AI COURSE

Supervised Learning - Part 1

Thanh Ha DO

AI Faculty
Posts and Telecommumications Institute of Technology

August 1, 2025

OUTLINE

Classification

Definition

Given a collection of records (training set) Each record is characterized by a tuple (**x**, y), where **x** is the attribute set and y is the class label

- Task: Learn a model that maps each attribute set **x** into one of the predefined class labels y
- Example:

Task	Attribute set, **x**	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam

Classification

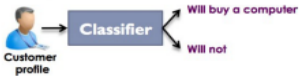
Classification vs. Prediction

Classification

Predicts categorical class labels (discrete or nominal)

Use labels of the training data to classify new data

Example

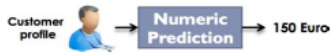


Classifier is contsructed to predict **categorical labels** such as *safe* or *risky* for a loan application data

Prediction

Models continuous-valued functions, i.e., predicts unknown or missing values

Example



Predict how much a given customer will spend during a sale

Unlike classification, it provides ordered values

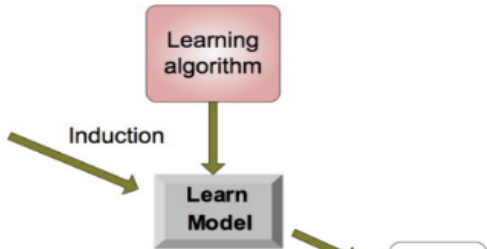
Regression analysis is used for prediction

Classification

General Approach for Building Classification Model

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

KNN
Nearest Neigghbors

Suppose were given a novel input vector x that wed like to classify. The idea is to find the nearest input vector to x in the training set and copy its label.

Can formalize nearest in terms of Euclidean distance

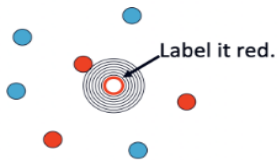
$$||\mathbf{x}^{(a)} - \mathbf{x}^{(b)}||_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

Algorithm:

1. Find example $(\mathbf{x}^*, t^*)$ (from the stored training set) closest to $\mathbf{x}$. That is:

$$\mathbf{x}^* = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\text{argmin}} \text{ distance}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. Output $y = t^*$



KNN
Nearest Neigghbors

Distance Metric $\left| \quad , \quad \right| \rightarrow \mathbb{R}$

L1 distance: $d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$

test image			
56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

\-

training image			
10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

pixel-wise absolute value differences			
46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

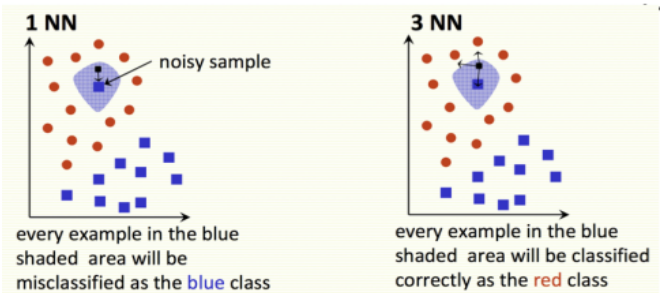
add → 456

KNN

Nearest Neigghbors

Nearest neighbors are sensitive to noise or mislabeled data (class noise).

Solution: Smooth by having k nearest neighbors vote



KNN

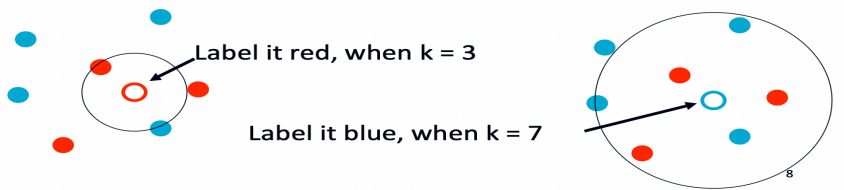
k-Nearest Neigghbors

Algorithm (kNN):

1. Find k examples $\{\mathbf{x}^{(i)}, t^{(i)}\}$ closest to the test instance $\mathbf{x}$
2. Classification output is majority class

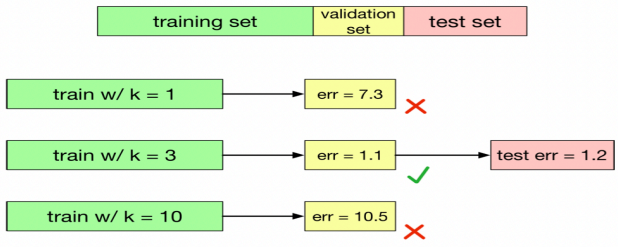
$$y = \underset{t^{(z)}}{\text{argmax}} \sum_{i=1}^{k} \mathbb{I}\{t^{(z)} = t^{(i)}\}$$

Istatement is the identity function equal to one whenever the statement is true. We could also write this as σ(t(z), t(i)) with σ(a, b) = 1 if a = b, 0 otherwise. I1.



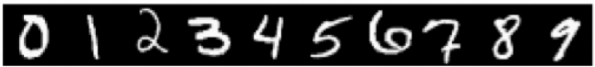
KNN
k value

- k is an example of a hyperparameter, something we cant fit as part of the learning algorithm itself
- We can tune hyperparameters using a validation set
- The test set is used only at the very end, to measure the generalization performance of the final configuration.



KNN

Example - Digit Classification



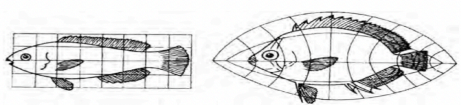
- Yann LeCunn – MNIST Digit Recognition
  - Handwritten digits
  - 28x28 pixel images: $d = 784$
  - 60,000 training samples
  - 10,000 test samples
- Nearest neighbour is competitive

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

KNN

Example - Digit Classification

- KNN can perform a lot better with a good similarity measure.
- Example: shape contexts for object recognition. They tried to warp one image to match the other to achieve invariance to image transformations.
  - Distance measure: average distance between corresponding points on warped images
- Achieved 0.63 % error on MNIST, compared with 3% for Euclidean KNN. Competitive with conv nets at the time, but required careful engineering.



[Belongie, Malik, and Puzicha, 2002. Shape matching and object recognition using shape contexts.]

KNN
Practical

- Practical in KNN Algorithm and others at the link `https://github.com/mantasu/cs231n/blob/master/assignment1/knn.ipynb`
- Python scikit-learn for Machine Learning `https://scikit-learn.org/stable/`

Reference - Python scikit-learn for Machine Learning

Classification	Module	Embedded Functions
Data example	sklearn.datasets	Dataset for practicing
Feature processing	sklearn.preprocessing	Pre-processing techniques(One-hot encoding, **normalization**, **scaling**, etc.)
	sklearn.feature_selection	Technique to search and select a feature that provides a significant impact to the model
	sklearn.feature_extraction	Feature extraction from source data The supporting API for feature extraction regarding image is present in the submodule image, while the supporting API for text data feature extraction is present in the submodule test.
Dimension reduction	sklearn.decomposition	Algorithms related to dimension reduction(PCA, NMF, Truncated SVD, etc.)
Validation, hyperparameter tuning, data separation	sklearn.model_selection	Validation, hyperparameter tuning, data separation, etc.(corss_validate, GridSearchCV, **train_test_split**, learning_curve, etc.)
Model evaluation	sklearn.metrics	Techniques to measure and evaluate model performance(accuracy, precision, recall, ROC curve, etc.)

Reference - Python scikit-learn for Machine Learning

Classification	Module	Embedded Functions
Machine learning algorithm	sklearn.ensemble	Ensemble algorithms (Random forest, AdaBoost, bagging, etc.)
	sklearn.linear_model	Linear algorithms (Linear regression, logistic regression, SGD, etc.)
	sklearn.naïve_bayes	Naive Bayes algorithms(Bernoulli NB, Gaussian NB, multinomial distribution NB, etc.)
	sklearn.neighbors	Nearest neighbor algorithms (K-NN, etc.)
	sklearn.svm	Support Vector Machine algorithms
	sklearn.tree	Decision tree algorithms
	sklearn.cluster	Unsupervised learning (clustering) algorithms(Kmeans, DBSCAN, etc.)
Utility	sklearn.pipeline	Serial conversion of feature processing and machine learning algorithms, etc.

Reference - Python scikit-learn for Machine Learning

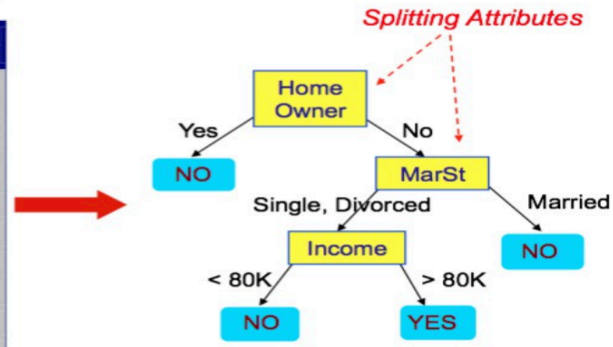
- To train a supervised learning model: myModel.fit(X_train, Y_train)
- To train a unsupervised learning model: myModel.fit(X_train)
- To predict using an already trained model: myModel.predict(X_test)
- To import a preprocessor as class:
  from sklearn.preprocessing import <a preprocessor>
- To split the dataset into a training set and a testing set:
  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=123)
- To calculate a performance metric (accuracy):
  metrics.accuracy_score(Y_test, Y_pred)
- To cross validate and do hyperparameter tuning at the same time:
  myGridCV = GridSearchCV(estimator, parameter_grid, cv=k)
  myGridCV.fit(X_train, Y_train)

Decision Tree

Example: A Decision Tree

	categorical	categorical	continuous	class
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

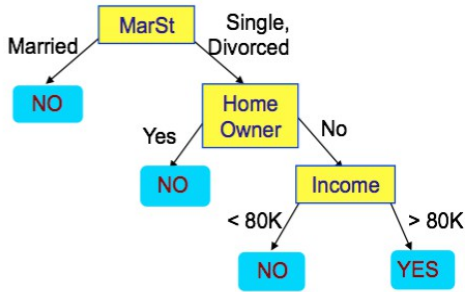


Model: Decision Tree

Decision Tree

Example: A Decision Tree

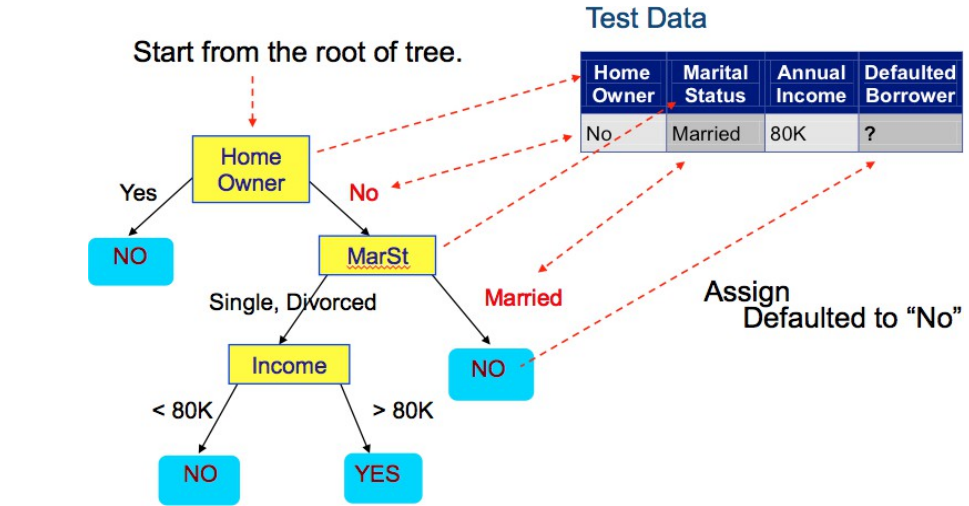
		categorical	categorical	continuous	class
	ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	1	Yes	Single	125K	No
	2	No	Married	100K	No
	3	No	Single	70K	No
	4	Yes	Married	120K	No
	5	No	Divorced	95K	Yes
	6	No	Married	60K	No
	7	Yes	Divorced	220K	No
	8	No	Single	85K	Yes
	9	No	Married	75K	No
	10	No	Single	90K	Yes



There could be more than one tree that fits the same data

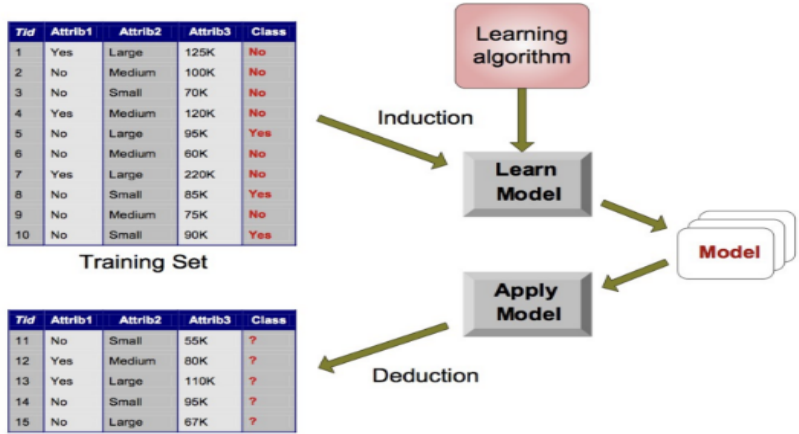
Decision Tree

Example: Apply Model to Test Data



Decision Tree

Decision Tree Classification Task: Learning Algorithm = Tree Induction Algorithm

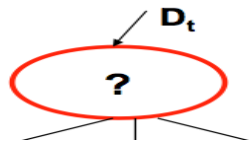


Decision Tree

Decision Tree Induction: Hunts Algorithm

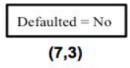
- Let D_t be the set of training records that reach a note t
- General Recursive Procedure:
  - If D_t contains records that belong to the same class y_t , then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class yd
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset
- Stopping condition: All the records in the subset belong to the same class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

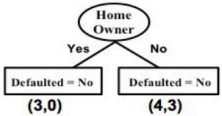


Decision Tree

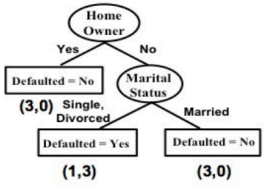
Hunts Algorithm (cont)



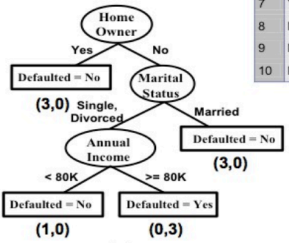
(a)



(b)



(c)



(d)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

Decision Tree

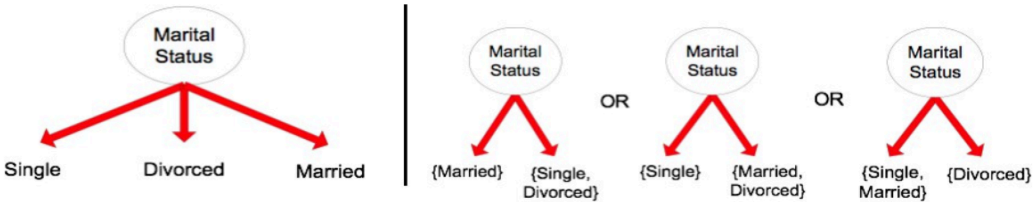
Design Issues of Decision Tree Induction

- How should training records be split?
  - Method for specifying test condition
    - Depending on attribute types: binary, nominal, ordinal, continuous
    - Depending on number of ways to split: 2-way split, multi-way split
  - Measure for evaluating the goodness of a test condition
- How should the splitting procedure stop?
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - Early termination

Decision Tree

Test Condition for Nominal Attributes

- Multi-way split
  - Use as many partitions as distinct values
- Binary split
  - Divides values into two subsets
  - Preserve order property among attribute values



Decision Tree

Splitting Based on Continuous Attributes

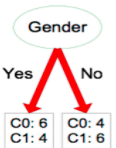
- Discretization to form an ordinal categorical attribute Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
  - Static – discretize once at the beginning
  - Dynamic – repeat at each node
- Binary Decision: $(A < v) or (A \leq v)$
  - consider all possible splits and finds the best cut
  - can be more compute intensive

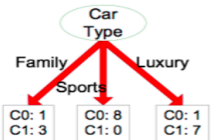
Decision Tree

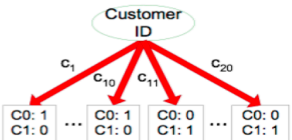
How to determine the best split

- Before Splitting:
  - 10 records of class 0
  - 10 records of class 1
- Whis test condition is the best?

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1







Decision Tree

How to determine the best split

- Greedy approach:
  - Nodes with purer class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Hight degree of impurity	Low level of impurity

Decision Tree

Measures of Node Impurity

- Gini Index

$$\text{GINI}(t) = 1 - \sum_j [p(j|t)]^2$$

- Entropy

$$\text{Entropy(t)} = -\sum p(j|t) \log p(j|t)$$

- Misclassification error

$$\text{Error}(t) = 1 - \max P(i|t)$$

Decision Tree

Finding the Best Split

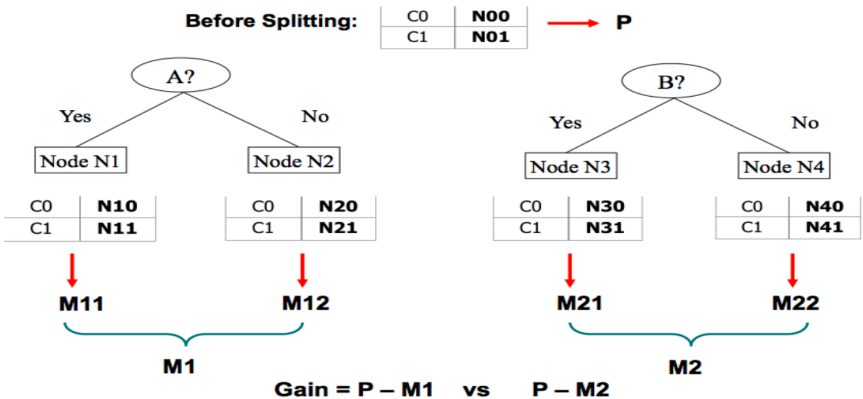
- Compute impurity measure (**P**) before splitting
- Compute impurity measure (**M**) after splitting
  - Compute impurity measure of each child node
  - **M** is the weighted impurity of children
- Choose the attribute test condition that produces the highest gain

$$\text{Gain} = \mathbf{P} - \mathbf{M}$$

or equivalently, lowest impurity measure after splitting (**M**)

Decision Tree

Finding the Best Split



Decision Tree

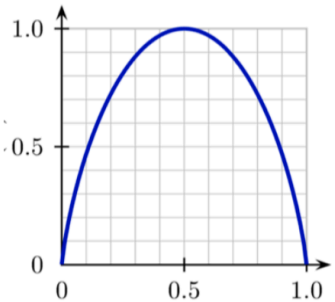
Measure of Impurity: Entropy

- Entropy at a given node t:

$$\text{Entropy(t)} = -\sum p(j|t) \log p(j|t)$$

NOTE: $p(j|t)$ is the relative frequency of class j at note t

- The higher the entropy, the less confident we are in the outcome



Decision Tree

Computing Entropy of a Single Node

$$\text{Entropy(t)} = -\sum p(j|t) \log p(j|t)$$

C1	**0**
C2	**6**

$\mathbf{P}(C1) = \frac{0}{6} = 0$; $\mathbf{P}(C2) = \frac{6}{6} = 1$
Entropy = -0 log0 -1 log1 = -0 -0 = 0

C1	**1**
C2	**5**

$\mathbf{P}(C1) = \frac{1}{6}$; $\mathbf{P}(C2) = \frac{5}{6}$
Entropy = $-\frac{1}{6} \log_2 (\frac{1}{6}) - \frac{5}{6} \log_2(\frac{5}{6}) = 0.65$

C1	**2**
C2	**4**

$\mathbf{P}(C1) = \frac{2}{6}$; $\mathbf{P}(C2) = \frac{4}{6}$
Entropy = $-\frac{2}{6} \log_2 (\frac{2}{6}) - \frac{4}{6} \log_2(\frac{4}{6}) = 0.92$

Decision Tree

Computing Information Gain After Splitting

- Information Gain:

$$\text{GAIN}_{split} = \text{Entropy}(p) - (\sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}(i))$$

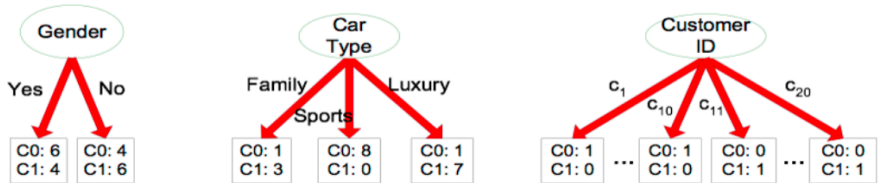
Parent Node, p is split into k partitions; n_i is number of records in partition i

- Choose the split that achieves most reduction (maximizes GAIN)

Decision Tree

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure



- Customer ID has highest information gain because entropy for all the children is zero

Decision Tree

Gain Ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{split}}{splitINFO} \quad \Big\| \quad splitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions; n_i is number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO)
  - Higher entropy partitioning (large number of small partitions) is penalized
  - SplitINFO = 1.52 (*Left*), 0.72 (*Middle*), and 0.97 (*Right*)

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10

Decision Tree

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
  - Space of possible decision trees is exponentially large. Greedy approaches are often unable to find the best tree.
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

Practical

- `https://scikit-learn.org/stable/modules/tree.html`

Predict the risk class of a car driver based on the following attributes

Attribute	Description	Values
time	time since obtaining a drivers license in years	{1-2, 2-7, >7}
gender	gender	{male, female}
area	residential area	{urban, rural}
risk	the risk class	{low, high}

For your analysis, you have the following manually classified training examples.

ID	time	gender	area	risk
1	1-2	m	urban	low
2	2-7	m	rural	high
3	>7	f	rural	low
4	1-2	f	rural	high
5	>7	m	rural	high
6	1-2	m	rural	high
7	2-7	f	urban	low
8	2-7	m	urban	low

Question:

(a) Construct a decision tree based on this training data ;

(b) Apply the decisi

ID	time	gender	area
A	1-2	f	rural
B	2-7	m	urban
C	1-2	f	urban

Naive Bayes

Bayes Classifiers

- A probabilistic framework for solving classification problems
- Conditional Probability

$$P(Y|X) = \frac{P(X,Y)}{P(X)}$$

$$P(X|Y) = \frac{P(X,Y)}{P(Y)}$$

- Bayes theorem: "X" is feature, "Y" is class

$$\underbrace{\mathrm{P(Class \mid Feature)}}_{\textbf{Posterior}} = \frac{\overbrace{\mathrm{P(Feature \mid Class)}}^{\textbf{Likelihood}}\overbrace{\mathrm{P(Class)}}^{\textbf{Prior}}}{\underbrace{\mathrm{P(Feature)}}_{\textbf{Evidence}}}$$

Naive Bayes

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M|S) = \frac{P(S|M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Naive Bayes

Using Bayes Theorem for Classification

- Consider each attribute and class label as random variables
- Given a record with attributes $(X_1, X_2, ..., X_d)$
  - Goal is to predict class Y
  - Specifically, we want to find the value of Y that maximizes $P(Y|X_1, X_2, ..., X_d)$
- Can we estimate $P(Y|X_1, X_2, ..., X_d)$ directly from data?
- For example:
  - Given $X = (Refund = No, Divorced, Income = 120K)$
  - Estimate $P(Evade = Yes|X)$ and $P(Evade = No|X)$?

Tid	Refund	Marital Status	Taxable Income	Evade
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

Naive Bayes

Using Bayes Theorem for Classification

- Approach:
  - compute posterior probability $P(Y|X_1X_2....X_d)$ using the Bayes theorem

$$P(Y|X_1X_2....X_d) = \frac{P(X_1X_2....X_d|Y)P(Y)}{P(X_1X_2....X_d)}$$

  - *Maximum a-posteriori*: Choose Y that maximizes

$$P(Y|X_1X_2....X_d)$$

  - Equivalent to choosing value of Y that maximizes

$$P(X_1X_2....X_d|Y)P(Y)$$

- How to estimate $P(X_1X_2...X_d|Y)$?

Naive Bayes
Example Data

- Given $X = (Refund = No, Divorced, Income = 120K)$
- Using Bayes Theorem:

$$P(Yes|X) = \frac{P(X|Yes)P(Yes)}{P(X)}$$

$$P(No|Y) = \frac{P(X|No)P(No)}{P(X)}$$

- How to estimate $P(X|Yes)$ and $P(X|No)$?

Tid	Refund	Marital Status	Taxable Income	Evade
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

Naive Bayes

Naive Bayes Classifier

Assume independence among attributes X_i when class is given:

- $P(X_1 X_2 ... X_d | Y_j) = P(X_1 | Y_j) P(X_2 | Y_j) ... P(X_d | Y_j)$
- Now we can estimate $P(X_i | Y_j)$ for all X_i and Y_j combinations from the training data
- New point is classified to Y_j if $P(Y_j) \prod P(X_i | Y_j)$ is maximal

Naive Bayes
Conditional Independence

- $\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ if $P(\mathbf{X}|\mathbf{YZ}) = P(\mathbf{X}|\mathbf{Z})$
- Example: Arm length and reading skills
  - Young child has shorter arm length and limited reading skills, compared to adults
  - If age is fixed, no apparent relationship between arm length and reading skills
  - Arm length and reading skills are conditionally independent given age

Naive Bayes

Naive Bayes on Example Data

Given $X = (Refund = No, Divorced, Income = 120K)$

$$P(X|Yes) = P(Refund = No|Yes) \times P(Divorced|Yes) \times P(Income = 120K|Yes)$$

$$P(X|No) = P(Refund = No|No) \times P(Divorced|No) \times P(Income = 120K|No)$$

Tid	Refund	Marital Status	Taxable Income	Evade
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

Naive Bayes

Naive Bayes on Example Data

- Class: $P(Y) = N_c/N$
  - e.g., $P(No) = \frac{7}{10}$; $P(Yes) = \frac{3}{10}$
- For categorical attributes: $P(X_i|Y_k) = |X_{ik}|/N_{ck}$
  - where $|X_{ik}|$ is number of instances having attribute value X_i and belonging to class Y_k
  - eg.:

$$P(Status = Married|No) = \frac{4}{7}$$

$$P(Refund = Yes|Yes) = 0$$

Tid	Refund	Marital Status	Taxable Income	Evade
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

Naive Bayes

Estimate Probabilities from Data

For continuous attributes:

- Discretization: Partition the range into bins:
  - Replace continuous value with bin value
    - Attribute changed from continuous to ordinal
- Probability density estimation:
  - Assume attribute follows a normal distribution
  - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
  - Once probability distribution is known, use it to estimate the conditional probability $P(X_i|Y)$

Naive Bayes

Estimate Probabilities from Data

- Normal distribution

$$P(X_i|Y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(X_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (X_i, Y_i) pair
- For (Income, Class = No):
  - If Class = No
    - sample mean = 110
    - sample variance = 2975

Tid	Refund	Marital Status	Taxable Income	Evade
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

$$P(Income = 120|No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Naive Bayes

Example of Naive Bayes Classifier

Given $X = (Refund = No, Divorced, Income = 120K)$

$$P(Refund = Yes|No) = 3/7$$
$$P(Refund = No|No) = 4/7$$
$$P(Refund = Yes|Yes) = 0$$
$$P(Refund = No|Yes) = 1$$
$$P(MaritalStatus = Single|No) = 2/7$$
$$P(MaritalStatus = Divorced|No) = 1/7$$
$$P(MaritalStatus = Married|No) = 4/7$$
$$P(MaritalStatus = Single|Yes) = 2/3$$
$$P(MaritalStatus = Divorced|Yes) = 1/3$$
$$P(MaritalStatus = Married|Yes) = 0$$

For Taxable Income:

$$If class = No : sample mean = 110; sample variance = 2975$$
$$= Yes : sample mean = 90; sample variance = 25$$

$$\begin{aligned} P(X|No) &= P(Refund = No|No) \\ &\times P(Divorced|No) \\ &\times P(Income = 120K|No) \\ &= \frac{4}{7} \times \frac{1}{7} \times 0.0072 = 0.0006 \end{aligned}$$

$$\begin{aligned} P(X|Yes) &\longrightarrow \text{Class} = \text{No} \\ &\times P(Divorced|Yes) \\ &\times P(Income = 120K|Yes) \\ &= 1 \times \frac{1}{3} \times \frac{1}{2} \times 10^{-9} = 4 \times 10^{-10} \end{aligned}$$

Naive Bayes

Example of Naive Bayes Classifier

Given $X = (Refund = No, Divorced, Income = 120K)$

$$P(Refund = Yes|No) = 3/7$$
$$P(Refund = No|No) = 4/7$$
$$P(Refund = Yes|Yes) = 0$$
$$P(Refund = No|Yes) = 1$$
$$P(MaritalStatus = Single|No) = 2/7$$
$$P(MaritalStatus = Divorced|No) = 1/7$$
$$P(MaritalStatus = Married|No) = 4/7$$
$$P(MaritalStatus = Single|Yes) = 2/3$$
$$P(MaritalStatus = Divorced|Yes) = 1/3$$
$$P(MaritalStatus = Married|Yes) = 0$$

For Taxable Income:

- $P(Yes) = \frac{3}{10}; P(No) = \frac{7}{10}$
- $P(Yes|Divorced) = \frac{1}{3} \times \frac{3}{10} / P(Divorced)$
- $P(No|Divorced) = \frac{1}{7} \times \frac{7}{10} / P(Divorced)$
- $P(Yes|Refund = No, Divorced) = 1 \times \frac{1}{3} \times \frac{3}{10} / P(Divorced, Refund = No)$
- $P(No|Refund = No, Divorced) = \frac{4}{7} \times \frac{1}{7} \times \frac{7}{10} / P(Divorced, Refund = No)$

Naive Bayes

Example of Naive Bayes Classifier

Given $X = (Refund = No, Divorced, Income = 120K)$

$$P(Refund = Yes|No) = 3/7$$
$$P(Refund = No|No) = 4/7$$
$$P(Refund = Yes|Yes) = 0$$
$$P(Refund = No|Yes) = 1$$
$$P(MaritalStatus = Single|No) = 2/7$$
$$P(MaritalStatus = Divorced|No) = 1/7$$
$$P(MaritalStatus = Married|No) = 4/7$$
$$P(MaritalStatus = Single|Yes) = 2/3$$
$$P(MaritalStatus = Divorced|Yes) = 1/3$$
$$P(MaritalStatus = Married|Yes) = 0$$

- $P(Yes) = \frac{3}{10}; P(No) = \frac{7}{10}$
- $P(Yes|Married) = 0 x \frac{3}{10} / P(Married)$
- $P(No|Married) = \frac{4}{7} x \frac{7}{10} / P(Married)$

For Taxable Income:

$If class = No : sample mean = 110; sample variance = 2975$
$= Yes : sample mean = 90; sample variance = 25$

Naive Bayes

Issues with Naive Bayes Classifier

$$P(Refund = Yes|No) = 2/6$$
$$P(Refund = No|No) = 4/6$$
$$P(Refund = Yes|Yes) = 0$$
$$P(Refund = No|Yes) = 1$$
$$P(MaritalStatus = Single|No) = 2/6$$
$$P(MaritalStatus = Divorced|No) = 0$$
$$P(MaritalStatus = Married|No) = 4/6$$
$$P(MaritalStatus = Single|Yes) = 2/3$$
$$P(MaritalStatus = Divorced|Yes) = 1/3$$
$$P(MaritalStatus = Married|Yes) = 0/3$$

For Taxable Income:

$$If class = No : sample mean = 91; sample variance = 685$$
$$= Yes : sample mean = 90; sample variance = 25$$

Tid	Refund	Marital Status	Taxable Income	Evade
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
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Naive Bayes

Issues with Naive Bayes Classifier

- If one of the conditional probabilities is zero, then the entire expression becomes zero
- Need to use other estimates of conditional probabilities than simple fractions
- Probability estimation:
  - Original: $P(A_i|C) = \frac{N_{ic}}{N_c}$
  - Laplace: $P(A_i|C) = \frac{N_{ic}+1}{N_c+c}$
  - m - estimate: $P(A_i|C) = \frac{N_{ic}+mp}{N_c+m}$

c: number of classes; p: prior probability of the class, m: parameter; N_c: number of instances in the class; N_{ic}: number of instances having attribute value A_i in class c

Naive Bayes

Naive Bayes Classifier (Summary)

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN) that *provides graphical representation of probabilistic relationships among a set of random variables*

Practical

- `https://scikit-learn.org/stable/modules/naive_bayes.html`
- Exercise: Given the training data in the table (by computer data)
  - Predict the class of the following new example using Naive Bayes Classification
    Age <= 30, income = medium, student = yes, credit-rating = fair

RID	age	income	student	credit_rating	Class: buys_computer
1	<=30	high	no	fair	no
2	<=30	high	no	excellent	no
3	31 . . . 40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	31 . . . 40	low	yes	excellent	yes
8	<=30	medium	no	fair	no
9	<=30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<=30	medium	yes	excellent	yes
12	31 . . . 40	medium	no	excellent	yes
13	31 . . . 40	high	yes	fair	yes
14	>40	medium	no	excellent	no

Support Vector Machines

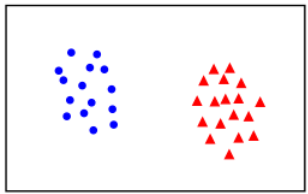
- derived from statistical learning theory by Vapnik and Chervonenkis (COLT-92)
- Supervised learning methods for classification and regression relatively new class of successful learning methods
- They can represent non-linear functions and they have an efficient training algorithm
- SVM got into mainstream because of their exceptional performance in Handwritten Digit Recognition

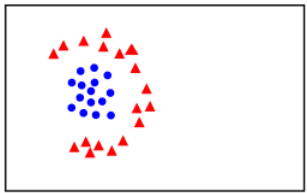
Binary Classification

- Given training data $(\mathbf{x}_i, y_i)$ for $i = 1, ..., N$, with $\mathbf{x}_i \in R^d$ and $y_i \in \{-1, 1\}$, learn a classifier $f(x)$ such as

$$f(\mathbf{x}_i) \begin{cases} \geq 0, & y_i = +1 \\ < 0, & y_i = -1 \end{cases}$$

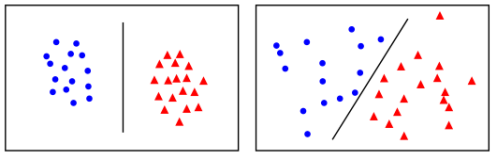
- i.e. $y_i f(\mathbf{x}_i) > 0$ for a correct classification



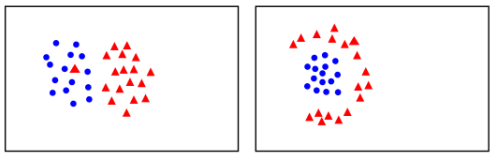


Linear separability

linear separable



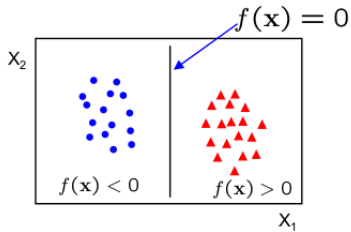
no linear separable



Linear separability

- A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

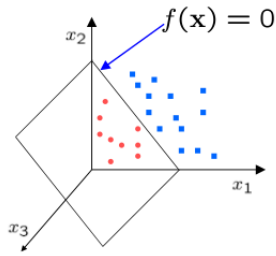


- in which:
  - in 2D the discriminant is a line
  - **w** is the normal to the line, and b the bias
  - **w** is known as the weight vector

Linear separability

- A linear classifier has the form

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$



- in which:
  - in 3D the discriminant is a plane, and in nD it is a hyperplane
  - for a linear classifier, the training data is used to learn **w** and then discarded. Only **w** is needed for classifying new data

The Perceptron Classifier

- Given linearly separable data $\mathbf{x}_i$ labelled into two categories $y_i = \{-1, 1\}$, find a weight vector $\mathbf{w}$ such that the following discriminant function separates the categories for $i = 1, ..., N$

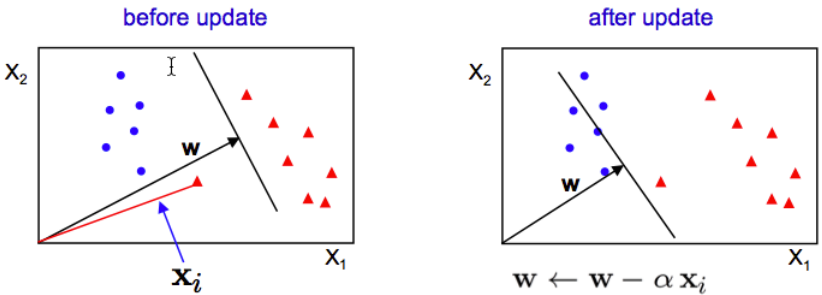
$$f(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i + b$$

The Perceptron Algorithm

- Write classifier as $f(\mathbf{x}_i) = \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i + w_0 = \mathbf{w}^T \mathbf{x}_i$ where $\mathbf{w} = (\tilde{\mathbf{w}}, w_0)$, $\mathbf{x}_i = (\tilde{\mathbf{x}}_i, 1)$
  - Initialize $\mathbf{w} = 0$
  - Cycle though the data points $\{\mathbf{x}_i, y_i\}$.
    - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \text{sign}(f(\mathbf{x}_i))\mathbf{x}_i$
  - Until all the data is correctly classified

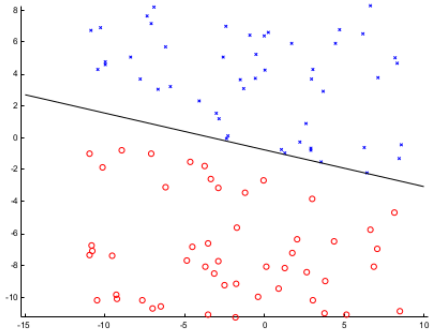
For example in 2D

- Initialize $\mathbf{w} = 0$
- Cycle though the data points $\{\mathbf{x}_i, y_i\}$.
  - if $\mathbf{x}_i$ is misclassified then $\mathbf{w} \leftarrow \mathbf{w} + \alpha \text{sign}(f(\mathbf{x}_i))\mathbf{x}_i$
- Until all the data is correctly classified



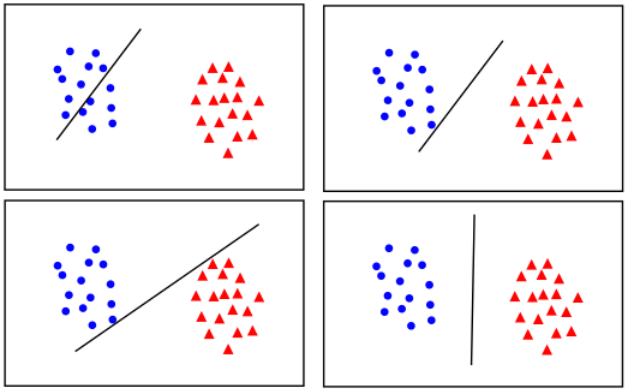
after convergence $\mathbf{w} = \sum_i^N \alpha_i \mathbf{x}_i$

Perceptron example



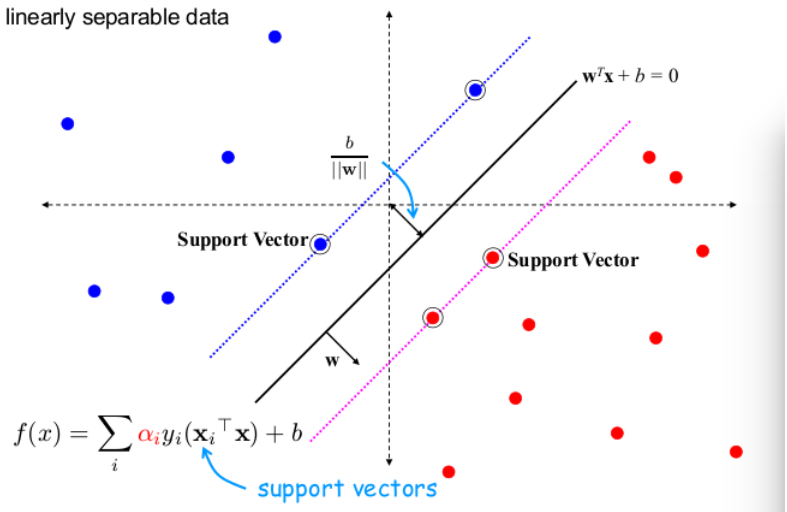
- if the data is linearly separable, then the algorithm will converge
- convergence can be slow ...
- separating line close to training data
- we would prefer a larger margin for generalization

What is the best w?



maximum margin solution: most stable under perturbations of the inputs

Support Vector Machine

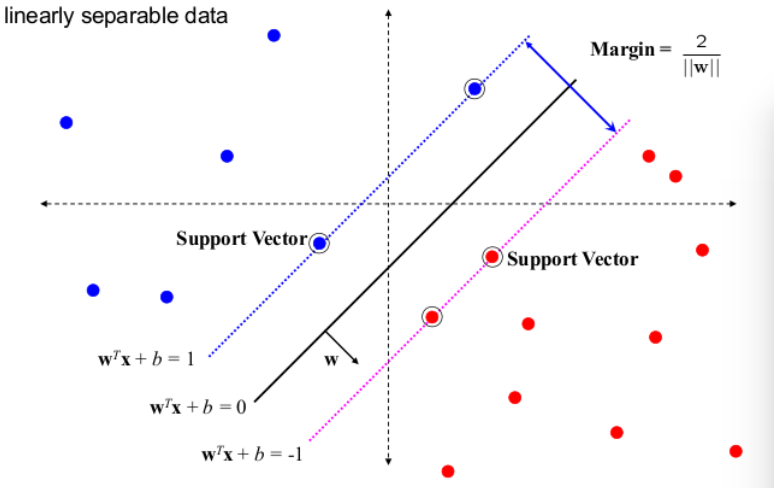


SVM - sketch derivation

- Since $\mathbf{w}^T\mathbf{x} + b = 0$ and $c(\mathbf{w}^T\mathbf{x} + b) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$
- Choose normalization such that $\mathbf{w}^T\mathbf{x}_+ + b = +1$ and $\mathbf{w}^T\mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively
- Then the margin is given by

$$\frac{\mathbf{w}}{||\mathbf{w}||} \cdot (\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^T(\mathbf{x}_+ - \mathbf{x}_-)}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$$

Support Vector Machine



SVM Optimization

- Learning the SVM can be formulated as an optimization:

$$\max_{\mathbf{w}} \frac{2}{||\mathbf{w}||} \text{ s.t } \mathbf{w}^T\mathbf{x}_i + b \begin{cases} \geq 1, & \text{if } y_i = +1 \\ \leq -1, & \text{if } y_i = -1 \end{cases} \text{ for } i = 1, ..., N$$

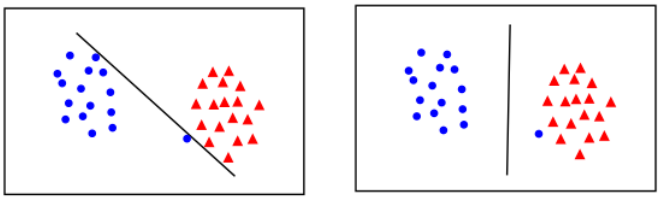
- Or equivalently

$$\min_{\mathbf{w}} ||\mathbf{w}||^2 \text{ s.t } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \text{ for } i = 1, ..., N$$

- This is a quadratic optimization problem subject to linear constraints and there is a unique minimum

Linear separability again: What is the best w?

- the points can be linearly separated but there is a very narrow margin
- but possibly the large margin solution is better, even though one constraint is violated

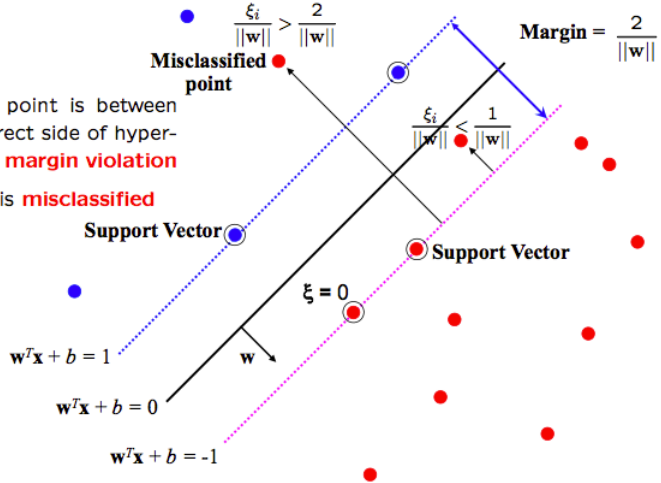


In general there is a trade off between the margin and the number of mistakes on the training data

Introduce slack variables

$\xi_i \geq 0$

- for $0 < \xi \leq 1$ point is between margin and correct side of hyper-plane. This is a **margin violation**
- for $\xi > 1$ point is **misclassified**



Soft margin solution

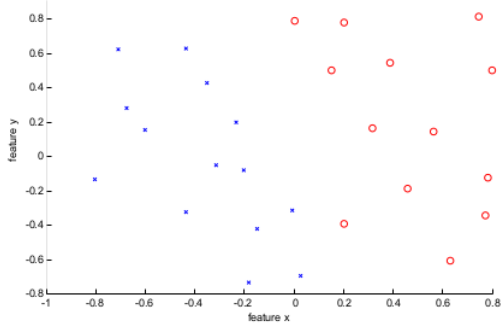
The optimization problem becomes

$$\min_{\mathbf{w}\in R^d, \xi_i \in R^+} ||\mathbf{w}||^2 + C\sum_i^N \xi_i$$

subject to

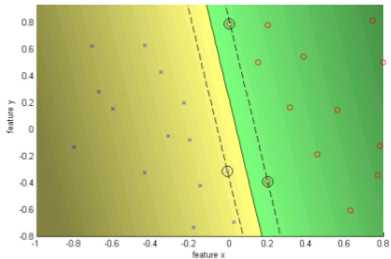
$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, ..., N$$

- Every constraint can be satisfied if ξ_i is sufficiently large
- C is a regularization parameter:
  - small C allows constraints to be easily ignored $\rightarrow$ large margin
  - large C makes constraints hard to ignore $\rightarrow$ narrow margin
  - $C = \infty$ enforces all constraints: hard margin
- This is still a quadratic optimization problem and there is a unique minimum. Note, there is only one parameter, C.

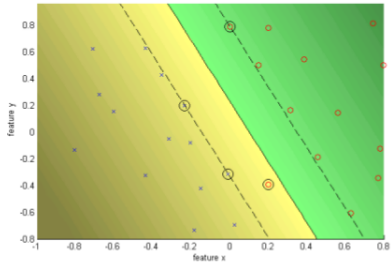


- data is linearly separable
- but only with a narrow margin

C = Infinity hard margin



C = 10 soft margin



Application: Pedestrian detection in Computer Vision

Objective: detect (localize) standing humans in an image

- cf face detection with a sliding window classifier

- reduces object detection to binary classification
- does an image window contain a person or not?

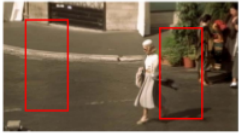
Method: the HOG detector

Training data and features

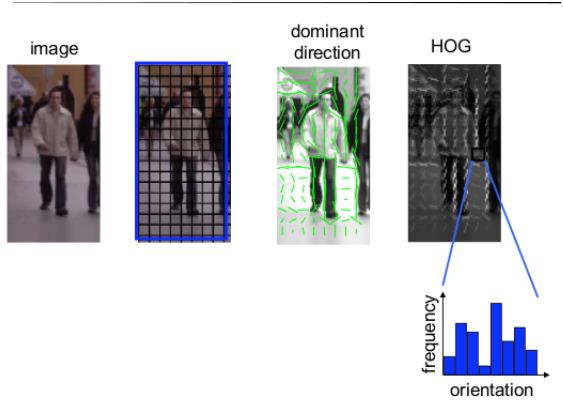
- Positive data 1208 positive window examples




- Negative data 1218 negative window examples (initially)

Training data and features



- tile window into 8×8 pixel cells
- each cell represented by HOG
- Feature vector dimension = 16×8 (for tiling) $\times$ 8 (orientations) = 1024

Algorithm

Training (Learning)

- Represent each example window by a HOG feature vector

$\mathbf{x}_i \in R^d$ with $d = 1024$

- Train a SVM classifier

Testing (Detection)

- Sliding window classifier

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

Dalal and Triggs, CVPR 2005

Optimization

- Learning an SVM has been formulated as a constrained optimization problem over $\mathbf{w}$ and ξ

$$\min_{\mathbf{w}\in R^d, \xi_i \in R^+} ||\mathbf{w}||^2 + C\sum_{i}^{N} \xi_i \text{ s. t } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, ..., N$$

- The constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\mathbf{x}_i) \geq 1 - \xi_i$$

  which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \max(0, 1 - y_i f(\mathbf{x}_i))$$

- Hence the learning problem is equivalent to the unconstrained optimization problem over $\mathbf{w}$

$$\min_{\mathbf{w}\in R^d} \underbrace{||\mathbf{w}||^2}_{\text{regularization}} + C\sum_{i}^{N} \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

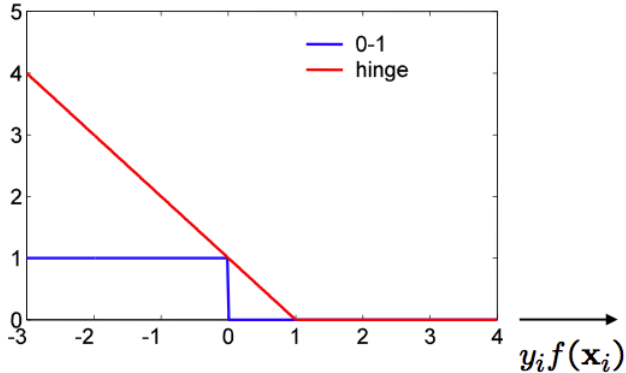
Loss function

$$\min_{\mathbf{w} \in R^d} ||\mathbf{w}||^2 + C \sum_{1}^{N} \underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

Points are in three categories:

- $y_i f(x_i) > 1$ Point is outside margin. No contribution to loss
- $y_i f(x_i) = 1$ Point is on margin. No contribution to loss. As in hard margin case.
- $y_i f(x_i) < 1$ Point violates margin constraint. Contributes to loss

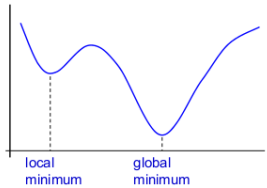
Loss function



- SVM uses hinge loss $\max(0, 1 - y_i f(\mathbf{x}_i))$
- an approximation to the 0-1 loss

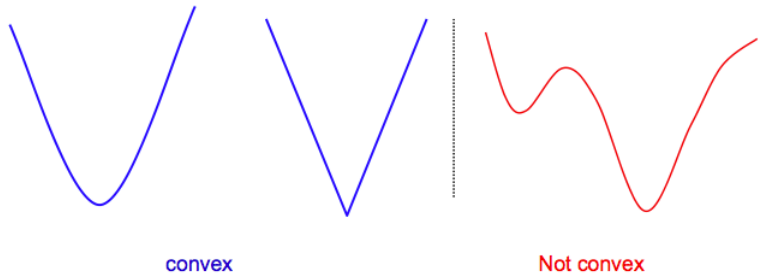
Optimization continued

$$\min_{\mathbf{w} \in R^d} C \sum_{i}^{N} \max(0, 1 - y_i f(\mathbf{x}_i)) + ||\mathbf{w}||^2$$

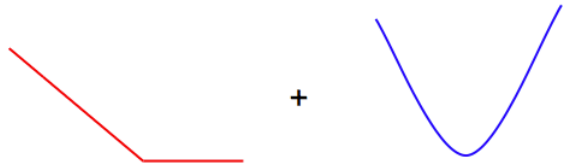


- Does this cost function have a unique solution?
- Does the solution depend on the starting point of an iterative optimization algorithm (such as gradient descent)?
- If the cost function is convex, then a locally optimal point is globally optimal

Convex function examples



- A non-negative sum of convex functions is convex



SVM

$$\min_{\mathbf{w}\in\mathbb{R}^d} C\sum_{i}^{N} \max\left(0, 1 - y_i f(\mathbf{x}_i)\right) + ||\mathbf{w}||^2 \qquad \text{convex}$$

Gradient (or steepest) descent algorithm for SVM

- To minimize a cost function $C(\mathbf{w})$ use the iterative update

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \nabla_{\mathbf{w}} C(\mathbf{w}_t)$$

  where η is the learning rate
- First, rewrite the optimization problem as an average

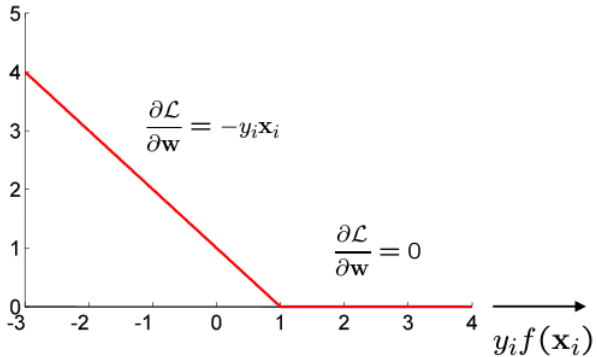
$$\min_{\mathbf{w}} C(\mathbf{w}) = \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{1}{N}\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

$$= \frac{1}{N}\sum_i^N (\frac{\lambda}{2}||\mathbf{w}||^2 + \max(0, 1 - y_i f(\mathbf{x}_i)))$$

  (with $\lambda = 2/(NC)$ up to an overall scale of the problem) and $f(\mathbf{x} = \mathbf{w}^T\mathbf{x} + b)$
- Because the hinge loss is not differentiable, a sub-gradient is computed

Sub-gradient for hinge loss

$$L(\mathbf{x}_i, y_i; \mathbf{w}) = \max(0, 1 - y_i f(\mathbf{x}_i))$$



Sub-gradient descent algorithm for SVM

$$C(\mathbf{w}) = \frac{1}{N}\sum_{i}^{N}(\frac{\lambda}{2}||\mathbf{w}||^2 + L(\mathbf{x}_i, y_i; \mathbf{w}))$$

- The iterative update is:

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta\nabla_{\mathbf{w}_t}C(\mathbf{w}_t) \\ &\leftarrow \mathbf{w}_t - \eta\frac{1}{N}\sum_{1}^{N}(\lambda\mathbf{w}_t + \nabla_{\mathbf{w}}L(\mathbf{x}_i, y_i; \mathbf{w}_t))\end{aligned}$$

  where η is the learning rate
- Then each iteration t involves cycling through the training data with the updates:

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t - \eta(\lambda\mathbf{w}_t - y_i\mathbf{x}_i) \text{ if } y_i f(\mathbf{x}_i) < 1 \\ &\leftarrow \mathbf{w}_t - \eta\lambda\mathbf{w}_t \text{ otherwise}\end{aligned}$$

- In the Pegasos algorithm the learning rate is set at $\eta_t = \frac{1}{\lambda_t}$

SVM -review

- We have seen that for an SVM learning a linear classifier

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

  is formulated as solving an optimization problem over $\mathbf{w}$:

$$\min_{\mathbf{w}\in R^d} ||\mathbf{w}||^2 + C\sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

- This quadratic optimization problem is known as the primal problem
- Instead, the SVM can be formulated to learn a linear classifier

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i^T\mathbf{x}) + b$$

  by solving an optimization problem over α_i
- This is know as the dual problem, and we will look at the advantages of this formulation

Sketch derivation of dual form

The Representer Theorem states that the solution $\mathbf{w}$ can always be written as a linear combination of the training data:

$$\mathbf{w} = \sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j$$

Primal and dual formulations

N is number of training points, and d is dimension of feature vector $\mathbf{x}$.

- Primal problem: for $\mathbf{w} \in R^d$

$$\min_{\mathbf{w} \in R^d} ||\mathbf{w}||^2 + C \sum_i^N \max(0, 1 - y_i f(\mathbf{x}_i))$$

- Dual problem for $\alpha \in R^N$

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j^T \mathbf{x}_k)$$

$$\text{s.t } 0 \leq \alpha_i \leq C \text{ for } \forall i \text{ and } \sum_i \alpha_i y_i = 0$$

  - Need to learn d parameters for primal, and N for dual
  - If $N \ll d$ then more efficient to solve for α than $\mathbf{w}$
  - Dual form only involves $(\mathbf{x}_j^T \mathbf{x}_k)$. We will return to why this is an advantage when we look at kernels.

Primal and dual formulations

- Primal version of classifier:

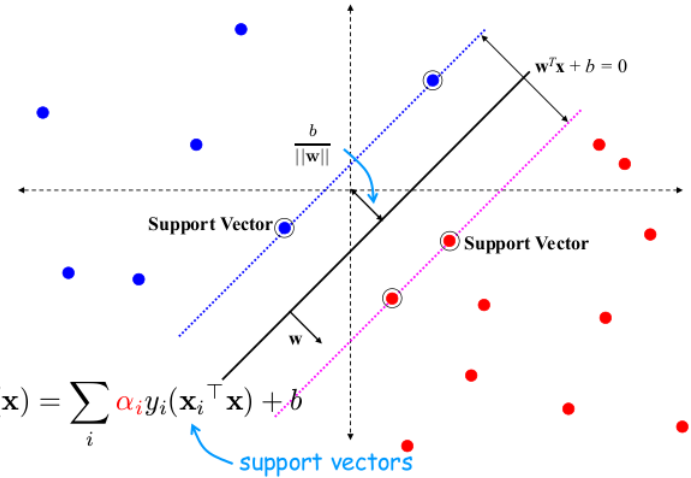
$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

- Dual version of classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i (\mathbf{x}_i^T \mathbf{x}) + b$$

In dual form, many of the α_is are zero. The ones that are non-zero define the support vectors $\mathbf{x}_i$.

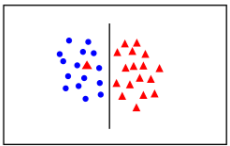
Support Vector Machine



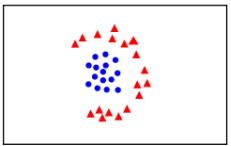
Handling data that is not linearly separable

- introduce slack variables

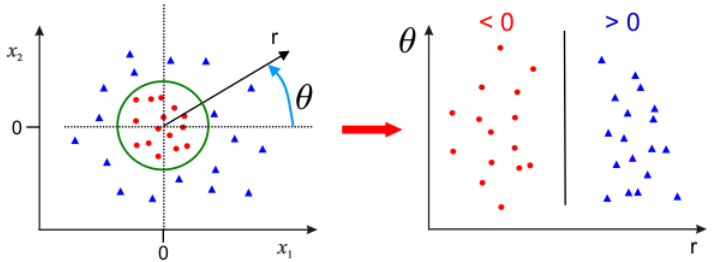
$$\min_{\mathbf{w}\in R^d, \xi_i \in R^+} ||\mathbf{w}||^2 + C\sum_{i}^{N} \xi_i \text{ s.t } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \text{ for } i = 1, ..., N$$



- linear classifier not appropriate



Solution 1: use polar coordinates

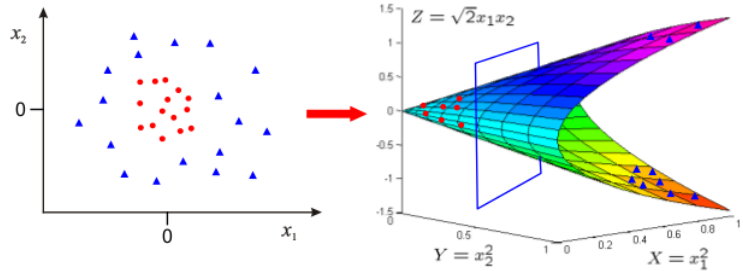


- Data is linearly separable in polar coordinates
- Acts non-linearly in original space

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} r \\ \theta \end{pmatrix}$$

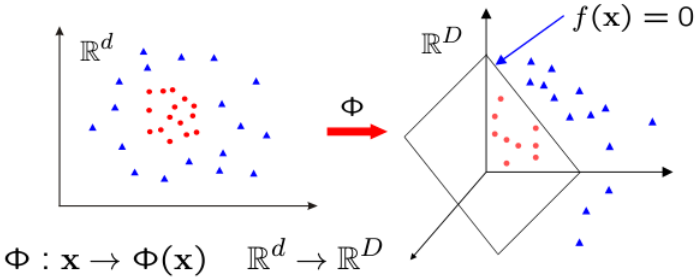
Solution 2: map data to higher dimension

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix}$$



- Data is linearly separable in 3D
- This means that the problem can still be solved by a linear classifier

SVM classifiers in a transformed feature space



Learn classifier linear in $\mathbf{w}$ for R^D

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

$\Phi(\mathbf{x})$ is a feature map

Primal Classifier in transformed feature space

Classifier, with $\mathbf{w} \in R^D$

$$f(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) + b$$

Learning, for $\mathbf{w} \in R^D$

$$\min_{\mathbf{w} \in R^D} ||\mathbf{w}||^2 + C \sum_{i}^{N} \max(0, 1 - y_i f(\mathbf{x}_i))$$

- Simply map $\mathbf{x}$ to $\Phi(\mathbf{x})$ where data is separable
- Solve for $\mathbf{w}$ in high dimensional space R^D
- If $D \gg d$ then there are many more parameters to learn for $\mathbf{w}$. Can this be avoided?

Dual Classifier in transformed feature space

Classifier:

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

$\rightarrow$

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i \Phi(\mathbf{x}_i)^T \Phi(\mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_{i} \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \mathbf{x}_j^T \mathbf{x}_k$$

$\rightarrow$

$$\max_{\alpha_i \geq 0} \sum_{i} \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k \Phi(\mathbf{x}_j)^T \Phi(\mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i \text{ and } \sum_{i} \alpha_i y_i = 0$$

Dual Classifier in transformed feature space

- Note, that $\Phi(\mathbf{x})$ only occurs in pairs $\Phi(\mathbf{x}_j)^T\Phi(\mathbf{x}_i)$
- Once the scalar products are computed, only the N dimensional vector α needs to be learnt; it is not necessary to learn in the D dimensional space, as it is for the primal
- Write $k(\mathbf{x}_j, \mathbf{x}_i) = \Phi(\mathbf{x}_j)^T\Phi(\mathbf{x}_i)$. This is known as a Kernel

Classifier:

$$f(\mathbf{x}) = \sum_i^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

Learning:

$$\max_{\alpha_i \geq 0} \sum_i \alpha_i - \frac{1}{2} \sum_{jk} \alpha_j \alpha_k y_j y_k k(\mathbf{x}_j, \mathbf{x}_k)$$

subject to

$$0 \leq \alpha_i \leq C \text{ for } \forall i \text{ and } \sum_i \alpha_i y_i = 0$$

Special transformations

$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix}$$

$$\begin{aligned} \Phi(\mathbf{x})^T\Phi(\mathbf{z}) &= (x_1^2, x_2^2, \sqrt{2}x_1x_2) \begin{pmatrix} z_1^2 \\ z_2^2 \\ \sqrt{2}z_1z_2 \end{pmatrix} \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= (x_1z_1 + x_2z_2)^2 \\ &= (\mathbf{x}^T\mathbf{z})^2 \end{aligned}$$

- Classifier can be learnt and applied without explicitly computing $\Phi(\mathbf{x})$
- All that is required is the kernel $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z})^2$

Example kernels

- Linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$
- Polynomial kernels $k(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^d$ for any $d > 0$
  - Contains all polynomials terms up to degree d
- Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp(-||\mathbf{x} - \mathbf{x}'||^2/2\sigma^2)$for $\sigma > 0$
  - Infinite dimensional feature space

SVM classifier with Gaussian kernel

N = size of training data

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

- Gaussian kernel $k(\mathbf{x}, \mathbf{x'}) = \exp(-||\mathbf{x} - \mathbf{x}'||^2/2\sigma^2)$
- Radial Basis Function (RBF) SVM

$$f(x) = \sum_{i}^{N} \alpha_i y_i \exp(-||\mathbf{x} - \mathbf{x}_i||/2\sigma^2) + b$$

Kernel Trick - Summary

- Classifiers can be learnt for high dimensional features spaces, without actually having to map the points into the high dimensional space
- Data may be linearly separable in the high dimensional space, but not linearly separable in the original feature space
- Kernels can be used for an SVM because of the scalar product in the dual form, but can also be used elsewhere they are not tied to the SVM formalism
- Kernels apply also to objects that are not vectors, e.g.

$$k(h, h') = \sum_k \min(h_k, h'_k) \text{ for histograms with bins } h_k, h'_k$$