University
Politehnica
of Bucharest

Faculty of
Automatic
Control and
Computers

Computer
Science and
Engineering
Department

Supervised Leaning

Ciprian-Octavian Truică
ciprian.truica@cs.pub.ro

Overview

- Naïve Bayes
- Support Vector Machines
- K-nearest neighbor
- Ensemble methods

Overview

- Naïve Bayes
- Support Vector Machines
- K-nearest neighbor
- Ensemble methods

Naïve Bayes Classifier

- This is a probabilistic classifier
- The algorithm is based on the Bayes theorem
- The probability of each element of the dataset to belong to a class is computed

Naïve Bayes Classifier

- Given a dataset of n elements $X = \{x_1, x_2, \dots, x_n\}$
- Each element from X is described by m attributes $A = \{A_1, A_2, \dots, A_m\}$. In other words, $x_i = \{a_{i1}, a_{i2}, \dots, a_{im}\}$, $i = \overline{1, n}$.
- And, given a set of k classes $C = \{c_1, c_2, \dots, c_k\}$
- The probability of an element x_i $(i = \overline{1, n})$ to belong to a class $c_j (j = \overline{1, k})$is:

$$P\left(C = c_j \middle| x = x_i\right)$$
$$= P\left(C = c_j \middle| A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im}\right)$$

- If classification is needed, the class with the highest probability may be assigned to that example

Naïve Bayes Classifier

- The Bayes theorem is:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{\mathrm{P}(B)}$$

- Where $P(A|B)$ means the probability of A given B

Naïve Bayes Classifier

- Example:
  - Students in a classroom are 60% from the AI M.Sc. module and 40% from other modules.
  - 20% of the students are placed in the first 2 rows of seats but for AI students this percent is 30%.
  - When the dean enters the class and sits somewhere in the first 2 rows, near a student, compute the probability that its neighbor is from AI?
- $P(AI) = 0.6$
- $P(2\ rows|AI) = 0.3$
- $P(2\ rows) = 0.2$
- $P(AI|2\ rows) = \frac{P(2\ rows|AI)\cdot P(AI)}{P(2\ rows)} = \frac{0.3\cdot 0.6}{0.2} = 0.9$

Naïve Bayes Classifier

- Building the classifier
- The objective is to compute

$$P(C = c_j | x = x_i) =$$
$$P(C = c_j | A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im})$$

- By applying the theorem

$$P(C = c_j | x = x_i) = \frac{P(x = x_i | C = c_j) \cdot P(C = c_j)}{P(x = x_i)}$$
$$= \frac{P(A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im} | C = c_j) \cdot P(C = c_j)}{P(A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im})}$$
$$= \frac{P(A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im} | C = c_j) \cdot P(C = c_j)}{\sum_{\beta=1}^{k} \left(P(A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im} | C = c_\beta) \cdot P(C = c_\beta) \right)}$$

Naïve Bayes Classifier

- Making the following assumption: "all attributes are conditionally independent given the class $C = c_j$" then:

$$P(x = x_i | C = c_j) = P(A_1 = a_{i1}, A_2 = a_{i2}, \dots, A_m = a_{im} | C = c_j)$$
$$= P(A_1 = a_{i1} | C = c_j) \cdot P(A_2 = a_{i2} | C = c_j) \cdot \dots \cdot P(A_m = a_{im} | C = c_j)$$
$$= \prod_{\alpha=1}^{m} P(A_\alpha = a_\alpha | C = c_j)$$

- Because of this assumption the method is called "naïve".
- Not in all situations the assumption is valid.
- The practice shows that the results obtained using this simplifying assumption are good enough in most of the cases.

Naïve Bayes Classifier

- Finally, replacing in the above expression we obtain:

$$P(C = c_j | x = x_i) = \frac{P(C = c_j) \cdot \prod_{\alpha=1}^{m} P(A_\alpha = a_\alpha | C = c_j)}{\sum_{\beta=1}^{k} \left(P(C = C_\beta) \cdot \prod_{\alpha=1}^{m} P(A_\alpha = a_\alpha | C = c_\beta) \right)}$$

- All probabilities in the above expression may be obtained by counting.

Naïve Bayes Classifier

- When only classification is needed, the denominator of the above expression may be ignored (is the same for all c_j) and the labeling class is obtained by maximizing the numerator:

$$C = argmax_{c_j} P(C = c_j) \cdot \prod_{\alpha=1}^{m} P(A_\alpha = a_\alpha | C = c_j)$$

Naïve Bayes Classifier

- Consider a simplified version of the Play Tennis table

Outlook	Wind	Play Tennis
Overcast	Weak	Yes
Overcast	Strong	No
Overcast	Absent	No
Sunny	Weak	Yes
Sunny	Strong	No
Rain	Strong	No
Rain	Weak	No
Rain	Absent	No

Naïve Bayes Classifier

P(Yes) = 2/8	P(No) = 6/8
P(Overcast | C = Yes) = 1/2	P(Overcast | C = No) = 2/6
P(Weak | C = Yes) = 2/2	P(Weak | C = No) = 1/6
P(Sunny | C = Yes) = 1/2	P(Sunny | C = No) = 1/6
P(Strong| C = Yes) = 0/2	P(Strong| C = No) = 3/6
P(Rain | C = Yes) = 0/2	P(Rain | C = No) = 3/6
P(Absent| C = Yes) = 0/2	P(Absent| C = No) = 2/6

- If the test example is:

Sunny	**Absent**	**???**

Naïve Bayes Classifier

- For $C = Yes$

$$P(Yes) * P(Sunny|Yes) * P(Absent|Yes) = \frac{2}{8} \cdot \frac{1}{2} \cdot \frac{0}{2} = 0$$

- For $C = No$

$$P(No) \cdot P(Sunny|No) \cdot P(Absent|No) = \frac{6}{8} \cdot \frac{1}{6} \cdot \frac{2}{6} = \frac{1}{24}$$

- The result is No (not a very wise result!)

Naïve Bayes Classifier

- Sometimes a class does not occur with a specific attribute value.
- This is problematic because it will result in a $P(A_i = a_i \mid C = c_j) = 0$ probability, which wipes out all the other probabilities
- For avoiding this situation, the smoothing is used and all the product terms are greater than zero
- The smooth equation is:

$$P(A_i = a_i | C = c_j) = \frac{a + s}{b + s \cdot r}$$

- Where
  - a is the number of training examples with $A_i = a_i$ and $C = c_j$
  - s is a multiplicative factor, commonly set to $s = \frac{1}{n}$, where n is the number of examples in the training set
  - b is the number of training examples with $C = c_j$
  - r is the number of distinct values for attribute A_i

Naïve Bayes Classifier

- $s = \frac{1}{8}$
- For attribute Outlook $r_{outlook} = 3$
- For attribute Wind $r_{wind} = 3$
- For $C = Yes$

$$P(Yes) * P(Sunny|Yes) * P(Absent|Yes) = \frac{2}{8} \cdot \frac{1}{2} \cdot \frac{0 + \frac{1}{8}}{2 + \frac{1}{8} \cdot 3} = \frac{1}{152}$$

- For $C = No$

$$P(No) \cdot P(Sunny|No) \cdot P(Absent|No) = \frac{6}{8} \cdot \frac{1}{6} \cdot \frac{2}{6} = \frac{1}{24}$$

- The result is No

Naïve Bayes Classifier

- If examples have missing values for attributes:
  - Ignore them
  - Replace them
- If the values for attributes are numerical:
  - Use discretization to make all the values categorical
  - Use Gaussian Naïve Bayes

Gaussian Naïve Bayes Classifier

- When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. (Wikipedia)
- Segment the data by each class c_j
- Compute for each continuous attributes A_i in class c_j its mean ($\mu_{A_i|c_j}$) and its variance $\sigma^2_{A_i|c_j}$
- For some observations V, the probability distribution of attribute $\mathrm{A_i} = v$ given a class c_j is:

$$P(A_i = v | C = c_j) = \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma^2_{A_i|c_j}}} e^{-\frac{(v - \mu_{A_i|c_j})^2}{2 \cdot \sigma^2_{A_i|c_j}}}$$

Overview

- Naïve Bayes
- Support Vector Machines
- K-nearest neighbor
- Ensemble methods

Support Vector Machines

- In this course is presented only the general idea of the Support Vector Machines (SVM) classification method.
- SVMs are described in detail in many articles and books, for example [Liu 2011] or [Han 2006].
- The method was discovered in the Soviet Union in '70 by Vladimir Vapnik and was developed in USA after Vapnik joined AT&T Bell Labs in early '90 (see [Cortes 1995]).

SVM

- Given a training datasetLabeled Dataset $D = \{(x_1, y_{1j}), (x_2, y_{2j}), \ldots, (x_n, y_{nj})\}$, where
  - $x_i = (a_{i1}, a_{i2}, \ldots, a_{im})$ is a vector in the R^m (all a_{ik} components are real numbers)
  - y_{ij} is the class label, $C = \{-1, +1\}$. If x_i is labeled with $+1$ then it belongs to the positive class, otherwise to the negative class

SVM – Linear

- A possible classifier is a linear function :

$$f(x) = < w \cdot x > + b$$

$$y_i = < w \cdot x_i > + b$$

- For example:

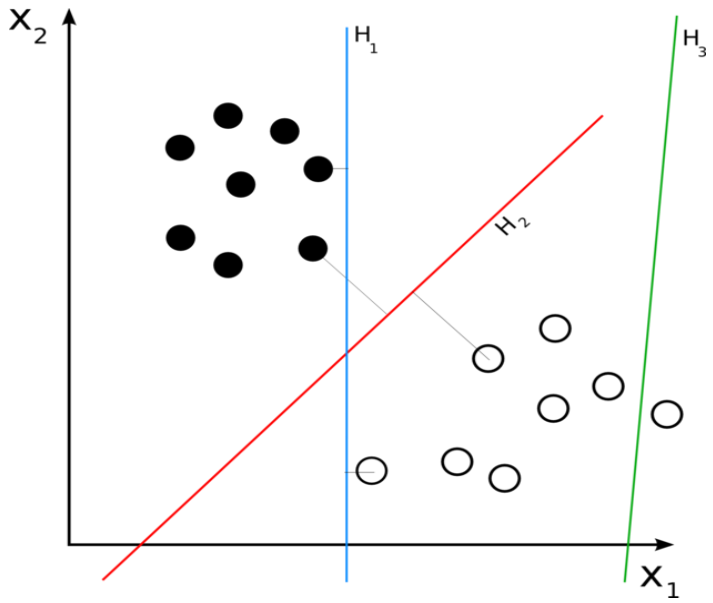
$$y_i = \begin{cases} 1 \; if \; < w \cdot x_i > + b \geq 0 \\ -1 \; if \; < w \cdot x_i > + b < 0 \end{cases}$$

- Where
  - w is a weight vector
  - $< w \cdot x >$ is the dot product of vectors $\boldsymbol{w}$ and $\boldsymbol{x}$
  - $\boldsymbol{b}$ is a real number
  - $\boldsymbol{w}$ and $\boldsymbol{b}$ may be scaled.

SVM – Linear

- The meaning of $\boldsymbol{f}$ is that the hyperplane $< w \cdot x > + b = 0$ separates the points of the training set D in two:
  - one half of the space contains the positive values and
  - the other half the negative values in D (like hyperplanes H_1 and H_2 in the next figure).
- All test examples can now be classified using f: the value of f gives the label for the example.

SVM – Linear

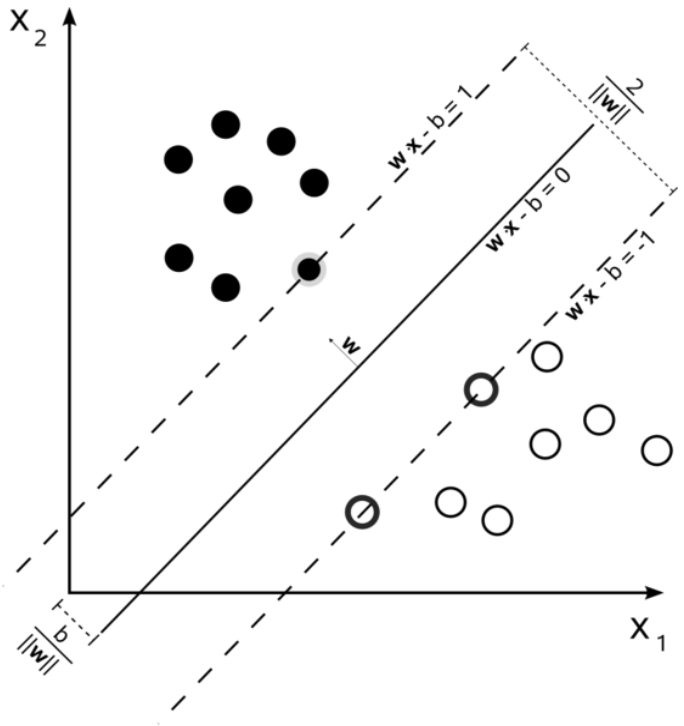


- Source Wikipedia

SVM – Linear

- SVM tries to find the ‘best’ hyperplane of that form.
- The theory shows that the best plane is the one maximizing the so-called ***margin*** (the minimum orthogonal distance between a positive and negative point from the training set – see next figure for an example.

SVM – Linear



- Source Wikipedia

SVM – Linear

- Consider X^+ and X^- the nearest positive and negative points for the hyperplane

$$< w \cdot X > + b = 0$$

- Then there are two other parallel hyperplanes, H_+ and H_- passing through X^+ and X^- and their expression is:

$$H_+ : < w \cdot X > + b = 1$$

$$H_- : < w \cdot X > + b = -1$$

- Note that **w** and **b** must be scaled such as:

$$< w \cdot X > + b = 1 \; for \; y_i = 1$$

$$< w \cdot X > + b = -1 \; for \; y_i = -1$$

SVM – Linear

- The margin is the distance between these two planes and may be computed using vector space algebra obtaining:

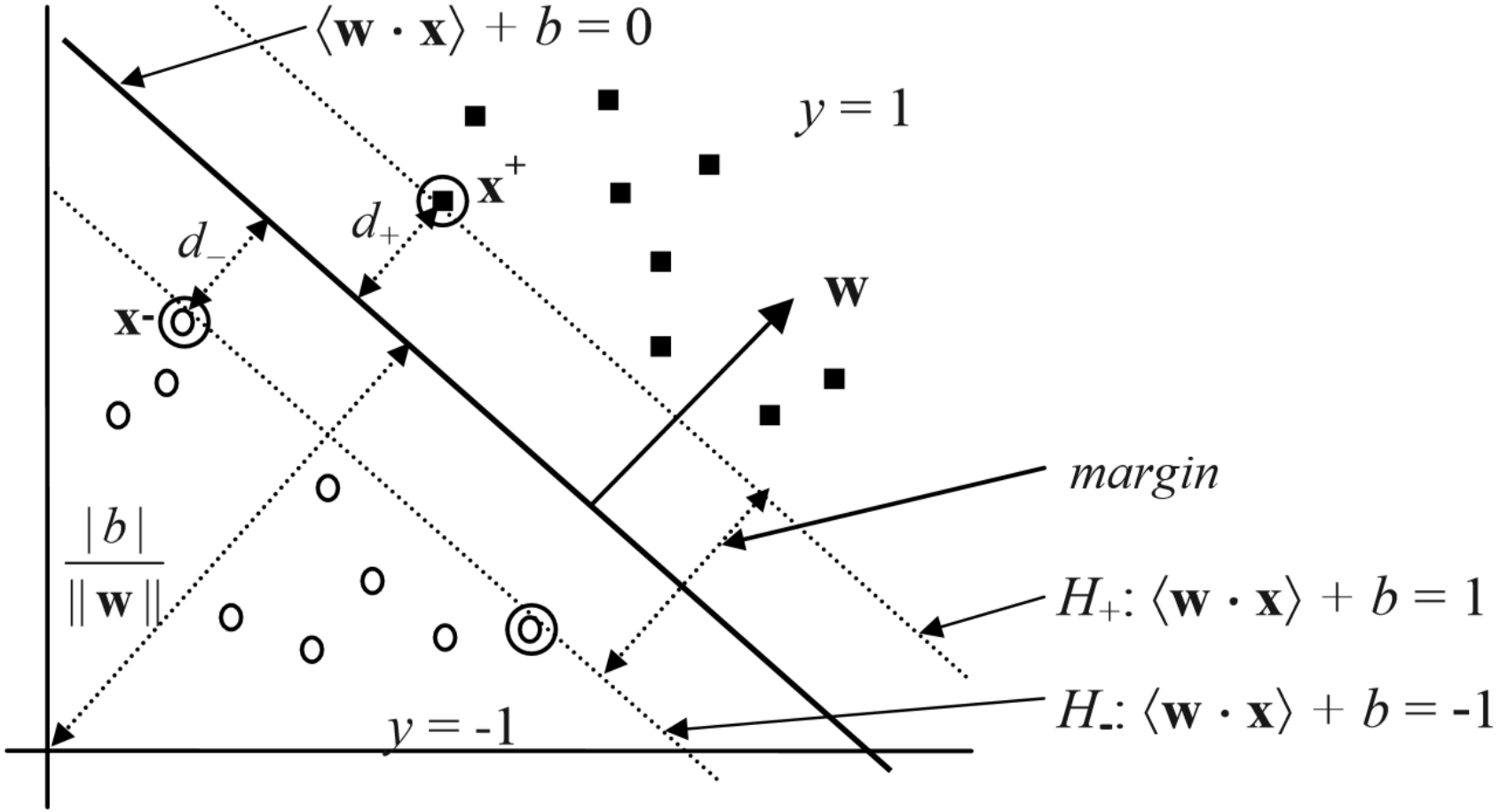
$$Margin = \frac{2}{||w||}$$

- Maximizing the margin means minimizing the value of

$$\frac{< w \cdot w >}{2} = \frac{\left||w|\right|^2}{2}$$

- The points X^+ and X^- are called ***support vectors*** and are the only important points from the dataset

SVM – Linear



SVM – Linear

- When positive and negative points are linearly separable, the SVM definition is the following:
  - Having a training data set $D = \{(x_1, y_{1j}), (x_2, y_{2j}), \dots, (x_n, y_{nj})\}$
  - Minimize the value of $\frac{<w \cdot w>}{2}$
  - With restriction: $y_i\ (=< w \cdot x_i > +b) \geq 1$, knowing the value of $y_i = \{-1, 1\}$
- This optimization problem is solvable by rewriting the above inequality using a Lagrangian formulation and then finding solution using Karush-Kuhn-Tucker (KKT) conditions.
- This mathematical approach is beyond the scope of this course.

SVM – Non-Linear

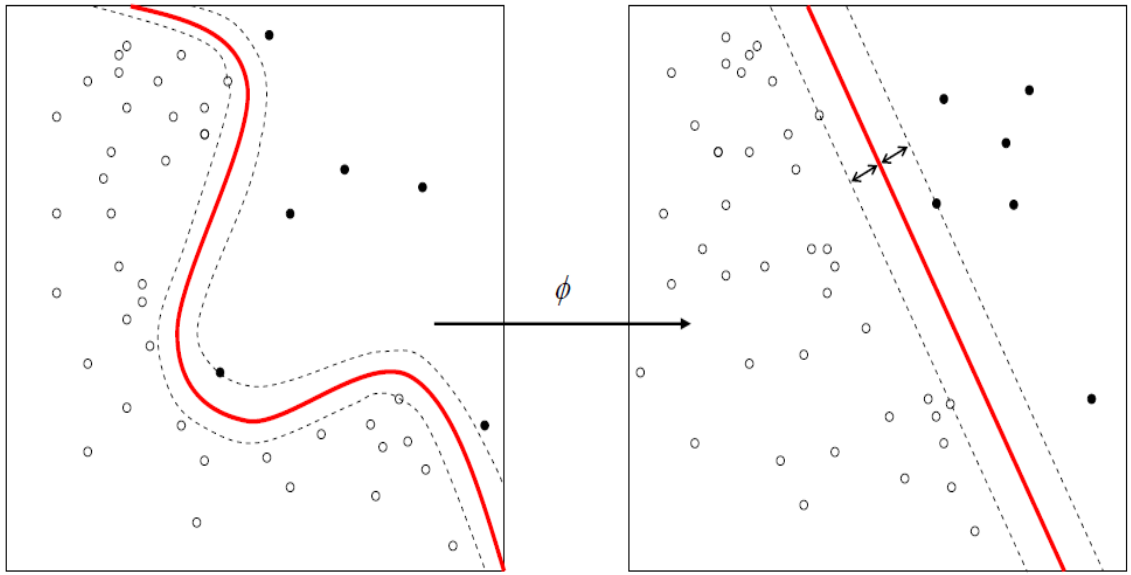
- In many situations there is no hyperplane for separation between the positive and negative examples.
- In such cases there is possible to map the training data points (examples) in another space, a higher dimensional one.
- Here data points may be linearly separable.
- The mapping function gets examples (vectors) from the input space X and maps them in the so-called ***feature space*** F:

$$\phi: X \rightarrow F$$

SVN – Non-Linear

- Each point x is mapped in $\phi(x)$.
- After mapping the whole dataset D, there is another training set, containing vectors from F and not from X, with $\dim(F) \geq n = \dim(X)$:
  $D = \{(\phi(x_1), y_{1j}), (\phi(x_2), y_{2j}), \dots, (\phi(x_n), y_{nj})\}$
- For an appropriate ϕ, these points are linearly separable.

SVM – Non-Linear



- Source Wikipedia

SVM – Non-Linear

- But how can we find this mapping function?
- In solving the optimization problem for finding the linear separation hyperplane in the new feature space F, all terms containing training examples are only of the form $\phi(x_i) \cdot \phi(x_j)$
- By replacing this dot product with a function in both x_i and x_j the need for finding ϕ disappears.
- Such a function is called a ***kernel function***:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

- For finding the separation hyperplane in F we must only replace all dot products with the chosen kernel function and then proceed with the optimization problem like in separable case.

SVM – Non-Linear

- Some of the most used kernel functions are:
  - Linear kernel:

$$K(v,u) = < u \cdot v > + b$$

  - Polynomial Kernel:

$$K(v,u) = (a \cdot < u \cdot v > + b)^p$$

  - Sigmoid Kernel (tanh - Hyperbolic tangent):

$$K(v,u) = \tanh(a \cdot < u \cdot v > + b)$$

SVM

- SVM deals with continuous real values for attributes.
  - When categorical attributes exists in the training data a conversion to real values is needed.
- When more than two classes are needed SVM can be used recursively.
  - First use separates one class; the second use separates the second class and so on. For N classes N-1 runs are needed.
- SVM are a very good method in hyper dimensional data classification.

Overview

- Naïve Bayes
- Support Vector Machines
- K-nearest neighbor
- Ensemble methods

K-Nearest Neighbor

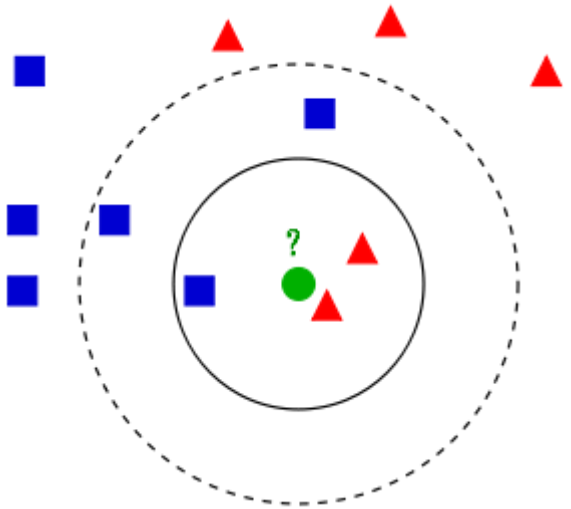
- K-Nearest Neighbor (kNN) ***does not produce a model*** but ***is a simple method*** for determining the class of an example based on the labels of its neighbors belonging to the training set.
- For running the algorithm a ***distance function*** is needed for computing the distance from the test example to the examples in the training set.
- A function $f(x, y)$ may be used as distance function if four conditions are met:
  - $f(x, y) \geq 0$
  - $f(x, x) = 0$
  - $f(x, y) = f(y, x)$
  - $f(x, y) \leq f(x, z) + f(z, y)$

K-Nearest Neighbor

- Input:
  - A labeled dataset $D = \{(x_1, c_{1i}), (x_2, c_{2i}), \dots, (x_n, c_{ni})\}$ (training set) with n observations.
  - A distance function f for measuring the dissimilarity between two examples
  - An integer k telling how many neighbors are considered
  - A dataset for testing the algorithm $T = \{t_1, t_2, \dots, t_m\}$
- Output
  - The classes for the observations in T
- Method
  - Use f to compute the distance between each point in T and each point in D
  - Select nearest k points
  - Assign to each t_i the class from the set of k nearest neighbor

K-Nearest Neighbor

- Example: get the class of the green point (Red or Blue)
- $k = 3$ then its class is Red
- $k = 5$ then its class is Blue



- kNN is very sensitive to the value of parameter k
- The best k can be determined using cross validation

K-Nearest Neighbor

- Distance functions for kNN for continuous variables:
- Euclidian distance

$$f(x,y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

- Manhattan distance

$$f(x,y) = \sum_{i=1}^{k} |x_i - y_i|$$

- Minkowski

$$f(x,y) = \left(\sum_{i=1}^{k} (|x_i - y_i|)^q \right)^{\frac{1}{q}}$$

Overview

- Naïve Bayes
- Support Vector Machines
- K-nearest neighbor
- Ensemble methods

Ensemble Methods

- Ensemble methods combine multiple classifiers to obtain a better one
- Combined classifiers are similar (use the same learning method) but the training datasets or the weights' examples are different
- Bagging (**B**ootstrap **Agg**regat**ing)**
- Boosting

Ensemble Methods
- Bootstrap -

- Bootstrap uses statistical methods (e.g. mean and standard deviation) for estimating a quantity from a data sample
- Given a sample of n values $x = \{x_1, x_2, \dots, x_n\}$ it's mean is: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- If n is small then the sample mean has an error
- The estimate of the mean can be improved by using bootstrap

Ensemble Methods
- Bootstrap -

- Bootstrap steps:
  1. Create multiple random sub-samples of the initial sample with replacement (meaning the same value can be selected multiple times): $x^1, x^2, \dots, x^m$
  2. Calculate the mean for each subsample: $\overline{x^1}, \overline{x^2}, \dots, \overline{x^m}$
  3. Calculate the average of all the sub-samples means and use it as the estimated mean for the initial sample: $\frac{1}{m}\sum_{j=1}^{m} \overline{x^j}$

Ensemble Methods
- Bagging -

- Bagging ([Breiman 1996]) is a machine learning ensemble algorithm designed to improve the stability and accuracy of machine learning.
- Bagging reduces the variance and helps to avoid overfitting.
- Bagging consists in getting a training set from the initial labeled data set by sampling with replacement

Ensemble Methods
- Bagging -

- Bagging consists in:
  - Starting with the original dataset, build $\boldsymbol{n}$ training datasets by sampling with replacement (bootstrap samples)
  - For each training dataset build a classifier using the same learning algorithm.
  - The final classifier is obtained by combining the results of each classifiers (by voting for example).
  - Bagging helps to improve the accuracy for unstable learning algorithms: decision trees, neural networks.
- It does not help for kNN, Naïve Bayesian classification or CARs.

Ensemble Methods
- Bagging -

- Bagging is an application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees
- Bagging steps:
  1. Create many random sub-samples of your dataset with replacement
  2. Train the classifier on each sample
  3. Given a new dataset, calculate the average prediction for each model
- For example: we have 5 bagged decision trees (CART) that made the following class predictions for a in input sample: blue, blue, red, blue and red, we would take the most frequent class and predict blue.

Ensemble Methods
- Random Forest -

- Random Forest [Ho 1995, Ho 1998, Breiman 2001] is an ensemble classifier consisting in a set of decision trees.
- The final classifier output the modal value of the classes output by each tree.
- Random Forests are an improvement over bagged decision trees.

Ensemble Methods
- Random Forest -

- A problem with decision trees like CART is that they are greedy.
- They choose which variable to split on using a greedy algorithm that minimizes error.
- As such, even with Bagging, the decision trees can have a lot of structural similarities and in turn have high correlation in their predictions.

Ensemble Methods
- Random Forest -

- Combining predictions from multiple models in ensembles works better if the predictions from the sub-models are uncorrelated or at best weakly correlated.
- Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all of the subtrees have less correlation.

Ensemble Methods
- Random Forest -

- It is a simple tweak. In CART, when selecting a split point, the learning algorithm is allowed to look through all variables and all variable values in order to select the most optimal split-point.
- The Random Forest algorithm changes this procedure so that the learning algorithm is limited to a random sample of features of which to search.

Ensemble Methods
- Random Forest -

- The number of features that can be searched at each split point (m) must be specified as a parameter to the algorithm.
- You can try different values and tune it using cross validation.
- The values determined experimentally are:
  - For classification a good default is: $m = \sqrt{p}$
  - For regression a good default is: $m = \frac{p}{3}$
  - Where:
    - m is the number of randomly selected features that can be searched at a split point
    - p is the number of input variables

Ensemble Methods
- Random Forest -

- Random Forest algorithm:
  1. Choose T - number of trees to grow.
  2. Choose m - number of variables used to split each node. m << M, where M is the number of input variables.
  3. Grow T trees. When growing each tree do the following:
     I. Construct a bootstrap sample from training data with replacement and grow a tree from this bootstrap sample.
     II. When growing a tree at each node select **m** variables at random and use them to find the best split.
     III. Grow the tree to a maximal extent. There is no pruning.
  4. Predict new data by aggregating the predictions of the trees (e.g. majority votes for classification, average for regression).

Ensemble Methods
- Extremely Randomized Trees -

- Extremely Randomized Trees add an other step of randomization.
- They are trained using bagging like random forest but the top-down splitting for each tree is randomized.
- This means that instead of computing the best attribute for the split using a function (e.g. information gain) a random value is selected for the split.
- This value is selected from the feature's empirical range

Ensemble Methods
- Boosting -

- Boosting consists in building a sequence of weak classifiers and adding them in the structure of the final strong classifier.
- Weak learner (classifier) – a classification algorithm with a substantial error rate which performance is not random.
- In other words, a weak leaner has an accuracy only slightly better than using random guessing

Ensemble Methods
- Boosting -

- The weak classifiers are weighted based on the weak learners' accuracy.
- Also data is reweighted after each weak classifier is built such as examples that are incorrectly classified gain some extra weight.
- The result is that the next weak classifiers in the sequence focus more on the examples that previous weak classifiers missed

Ensemble Methods
- AdaBoost -

- AdaBoost [Freund 1997] (Adaptive Boosting) uses weak learners output and combine it into a weighted sum that represents the final output of the classifier
- Construct a strong classifier as a linear combination of week classifiers

Ensemble Methods
- AdaBoost -

- Advantages:
  - It helps you choose the training set for each new classifier that you train based on the results of the previous classifier.
  - It determines how much weight should be given to each classifier's proposed answer when combining the results.

Ensemble Methods
- AdaBoost -

- AdaBoost assigns a “weight” to each training example, which determines the probability that each example should appear in the training set [Akerkar 2016] .
- Examples with higher weights are more likely to be included in the training set, and vice versa [Akerkar 2016] .

Ensemble Methods
- AdaBoost -

- After training a classifier, AdaBoost increases the weight on the misclassified examples so that these examples will make up a larger part of the next classifiers training set, and hopefully the next classifier trained will perform better on them [Akerkar 2016].

Ensemble Methods - AdaBoost -

- AdaBoost algorithm:

1. Input:
   - A training dataset $D = \{(x_1, y_{1i}), (x_2, y_{2i}), \dots, (x_n, y_{ni})\}$ (training set) with n observations
     - $x_i = \{a_{i1}, a_{i2}, \dots, a_{im}\}$ is a an observation with m attributes $A = \{a_1, a_2, \dots, a_m\}$
     - $Y = \{y_1, y_2, \dots, y_k\}$ is a set of k classes, and y_j is a class label
   - The maximum number of iterations T
   - A classifier C
2. Initialization:
   - Initialize the weight distribution $\delta^1(w_i) = \frac{1}{n}$ for $i = \overline{1, n}$
   - Now, the data set is $D_1 = \{(x_1, y_{1i}, w_1), (x_2, y_{2i}, w_2), \dots, (x_n, y_{ni}, w_n)\}$
   - The $\sum_{i=1}^{n} w_i = 1$

Ensemble Methods
- AdaBoost -

- AdaBoost algorithm [Freund 1996, Liu]:

3. For $t \in \{1, 2, \dots, T\}$ do
   - Train classifier C_t ($h_t: R^m \rightarrow Y$) using the weight distribution $\delta^t(w_i)$
   - Get the training error ϵ_t for classifier $C_t (or\ h_t)$ measured by using $\delta^t(w_i)$
   - Compute $\epsilon_t = \sum_{i=1}^{n} \delta^t(w_i)$ only for $h_t(x_i) \neq y_i$
   - If $\epsilon_t > \frac{1}{2}$ then $T = t - 1$ and exit loop
   - Else
     - Compute the output weight $\beta_t = \frac{1-\epsilon_t}{\epsilon_t}$
     - Update the weight distribution $\delta^{t+1}(w_i) = \delta^t(w_i) \cdot \begin{cases} \beta_t \ if\ h_t(x_i) = y_i \\ 1\ otherwise \end{cases}$
     - Normalization the weights $\delta^{t+1}(w_i) = \frac{\delta^{t+1}(w_i)}{\sum_{i=1}^{n} \delta^{t+1}(w_i)}$

4. Output:
   - the strong classifier is $H(x) = argmax_{y_i \in Y} \sum_{t=1}^{T} \log \frac{1}{\beta_t}$ for $h_t(x) = y_i$

Summary

- This course presented:
  - Naïve Bayes classifier: Bayes theorem, Naïve Bayes algorithm for building classifiers, Gaussian Naïve Bayes.
  - An introduction to support vector machines (SVMs): model, definition, kernel functions.
  - K-nearest neighbor
  - Ensemble methods
    - Bootstrap
    - Bagging
    - Random Forest
    - Extremely Randomized Trees
    - Boosting
    - AdaBoost

References

- [Liu 2011] Bing Liu, 2011. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data, Second Edition, Springer, chapter 3.
- [Han 2006] Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Second Edition, Morgan Kaufmann Publishers, 2006
- [Cortes 1995] Cortes, Corinna; and Vapnik, Vladimir N.; "Support-Vector Networks", Machine Learning, 20, 1995.
- [Akerkar 2016] Rajendra Akerkar, Priti Srinivas Sajja, Intelligent Techniques for Data Science, Springer International Publishing, 2016

References

- [Liu 2007] Bing Liu. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.
- [Breiman 1996] Leo Breiman. *Bagging predictors*. *Machine learning,* 24(2), 123-140, 1996.
- [Breiman 2001] Leo Breiman. *Random Forests*, Machine learning, 45(1), 5-32, 2001
- [Freund 1996] Yoav Freund, Robert E. Schapire. *Experiments with a new boosting algorithm.* International Conference on Machine Learning, 148-156, 1996.

References

- [Freund 1997]Yoav Freund, Robert E. Schapire. *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences. 55(119), 23-37, 1997
- [Geurts 2006] Pierre Geurts, Damien Ernst, Louis Wehenkel. *Extremely randomized trees,* Machine Learning, 63(1), 2006
- [Ho 1995] Tin Kam Ho. *Random Decision Forests*, International Conference on Document Analysis and Recognition, 278-282, 1995
- [Ho 1998] Tin Kam Ho. *The random subspaces method for constructing decision forests,* IEEE transactions on pattern analysis and machine intelligence, 20(8), 832-844, 1998