CSCAR CONSULTING FOR STATISTICS, COMPUTING & ANALYTICS RESEARCH
UNIVERSITY OF MICHIGAN

Supervised Machine Learning

Random Forests
Support Vector Machines (SVMs)

Data Science Skills Series

Márcio Mourão

References

http://scikit-learn.org/

http://scikit-learn.org/stable/modules/svm.html/

www.cs.kent.edu/~jin/DM07/ClassificationDecisionTree.ppt

http://www.svm-tutorial.com/2014/11/svm-understanding-math-part-1/

http://cs.haifa.ac.il/hagit/courses/seminars/visionTopics/Presentations/SVM_Lecture.ppt/

Summary

- Introduction to Random Forests

- Use Random Forests to model available census data and predict whether someone makes more than 50K a year

- Introduction to Support Vector Machines (SVMs)

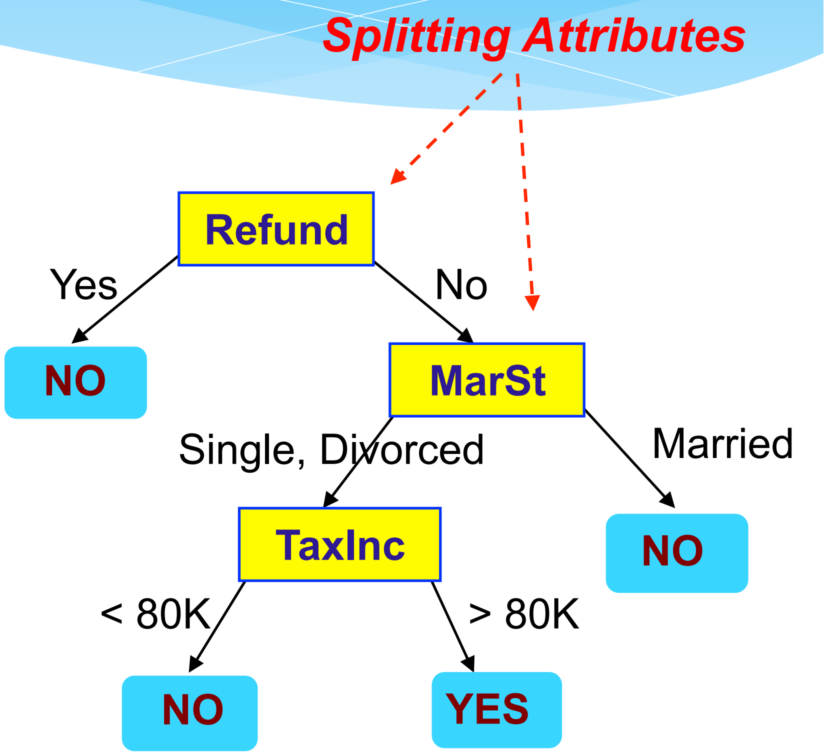
- Use SVMs to predict handwritten digits

Random Forests

Create a decision tree from training data

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

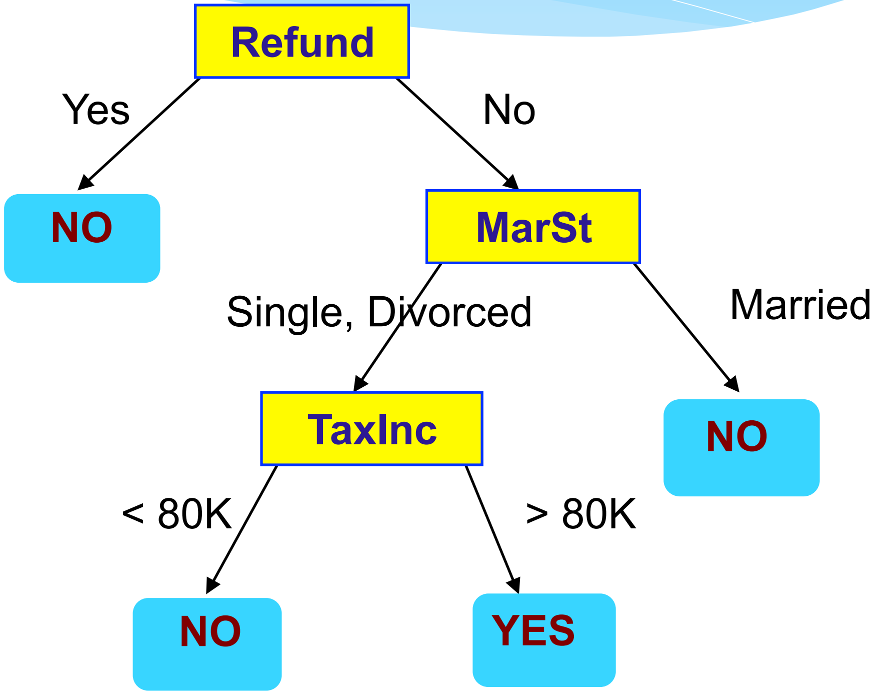


Model: Decision Tree

Apply model to test data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**



Decisions and algorithms

- Determine how to split
  - Specify attribute test condition
    - Attribute type: nominal, ordinal, continuous
    - Splitting: 2-way split, multi-way split

- Determine best split
  - Need a measurement of impurity: Gini Index, Entropy, Misclassification Error

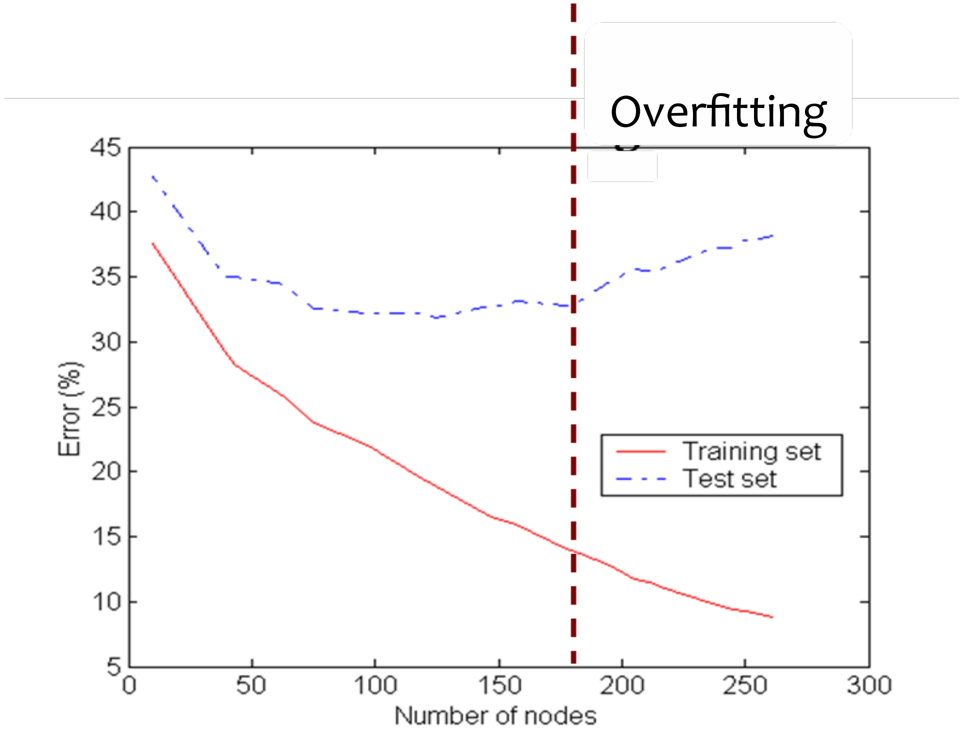
- There are many algorithms available
  - Hunt's, CART, ID3, C4.5, SLIQ, SPRINT

Advantages

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Easily handles mixed data
- Scalable
- Accuracy is comparable to other classification techniques for many simple data sets

Disadvantages

- Prone to produce decision trees that are more complex than necessary (overfitting)
- In overfitting, trees are highly sensitive to noise in the training set
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records


Overfitting

Random Forests

- Random Forests correct decision's tree habit of overfitting

- Random Forests fall under a class of learners called "Ensemble Methods"

- For what problems can you apply this method?
  - Regression/Prediction for Continuous Outcome Variables
  - Classification for Categorical Outcome Variables (2 or more levels)

Methodology

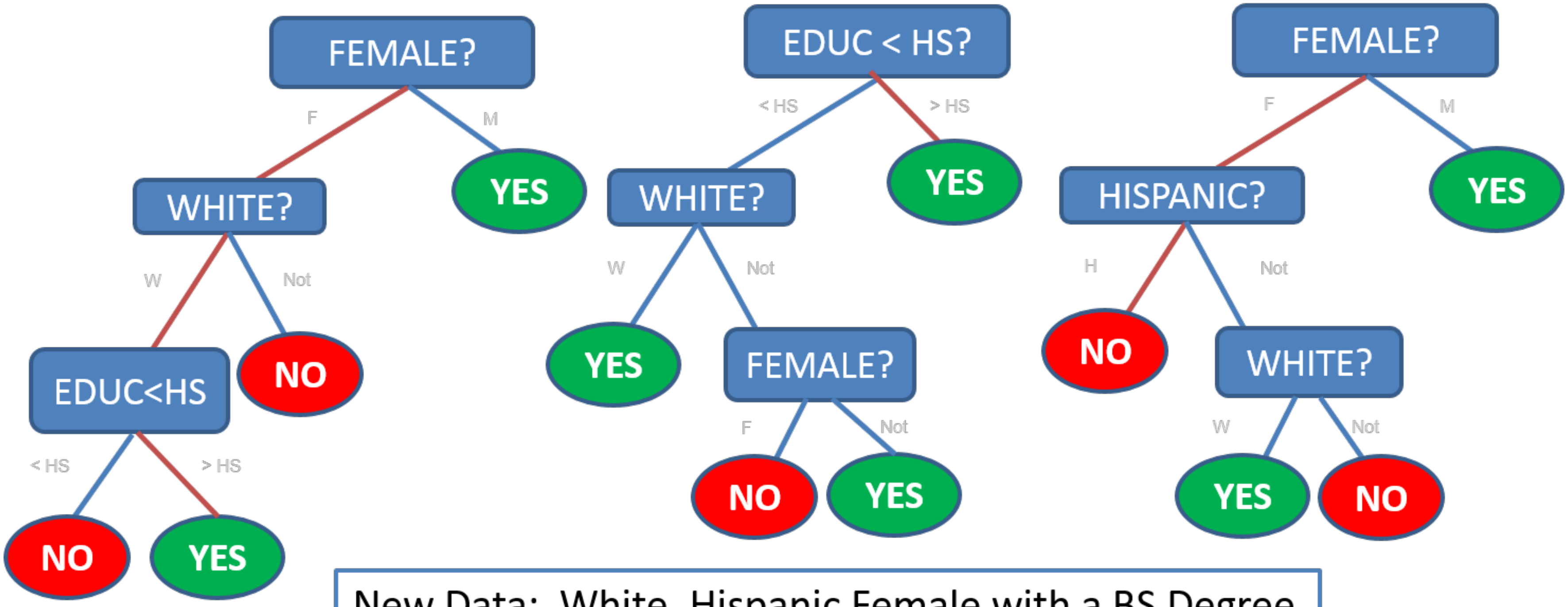
- **Step 1:** Establish the Forest (bagging)
  - Generate ntree bootstrap samples from the original data set, with replacement, and having the same size as the input data set.
    - Reduces correlation between trees by using different data sets
    - Decreases the variance of the model

- **Step 2:** Grow Trees
  - For each bootstrap sample from Step 1, grow a classification or regression tree to full size (with no pruning)
  - At each node, randomly select predictor variables to use for splitting
    - Splits will be based on the BEST variable from those randomly selected for a given node

Methodology

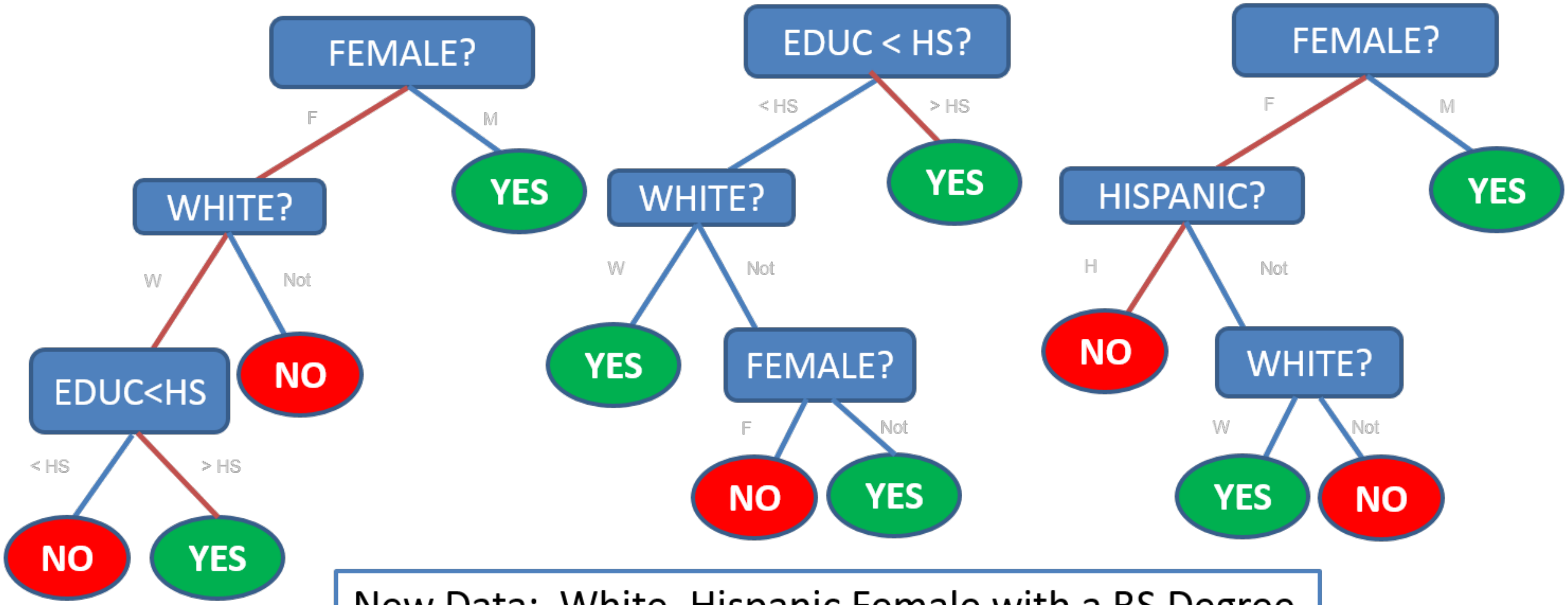
- **Step 3:** Prediction
  - After all trees in the forest have been grown, form a prediction for new data from each tree in the forest
  - Final predictions in classification problems are based on the majority
  - Final predictions in regression problems are means

Illustration



New Data: White, Hispanic Female with a BS Degree

Illustration



New Data: White, Hispanic Female with a BS Degree

Tree 1: YES

Tree 2: YES

Tree 3: NO

Pros and Cons of Random Forests

Pros

- Can handle predictors that are continuous, categorical, skewed and sparse data as well as missing data
- Aptly suited for the "large p, small n" scenario
- Very effective for estimating outcomes that are a complex functions of predictors with many interactions or possibly non-linear functions of the parameters

Cons

- Random Forests can be **extremely computationally intensive**
- Unlike Trees, Random Forests are **not easily visualized**

Random Forests using Scikit-Learn in Python

```
#Import Library
from sklearn.ensemble import RandomForestClassifier

# Create SVM classification object
model = RandomForestClassifier(n_estimators=10, criterion='gini’)

# Train the model using the training sets
model.fit(X, y)

#Predict Output
predicted= model.predict(x_test)
```

Case study: 1994 census dataset

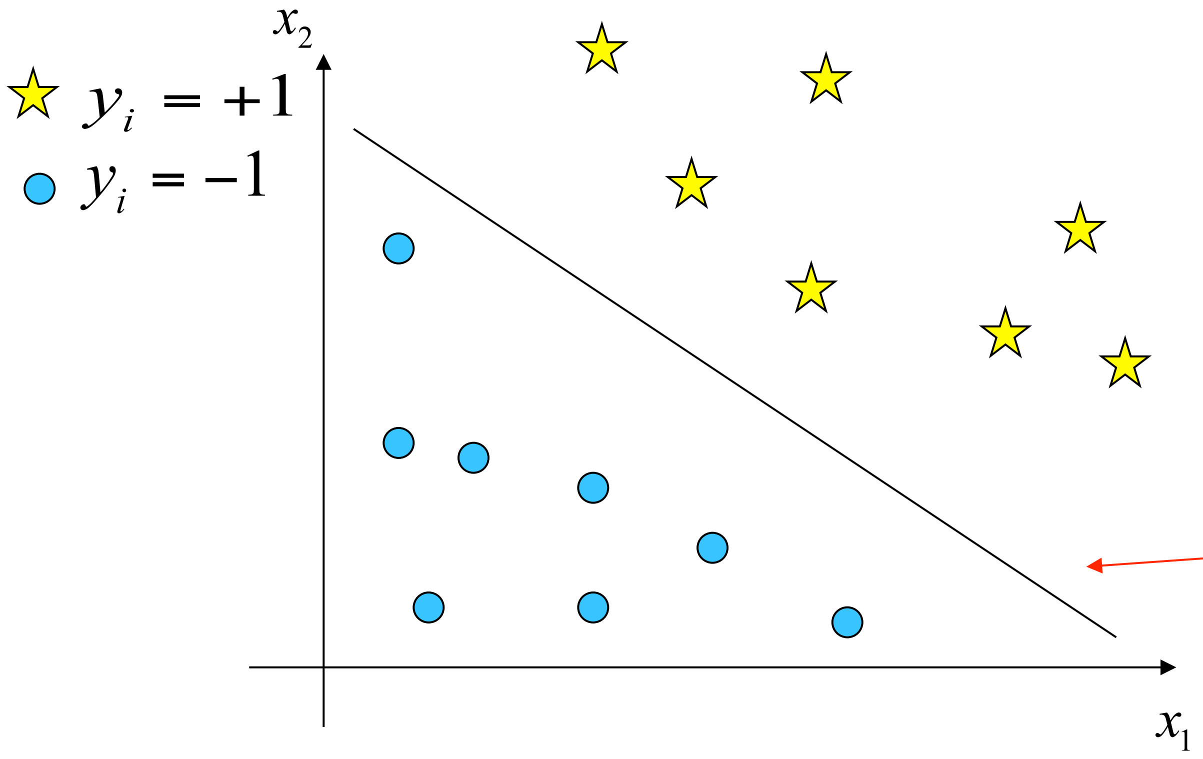
- This data was extracted from the 1994 Census bureau database by Ronny Kohavi and Barry Becker (Data Mining and Visualization, Silicon Graphics)

- A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0))

- **The prediction task is to determine whether a person makes over $50K a year**

Support Vector Machines (SVMs)

The goal of SVM is to find an hyperplane that separates two classes

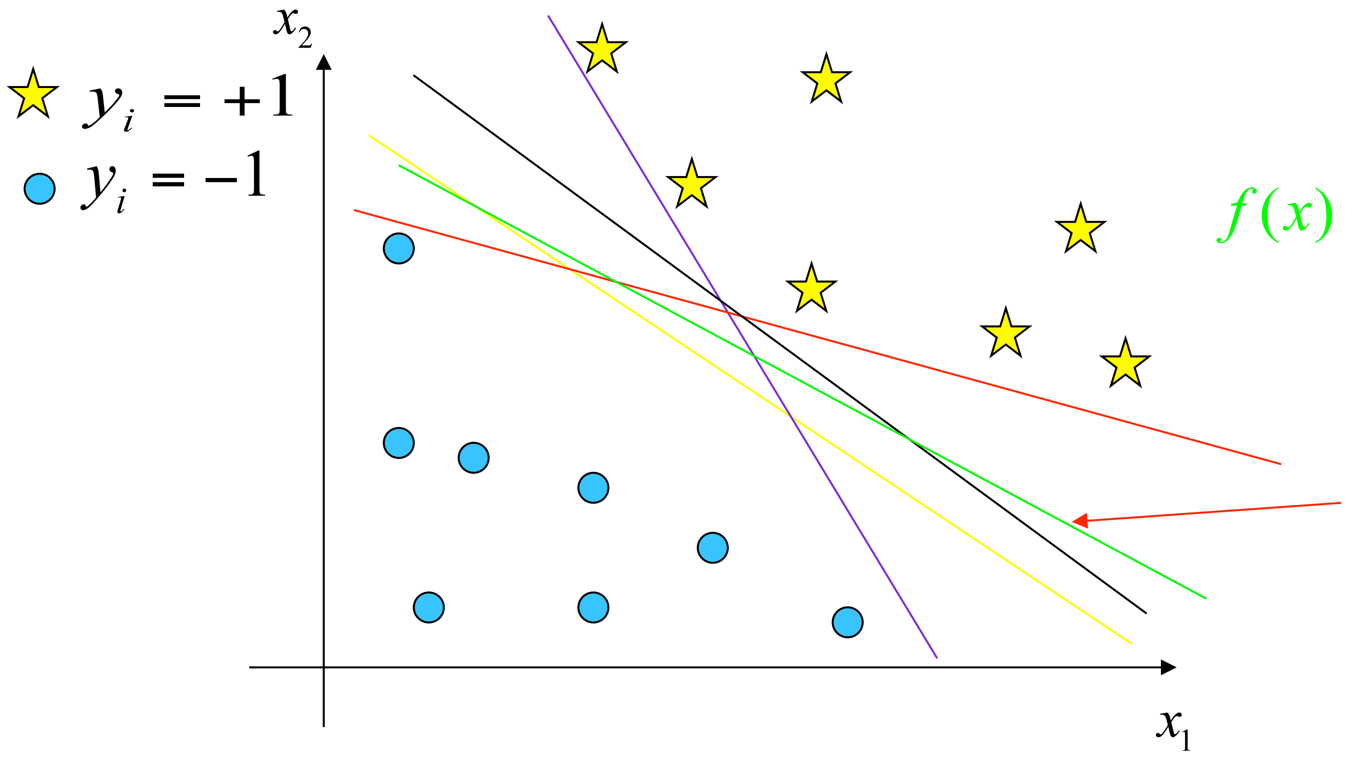


$f(x) = sign(w \cdot x + b)$

A separating hyperplane

$w \cdot x + b = 0$

There are many hyperplanes separating the two classes



$$f(x) = sign(w \cdot x + b)$$

A separating hyperplane

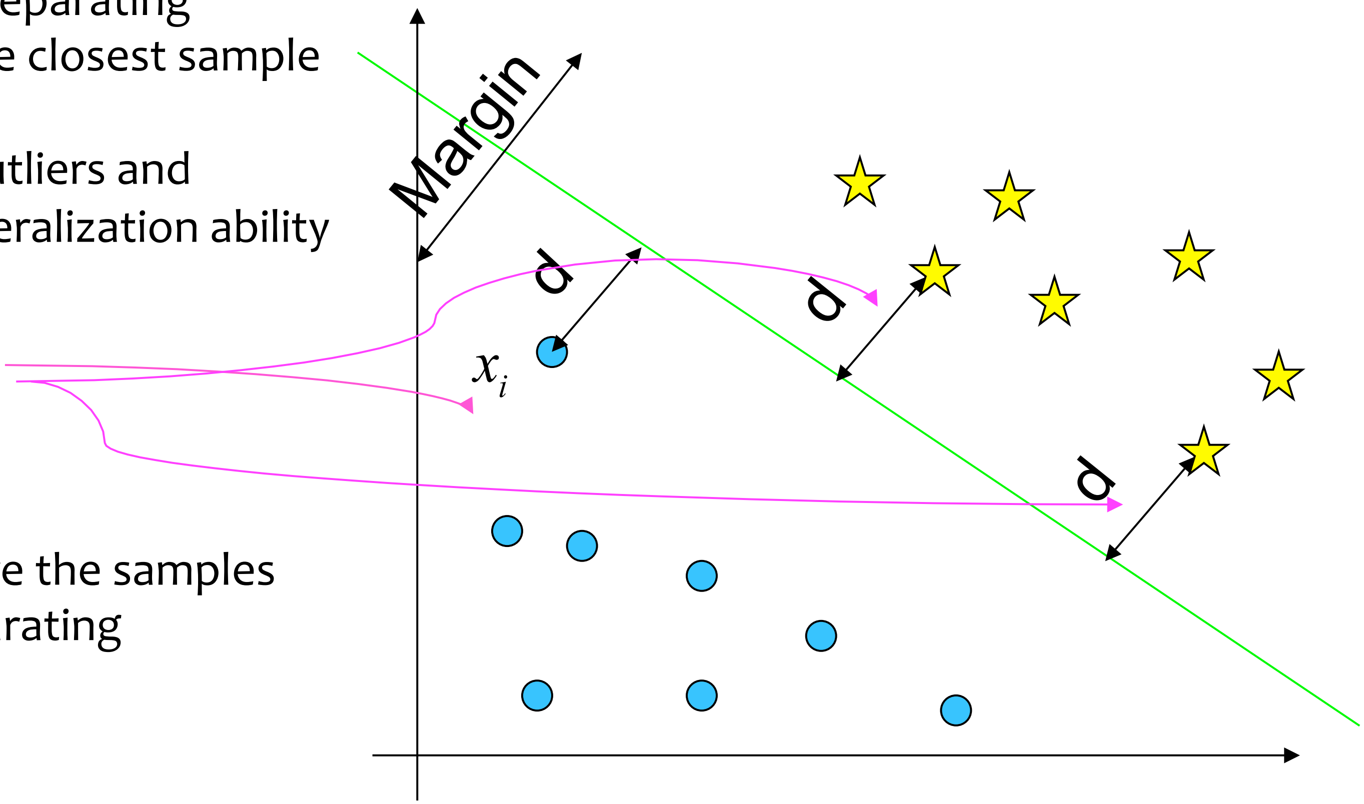
$$w \cdot x + b = 0$$

SVM’s goal is to maximize the Margin

The Margin is twice the distance “d” between the separating hyperplane and the closest sample

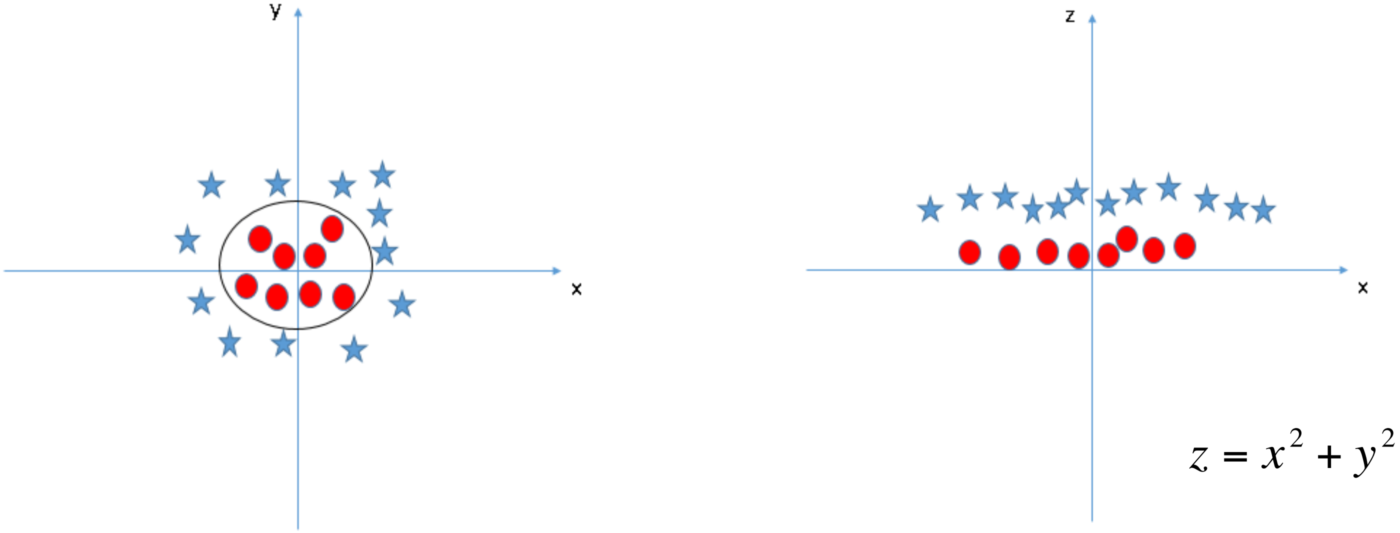
This is robust to outliers and hence, strong generalization ability

Support vectors



Support vectors are the samples closest to the separating hyperplane

What if the points are not linearly separable?



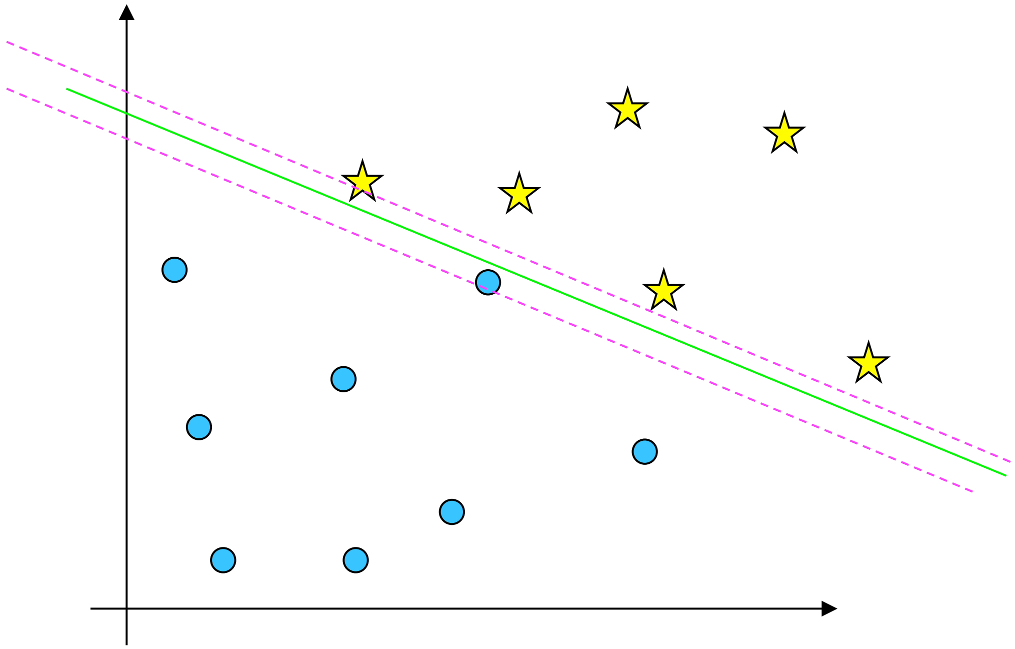
$$z = x^2 + y^2$$

Key idea: map the points to a space of sufficiently high dimension so that they will be separable by a hyperplane: SVM has Kernel functions which take low dimensional input space and transforms it to a higher dimensional space

Higher dimensional spaces introduce additional problems

A strong kernel ,which lifts the data to a great number of dimensions, sometimes may lead us to the severe problem of overfitting:

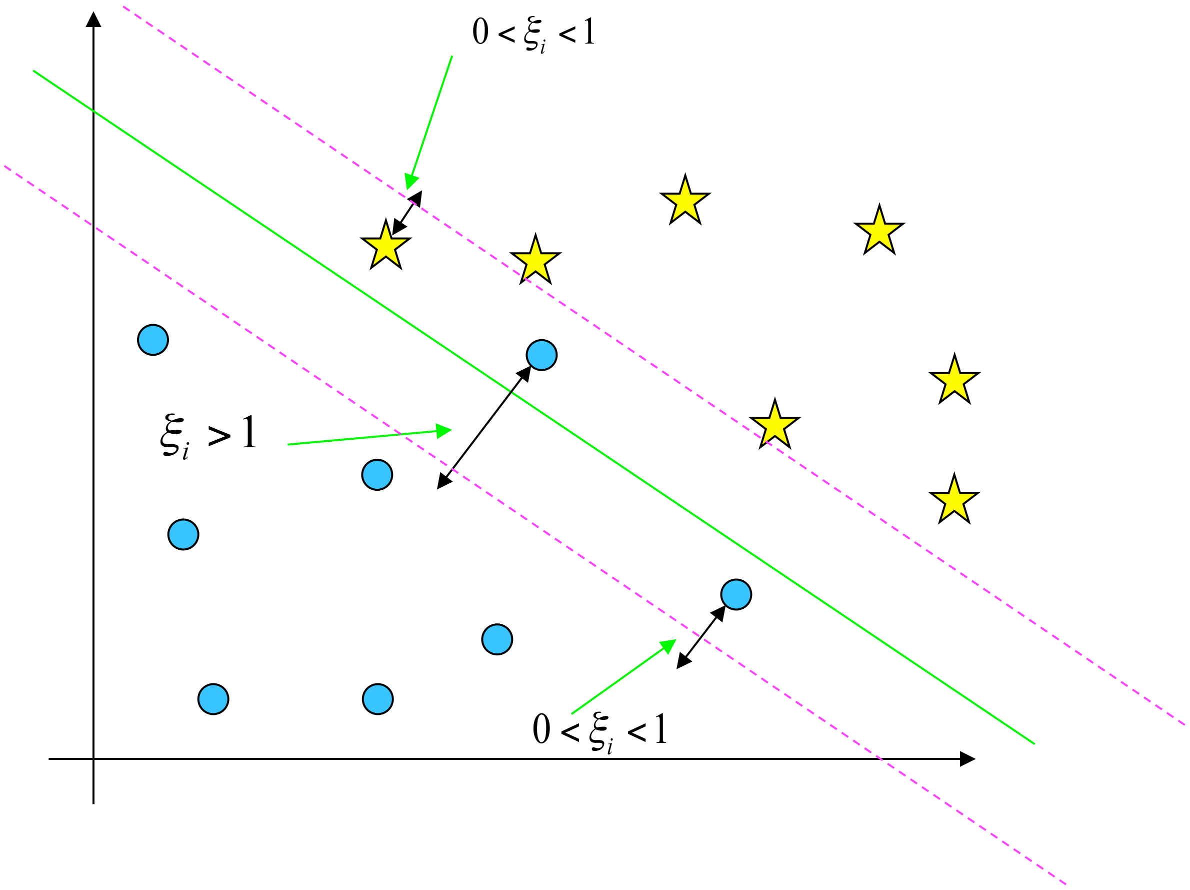
- Low margin → poor classification performance
- Large number of support vectors→ Slows down the computation



Overfitting can be partially solved by introducing soft margins

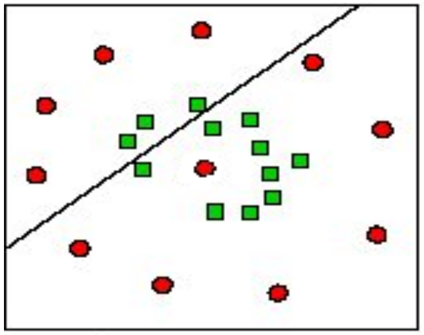
Soft margins allows for some error in classification

Introduction of parameter C for controlling the error term: this controls the trade off between a soft boundary and classifying the training points correctly



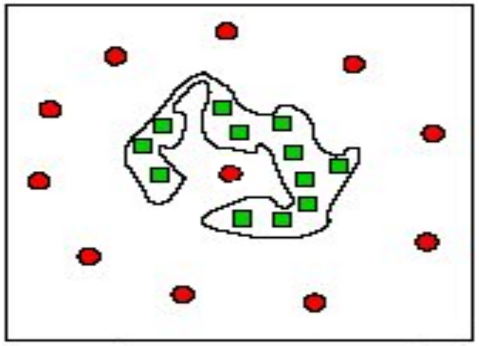
You need to consider the tradeoff between underfitting and overfitting

Under-Fitting

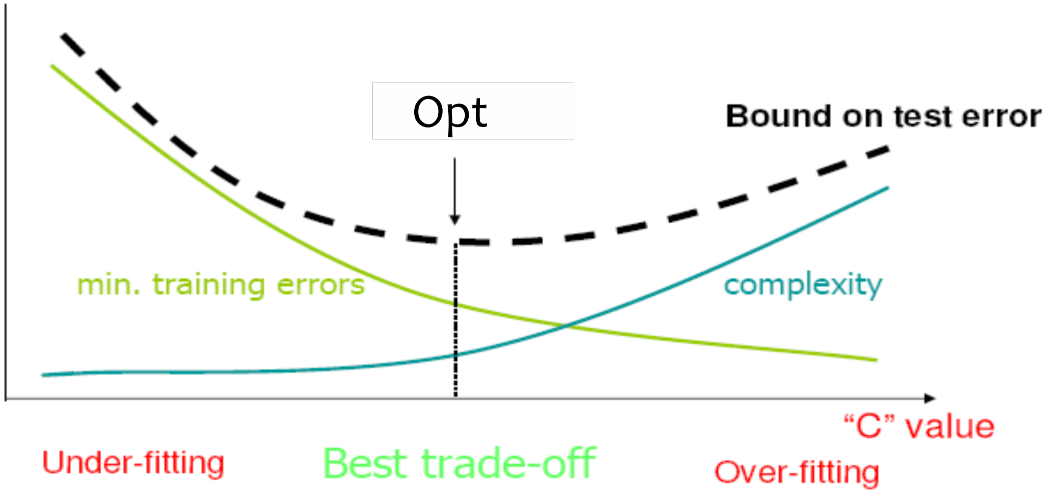


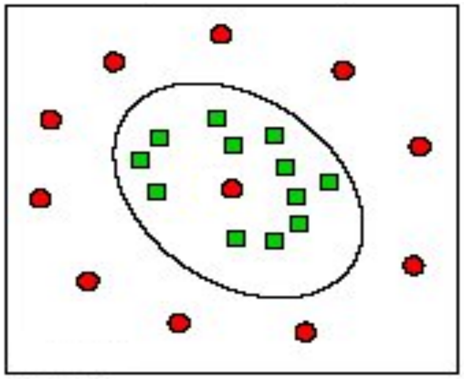
Too much simple!

Over-Fitting



Too much complicated!





Trade-Off

Pros and Cons with SVM

Pros

- It works really well with clear margin of separation
- It is effective in high dimensional spaces
- It is effective in cases where number of dimensions is greater than the number of samples
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient

Cons

- It doesn't perform well when we have large data set because the required training time is higher
- It also doesn't perform very well when the data set has more noise i.e. target classes are overlapping

SVM using Scikit-Learn in Python

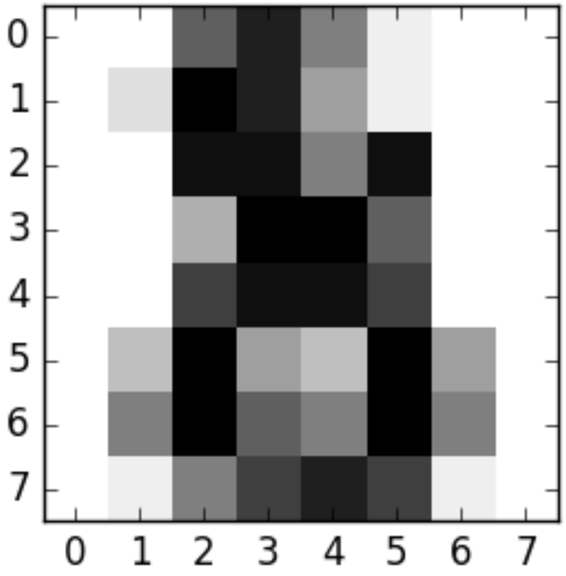
```
#Import Library
from sklearn import svm

# Create SVM classification object
model = svm.SVC(kernel='linear', c=1)

# Train the model using the training sets
model.fit(X, y)

#Predict Output
predicted= model.predict(x_test)
```

Case study: the digits dataset

- This dataset is made up of 1797 8x8 images

- Each image, like the one shown below, is of a hand-written digit

- **The goal is to recognize handwritten digits**


0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7