# Resources

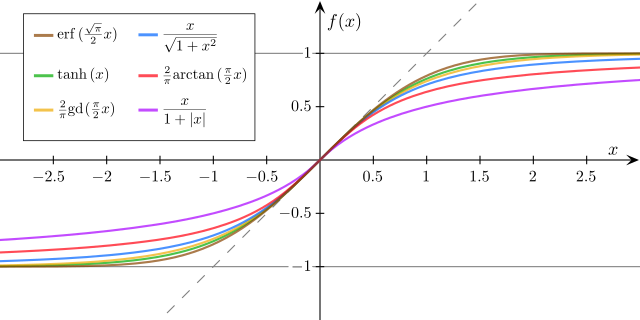
Andrew Moore’s slides: <http://www.cs.cmu.edu/~./awm/tutorials/index.html>

# Logistic Regression 4/15/2019

uses a logistic / sigmoid function (of a linear function) to do binary classification

### Sigmoid function

The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

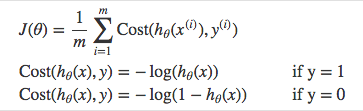


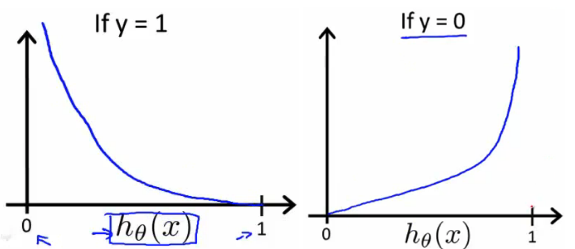
same update rule as least mean square

### Loss function:

Can’t use MSE because of sigmoid, the MSE results in a non-convex function with many local minimums, thus cannot use gradient descent to find the global optimal solution.

Use cross-entropy loss instead.





Cost function



### Multiclass logistic regression

Instead of \(y = {0,1}\) we will expand our definition so that \(y = {0,1...n}\). Basically we re-run binary classification multiple times, once for each class.

Procedure

1. Divide the problem into n+1 binary classification problems (+1 because the index starts at 0?).
2. For each class…
3. Predict the probability the observations are in that single class.
4. prediction = <math>max(probability of the classes)

For each sub-problem, we select one class (YES) and lump all the others into a second class (NO). Then we take the class with the highest predicted value.

## MLE vs MAP

Maximum likelihood estimate: argmax P(Y | X, W)

Maximum a posterior: argmax P(W | X, Y)

Bayes Rule

MAP same as MLE when

* infinite amount of data
* uniform prior (infinitely weak prior belief)

MLE used with regularization, MAP uses prior like a regularization

Generative vs Discriminative

Generative model: joint probability p(x, y)

discriminative mode: learn p(y | x) directly

## TODO

Generative vs Discriminative

* Naive Bayes
* Logistic Regression

Convex vs Non-convex

This looks like a good resource for future study

<https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html>

# HDFS 5/3/2019

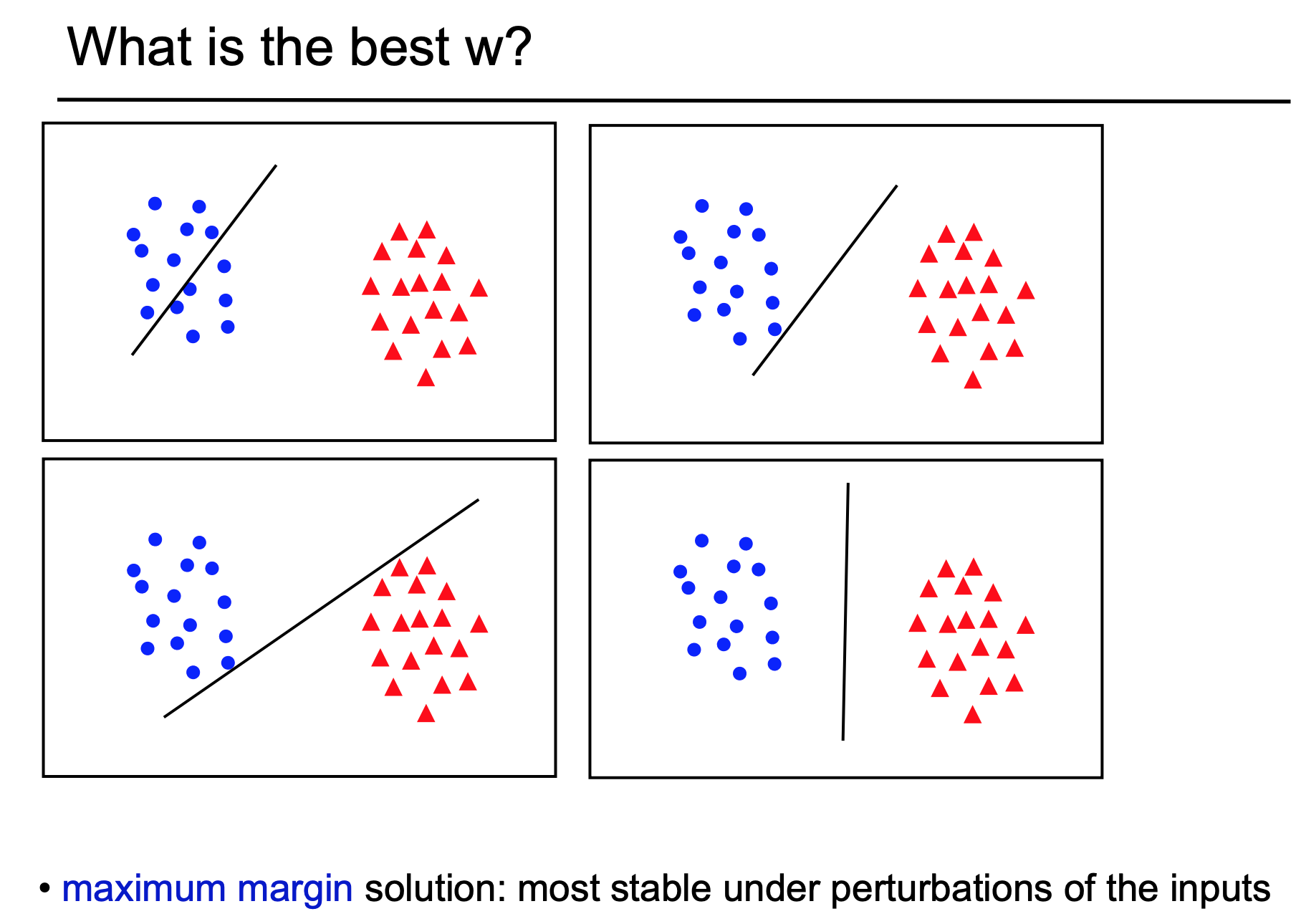
Homework Questions:

1. Can you update a file on HDFS. If yes, how is it done? [the block may not be pointed to any longer but what if you update a small chunk of the block?]
2. What exactly is in fsimage? What is the merge of edit logs and fsimage?
3. How do we know what the ‘closest’ location to the user is?

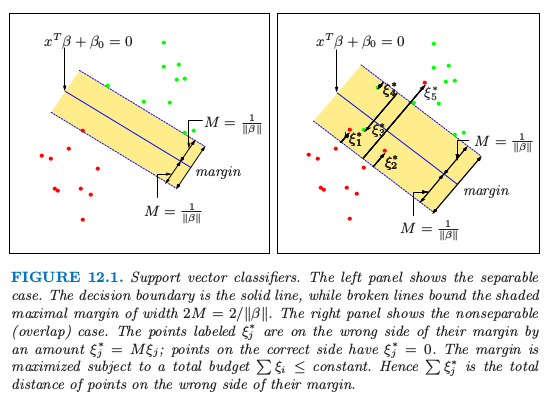
# SVM 5/13/2019

### Definition

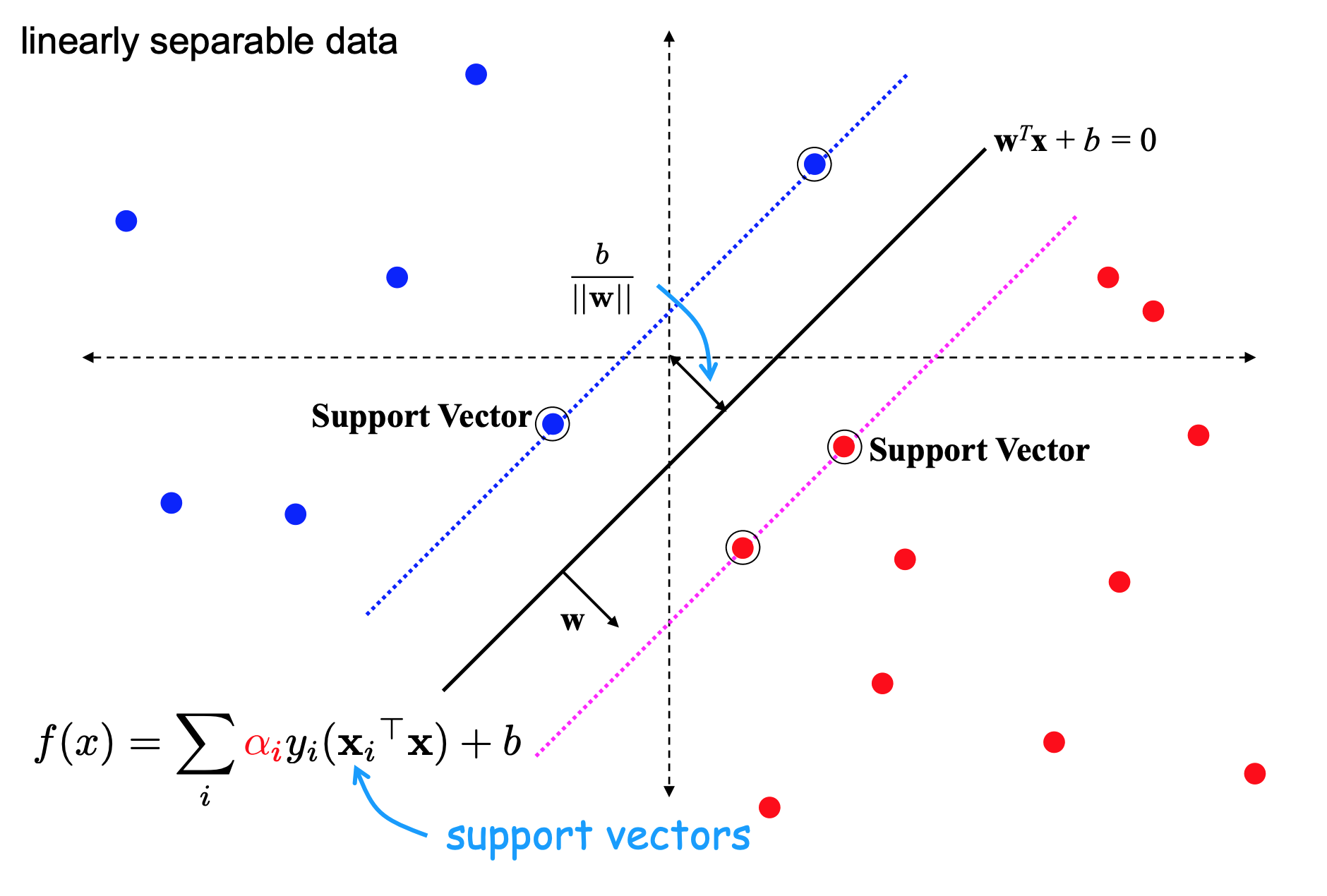
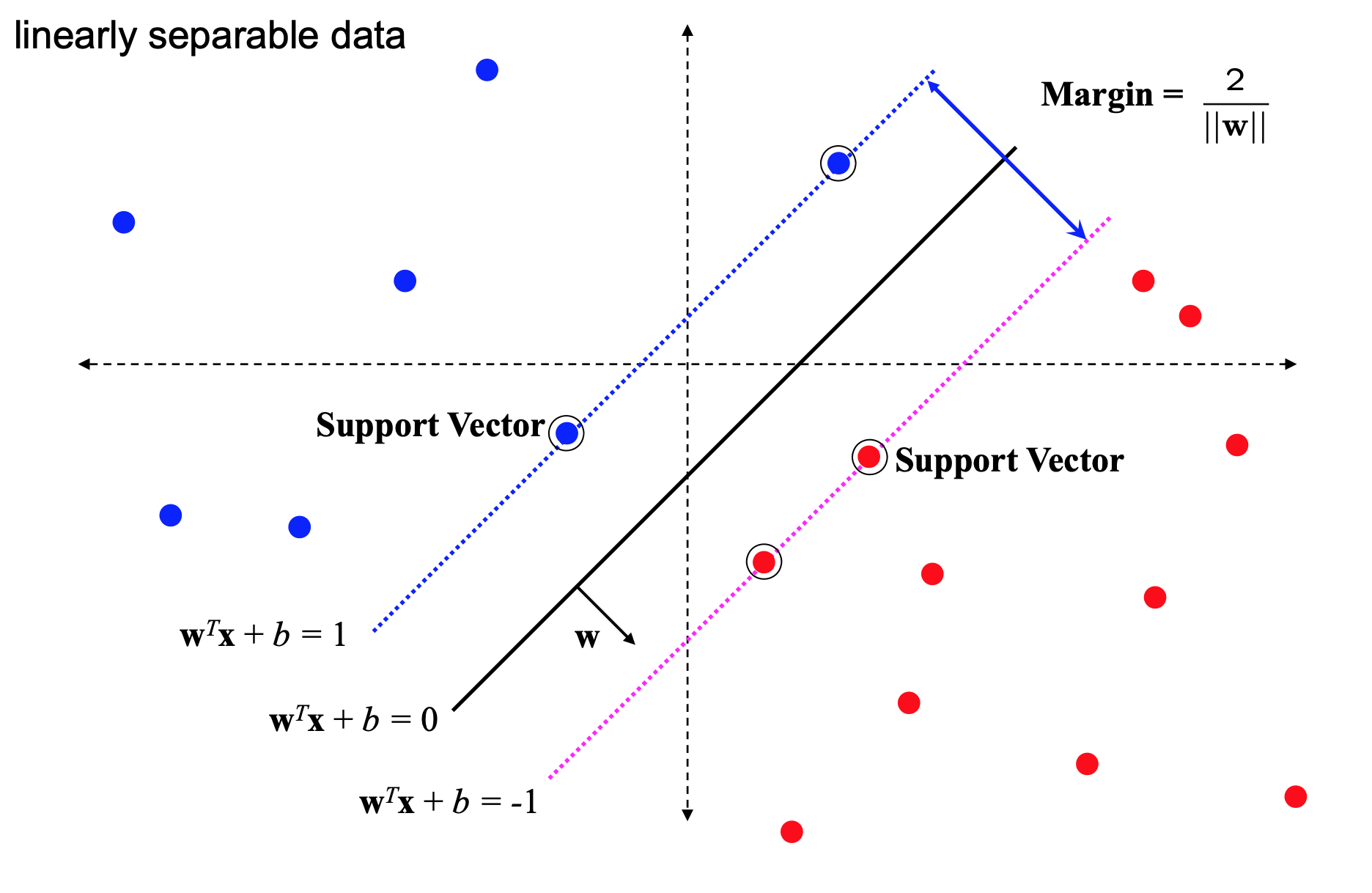
* Maximum margin Classification
  + fit the maximum-margin hyperplane in a transformed feature space
  + Produces nonlinear boundaries by constructing a linear boundary in a large, transformed version of the feature space
  + linear or non-linear
  + Find a hyperplane with a large margin for all classes
  + a line L1 is said to be a better classifier than line L2, if the “margin” of L1 is larger i.e., L1 is farther from both classes.
  + Generalize better when margin is large



* Support vector
  + Data points on the wrong side of the boundary
  + Data points on the correct side of the boundary but close to it (in the margin)

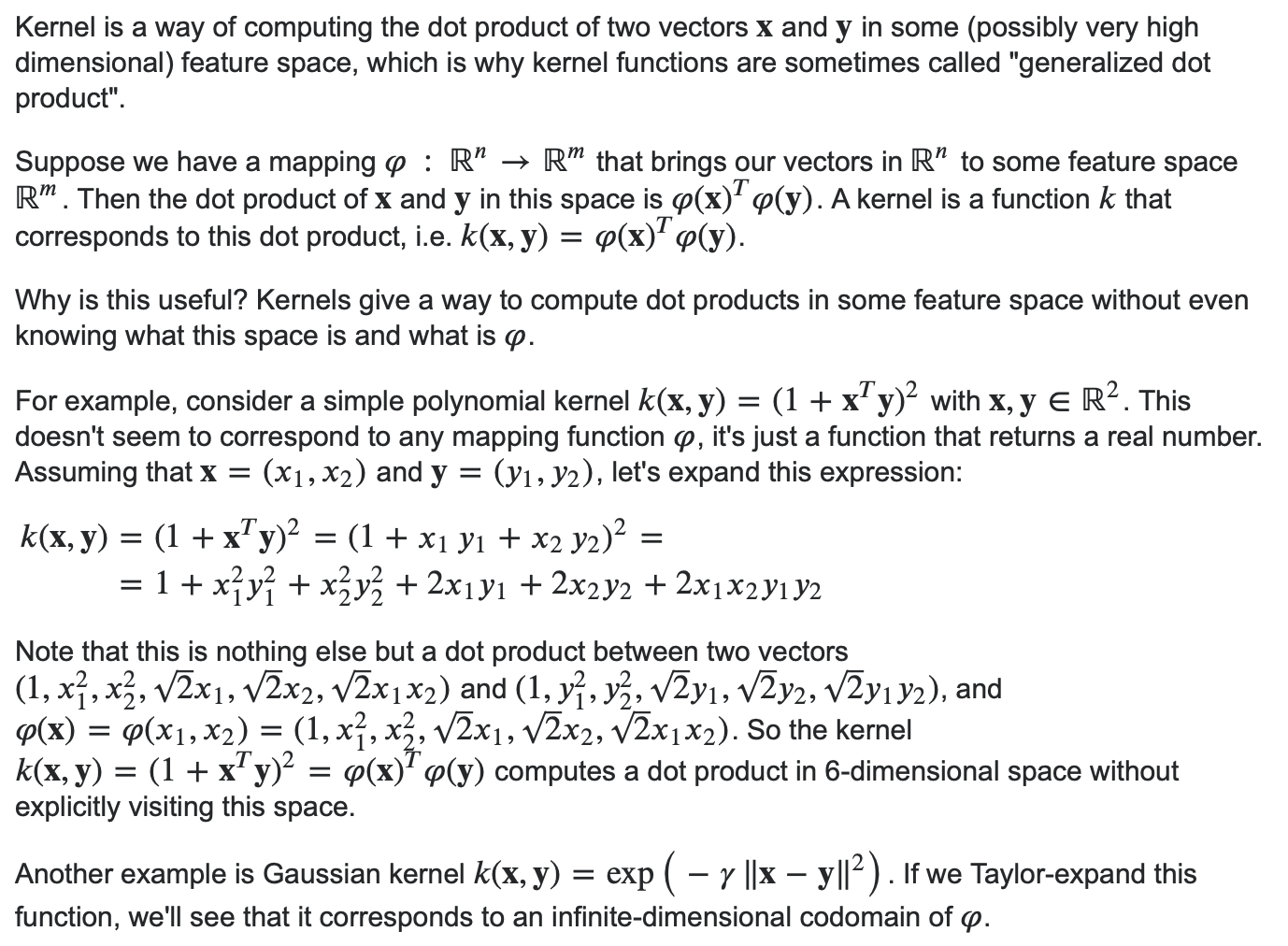


* Kernel
  + Feature transformation



* Can also be used for regression: Support Vector Regression (SVR)

### Parameters

* Regularization - C
  + Trade off between margin’s size and #misclassifications in training set
  + larger -> smaller margin, less regularization, less misclassification, more overfitting, bias
  + smaller -> larger margin, more regularization, more misclassification, less overfitting, variance, smooth decision surface
  + C = 1/lambda where lambda is the regularization parameter
  + Choose through Cross validation
* Kernel
  + transformation to a higher dimensional space with a linear boundary
  + Generalized inner product
  + Linear: ⟨x,x′⟩
  + Polynomial
    - (gamma⟨x,x′⟩+r)^d, d is specified by keyword degree, r by coef0.
  + Rbf: radial basis function
    - A radial basis function (RBF) is a real-valued function whose value depends only on the distance from the origin; or the distance from some other point , called a center
    - exp⁡(−gamma‖x−x′‖^2)
    - usually Euclidean distance
    - The Gaussian kernel is a specific example of a radial basis function
  + Sigmoid
    - tanh(gamma(x, x’) + r
  + 
  + A very simple and intuitive way of thinking about kernels (at least for SVMs) is a similarity function. Given two objects, the kernel outputs some similarity score. The objects can be anything starting from two integers, two real valued vectors, trees whatever provided that the kernel function knows how to compare them.

The arguably simplest example is the linear kernel, also called dot-product. Given two vectors, the similarity is the length of the projection of one vector on another.

Another interesting kernel examples is Gaussian kernel. Given two vectors, the similarity will diminish with the radius of 𝜎. The distance between two objects is "reweighted" by this radius parameter.

The success of learning with kernels (again, at least for SVMs), very strongly depends on the choice of kernel. You can see a kernel as a compact representation of the knowledge about your classification problem. It is very often problem specific.

I would not call a kernel a decision function since the kernel is used *inside* the decision function. Given a data point to classify, the decision function makes use of the kernel by comparing that data point to a number of support vectors weighted by the learned parameters 𝛼. The support vectors are in the domain of that data point and along the learned parameters 𝛼 are found by the learning algorithm.

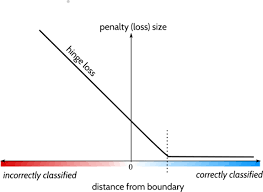
* Gamma?
  + how close a point needs to be from the hyperplane to be included in the calculation
  + How much influence a single training example has
  + Support vector: subset of the training data
  + Controls the position and orientation of the hyperplane

### Loss Function - Hinge loss



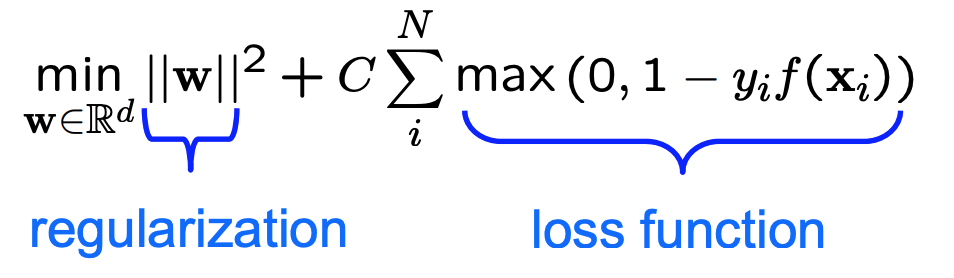


* Convex
  + Local optimal == global optimal
* Not differentiable (not smooth) -> can’t be used with gradient descent
* Correctly classified points lying outside the margin boundaries of the support vectors are not penalized, whereas points within the margin boundaries or on the wrong side of the hyperplane are penalized in a linear fashion compared to their distance from the correct boundary.



### Optimization

* Regularization
  + L2 norm
  + increase the cost penalty if the values for the weights are high
* Loss function - hinge loss



* Weights
  + For a general kernel it is difficult to interpret the SVM weights, however for the linear SVM there actually is a useful interpretation:
  + 1) Recall that in linear SVM, the result is a hyperplane that separates the classes as best as possible. The weights represent this hyperplane, by giving you the coordinates of a vector which is orthogonal to the hyperplane - these are the coefficients given by svm.coef\_. Let's call this vector w.
  + 2) What can we do with this vector? It's direction gives us the predicted class, so if you take the dot product of any point with the vector, you can tell on which side it is: if the dot product is positive, it belongs to the positive class, if it is negative it belongs to the negative class.
  + 3) Finally, you can even learn something about the importance of each feature. This is my own interpretation so convince yourself first. Let's say the svm would find only one feature useful for separating the data, then the hyperplane would be orthogonal to that axis. So, you could say that the absolute size of the coefficient relative to the other ones gives an indication of how important the feature was for the separation. For example if only the first coordinate is used for separation, w will be of the form (x,0) where x is some non zero number and then |x|>0.
* Support Vector Machine algorithms are not scale invariant, so **it is highly recommended to scale your data**. For example, scale each attribute on the input vector X to [0,1] or [-1,+1], or standardize it to have mean 0 and variance 1. Note that the *same* scaling must be applied to the test vector to obtain meaningful results.

### Comparison with logistic regression

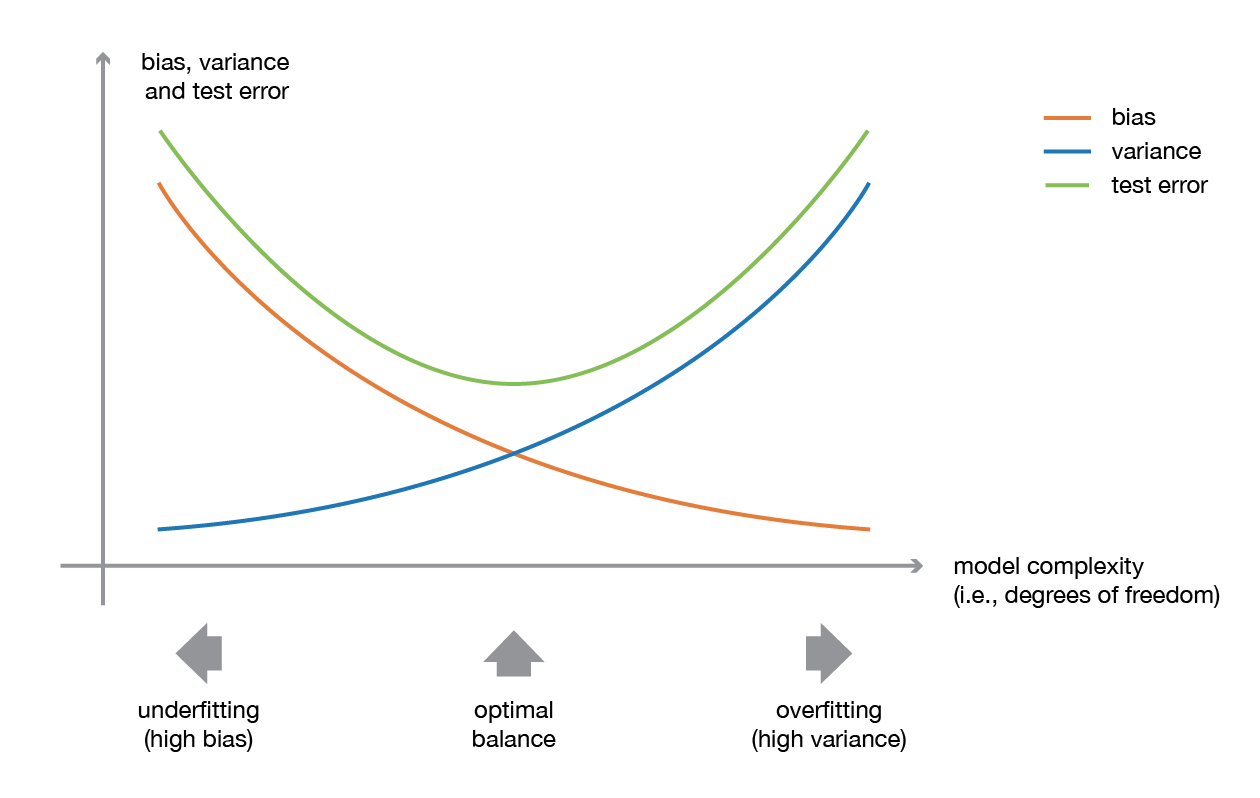
* SVM is less sensitive to outliers because logistic loss is much higher than hinge lss

### Epoch vs iterations vs batch size

* Epoch: times the whole dataset is used
* Iterations: number of batches
* Batch size: number of training examples used in each iteration

### Bias vs Variance

* j



## Question

* Can SVM work on multi-classification problem?

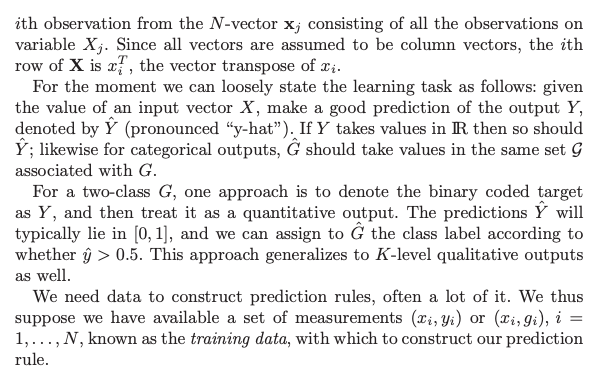
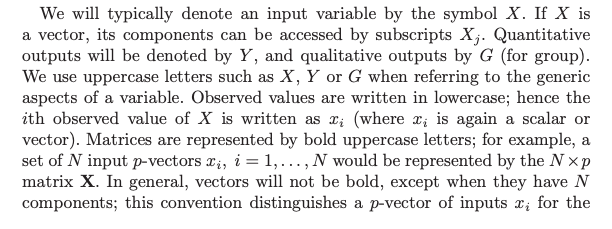
Yes, train multiple binary SVM and do one-vs-all or one-vs-one

SVC: one-vs-one for multi-class

LinearSVC: one-vs-all for multi-class

* <https://www.quora.com/What-are-kernels-in-machine-learning-and-SVM-and-why-do-we-need-them>
* <http://www.cs.cmu.edu/~./awm/tutorials/svm15.pdf>

Notes on Notation for Chinmayee



Notes:

# Linear Regression

## Assumptions

1. Linearity
   1. Linear relationship between variables and target
   2. sensitive to outliers -> should remove outliers
2. Fff
   1. Variance of errors should be constant
3. No correlation between errors
   1. Independent errors
4. Normality of errors distribution
   1. Most errors are closer to 0 with few outliers

## Problems

1. Overfitting
   1. Regularization: favors simpler linear model with fewer parameters
2. Multicollinearity
   1. Correlation between 2 or more variables
   2. Difficult to determine the impact of individual variables
   3. Use domain knowledge to choose variables and trial & error

# Bias 6/17/2019

### Bias (ethics/fairness)

Stereotyping, prejudice or favoritism towards some things, people, or groups over others. These biases can affect collection and interpretation of data, the design of a system, and how users interact with a system. Forms of this type of bias include:

* automation bias
* confirmation bias
* experimenter’s bias
* group attribution bias
* implicit bias
* in-group bias
* out-group homogeneity bias

### 2. Systematic error introduced by a sampling or reporting procedure.

* coverage bias
* non-response bias
* participation bias
* reporting bias
* sampling bias
* selection bias

### 3. Bias term

An intercept or offset from an origin. Bias (also known as the bias term) is referred to as b or w\*\*0 in machine learning models. For example, bias is the b in the following formula:

y′=b+w1x1+w2x2+…wnxn

If the expected value of your outcome variable is not 0, then we would also need to estimate the bias weight (i.e., the shift from 0) along with the feature weights. If we don't account for this bias, then any predictions from our estimated model will be off.

In addition, you could also force the expected value of your outcomes to equal 0 by "demeaning" (i.e., subtracting the mean of your outcomes from every outcome).

In linear regression, without the bias term your solution has to go through the origin. That is, when all of your features are zero, your predicted value would also have to be zero. Adding a bias weight that does not depend on any of the features allows the hyperplane desbribed by your learned weights to more easily fit data that doesn't pass through the origin.

Because different features do not have similar ranges of values and hence gradients may end up taking a long time and can oscillate back and forth and take a long time before it can finally find its way to the global/local minimum. To overcome the model learning problem, we normalize the data. We make sure that the different features take on similar ranges of values so that gradient descents can converge more quickly.

The bias value allows the activation function to be shifted to the left or right, to better fit the data. Hence changes to the weights alter the steepness of the sigmoid curve, whilst the bias offsets it, shifting the entire curve so it fits better. Note also how the bias only influences the output values, it doesn’t interact with the actual input data.

### 4. Model Bias

is also used to refer to the systematic errors in a model, the tendency to keep getting the same results incorrectly. This tends to arise when a model is too simple, and so fails to represent the complexity of the underlying data, causing an algorithm to miss relevant relations between salient features and the expected outputs, despite having more than enough training data — a problem called underfitting.

## Normalization

It's simply a case of getting all your data on the same scale: if the scales for different features are wildly different, this can have a knock-on effect on your ability to learn (depending on what methods you're using to do it). Ensuring standardized feature values implicitly weights all features equally in their representation.

It is true that preprocessing in machine learning is somewhat a very black art. It is not written down in papers a lot why several preprocessing steps are essential to make it work. I am also not sure if it is understood in every case. To make things more complicated, it depends heavily on the method you use and also on the problem domain.

Some methods e.g. are affine transformation invariant. If you have a neural network and just apply an affine transformation to your data, the network does not lose or gain anything in theory. In practice, however, a neural network works best if the inputs are centered and white. That means that their covariance is diagonal and the mean is the zero vector. Why does it improve things? It is only because the optimization of the neural net works more gracefully, since the hidden activation functions don't saturate that fast and thus do not give you near zero gradients early on in learning.

Other methods, e.g. K-Means, might give you totally different solutions depending on the preprocessing. This is because an affine transformation implies a change in the metric space: the Euclidean distance between two samples will be different after that transformation.

At the end of the day, you want to understand what you are doing to the data. E.g. whitening in computer vision and sample wise normalization is something that the human brain does as well in its vision pipeline.

### Standardization vs Normalization

Normalization rescales the values into a range of [0,1]. This might be useful in some cases where all parameters need to have the same positive scale. However, the outliers from the data set are lost.

Standardization rescales data to have a mean (𝜇) of 0 and standard deviation (σ) of 1 (unit variance).

For most applications standardization is recommended.

### When does normalization matter? Is it only distance based methods?

**SVM**

The answer to your question depends on what similarity/distance function you plan to use (in SVMs). If it's simple (unweighted) Euclidean distance, then if you don't normalize your data you are unwittingly giving some features more importance than others.

For example, if your first dimension ranges from 0-10, and second dimension from 0-1, a difference of 1 in the first dimension (just a tenth of the range) contributes as much in the distance computation as two wildly different values in the second dimension (0 and 1). So by doing this, you're exaggerating small differences in the first dimension. You could of course come up with a custom distance function or weight your dimensions by an expert's estimate, but this will lead to a lot of tunable parameters depending on dimensionality of your data. In this case, normalization is an easier path (although not necessarily ideal) because you can at least get started.

Finally, still for SVMs, another thing you can do is come up with a similarity function rather than a distance function and plug it in as a kernel (technically this function must generate positive-definite matrices). This function can be constructed any way you like and can take into account the disparity in ranges of features.

**Random Forests**

For random forests on the other hand, since one feature is never compared in magnitude to other features, the ranges don't matter. It's only the range of one feature that is split at each stage.

Scaling is done to Normalize data so that priority is not given to a particular feature. Role of Scaling is mostly important in algorithms that are distance based and require Euclidean Distance. Random Forest is a tree-based model and hence does not require feature scaling.This algorithm requires partitioning, even if you apply Normalization then also> the result would be the same.

Random Forest uses information gain / gini coefficient inherently which will not be affected by scaling unlike many other machine learning models which will (such as k-means clustering, PCA etc). However, it might 'arguably' fasten the convergence as hinted in other answers.

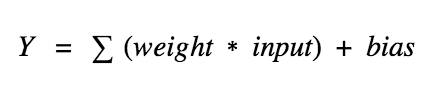
**Discussion:** Writing better Python code. Learning OOP Python. Design Patterns. Python Morsels.

Link: <https://python-patterns.guide/>

## 06/25/2019

### Activation Function

What does artificial neuron do? Simply put, it calculates a “weighted sum” of its input, adds a bias and then decides whether it should be “fired” or not .



What does activation function do?

To check the Y value produced by a neuron and decide whether outside connections should consider this neuron as “fired” or not. Or rather let’s say — “activated” or not.

## 06/26/2019

### Relational Database

Relation: relation between rows and columns

Used for tidy data:

<https://vita.had.co.nz/papers/tidy-data.pdf>

* Each variable forms a column
* Each observation forms a row/record
* Each type of observation unit forms a table

Use relational database when…

* The data is already in a DB
* You have a large amount of data, can’t fit into local memory
  + Can do operations and even machine learning (BigQuery) in the DB
* Many people need to access/change the data

Don’t use relational database when…

* Very small amount of data
* Not tabular data: images, text, hierarchical data structures -> NoSQL/blobs (for binary files)

BigQuery

* Relational database service from Google
* a web service that lets you apply SQL to huge datasets
* Spotify, Disney uses BigQuery