Lecture 7: Supervised Learning Pt. 1

Linear Classifiers and Cross Validation
INFO 1998: Introduction to Machine Learning

CDS Education

Agenda

1. Linear Classifiers
    - Linear Perceptron
    - Support Vector Machines (SVMs)
2. Cross Validation (K-Fold)

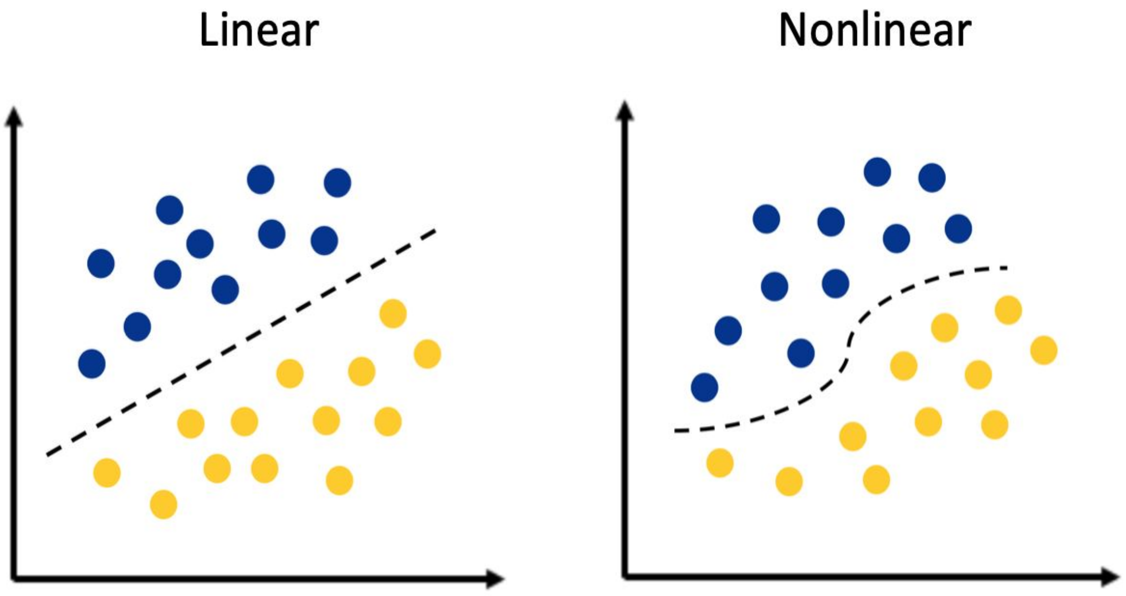
Linear Classifiers

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 best hyper plane for our data.



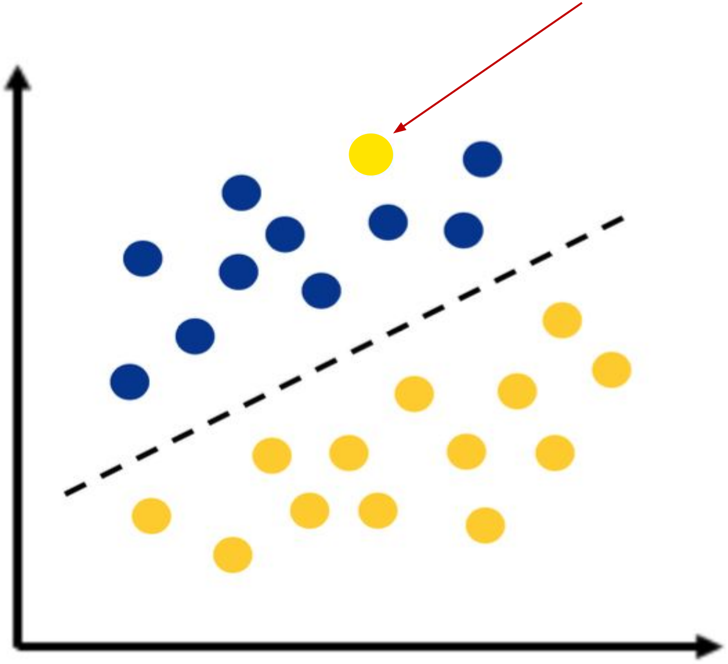
Source

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:
        Adjust classifier slightly so that it classifies x correctly
                (or, is a little closer to classifying x correctly)
End While
```

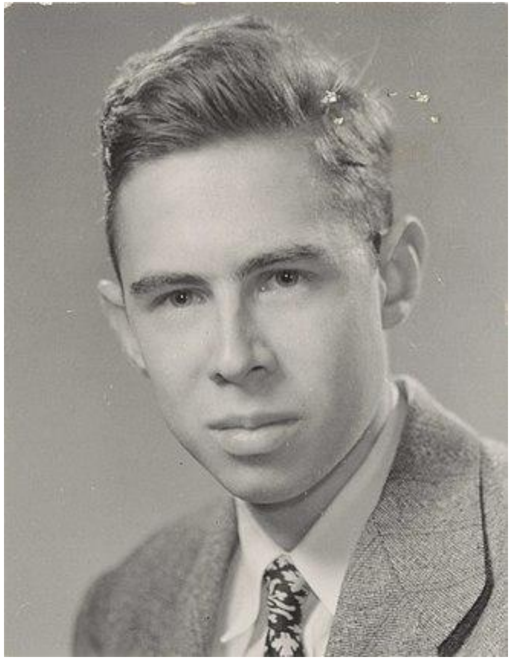
"Use your mistakes as your stepping stones"

History of the Linear Perceptron

Frank Rosenblatt was first to implement perceptron!
→ Cornell professor and alum PHD ‘56 🐻❗

Gave him the title of ‘Father of Deep Learning’

Deep Learning
→ Neural Networks a.k.a. Multilayer Perceptrons



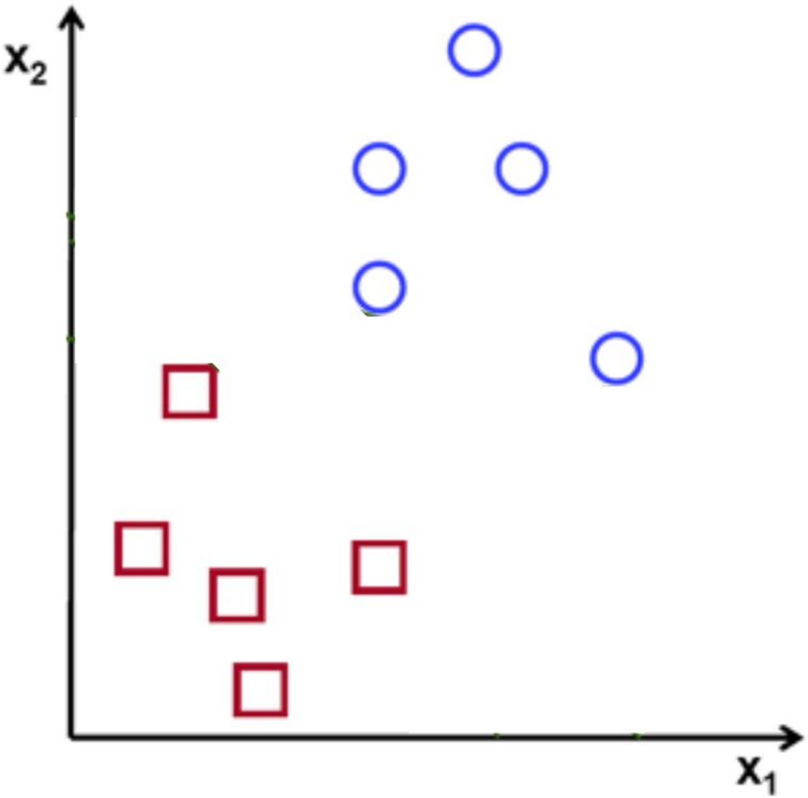
Limitations of Perceptron

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

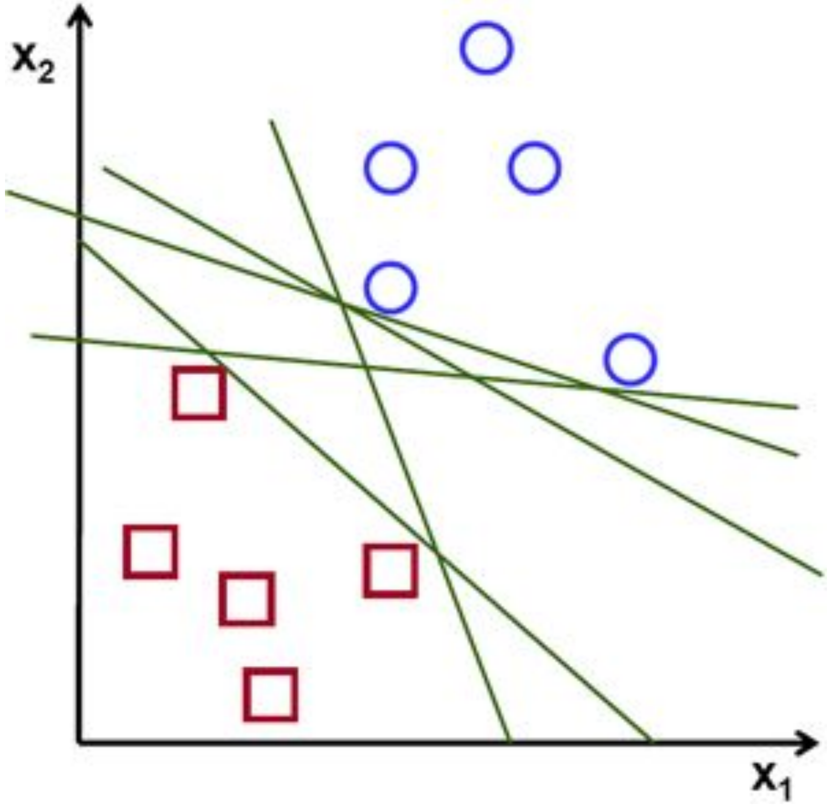
Is a great model to understand the intuition behind the training of a linear classifier: iteratively improve classifier by using misclassified points 😄

Support Vector Machines

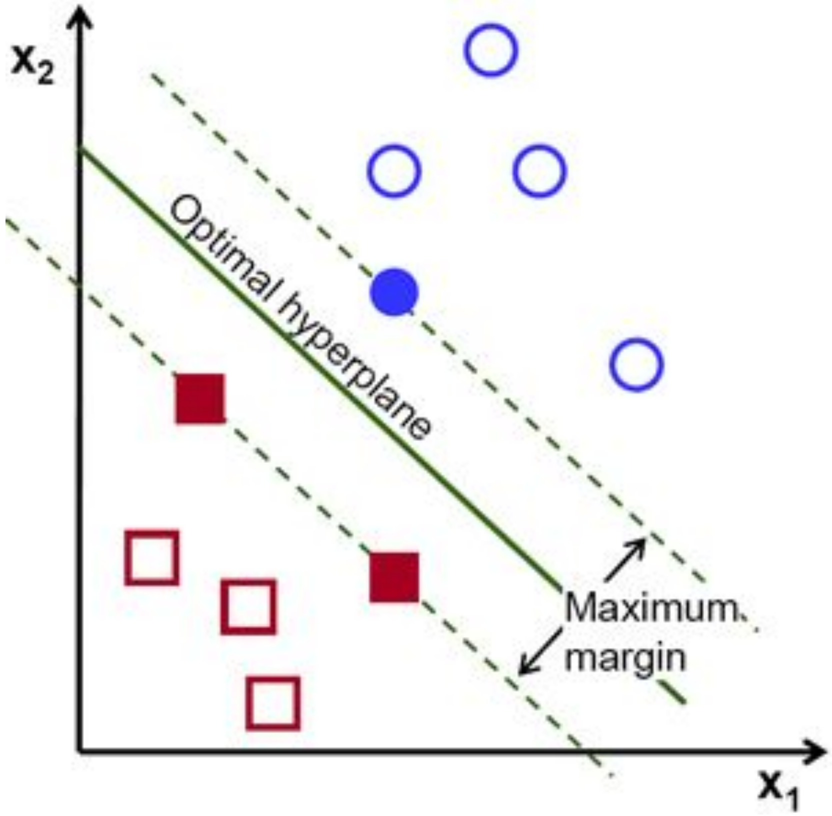
Classify (+) and (-)



Which Hyperplane?



Optimal Hyperplane



Support Vector Machine

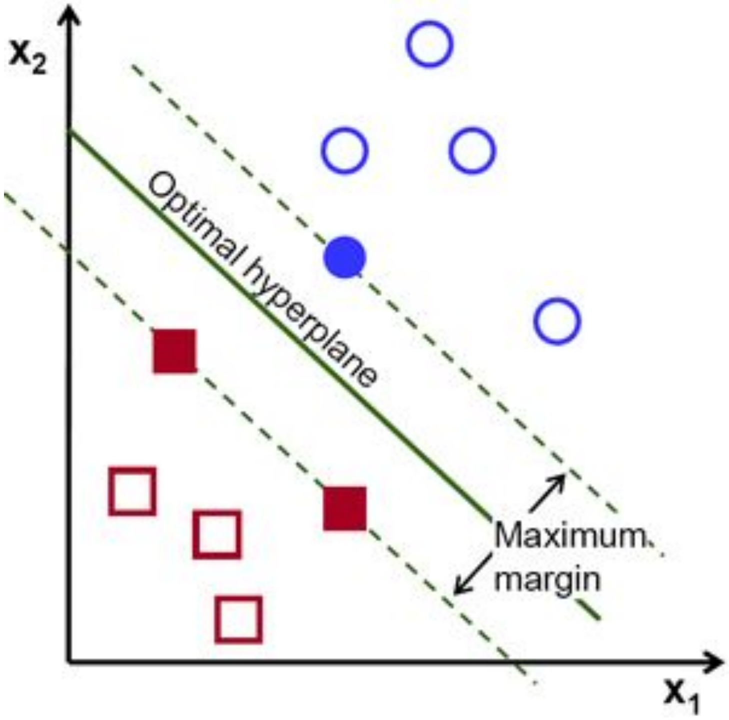
Memory efficient

Effective in a higher dimensions

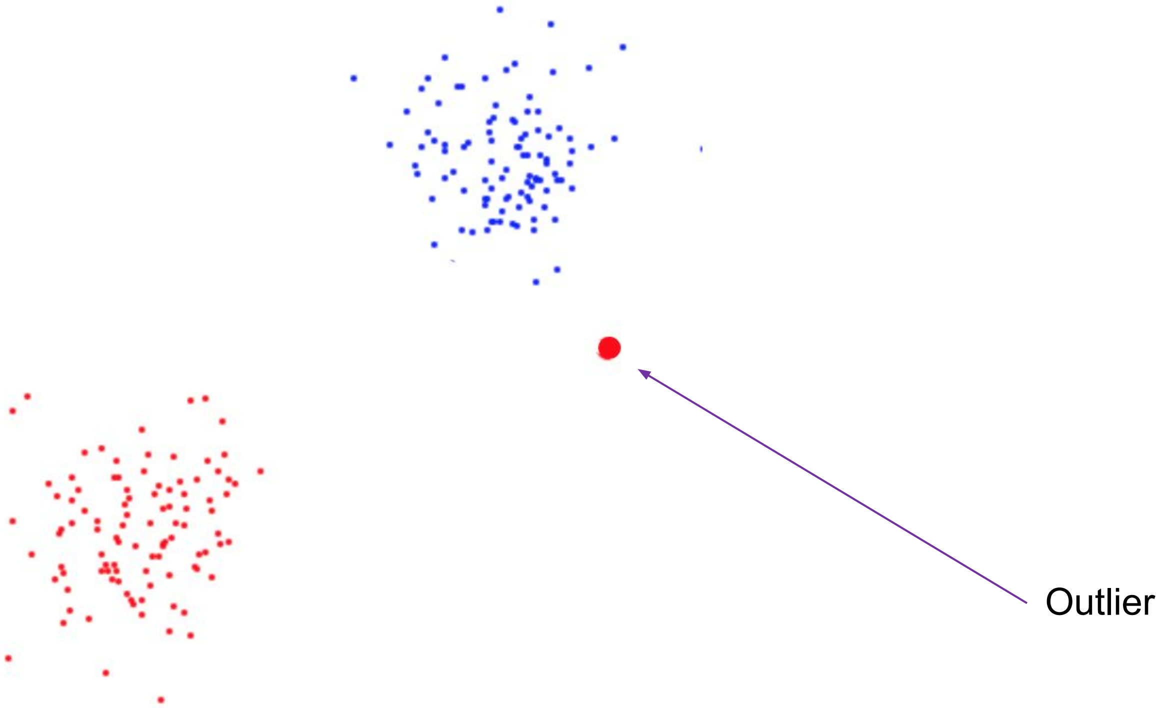
Slow calculation time

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

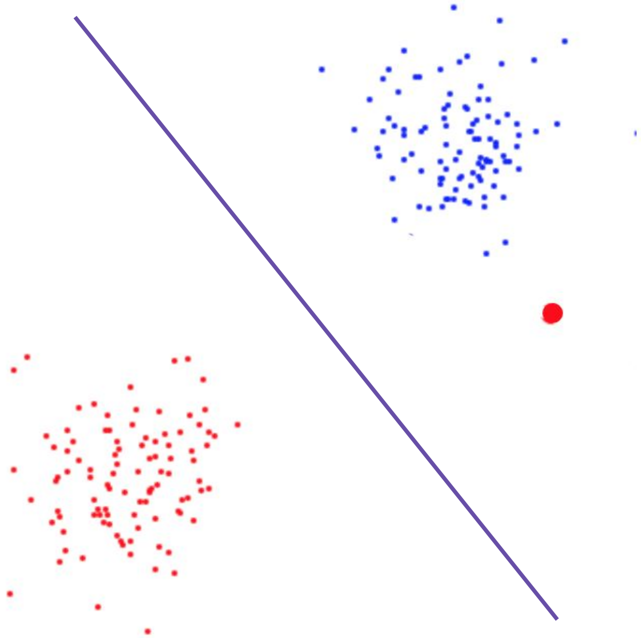


What if…

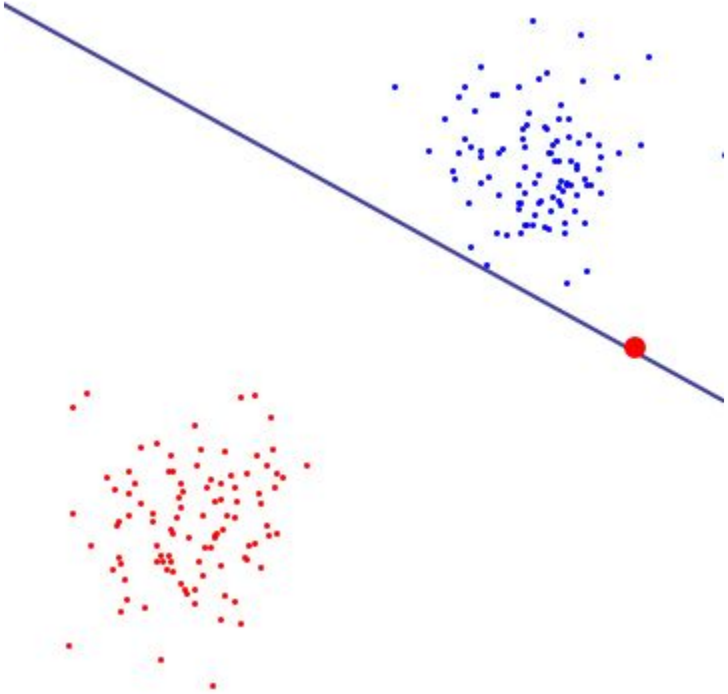


Which Decision Boundary is better?

Boundary 1



Boundary 2



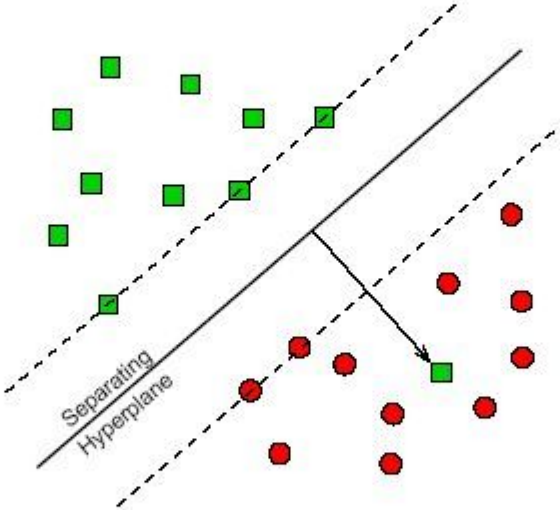
Margins

Use cost function to penalize misclassified points

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

Non-separable training sets

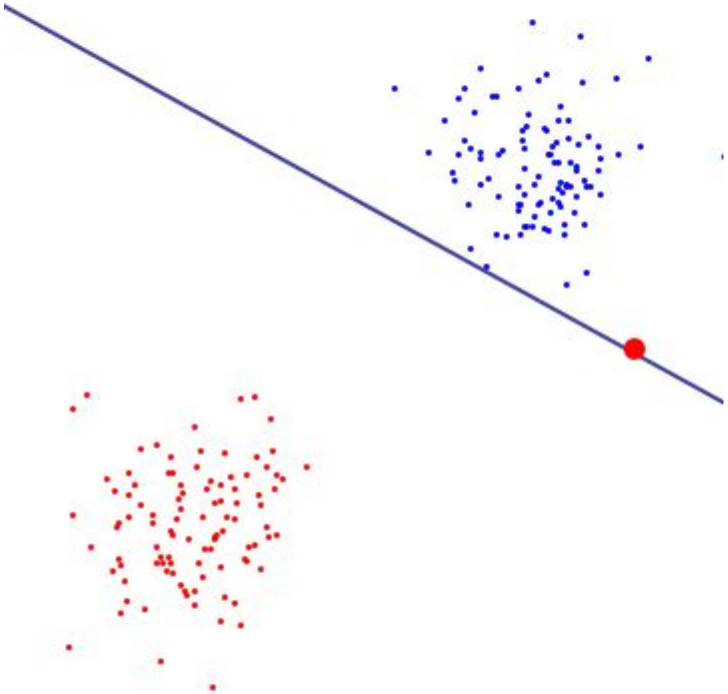
Use linear separation, but admit training errors.



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

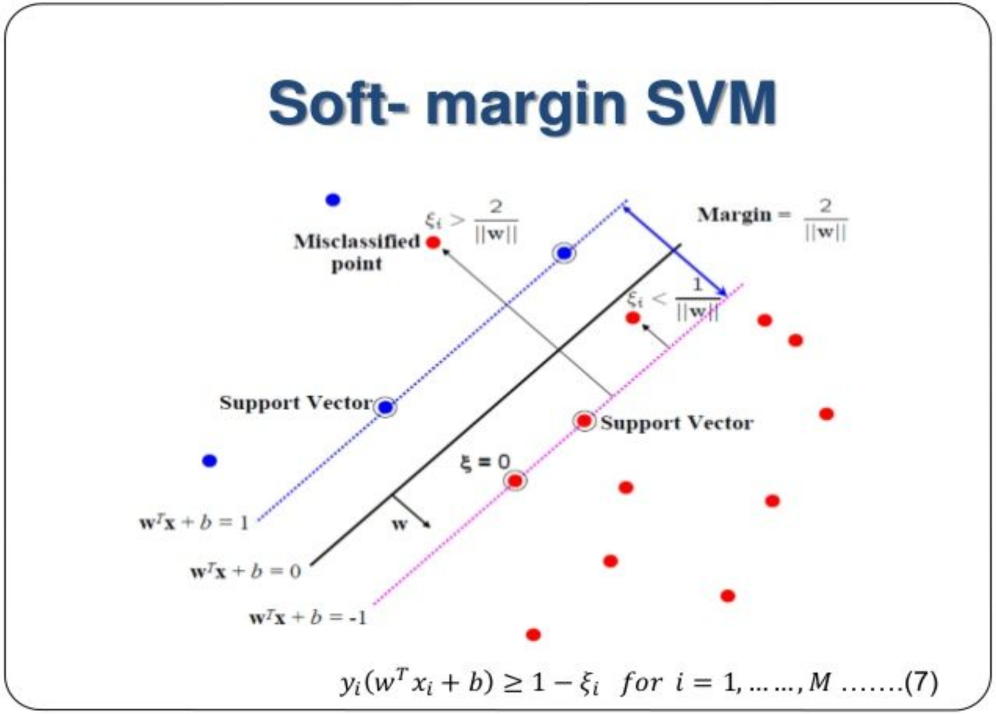
Hard Margins

- High penalty value
- The hyperplane can be dictated by a single outlier

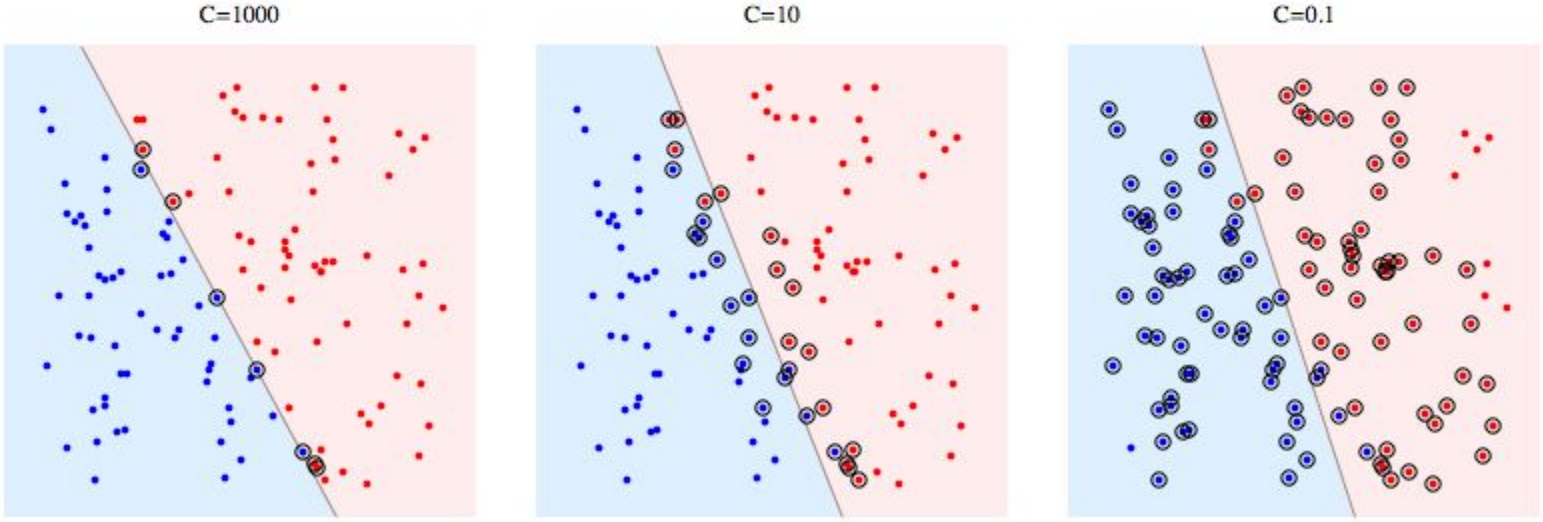


Soft Margins

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

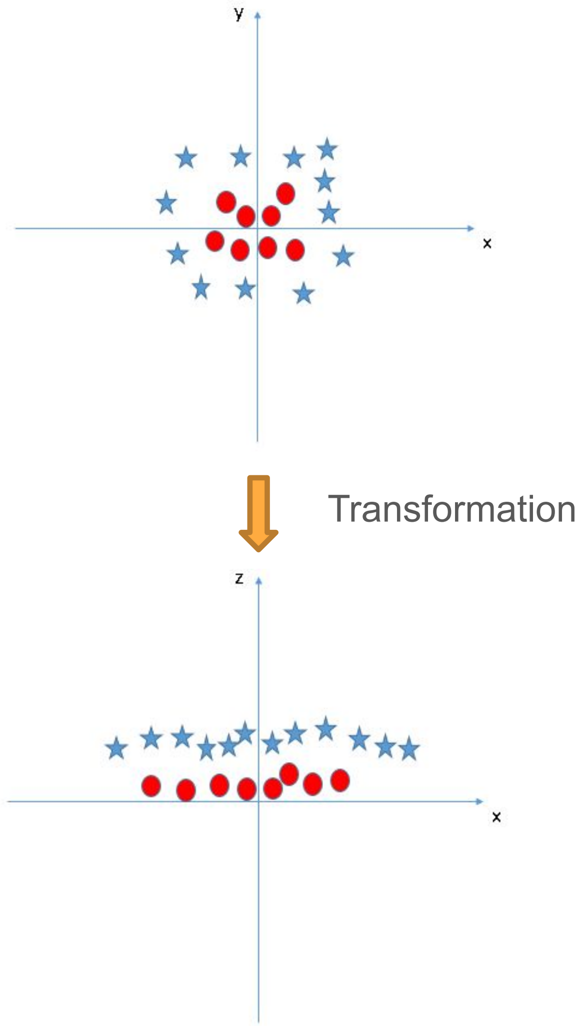


Misclassification Penalty C

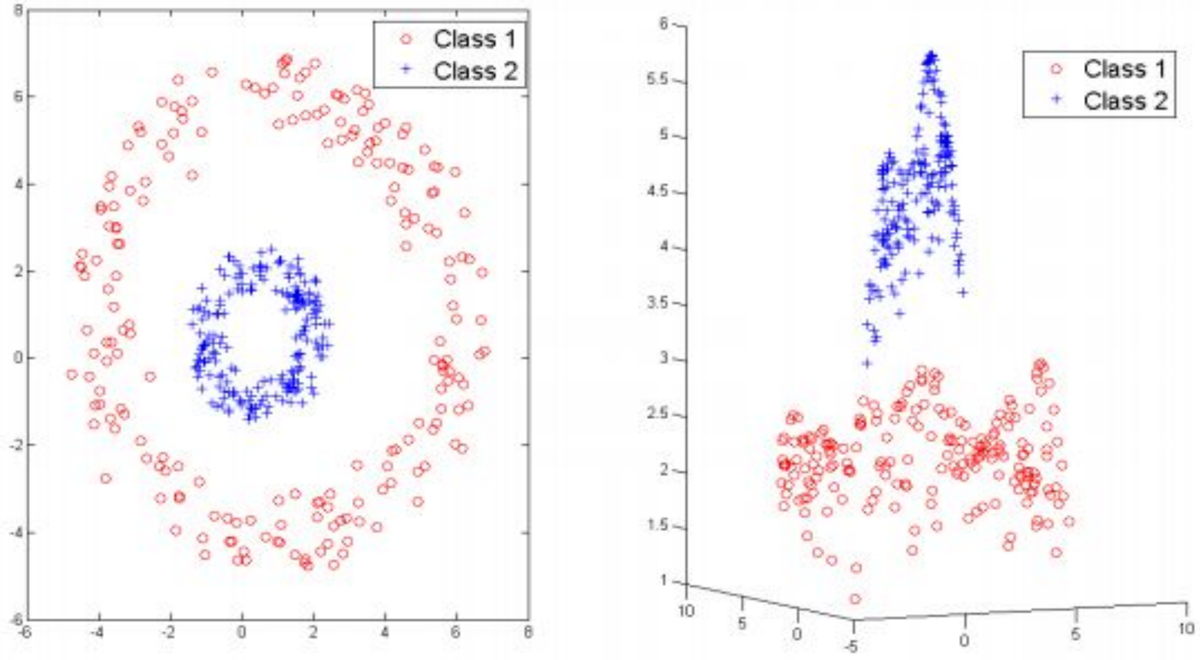


Kernels

- 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.



Kernels

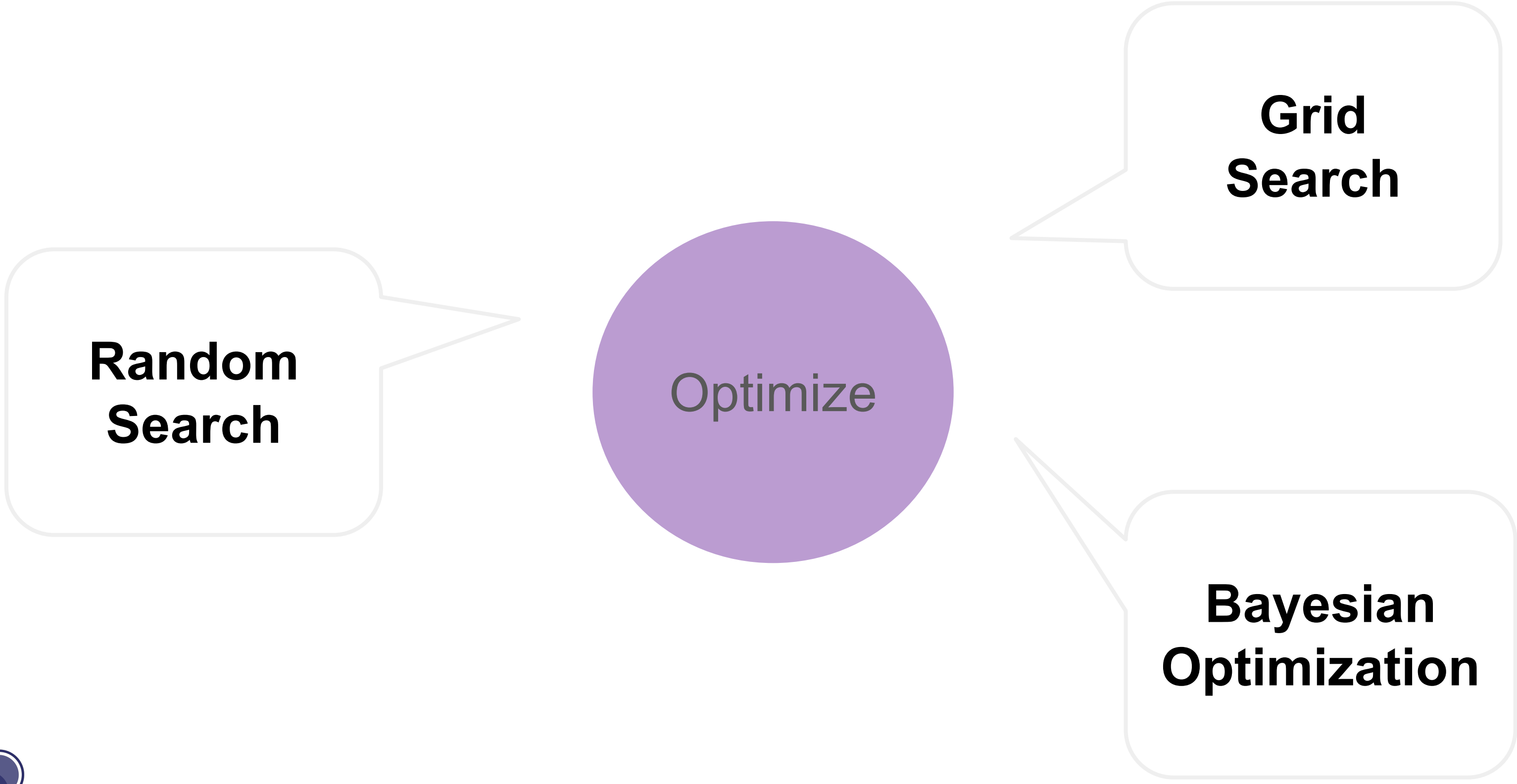

Class 1
Class 2
8
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6
8
Class 1
Class 2
6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
10
5
0
-5
0
5
10

SVM has MANY Hyperparameters

SVM

C	Gamma	Kernels
The “penalty cost” for misclassifications (soft margins)	How far the influence of a single training example reaches	Method of transforming our data set

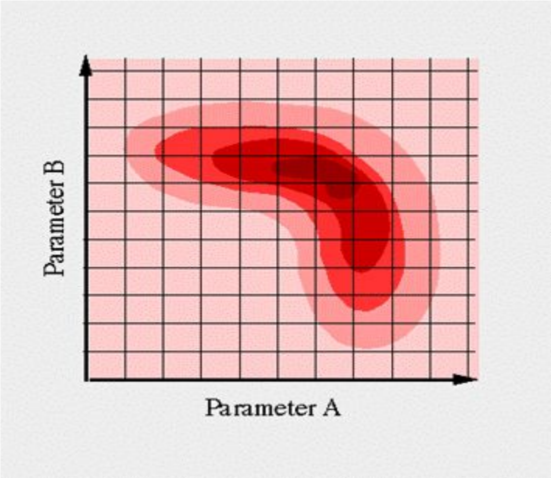
Finding the Best Hyper Parameters



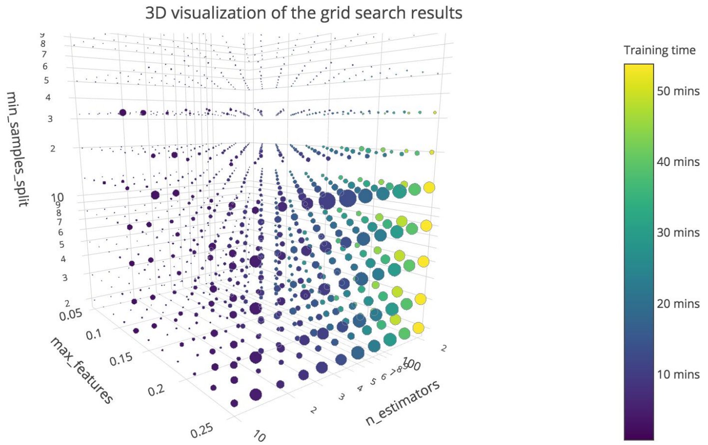
source

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

Cross Validation

K-fold Cross Validation

Often used in practice with k=5 or k=10.

Create equally sized k partitions, or **folds**, of training data

For each fold:

- Treat the k-1 other folds as training data.
- Test on the chosen fold.

The average of these errors is the validation error

K-fold Cross Validation

Dataset

Suppose K = 5,
5-Fold CV

K-fold Cross Validation

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

K-fold Cross Validation

Test Sample	Training Sample	Training Sample	Training Sample	Training Sample

Calculate MSE = mse1

K-fold Cross Validation

Training Sample	Test Sample	Training Sample	Training Sample	Training Sample

Calculate MSE = mse2

K-fold Cross Validation

Training Sample | Training Sample | Test Sample | Training Sample | Training Sample

Calculate MSE = mse3

K-fold Cross Validation

And so on

K-fold Cross Validation

Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

MSE = Avg(mse1...5)

K-fold Cross Validation

Matters less how we divide up

Selection bias not present

Leave-1-Out Cross Validation

For each sample:

- Treat all other data as training data.
- Test on that one sample

The average of these errors is the validation error

Pro: Better on small datasets

Pro: More realistic (trained on most of the data)

Con: Takes longer to run

Coming Up

- **Assignment 6:** Due tonight at 11:59pm
- **Assignment 7:** Due next Wednesday at 11:59pm
- **Mid-Semester Check-Ins:** Due today! Please get them done ASAP :)
- **Next Lecture:** More Supervised Learning! (Decision Trees & Logistic Regression)