Lecture 8: Linear Classifiers and More Model Validation

INFO 1998: Introduction to Machine Learning

CDS Education

We explore, learn, and educate big minds.

Agenda

1. Perceptron + SVM
2. More Cross-Validation techniques

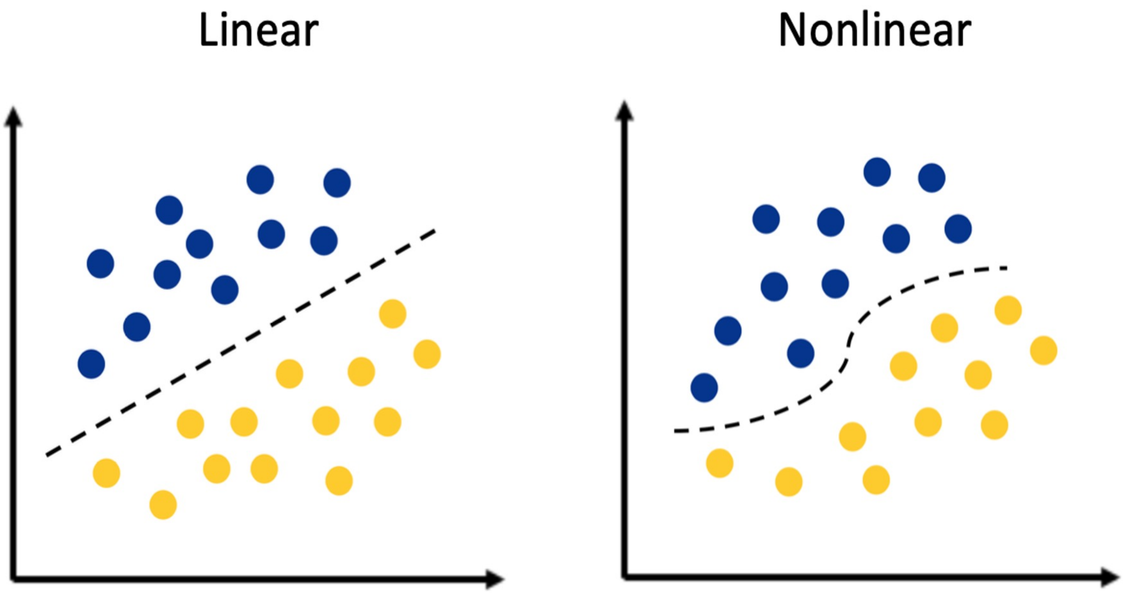
Linear Classifiers

Source

Linear Classifiers

A linear classifier is a hyper plane that is used to classify our data points

A hyperplane is our **decision boundary** and our goal is to find the hyper plane that best classifies our data



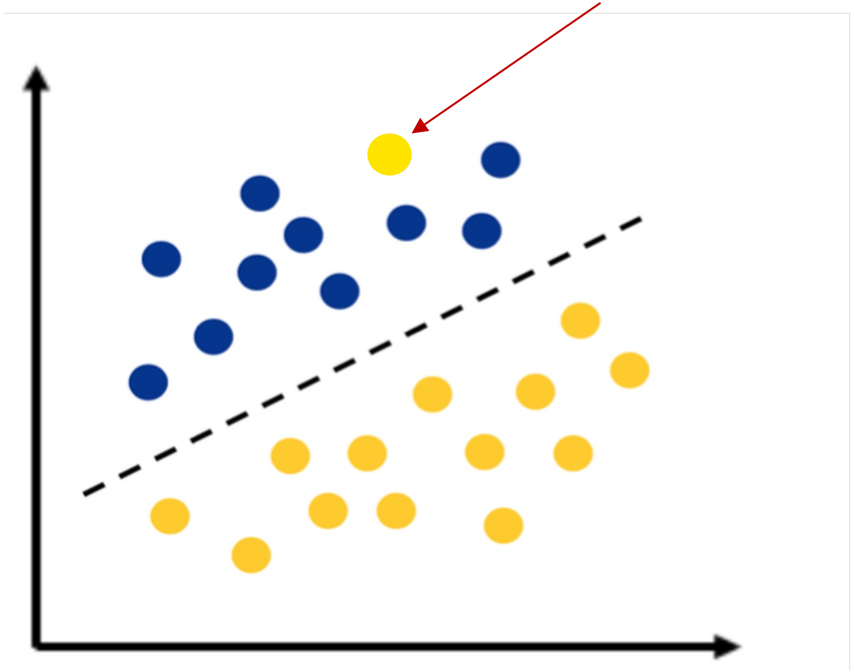
Source

Linearly Separable

In this example, we cannot partition our dataset into yellow and purple with a linear decision boundary. This means that our data is not **linearly separable.**

Outliers are frequently the reason a data set is not linearly separable.

This data set is not linearly separable because of an outlier



Perceptron Learning Algorithm

Goal: find a normal vector w that perfectly classifies all the points in our data set

Algorithm:

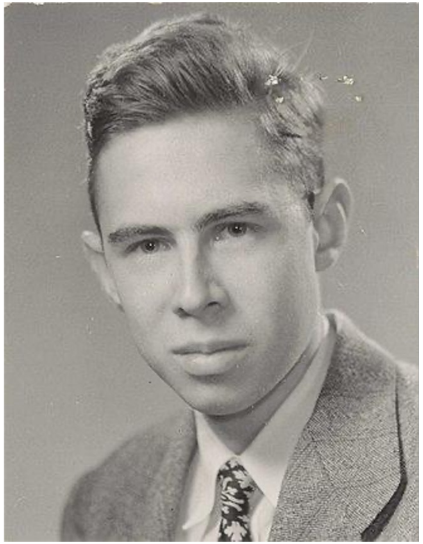
Initialize classifier as some random hyperplane
While there exists a misclassified point x:
    Tilt classifier slightly so that it classifies x correctly
    (or, is a little closer to classifying x correctly)
End While

"Use your mistakes as your stepping stones"

Perceptron in action here

Also, Frank Rosenblatt was first to implement perceptron

Gave him the title of ‘Father of Deep Learning’



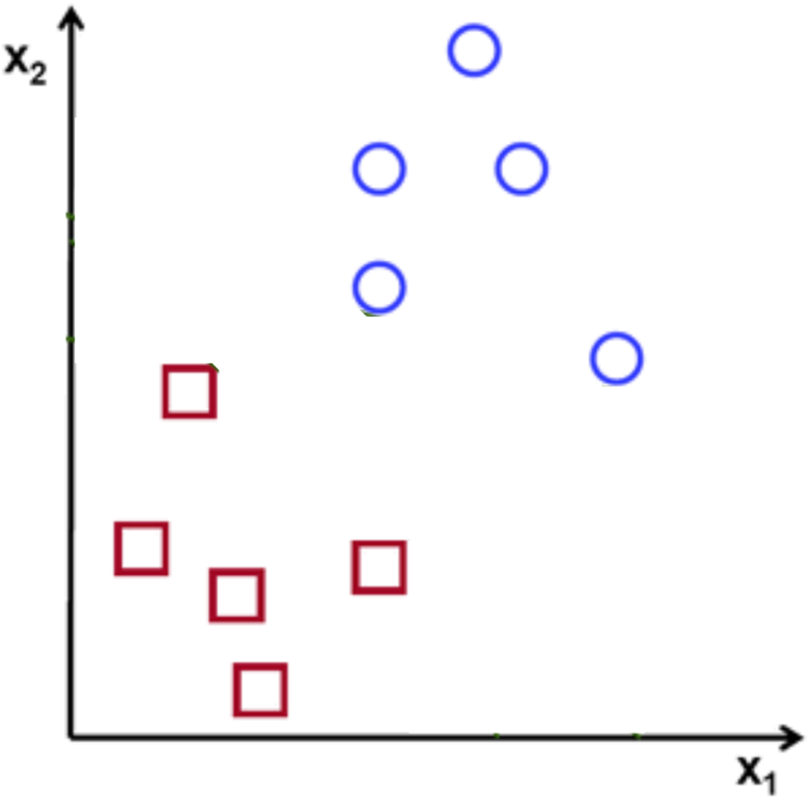
Limitations of Perceptron

The training algorithm will never terminate if your training dataset is not linearly separable ☹️

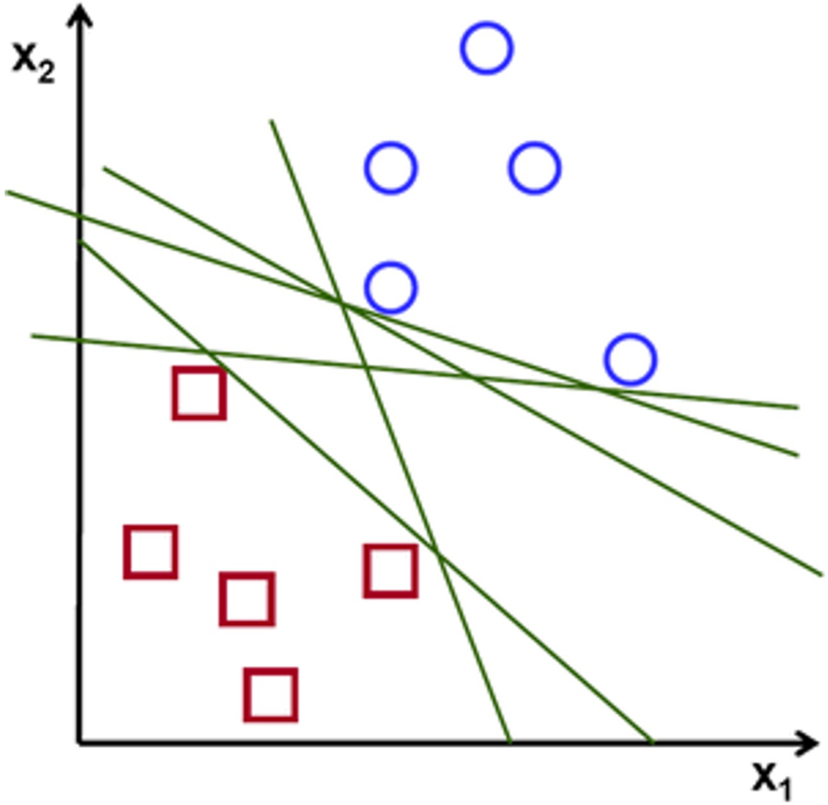
Is a great model to understand the intuition behind the training of a linear classifier: iteratively improve classifier by using misclassified points ☺

SVM

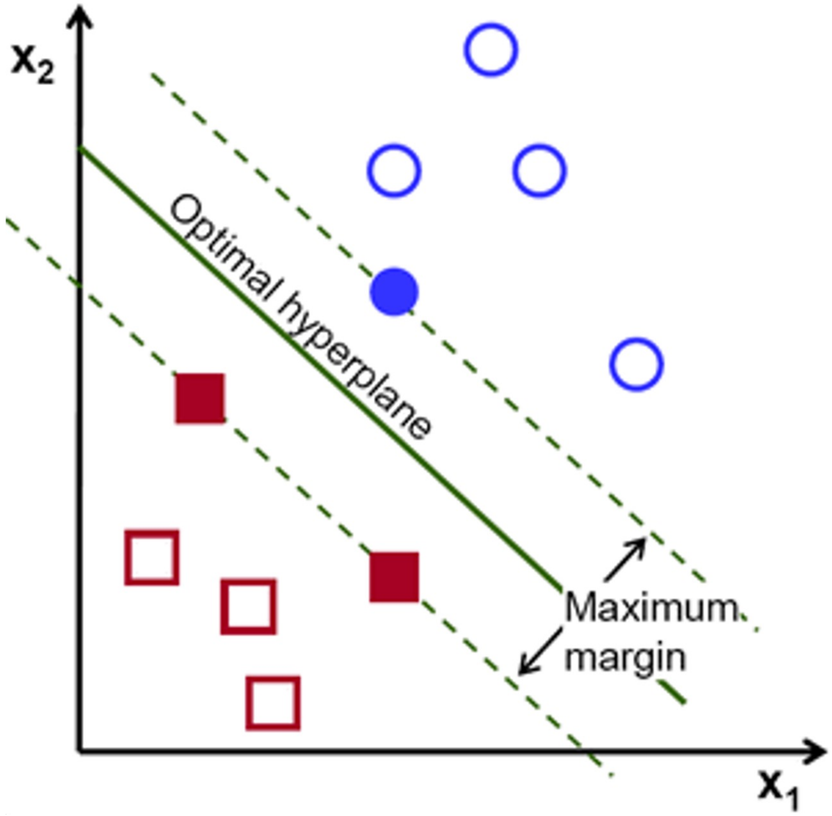
Classify (+) and (-)



Which Hyperplane?

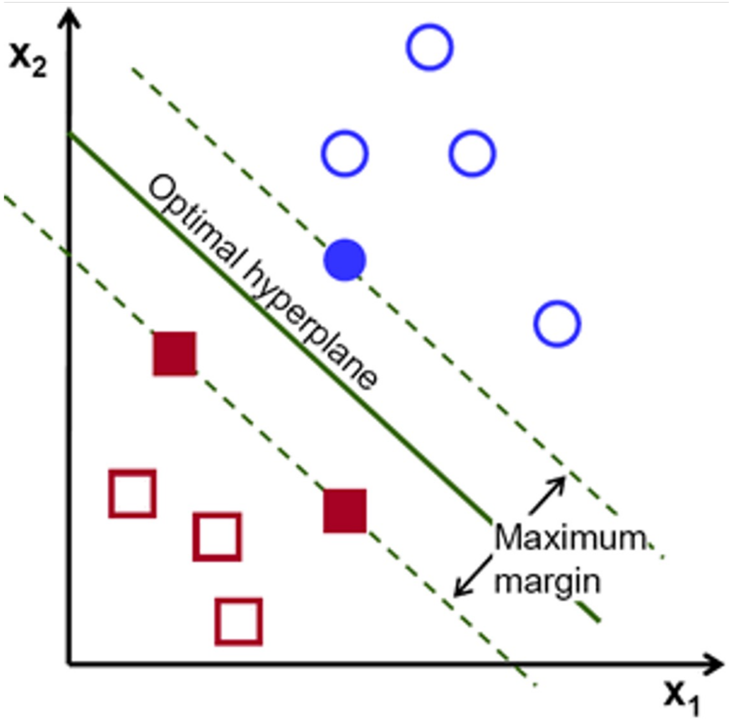


Optimal Hyperplane

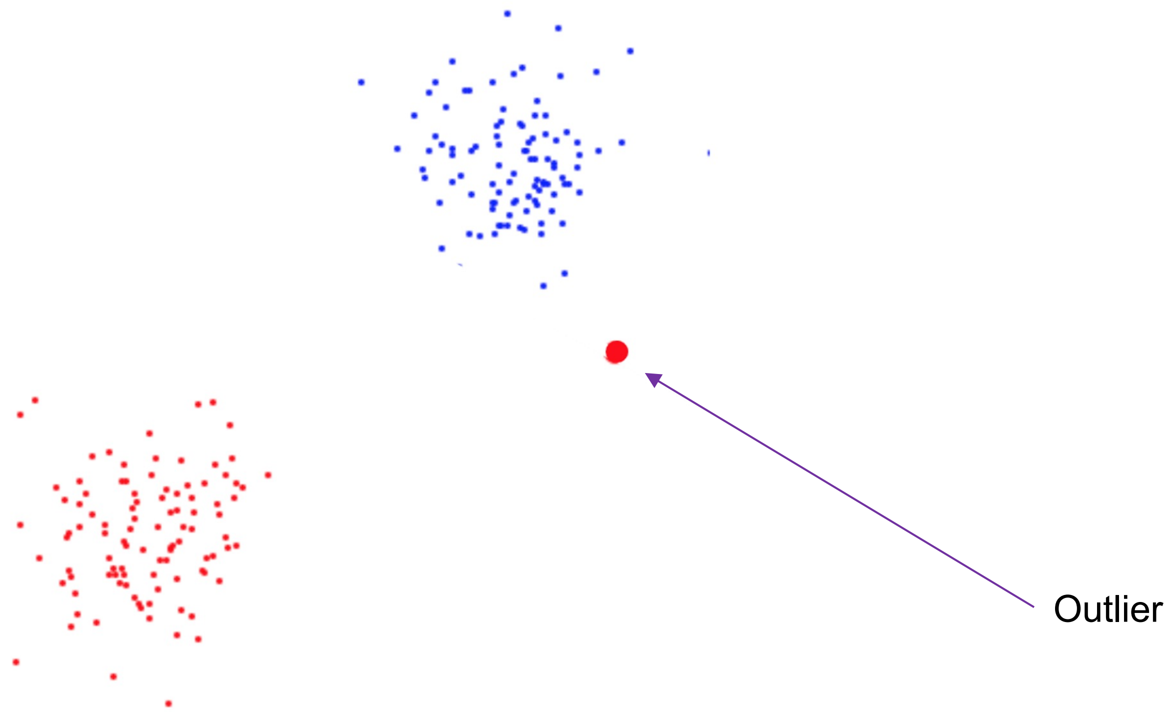


Maximal Margin Classifier

- We want to find a **separating hyperplane**
- Once we find candidates for the hyperplane, we try to maximize the **margin**, the normal distance from borderline points
  - Only **Support Vectors** matter

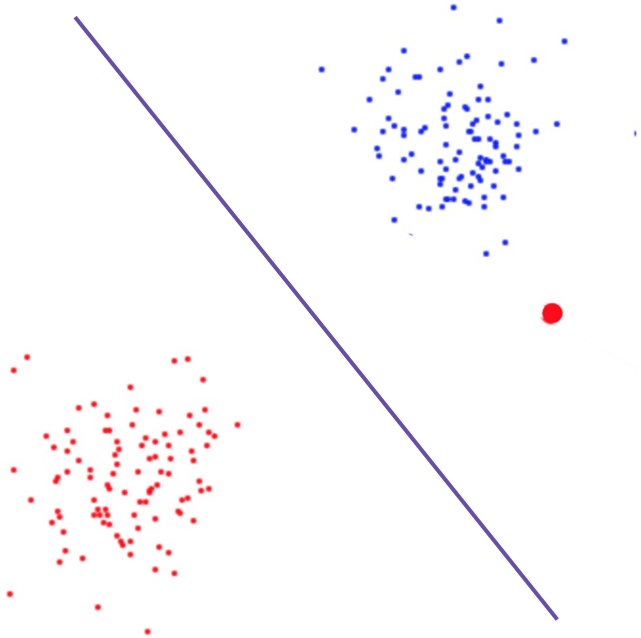


What if…

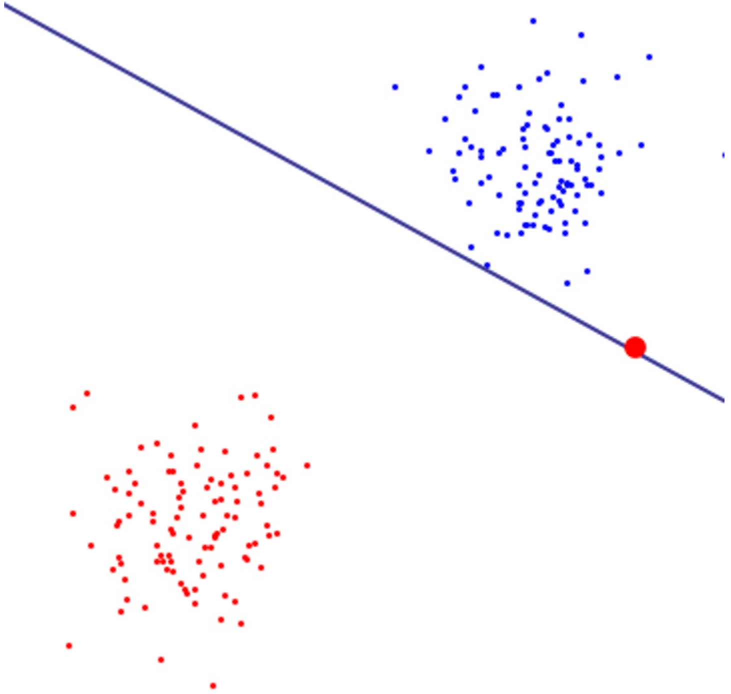


Which Decision Boundary is better?

Boundary 1



Boundary 2



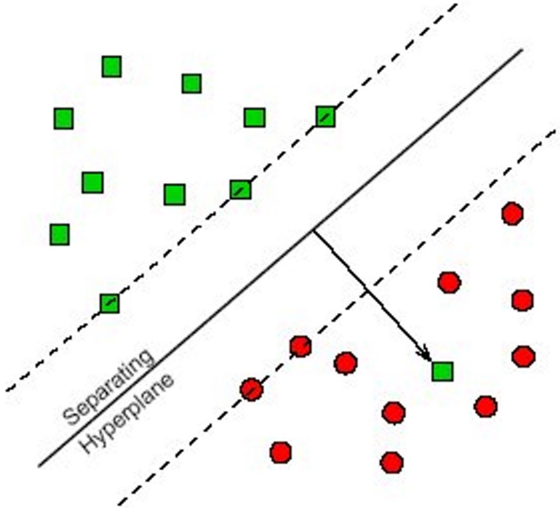
Margins

Use cost function to penalize misclassified points

Choice of cost function makes margin “hard” vs. “soft”

Non-separable training sets

Use linear separation, but admit training errors.

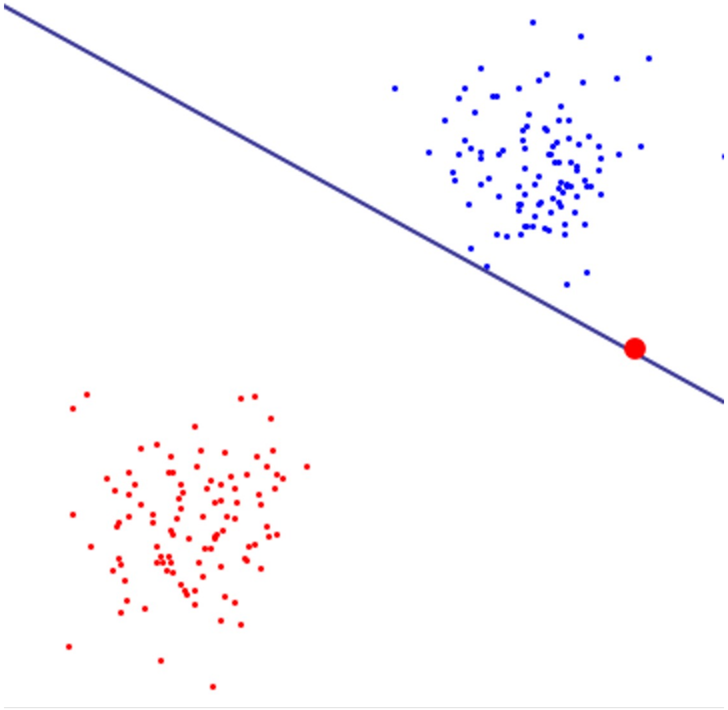


Penalty of error: distance to hyperplane multiplied by *error cost* C.

Hard Margins

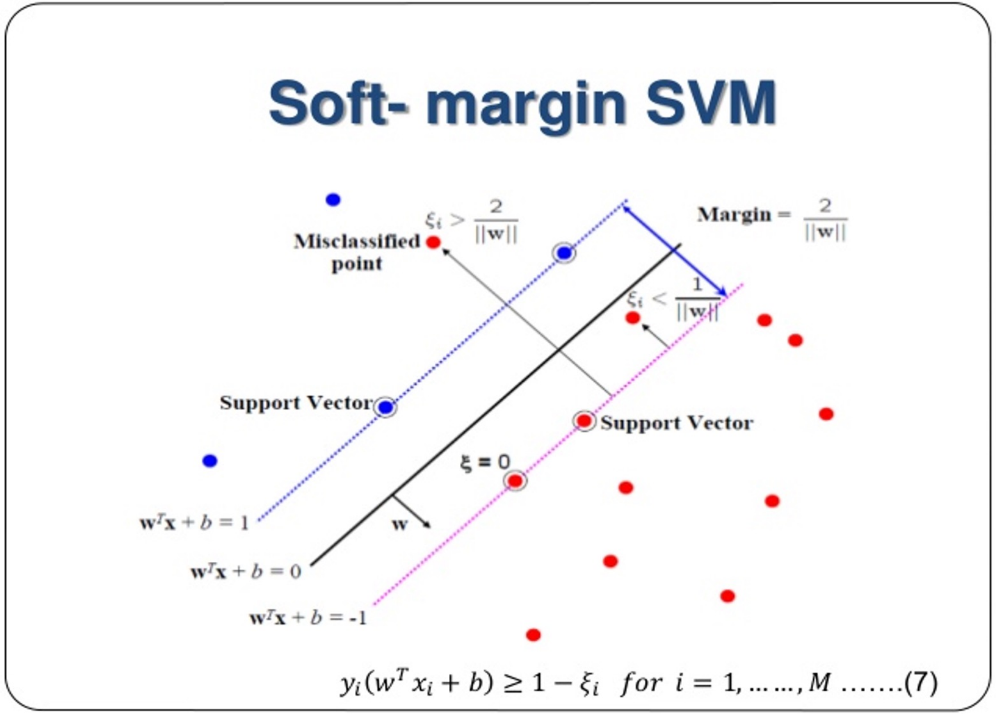
- High penalty value

- The hyperplane can be dictated by a single outlier

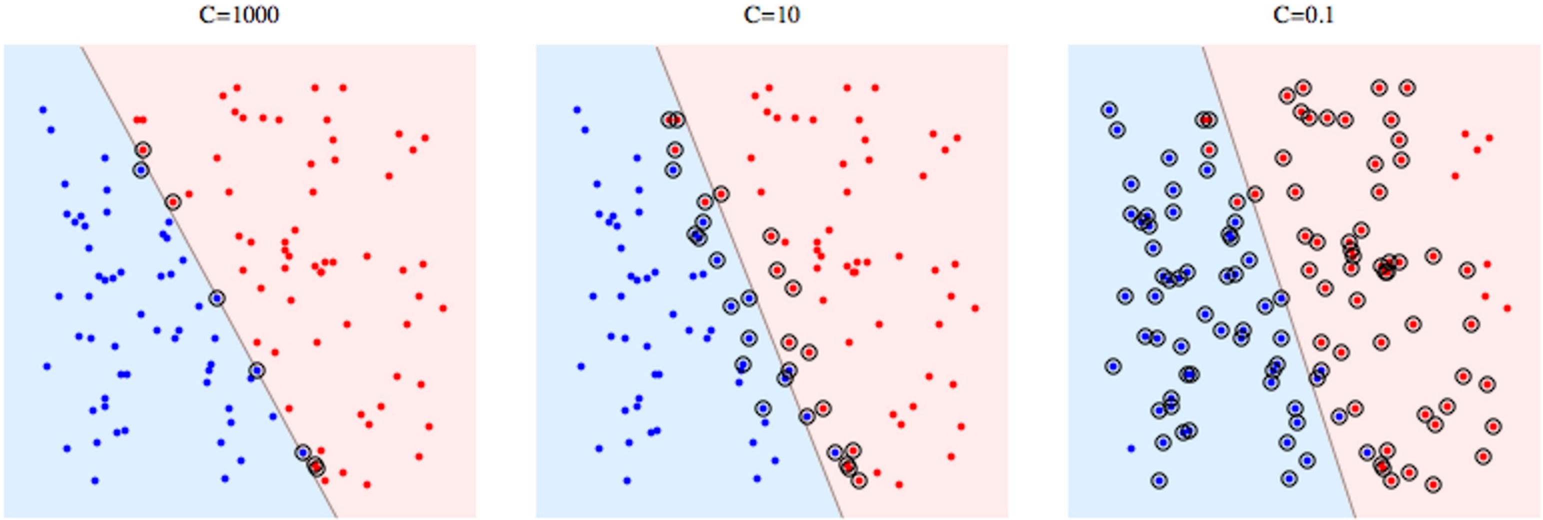


Soft Margins

- Used in non-linearly separable datasets
- Allow for misclassification
- Can account for "dirty" boundaries

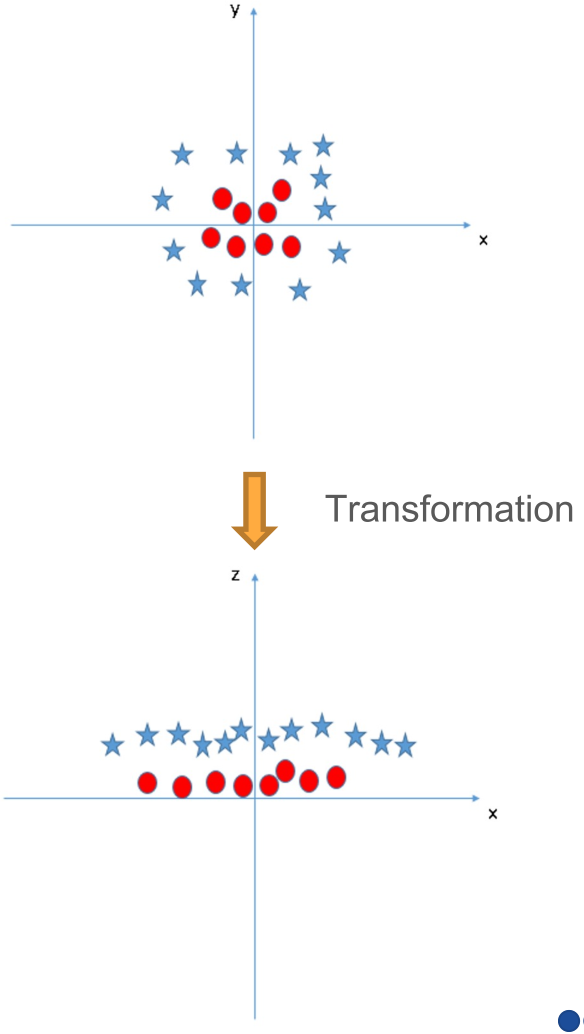


Misclassification Penalty C

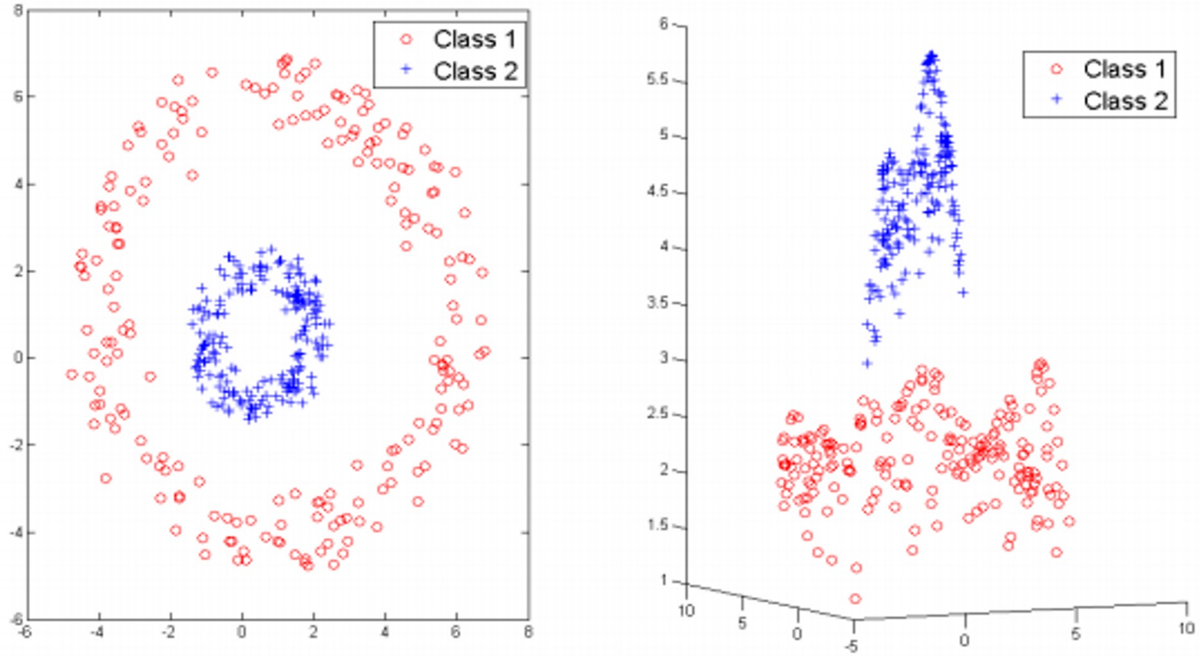


Kernels

- You cannot linearly divide the 2 classes on the *xy* plane at right
- Introduce new feature, $z = x^2 + y^2$ (**radial kernel**)
- Map 2 dimensional data onto 3 dimensional data. Now a hyperplane is easy to find. (Imagine slicing a cone!)



Kernels



SVM has MANY Hyperparameters

SVM

C

The "penalty cost" for misclassifications (soft margins)

Gamma

How far the influence of a single training example reaches

Kernels

Method of transforming our data set

Finding the Best Hyper Parameters

Grid Search

Random Search

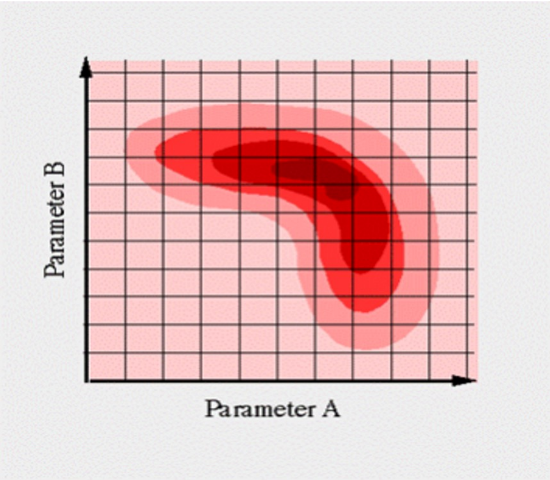
Optimize

Bayesian Optimization

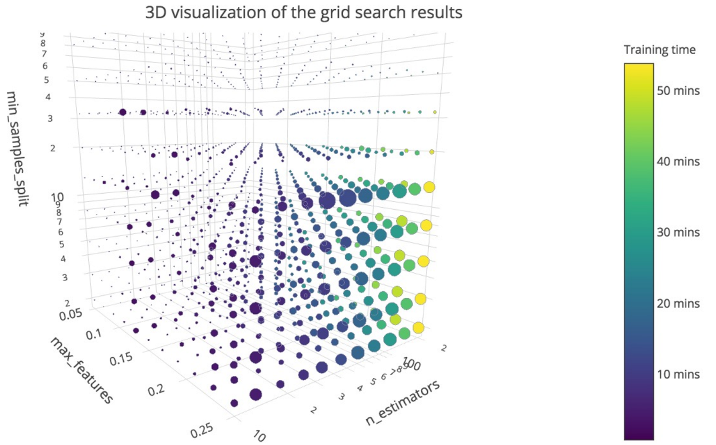
source

Curse of Dimensionality

Our search space for the optimal hyper-parameters increases **exponentially** as the number of hyper parameters we are considering increases



Add dimension



Overview

Perceptron	SVM
• A very simple model • Will perform poorly if data is not linearly separable	• More complex model because we have to choose the "penalty cost" associated with misclassifications • Can transform feature space by choosing a Kernel

Demo

More Validation Techniques

Leave-P-out

Let **D** be our whole dataset

Choose a **P**

For every combination of **P** points in **D**:

Use a train/test split with those **P** points as test, the rest as train

Leave-P-out: different from K-fold!

Let's say **D** has a size of 4. There are four data points: *a*, *b*, *c*, and *d*.

K-fold:

- K = 2.
- Each fold has a size of 2: {*a*,*b*} and {*c*,*d*}
- So, we only have 2 possible test sets:

  {*a*,*b*} and {*c*,*d*}

Leave-P-out:

- P = 2.
- We have 6 possible test sets:

  {*a*,*b*}, {*a*,*c*}, {*a*,*d*}, {*b*,*c*}, {*b*,*d*}, and {*c*,*d*}

Leave-P-out

Pros:

- Dependable (not random)
- Representative — checks all combinations

Cons:

- Slow!
  - Runtime increases with larger datasets
  - Runtime explodes with larger P

Monte Carlo CV

- Getting accuracy **1** time doesn’t tell us much
- Getting accuracy **2** times tells us a bit
- Getting accuracy **3** times tells us a bit more
- …
- Getting accuracy **N** times might be good enough!

Take the average of those **N** times

Monte Carlo CV

- Need to use **new**, **random** train/test split each time
  - If you use the same train/test split each time, you’re not getting any new information!
- Pros:
  - easy to implement
  - easy to make faster/slower by changing number of iterations
- Cons:
  - random -> train/test splits not guaranteed to be representative of dataset
  - harder to calculate how many iterations you need

The Bootstrap

What if we don't have enough data?

- Use **bootstrap datasets** to approximate the test error
- **Sample with replacement** from the original training dataset (with n samples) to generate **bootstrap datasets** of size n
  - Some data points may appear more than once in the generated data
  - Some data points may not appear
- Estimate of test error = average error among bootstrap datasets

Bootstrap vs. k-fold

In K-fold validation, each of the K folds is distinct from the other $(K - 1)$ folds used for training: there is **no overlap**.

This is crucial for its success in estimating prediction error.

Why we still use bootstrap

- Bootstrap allows us to use a computer to mimic the process of obtaining new data sets.
- Can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- Provides an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.
    - i.e., the variability of the model!

Bagging (Bootstrap Aggregating)

What if we don't have enough data?

- Bagging is a common technique that builds on Bootstrapping
- Main Idea: Do Bootstrapping a bunch and make a classifier for each bootstrap, then majority prediction wins.
- Many weak learners aggregated typically outperform a single learner over the entire set, and overfits less.
  - Principle behind Random Forests

Demo

Coming Up

- **Assignment 8**: Due midnight of next class
- **Next Lecture**: Applications of *Un*supervised Learning

CDS Education

We explore, learn, and educate big minds.