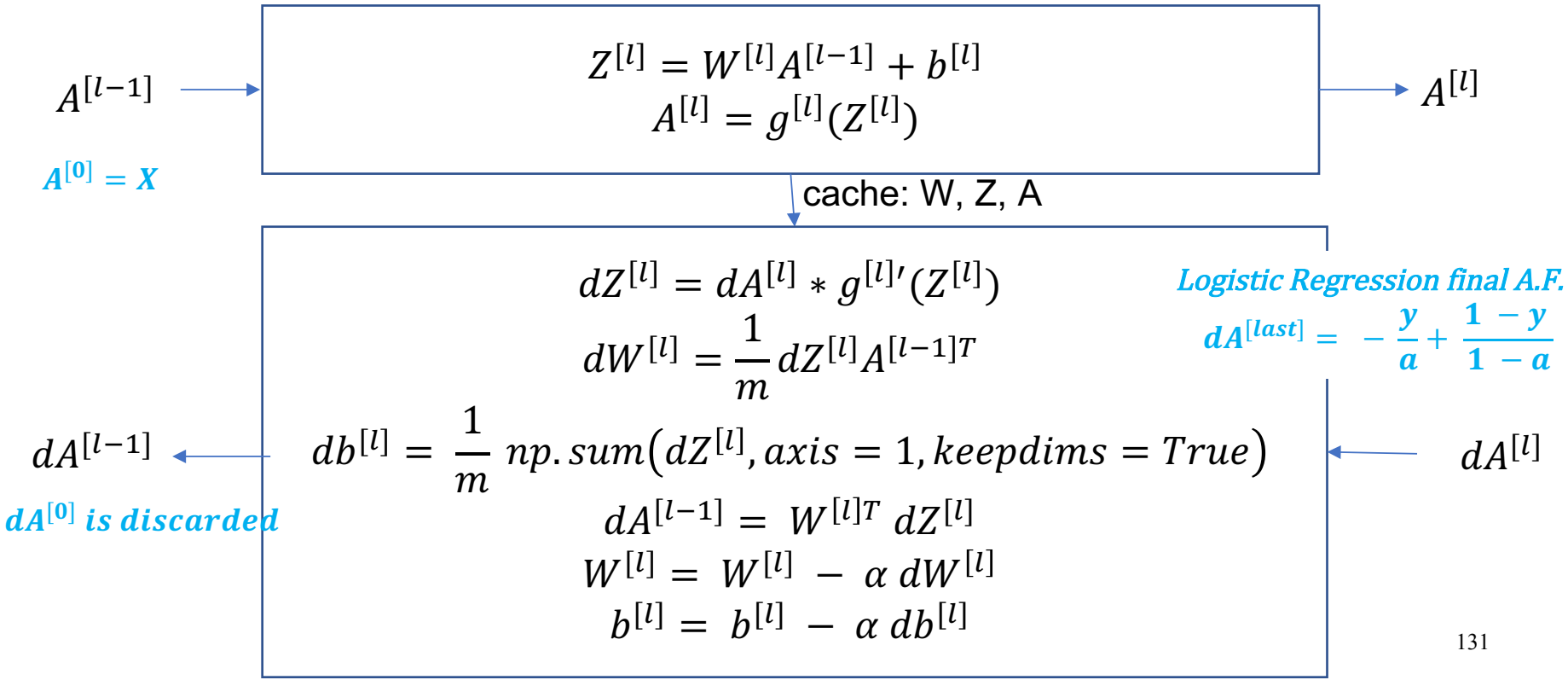
L2-regularization relies on the assumption that a model with small weights is simpler than a model with large weights. Thus, by penalizing the square values of the weights in the cost function you drive all the weights to smaller values. It becomes too costly for the cost to have large weights! This leads to a smoother model in which the output changes more slowly as the input changes.

Effect of Regularization



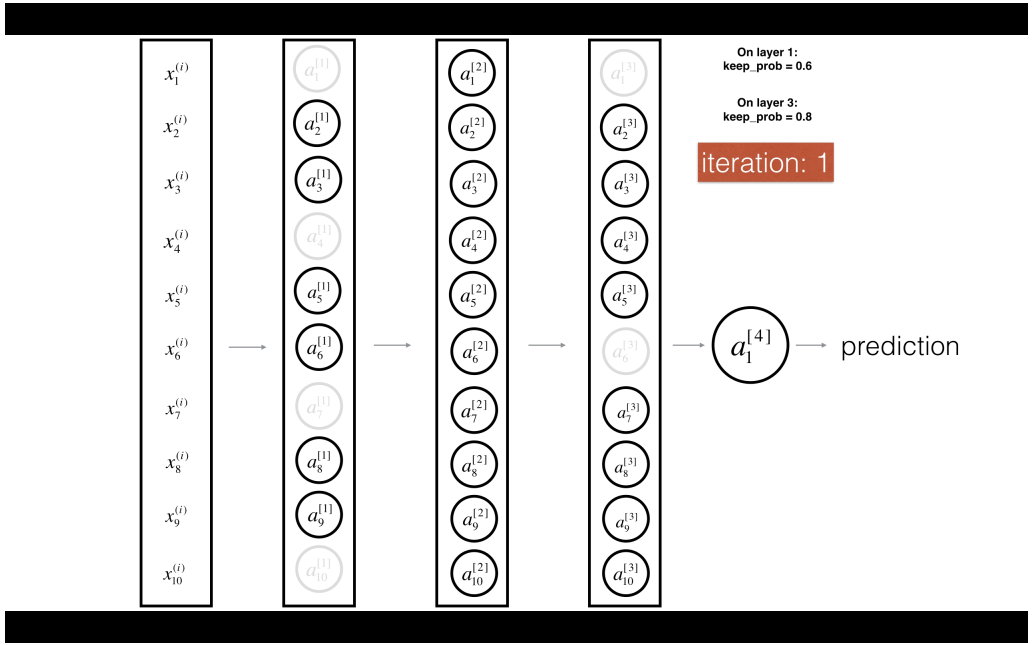
L2 regularization makes your decision boundary smoother. If λ is too large, it is also possible to "oversmooth", resulting in a model with high bias.

What happens in each layer after regularizarion

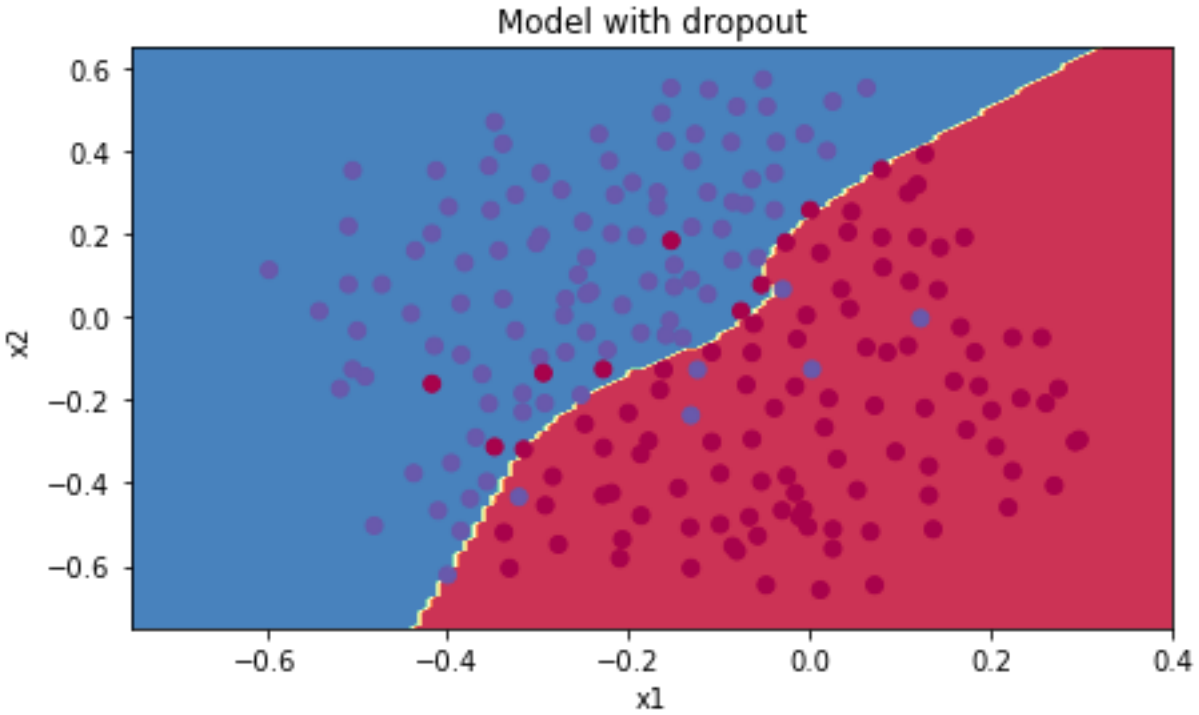


Dropout

- Dropout is a widely used regularization technique that is specific to deep learning.
- It randomly shuts down some neurons in each iteration.
- The dropped neurons don't contribute to the training in both the forward and backward propagations of the iteration.
- The idea behind drop-out is that at each iteration, you train a different model that uses only a subset of your neurons. With dropout, your neurons thus become less sensitive to the activation of one other specific neuron, because that other neuron might be shut down at any time.



Effect of Dropout



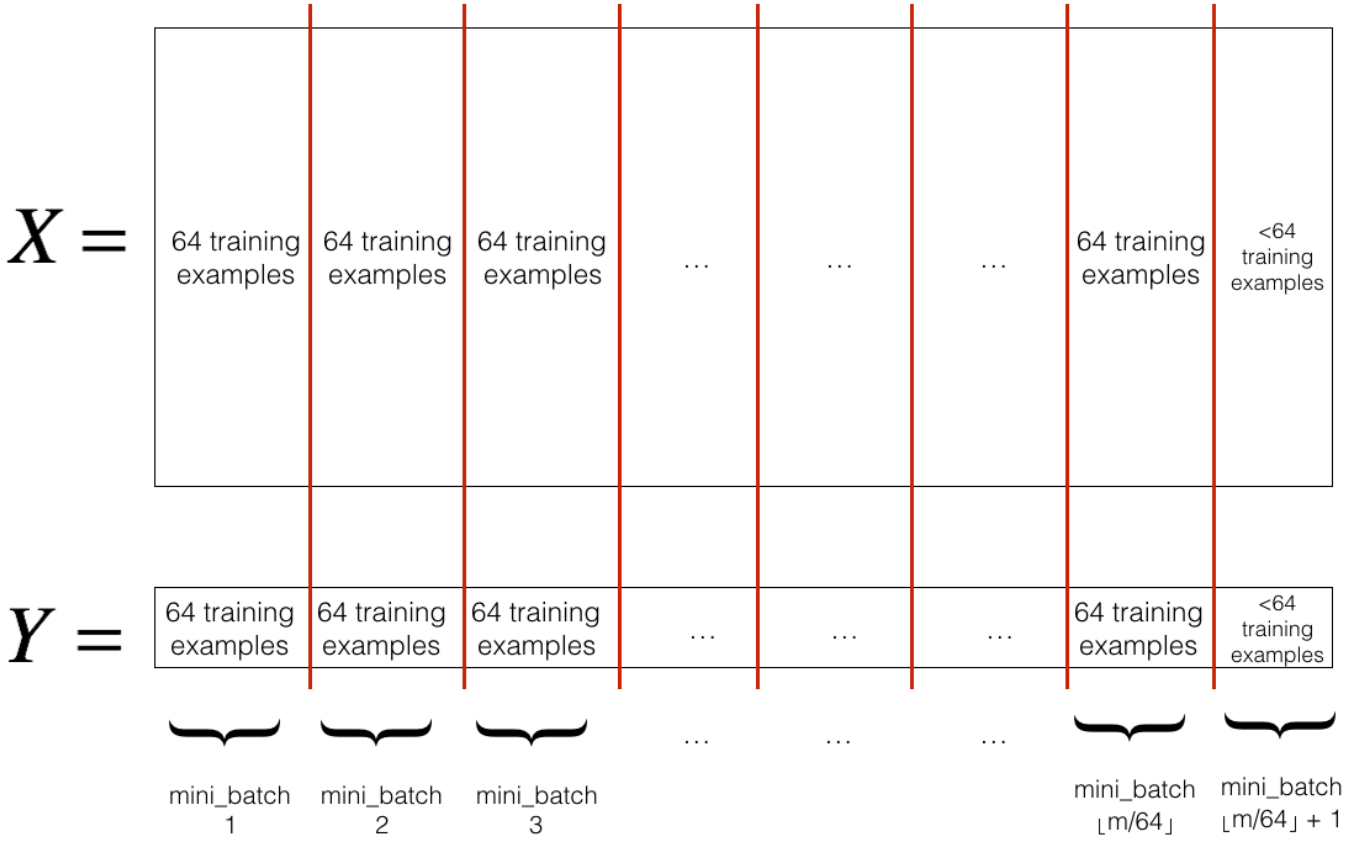
Regularization Summary Table

Model	Train Accuracy	Test Accuracy
3-layer NN without regularization	95%	91.5%
3-layer NN with L2-regularization	94%	93%
3-layer NN with dropout	93%	95%

Note that regularization hurts training set performance! This is because it limits the ability of the network to overfit to the training set. But since it ultimately gives better test accuracy, it is helping your system.

Gradient descent optimizers

Mini Batch

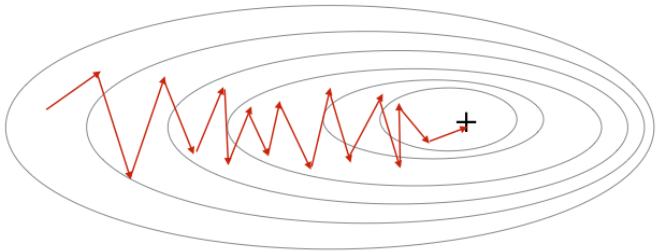

X =
64 training examples
64 training examples
64 training examples
…
…
…
64 training examples
<64 training examples
Y =
64 training examples
64 training examples
64 training examples
…
…
…
64 training examples
<64 training examples
mini_batch 1
mini_batch 2
mini_batch 3
mini_batch ⌊m/64⌋
mini_batch ⌊m/64⌋ + 1

Mini Batch, Stochastic and Batch

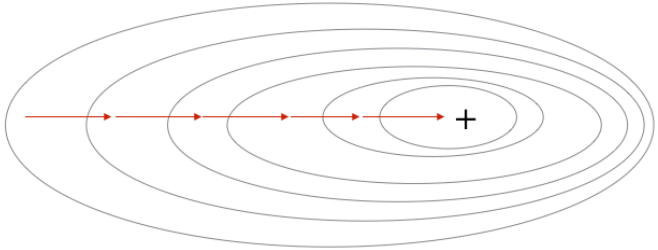
Batch Gradient Descent	Mini Batch	Stochastic
• At each iteration, calculate Δθ over entire training dataset and update θ:=θ - αθ • Takes longer for iteration • Data maybe too much to to fit in memory • Suitable for training set < 2000 records	• At each iteration, calculate Δθ over a small batch of training examples (subset from the entire training set) and update θ:=θ - αθ • Batch size is generally between 64-512 (2^n) • Oscillates near the global minimum • Takes advantage of Vectorized operations • Suitable for big dataset	• It is a mini batch with batch size = 1 • Lose the speed of vectorized operation • Converges faster to minimum faster since the θ are update more often • Oscillates around the global minimum

Stochastic vs Mini Batch vs Batch

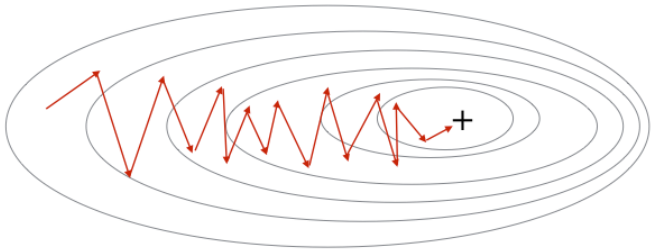
Stochastic Gradient Descent



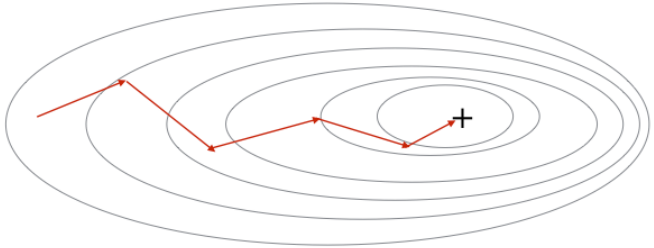
Gradient Descent



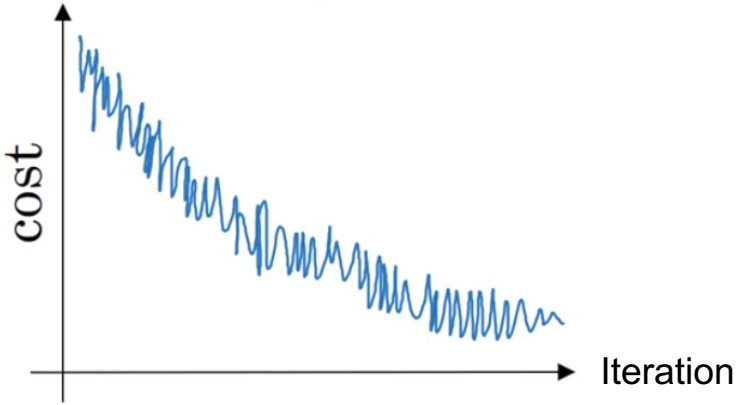
Stochastic Gradient Descent



Mini-Batch Gradient Descent



Cost decay over iteration



Mini Batch Gradient Descent

Gradient descent with momentum

$v_{dW} = \beta\, v_{dW} + (1-\beta)dW$
$v_{db} = \beta\, v_{db} + (1-\beta)db$
$W = W - \alpha v_{dW}$
$b = b - \alpha v_{db}$

Hyper parameters: α, β
β is usually 0.9, α is learning rate

RMS Prop

$$S_{dW} = \beta S_{dW} + (1-\beta) dW^2$$

$$S_{db} = \beta S_{db} + (1-\beta) db^2$$

$$W = W - \alpha \frac{dW}{\sqrt{S_{dW}} + \varepsilon}$$

$$b = b - \alpha \frac{db}{\sqrt{S_{db}} + \varepsilon}$$

Hyper parameters: α, β
β is usually 0.9, α is learning rate
ε is added to avoid divide by zero and usually 10^{-8}

Adam Optimization

Adam = Adaptive Moment Estimation

- Adam combines the advantages of RMSProp and momentum and used for mini batch gradient descent
- The learning rate parameter is usually tuned
- Usually use default values for hyper parameters β_1, β_2 and ε ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\varepsilon = 10^{-8}$)

$$v_{dW^{[l]}} = \beta_1 v_{dW^{[l]}} + (1 - \beta_1)\frac{dJ}{dW^{[l]}}$$

$$v_{dW^{[l]}}^{corrected} = \frac{v_{dW^{[l]}}}{1 - \beta_1^2}$$

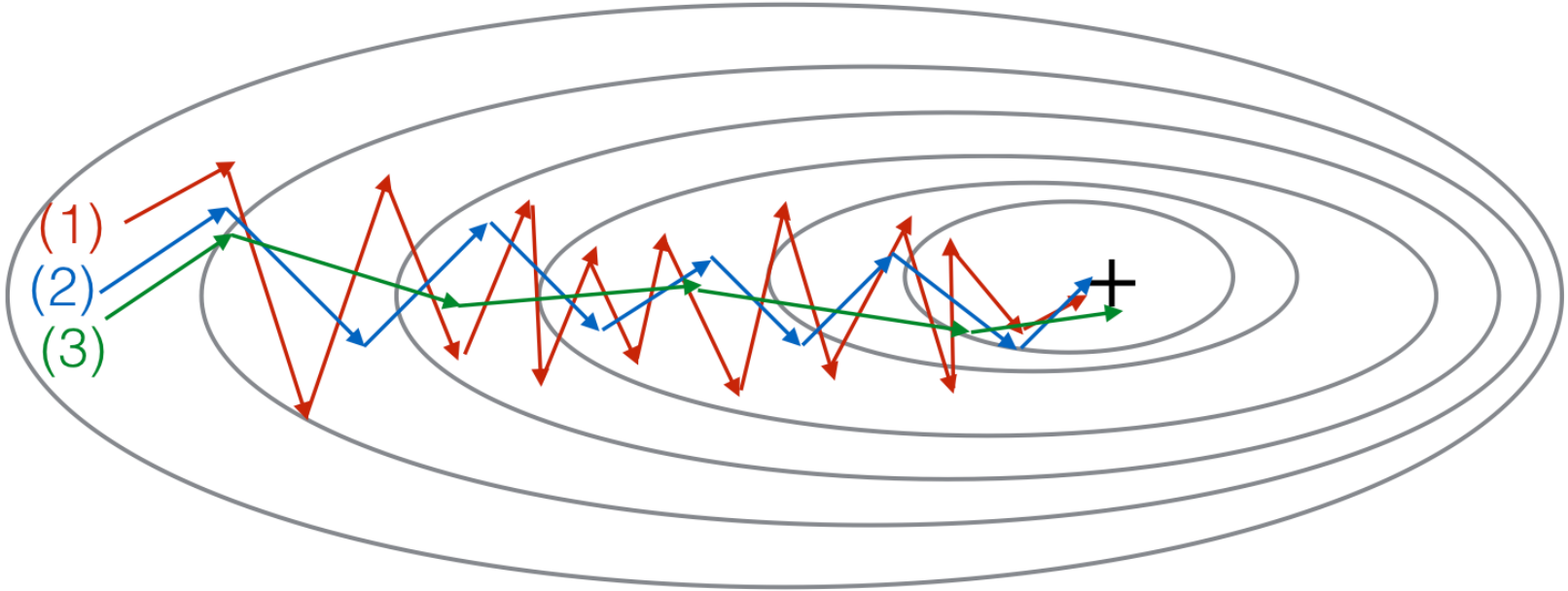
$$S_{dW^{[l]}} = \beta_2 S_{dW^{[l]}} + (1 - \beta_2)(\frac{dJ}{dW^{[l]}})^2$$

$$S_{dW^{[l]}}^{corrected} = \frac{S_{dW^{[l]}}}{1 - \beta_2^2}$$

$$W^{[l]} = W^{[l]} - \alpha \frac{v_{dW^{[l]}}^{corrected}}{\sqrt{S_{dW^{[l]}}^{corrected}} + \varepsilon}$$

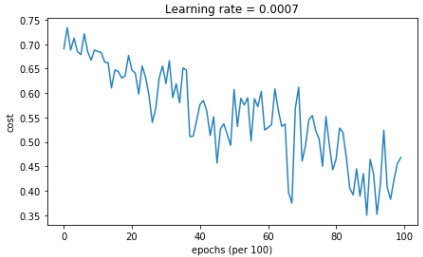
Adam paper: https://arxiv.org/pdf/1412.6980.pdf

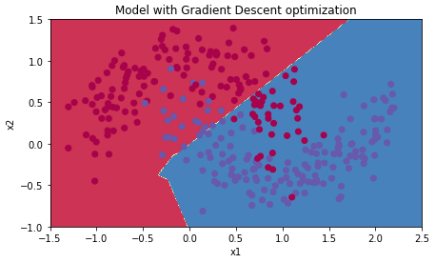
Gradient Descent Optimizer



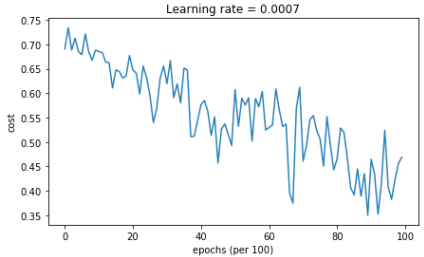
(1) gradient descent with momentum (small β)
(2) gradient descent with momentum (small β),
(3) gradient descent

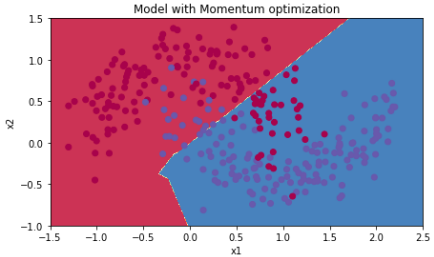
Mini Batch, With Momentum and Adam



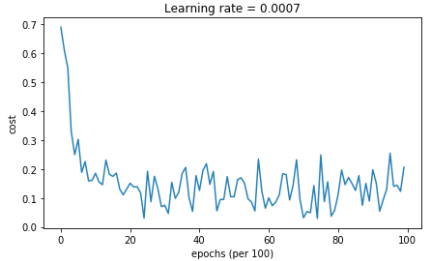


Gradient Descent with Mini Batch





Gradient Descent with Momentum Optimizer



Gradient Descent with Adam Optimizer

Accuracy Scores

Mini - 79.7%
Momentum - 79.7%
Adam - 94%

What if the cost functions remains flat

- Try better random initialization for the weight (He method)
- Try tuning the learning rate
- Try using Adam
- Try mini batch gradient descent

Learning Rate Decay

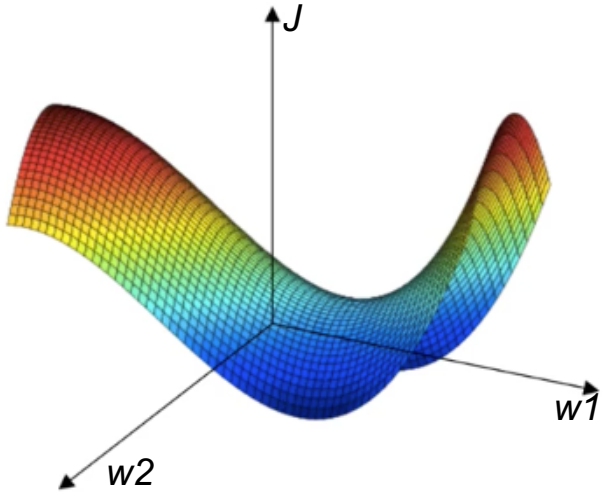
$\alpha = \frac{1}{1+decay\ rate\ *t}\alpha_0$ decay rate: 1, 2

$\alpha = \frac{1}{\sqrt{t}}\alpha_0$

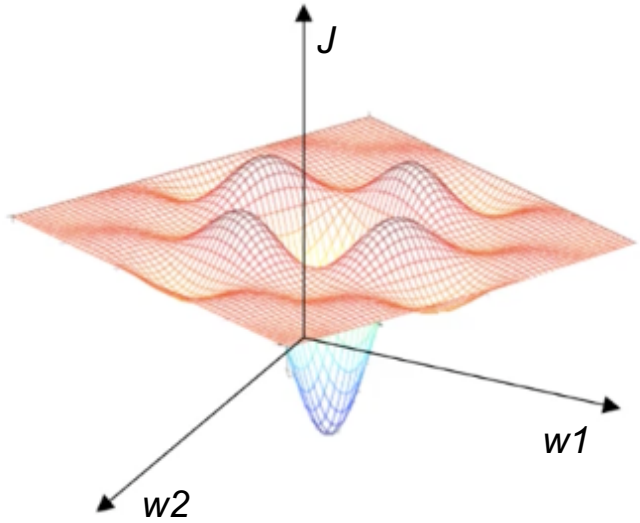
$\alpha = 0.95^t\alpha_0$

t is the epoch number, epoch is one pass over entire training set

Problem of Local Optima and Saddle Points



Saddle Point



Local Optima

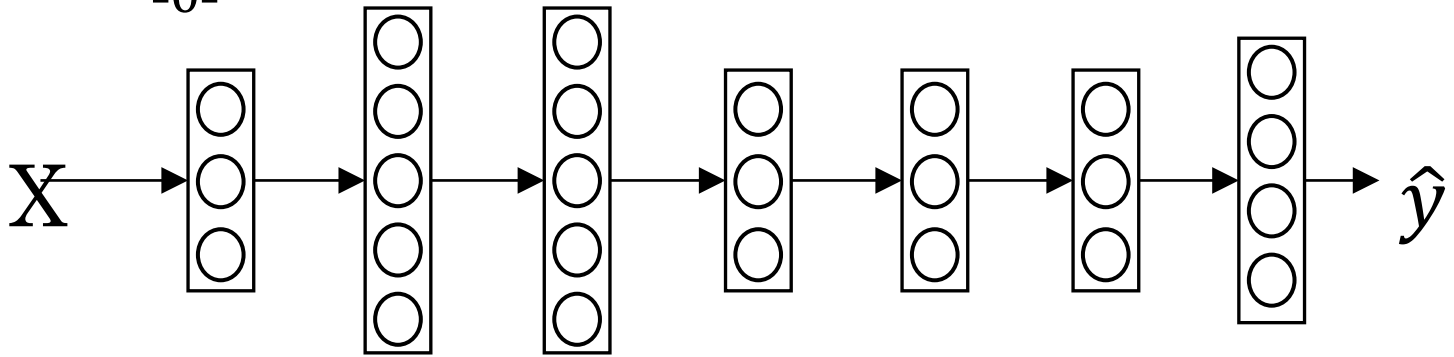
Batch Normalization

- Normalization: local response normalization across channels (LRN), batch normalization
-

Softmax Classifier

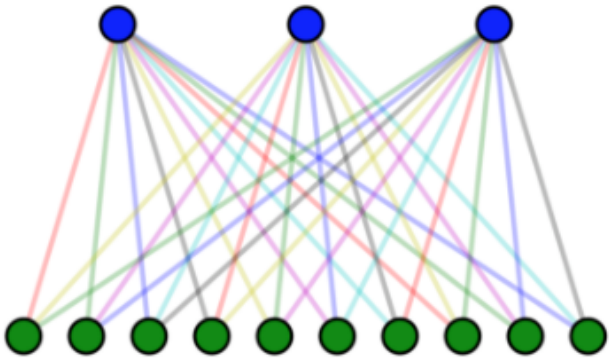
Softmax classifier is used as activation function at the final layer for multi class classifications problems. It gives out a vector of probabilities for the classes.

$y^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, AF = $\frac{e^{Z^{[l](i)}}}{\sum_{i=1}^{c} e^{Z^{[l](i)}}}$, loss function = $\sum_{i=1}^{c} y_i log\hat{y}_i$



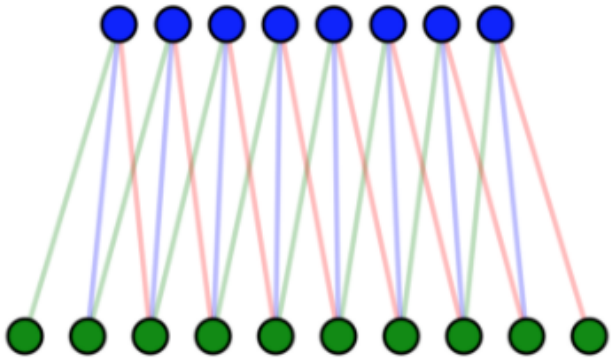
Fully Connected vs CNN

Fully Connected



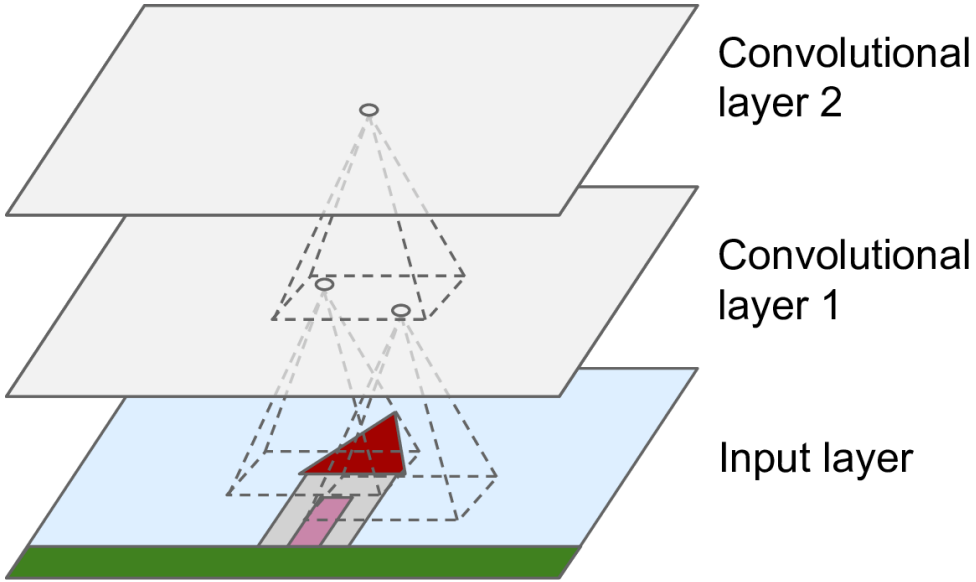
Each unit is connected to all of the units in the previous layer

Convolutional Layer



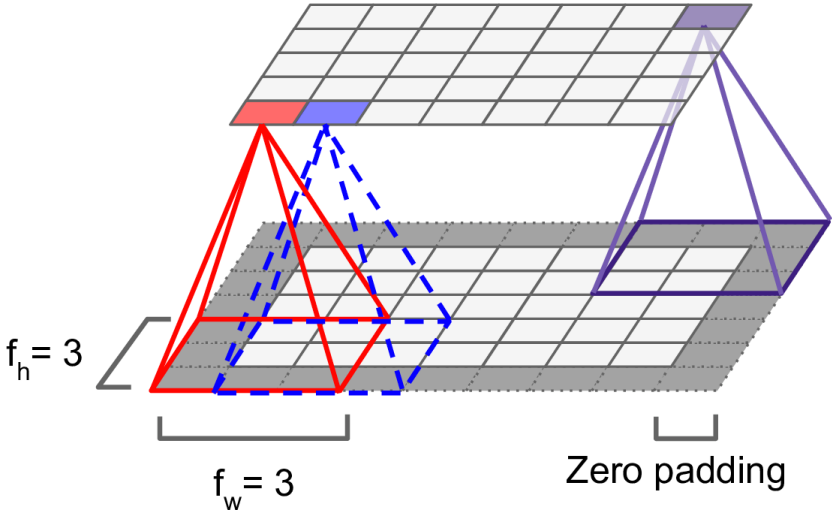
Each unit is connected to a (typically small) number of nearby units in the previous layer. Furthermore, all units are connected to the previous layer in the same way, with the exact same weights and structure.

CNN layers with rectangular local receptive fields



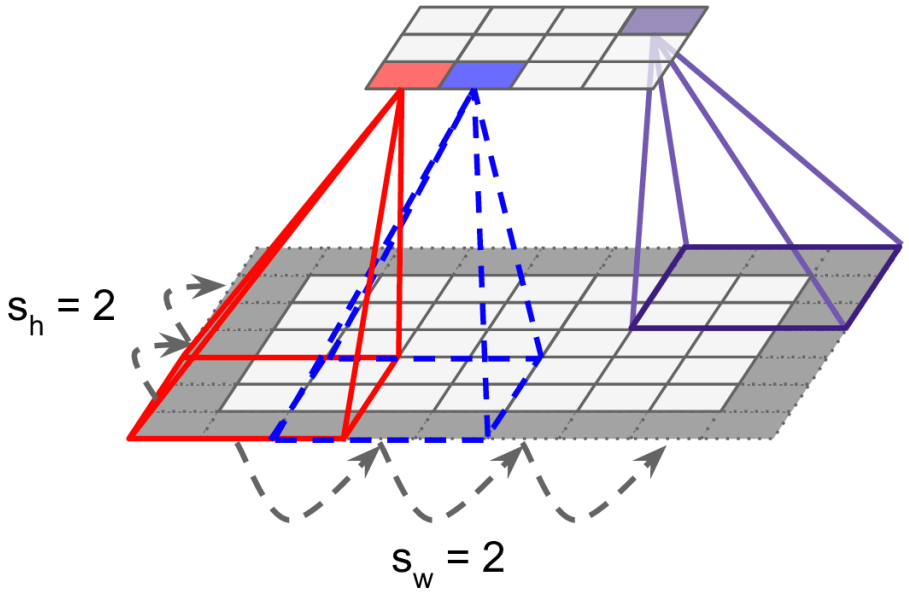
Connections between layers and zero padding

A neuron located in row i, column j of a given layer is connected to the outputs of the neurons in the previous layer located in rows i to $i + f_h - 1$, columns j to $j + f_w - 1$, where f_h and f_w are the height and width of the receptive field. In order for a layer to have the same height and width as the previous layer, it is common to add zeros around the inputs, as shown in the diagram. This is called **zero padding**.

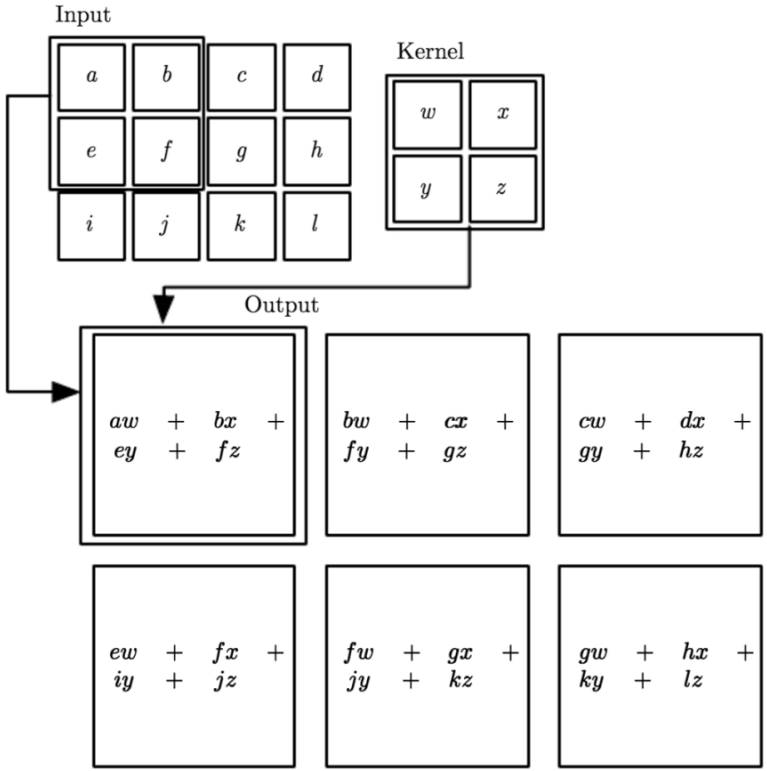


Reducing dimensionality using a stride of 2

It is also possible to connect a large input layer to a much smaller layer by spacing out the receptive fields. The distance between two consecutive receptive fields is called the **stride**. In the diagram, a 5 × 7 input layer (plus zero padding) is connected to a 3 × 4 layer, using 3 × 3 receptive fields and a stride of 2 (in this example the stride is the same in both directions, but it does not have to be so). A neuron located in row i, column j in the upper layer is connected to the outputs of the neurons in the previous layer located in rows $i \times s_h$ to $i \times s_h + f_h - 1$, columns $j \times s_w + f_w - 1$, where s_h and s_w are the vertical and horizontal strides.

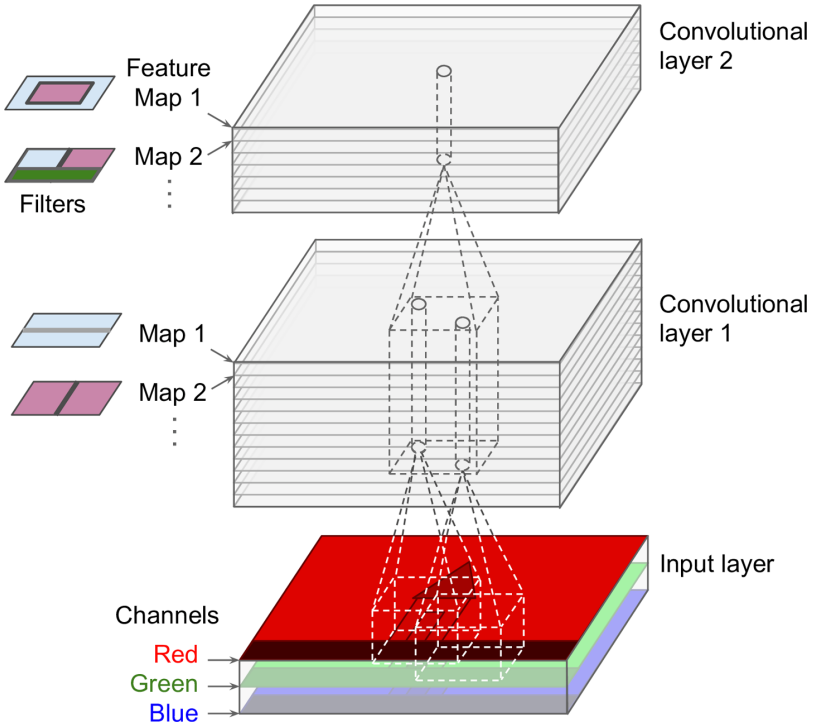


Filters


Input
a b c d
e f g h
i j k l
Kernel
w x
y z
Output
aw + bx +
ey + fz
bw + cx +
fy + gz
cw + dx +
gy + hz
ew + fx +
iy + jz
fw + gx +
jy + kz
gw + hx +
ky + lz

Convolution layers with multiple feature maps, and images with three channels

A neuron located in row i, column j of the feature map k in a given convolutional layer l is connected to the outputs of the neurons in the previous layer $l-1$, located in rows $i \times s_h$ to $i \times s_h + f_h - 1$ and columns $j \times s_w$ to $j \times s_w + f_w - 1$, across all feature maps (in layer $l-1$). Note that all neurons located in the same row i and column j but in different feature maps are connected to the outputs of the exact same neurons in the previous layer.



Computing the output of a neuron in a convolutional layer

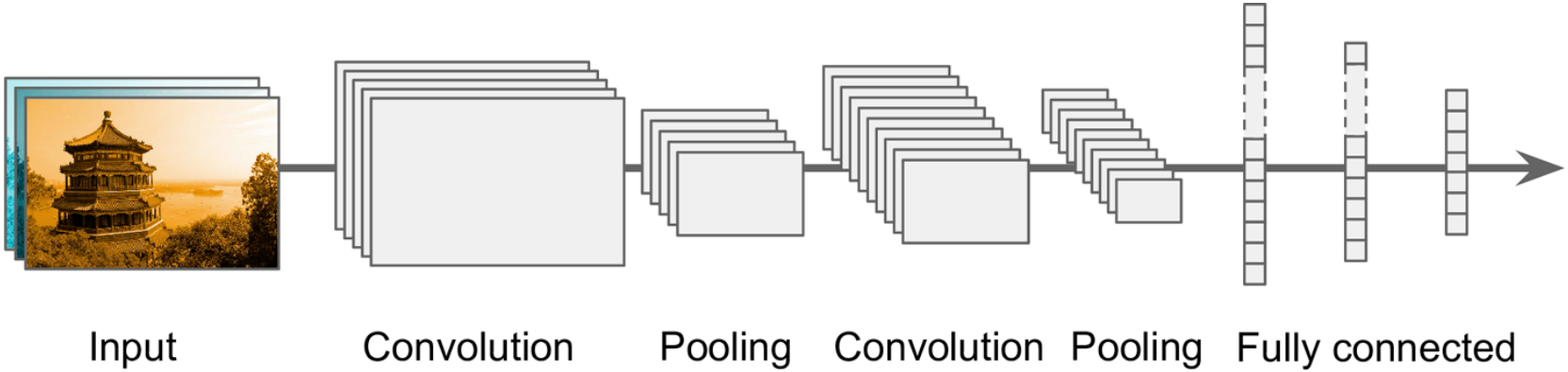
$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} . w_{u,v,k',k} \quad \text{with} \begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases}$$

- $z_{i, j, k}$ is the output of the neuron located in row *i*, column *j* in feature map *k* of the convolutional layer (layer *l*).
- As explained earlier, s_h and s_w are the vertical and horizontal strides, f_h and f_w are the height and width of the receptive field, and $f_{n'}$ is the number of feature maps in the previous layer (layer *l* – 1).
- $x_{i', j', k'}$ is the output of the neuron located in layer *l* – 1, row *i*′, column *j*′, feature map *k*′ (or channel *k*′ if the previous layer is the input layer).
- b_k is the bias term for feature map *k* (in layer *l*). You can think of it as a knob that tweaks the overall brightness of the feature map *k*.
- $w_{u, v, k', k}$ is the connection weight between any neuron in feature map *k* of the layer *l* and its input located at row *u*, column *v* (relative to the neuron's receptive field), and feature map *k*′.

TensorFlow Implementation

Each input image is typically represented as a 3D tensor of shape [height, width, RGB channels]. A mini-batch is represented as a 4D tensor of shape [f_h, f_w, $f_{n'}$, f_n]. The bias terms of a convolutional layer are simply represented as a 1D tensor of shape [f_n].

CNN Architecture



CNN Architecture

Typical CNN architectures stack a few convolutional layers (each one generally followed by a ReLU layer), then a pooling layer, then another few convolutional layers (+ReLU), then another pooling layer, and so on. The image gets smaller and smaller as it progresses through the network, but it also typically gets deeper and deeper (i.e., with more feature maps) thanks to the convolutional layers At the top of the stack, a regular feedforward neural network is added, composed of a few fully connected layers (+ReLUs), and the final layer outputs the prediction (e.g., a softmax layer that outputs estimated class probabilities).

A common mistake is to use convolution kernels that are too large. You can often get the same effect as one convolutional layer with a 9 × 9 kernel by stacking two layers with 3 × 3 kernels, for a lot less compute and parameters.

LeNet5 architecture for MNIST dataset

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg Pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg Pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

Why not use fully connected NN?

- Unfortunately, although fully connected neural network works fine for small images (e.g., MNIST), it breaks down for larger images because of the huge number of parameters it requires.
- For example, a 100 × 100 image has 10,000 pixels, and if the first layer has just 1,000 neurons (which already severely restricts the amount of information transmitted to the next layer), this means a total of 10 million connections. And that’s just the first layer. CNNs solve this problem using partially connected layers.

Recurrent Neural Networks

Models built to predict the future, for example,

- Analyze time series data such as stock prices, and tell you when to buy or sell
- In autonomous driving systems, they can anticipate car trajectories and help avoid accidents
- Analyze sentences, documents, or audio samples as input, use them for automatic translation, speech-to-text, sentiment analysis, generate sentences, image captions,

Deep Learning

- Deep learning is use a cascade of many layers of nonlinear processing units for feature extraction and transformation.
- Each successive layer uses the output from the previous layer as input.
- The algorithms may be supervised or unsupervised and applications include pattern analysis and classification.
- It is based on the learning of multiple levels of features or representations of the data.
- Higher level features are derived from lower level features to form a hierarchical representation.

Deep Learning

- It is a part of the broader machine learning field of learning representations of data.
- It learns multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- The composition of a layer of nonlinear processing units used in a deep learning algorithm depends on the problem to be solved.
- Layers that have been used in deep learning include hidden layers of an artificial neural network and sets of complicated propositional formulas.
- It may also include latent variables organized layer-wise in deep generative models such as the nodes in Deep Belief Networks and Deep Boltzmann Machines.

Deep Learning

- Deep learning algorithms are based on distributed representations.
- The underlying assumption behind distributed representations is that observed data are generated by the interactions of factors organized in layers.
- Deep learning adds the assumption that these layers of factors correspond to levels of abstraction or composition.
- Varying numbers of layers and layer sizes can be used to provide different amounts of abstraction.

Deep Learning

- Deep learning exploits this idea of hierarchical explanatory factors where higher level, more abstract concepts are learned from the lower level ones.
- These architectures are often constructed with a greedy layer-by-layer method.
- Deep learning helps to disentangle these abstractions and pick out which features are useful for learning.
- For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures which remove redundancy in representation.

Deep Learning

- Many deep learning algorithms are applied to unsupervised learning tasks.
- This is an important benefit because unlabeled data are usually more abundant than labeled data.
- Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks.

DNN As Service

Company	Cloud-Based ML Platform	DL Technology (Open Sourced)
Amazon	Amazon Machine Learning	DSSTNE
Baidu	-	Deep Speech 2 PaddlePaddle
Facebook	-	TorchNet, FastText
Google	NEXT Cloud	TensorFlow
IBM	Watson	IBM system
Microsoft	Azur	CNTK
Twitter	-	Cortex

- http://mscoco.org/home/
- http://www.image-net.org/

Image Databases

- MNIST dataset for digits (60,000 training and 10,000 testing)
- NORB database
- HWDB1.0 dataset for Chinese characters
- CIFAR10 (dataset of 60,000 32x32 labeled RGB images)
- ImageNet dataset
- QMUL Multiview Face Dataset

Deep Learning in Text Analytics

Dataset	Source	Labels	Statistical Models	CNN
Flipkart Twitter Sentiment	Twitter	Pos, Neg	85%	96%
Flipkart Twitter Sentiment	Twitter	Pos, Neg, Neu	76%	89%
Fine grained sentiment in Emails	Emails	Angry, Sad, Complaint, Request	55%	68%
SST2	Movie Reviews	Pos, Neg	79.4%	87.5%
SemEval Task 4	Restaurant Reviews	food / service / ambience / price / misc	88.5%	89.6%

https://youtu.be/04ev55WnvSg?t=27m31s

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

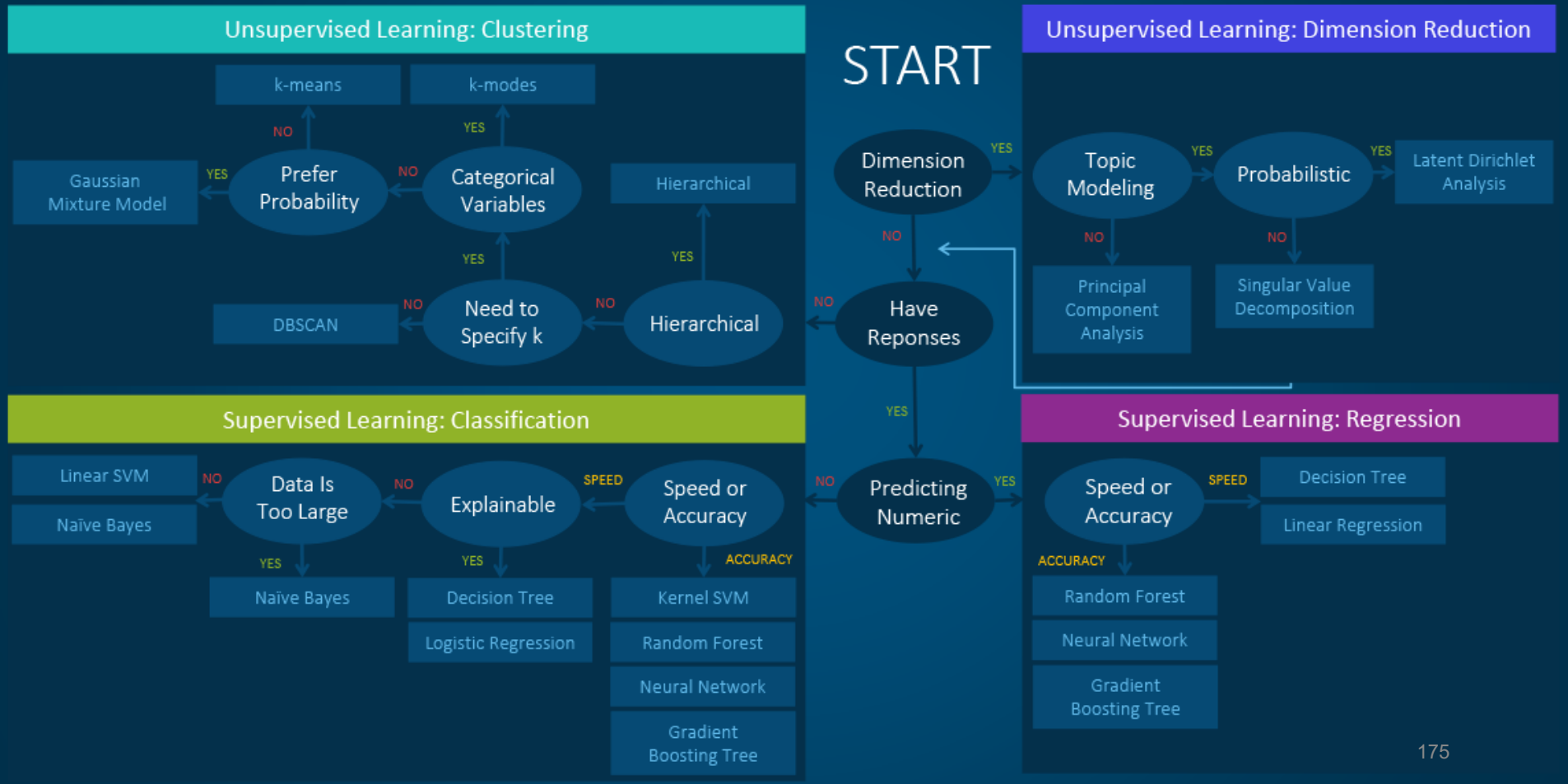
Image classification datasets

- http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#43494641522d3130 (Image classification datasets)

ML Platform

- ML Platform to serve offline (model training) and the online (model serving) needs
- On the online side, the ML Platform owns the systems that help ML engineers to build and deploy high performance, cost efficient, real time machine learning systems with high reliability and availability.
- On the offline side, the ML Platform team enables ML engineers to build data pipelines, do feature generation and train models in a fast, standardized and reusable way.

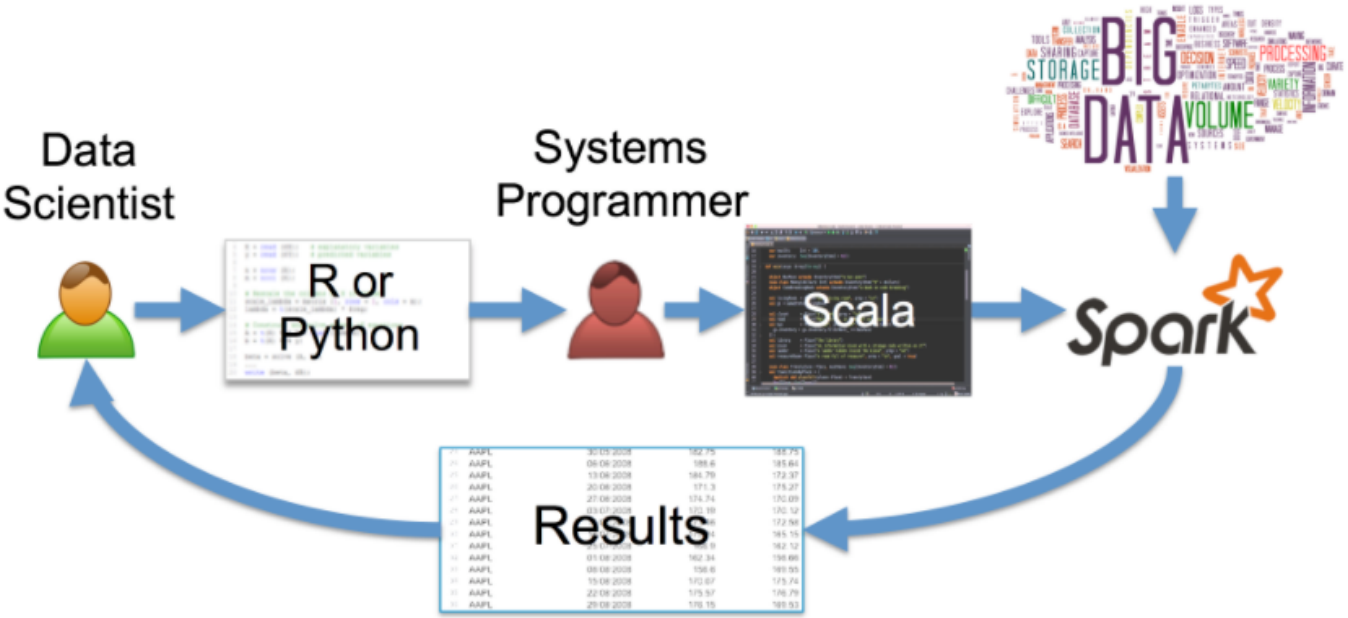
Machine Learning Algorithms Cheat Sheet



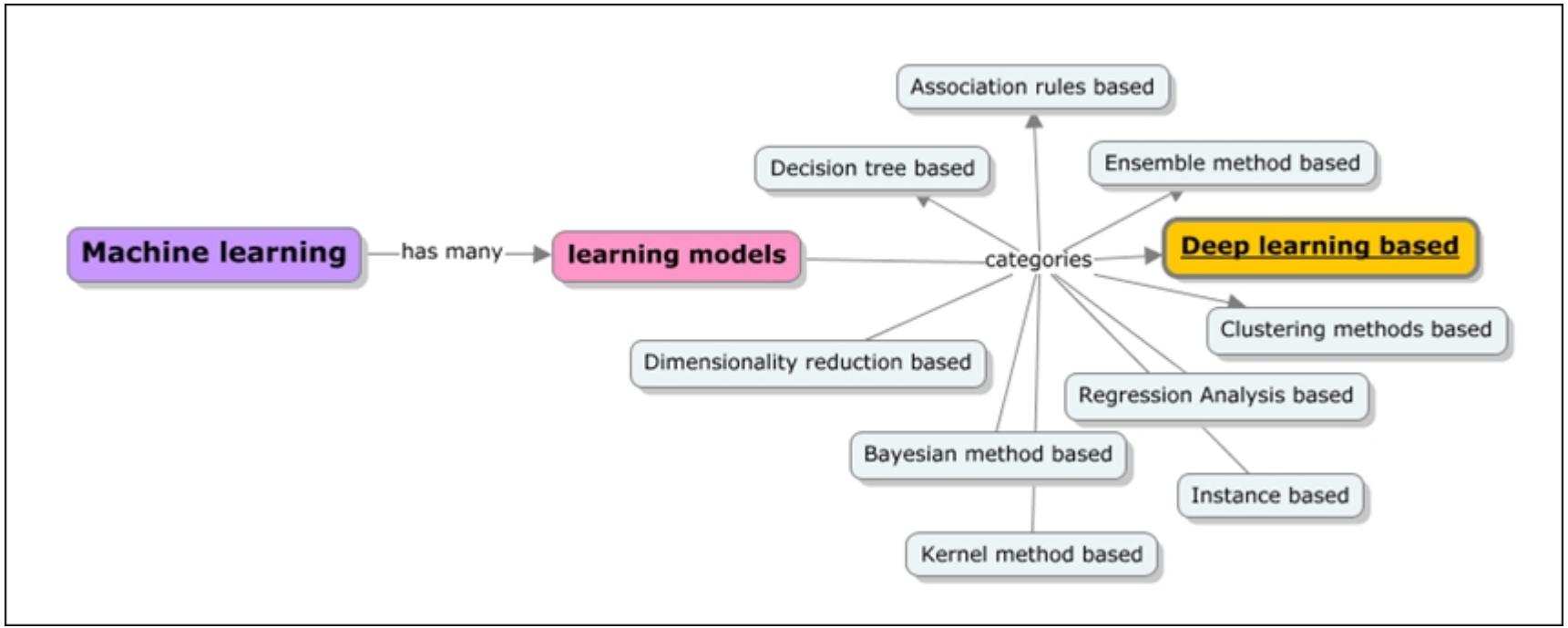
Challenges with machine learning in practice

- Training time
- Prediction time
- Performance metrics
- Training speed
  - By data volume, number of iterations, time
- Max memory requirement
- V-CPU-seconds
- Partial training
- CPU architecture
- Data volume
- Data formats
- Online training
- Serving machine learning model
  - Batch prediction
  - Online prediction
- Model portability
- Formation of training examples
- Confidence on training examples

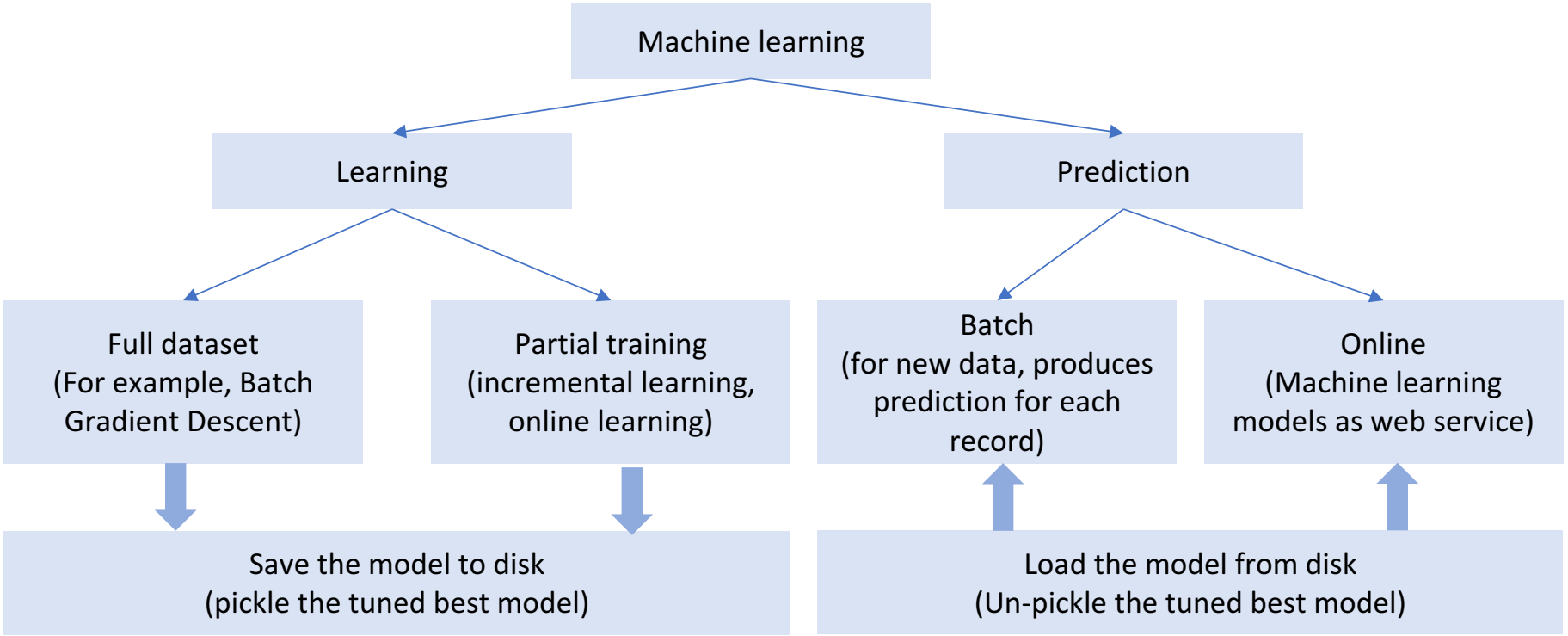
Data analysis pipeline



Concept Model of Machine Learning



Deployment Strategy



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 - scientific tools for Python

- 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

TensorFlow

- Processing is expressed in the form of computation graph
- TensorFlow automatically calculates the gradients
- TensorFlow can take advantage of multi core CPUs, GPUs and specialized TPU
- TensorFlow parts
  - Placeholders - takes input
  - Model variables - parameters that need to optimized
  - Cost measure - that is optimized
  - Optimization method

References

- Machine Learning Lecture Notes from MIT
- Machine Learning Lecture Notes from Stanford - Andrew Ng's class
- Challenges in Machine Learning and Data Mining
- Introduction to Statistical Learning
- Machine Learning Slides from Edx
- Matrix Cookbook
- Deep Learning Book
- Michael Nielsen's tutorials
- Convolutional Neural Network for Visual Recognition
- Basic Derivative Formula
- Oxford University Deep learning Course Material
- Deep Learning Mind Map
- MIT Course - Deep Learning for Self Driving Car
- CS231n: Convolutional Neural Networks for Visual Recognition
- Artificial Intelligence - A modern approach
- Mathematical Summary
- Kernel functions in machine learning
- Introduction to linear algebra
- Jupyter Notebook Tips
- Parallel Algorithms

Popular blogs of AI, ML

- https://machinelearningmastery.com/blog/
- https://blog.algorithmia.com/
- https://aitopics.org/search
- https://machinelearnings.co/
- https://chatbotsmagazine.com/
- https://chatbotslife.com/
- http://www.33rdsquare.com/
- https://openai.com/blog
- https://intelligence.org/blog/
- https://www.reddit.com/r/artificial/
- https://aitopics.org/search
- https://machinelearnings.co/
- https://www.artificial-intelligence.blog/

- Allen Institute for Artificial Intelligence
- **https://www.reddit.com/r/singularity/**
- **http://research.baidu.com/**
- **https://www.artificiallawyer.com/**
- **http://www.expertsystem.com/blog/**
- *https://aws.amazon.com/blogs/ai*
- **https://medium.com/ai-roadmap-institute**
- **https://deepmind.com/blog/**
- **https://www.inbenta.com/en/blog/**
- **https://blog.clarifai.com/**
- **https://www.datarobot.com/blog/**
- **https://medium.com/archieai**
- **https://www.singularityweblog.com/blog/**
- **https://iris.ai/blog/**
- **https://blogs.nvidia.com/**

Statistics vs Machine Learning

OpenIntro 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 Course - Andrew Ng

- 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

Statistics vs Machine Learning

Statistics

In Statistics a prime focus is often in understanding the data and relationships in terms of models giving approximate summaries such as linear relations or independencies.

Machine Learning

The goals in machine learning are primarily to make predictions as accurately as possible and predictions to understand the behavior of learning algorithms.

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

Deep Learning - Goodfellow