VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Machine Learning

Classification

Jan Platoš

November 22, 2023

Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB - Technical University of Ostrava

Classification

Classification

Basic questions:

- What it is?
- What it needs?
- What it produces?

Classification

Basic questions:

- What it is?
- What it needs?
- What it produces?

Definition
Given a set of training data points, each of which is associated with a class label, determine the class label of one or more previously unseen test instances.

Classification

Phases of classification:

- Training phase - construction of models from the training instances.
- Testing phase - determining class labels of one or more training instances.

Output of classification:

- Label prediction - one fixed label is predicted.
- Numerical score - numerical evaluation of each label assignment to the instance.

Feature Selection

Feature Selection

- Selection of the attributes subset for classification.
- Three types of models:
    1. Filter models – crisp mathematical criterion is used to evaluate each subset of attributes.
    2. Wrapper models – the model is run on each candidate subset to evaluate its efficiency.
    3. Embedded models – The model information is used to prune irrelevant attributes.

Feature Selection - Filter models

Gini index:

- Measures the discriminative power of a particular attributes subset.
- Usually used to categorical data/discretized numerical data.

Feature value index:

$$G(v_i) = 1 - \sum_{j=1}^{k} p_j^2$$

- $v_1, v_2, \ldots, v_r$ are r values of a particular attribute.
- p_j is the fraction of points that contains attribute v_i that belong to the class j for k possible classes.

Feature index:

$$G = \frac{1}{n} \sum_{i=1}^{r} n_i G(v_i)$$

Feature Selection - Filter models

Entropy:

- Measures the information gain from fixing a specific attribute value.

Feature value entropy:

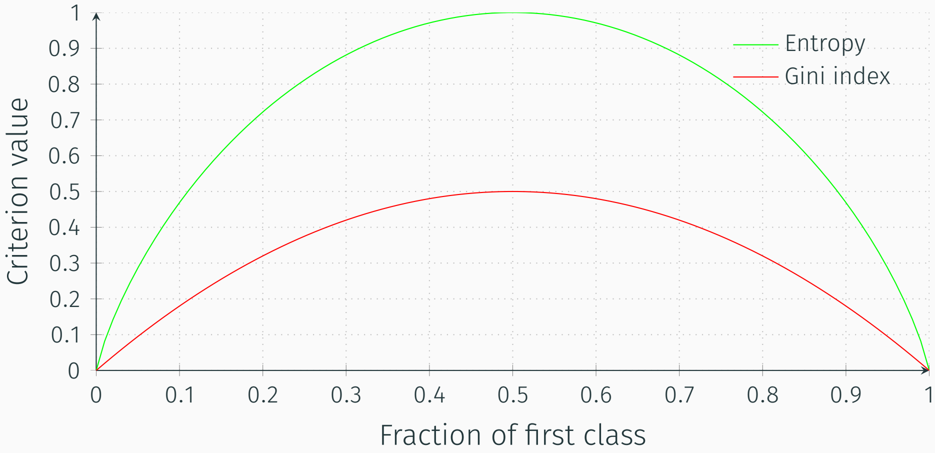
$$E(v_i) = -\sum_{j=1}^{k} p_j \log(p_j)$$

- $v_1, v_2, \ldots, v_r$ are r values of a particular attribute.
- p_j is the fraction of points that contains attribute v_i that belong to the class j for k possible classes.

Feature entropy:

$$E = \frac{1}{n}\sum_{i=1}^{r} n_i E(v_i)$$

Feature Selection - Filter models



Feature Selection - Filter models

Fisher Score:

- Naturally designed for numeric attributes.
- Measures the the ratio of the average interclass separation to the average intraclass separation.

$$F = \frac{\sum_{j=1}^{k} p_j(\mu_j - \mu)^2}{\sum_{j=1}^{k} p_j \sigma_j^2}$$

- p_j is the fraction of data points belonging to class j.
- μ_i, σ_j is the mean and standard deviation of data points belonging to class j for a particular feature.
- μ is the global mean of the data points on the feature being evaluated.

Feature Selection - Wrapper models

- Different classification models are more accurate with *different* sets of features.
- Filter models are agnostic to the particular classification algorithm being used.
- The characteristics of the specific classification algorithm is used to select features.
    - Linear classifier work more effectively with a set of features where the classes are best modeled with linear separators.
    - Distance based classifier works well with features in which distances reflect class distributions.
- A specific classification algorithm is used as an input to the feature selection.
- Wrapper models then optimize the feature selection process to the classification algorithm.
- The basic strategy in wrapper models is to iteratively refine a current set of features F by successively adding features to it.

Feature Selection - Wrapper models

- The algorithm starts with empty feature set $F = \emptyset$.
- The strategy may be summarized as follows:
    - Create an augmented set of features F by adding one or more features to the current feature set.
    - Use a classification algorithm A to evaluate the accuracy of the current set of features F.
    - Use the accuracy to either accept or reject the augmentation of F.
- The augmentation of F can be performed in many different ways.
    - Greedy strategy - the set of features in the previous iteration is augmented with an additional feature with the greatest discriminative power with respect to a filter criterion).
    - Random sampling - features may be selected for addition via random sampling.

Feature Selection - Wrapper models

- The accuracy of the classification algorithm A is used to determine the acceptance/rejection of the features.
- The rejected features are removed from the set and another augmentation is tested.
- This approach is continued until there is no improvement in the current feature set for a defined minimum number of iterations.
- The final set of featured is sensitive to the choice of the algorithm A.

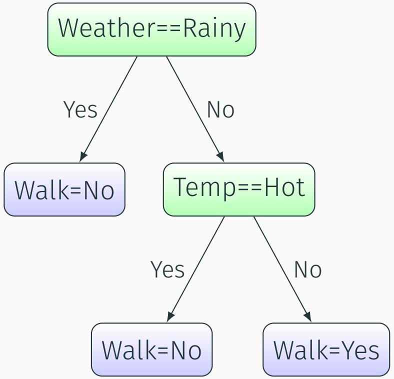
Decision trees

Decision Trees

- Classification is modeled using hierarchical decisions on the features that are arranged in tree-like structure.
- The decision at a particular node, called split criterion, is a relational condition on one or more features and their values.
- The goal is to identify a split criterion that minimizes the mixing of classes in each branch.
- Works on binary, numeric and categorical attributes.
- Each sub-space (region) is recursively split until terminal conditions are reached.
- Univariate or Multivariate split is possible.

Decision Trees

Weather	Temp	Walk?
Sunny	Cold	Yes
Sunny	Warm	Yes
Sunny	Hot	No
Cloudy	Cold	Yes
Cloudy	Warm	Yes
Cloudy	Hot	No
Rainy	Cold	No
Rainy	Warm	No
Rainy	Hot	No



Decision Trees

Split Criteria:

- The goal is to maximize separation of the different classes among the children nodes.
- Binary attribute – only one type of split is possible.
- Categorical attribute with r values
  - r-way split,
  - binary split on $2^r - 1$ possibilities (all combinations except $\emptyset$),
  - binary split on r possibilities (one-to-rest strategy).
- Numeric attribute
  - A split is made between two values with < or <= relation.
  - All values or selected values only may be tested.

Decision Trees - Split Criteria

Definitions:

- S is a set of points in a branch of a tree.
- $|S|$ is size of the set (number of points in a set).
- r-way split has r subsets $S_1, \ldots, S_r$ of set S.
- k is the number of classes.

Decision Trees - Split Criteria

Error rate:

- On a set:

$$Err\left(S\right) = 1 - p$$

    - where the p is a fraction of points that belongs to the dominant class from S.

- On r-way split:

$$Err\left(S \Rightarrow S_1, \ldots, S_r\right) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} \left(1 - p\right)$$

Decision Trees - Split Criteria

Gini index:

- On a set:

$$G(S) = 1 - \sum_{j=1}^{k} p_j^2$$

  - where the p_j is a fraction of points that belongs to the class j from S.
- On r-way split:

$$G(S \Rightarrow S_1, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} G(S_i)$$

Decision Trees - Split Criteria

Entropy:

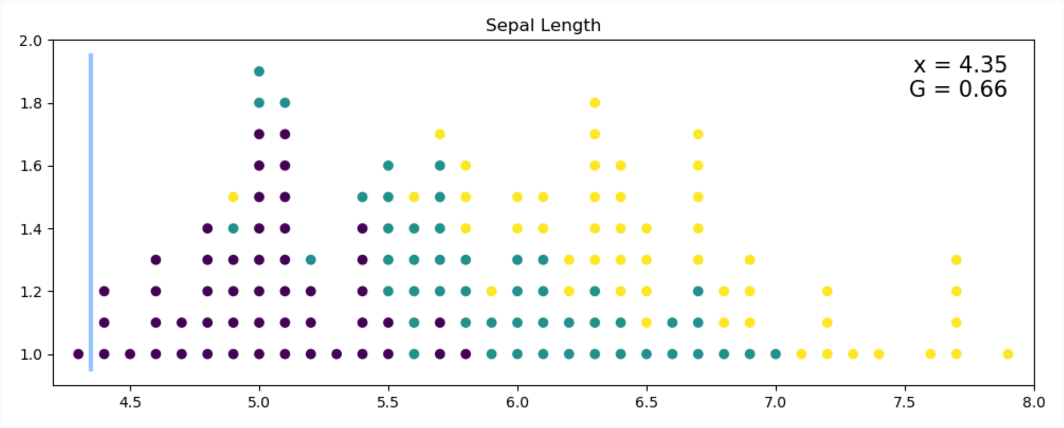
- On a set:

$$E(S) = -\sum_{j=1}^{k} p_j \log_2 (p_j)$$

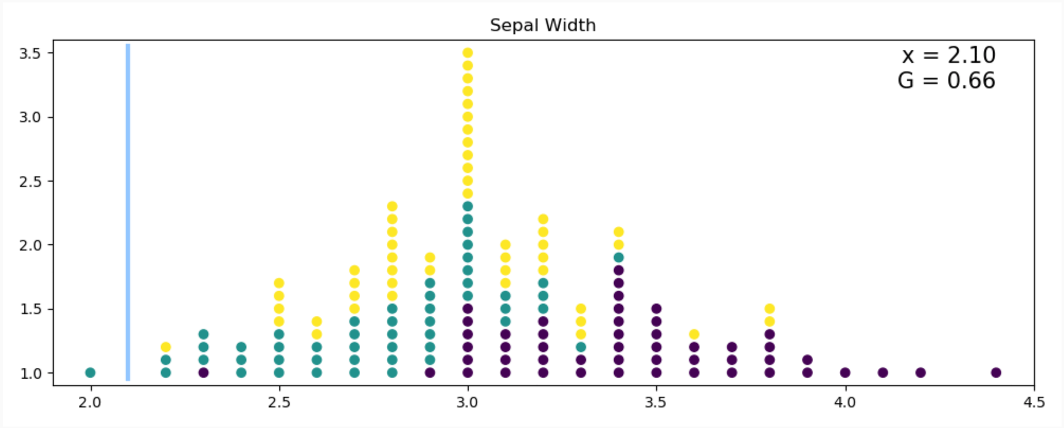
  - where the p_j is a fraction of points that belongs to the class j from S.
- On r-way split:

$$E(S \Rightarrow S_1, \ldots, S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} E(S_i)$$

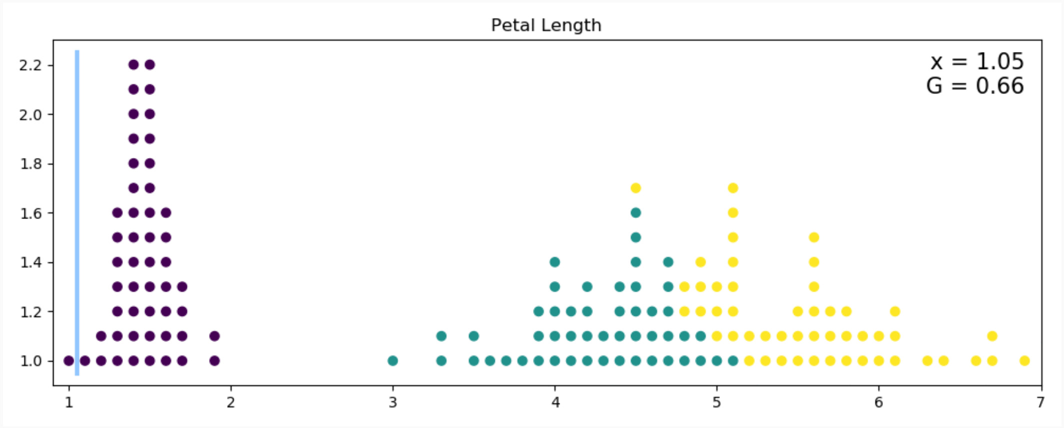
Decision Trees - Split Criteria - IRIS Example



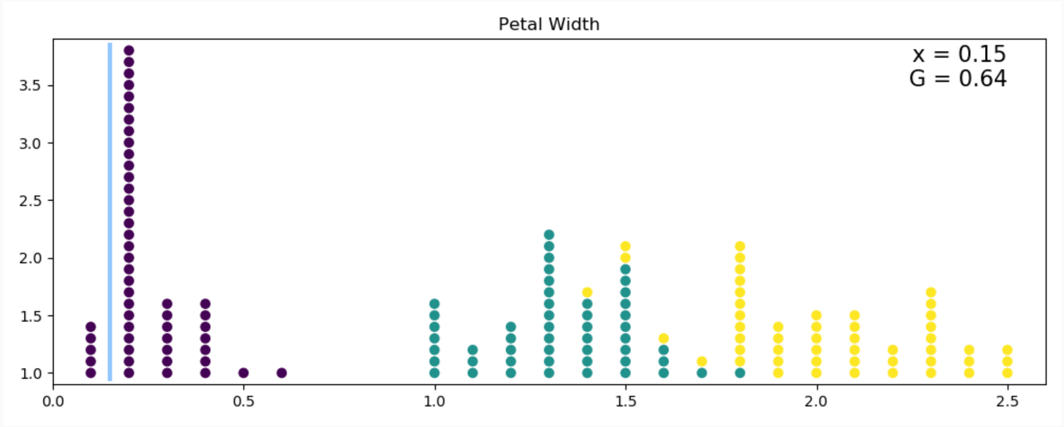
Decision Trees - Split Criteria - IRIS Example



Decision Trees - Split Criteria - IRIS Example



Decision Trees - Split Criteria - IRIS Example



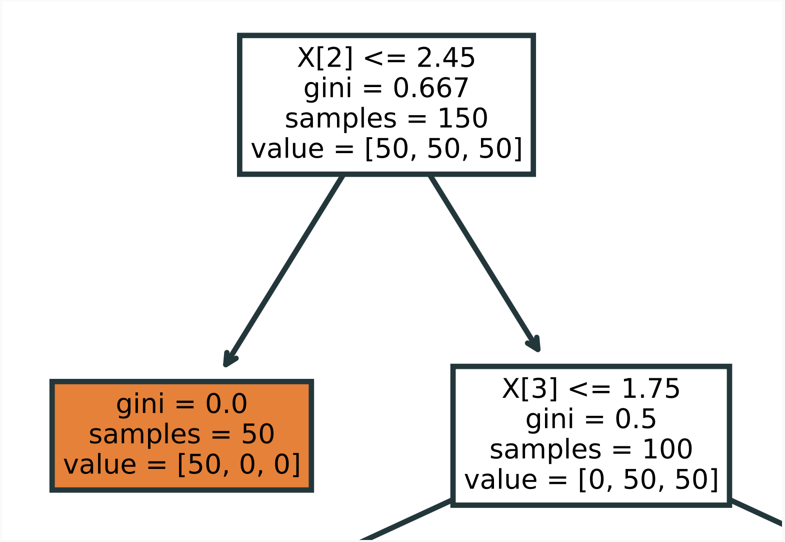
Decision Trees - Split Criteria - IRIS Example

- index = 0, Sepal Length $<$ 5.45, Gini = 0.44
- index = 1, Sepal Width $<$ 3.35, Gini = 0.54
- index = 2, Petal Length $<$ 2.45, Gini = 0.33
- index = 3, Petal Width $<$ 0.80, Gini = 0.33

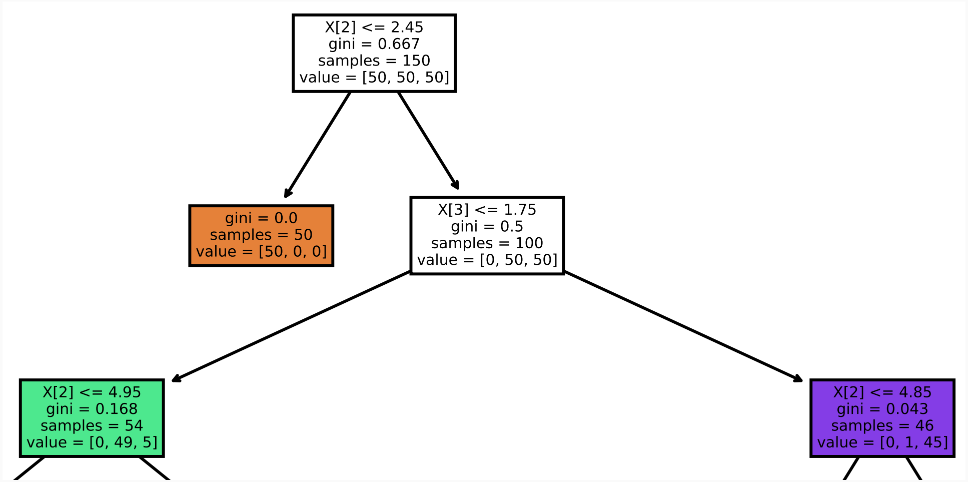
Decision Trees - Split Criteria - IRIS Example

- index = 0, Sepal Length $<$ 5.45, Gini = 0.44
- index = 1, Sepal Width $<$ 3.35, Gini = 0.54
- index = 2, Petal Length $<$ 2.45, Gini = 0.33
- index = 3, Petal Width $<$ 0.80, Gini = 0.33

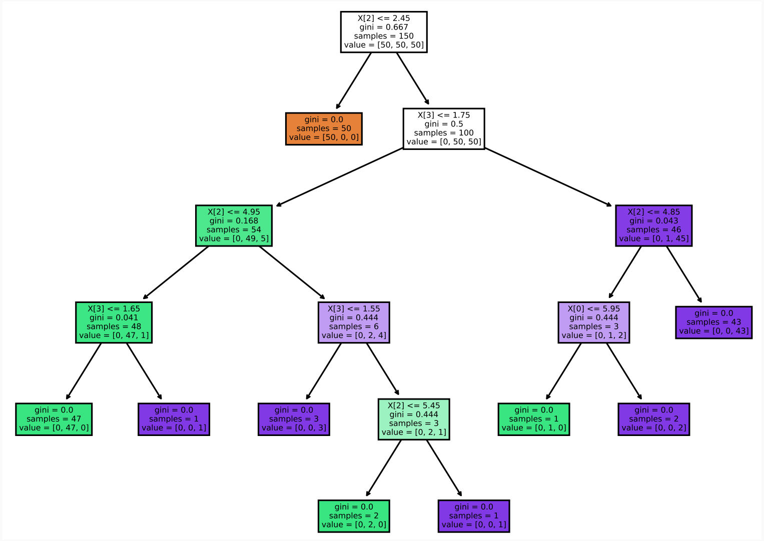
Decision Trees - Split Criteria - IRIS Example



Decision Trees - Split Criteria - IRIS Example



Decision Trees - Split Criteria - IRIS Example



Decision Trees

Stopping criterion:

- Very difficult to stop during the tree growth.
- Single class in a leaf node is the final condition.
- Such tree has 100% precision on Training data.
- But, such tree is over-fitted (unable to generalize to unseen data).
- Over-fitting is done by lower nodes with less number of points.

Decision Trees

Pruning:

- Shallow trees are more preferable is they produces the same error on training data.
- Nodes/Trees are evaluated using a criterion that penalizes the more complex tress without satisfactory improvement in precision.
- Usually a holdout set (e.g. 20% of training set) is used for pruning.
- A node is prunes is its removing improves the precision on the holdout.
- A leaf node are pruned iteratively until no node should be removed.

Rule-based classification

Rule-based classification

- A generalization of the Decision Trees.
- A set of rules in a form:

IF Condition THEN Conclusion

- *Condition* or *Antecedent* is a combination of relational, set and logical operators over features.
- *Conclusion* or *Consequent* is a class label.
- A rule cover the training instance is the condition match the instance.

Rule-based classification

Rule types:

- Mutually exclusive rules
    - Each rule covers disjoin set of instances.
    - Each instance trigger at most one rule.
- Exhaustive rules
    - The entire data space is covered by at least one rule.
    - Simple exhaustive rule assign dominant class do anything (catch-all).
- Non mutually exclusive rules brings problems with rule evaluation.

Rule-based classification

Rule ordering:

- Ordered rules
  - Rules are ordered by priority, such as quality measure.
  - Rules may be ordered by class-based principle.
  - Only the first triggered rule vote, its consequent is the result.
  - The rare classes are usually ordered first.
- Unordered rules
  - There is no priority on rules.
  - The dominant class of the all triggered rules is selected.
  - Simplifies the learning phase.

Rule-based classification

Rule generation:

- The goal is to generate rules that covers the instances from the training data.
- Two major algorithm exists:
  - Generation using Decision Trees.
  - Sequential Covering Algorithm.

Rule-based classification - Rule generation

Rule generation using Decision Trees:

- Trees are used for generation of the rules.
- Each leaf node represent one rule with its sequence of splits that lead to this leaf from root.
- The pruning is not made on tree, but on rules.
- Each rule is processed separately and pruned to get the most precise rule on the holdout set.
- The pruning process is more flexible because any part of the antecedent may be pruned.
- Duplicate rules are removed.
- The rules after pruning are not mutually exclusive.
- The ordering of the rules is necessary.
- Rare classes and less complex rules or rules with less false positives are prioritized

Rule-based classification - Rule generation

Sequential Covering Algorithm:

- An algorithm for creation of ordered set of rules.
- An 2-step iterative algorithm:
    - **Learn-one-rule** – select particular class and determine the “best” rule from the current training instances S with this class as a consequent. Add this rule to the bottom of the ordered rule list.
    - **Prune training data** – Remove training instances in S that are covered by the rule generated in previous step. The detection is based on the antecedent only, that consequent of the instances is ignored.

Rule-based classification - Rule generation - Sequential Covering Algorithm

The ordering of the generated rules:

- Class-based ordering
    - All rules for particular class are put together.
    - Rare classes may be prioritizes.
    - All rules for this particular class are generated continuously, until a termination criterion is met.
    - For k-class problem, $k-1$ rule sets is generated and the final catch-all rule covers the last class.
- Quality-based ordering
    - The rule are selected according a measure, such as confidence or support.
    - The catch-all rule corresponds to the dominant class among remaining instances.
    - The quality of very difficult to measure.

Rule-based classification - Rule generation - Sequential Covering Algorithm

Learn-one-rule step:

- Iterative algorithm that grows a rule with best conjunct according the quality measure.
- The simplest quality is the precision/accuracy.
- Each split choice (conjunct) is evaluated the same was as it is in trees.
- Several best options may be maintained to reduce the possibility of the mistakes and suboptimal rules.
- The ideal quality measure must combine accuracy and coverage, e.g. Laplace smoothing, like-hood ratio statistics, FOIL information gain.

Rule-based classification - Rule generation

Rule pruning:

- An Minimum description length (MDL) principle is one option.
- A penalty based on MDL may be used in rule-growth phase.
- An holdout set is another good principle.
- A greedy algorithm may be used for conjunct evaluation.

Naïve Bayes Classifier

Naïve Bayes Classifier

- Based on the Bayes theorem for conditional probabilities.

$$P(D|E) = \frac{P(E|D)P(D)}{P(E)}$$

- This theorem is useful when it is hard to estimate $P(D|E)$ but others probabilities are easy to get from input data.
- When E is a single attribute, everything is simple.
- When E is complex, $P(E|D)$ may be missing in the data or appear only few times.

Naïve Bayes Classifier

- Let C be a class variable.
- Let $\overline{X}$ is a d-dimensional instance $\overline{X} = (a_1, \ldots, a_d)$.
- Let the random d-dimensional variable is $\overline{X} = (x_1, \ldots, x_d)$.
- The goal is to estimate $P(C = c|\overline{X} = (a_1, \ldots, a_d))$ or $P(C = c|x_1 = a_1, \ldots, x_d = a_d))$ resp.

$$P(C = c|x_1 = a_1, \ldots, x_d = a_d) = \frac{P(x_1 = a_1, \ldots, x_d = a_d|C = c)P(C = c)}{P(x_1 = a_1, \ldots, x_d = a_d)}$$

- The denominator is independent of the class and may be removed.

Naïve Bayes Classifier

- The estimation of $P(x_1 = a_1, \ldots, x_d = a_d | C = c)$ is crucial and difficult.
- The Naïve approach assumes that the features are independent!!!
- Then

$$P(x_1 = a_1, \ldots, x_d = a_d | C = c) = \prod_{j=1}^{d} P(x_j = a_j | C = c)$$

$$P(x_j = a_j | C = c) = \frac{q(a_j, c) + \alpha}{r(c) + \alpha \cdot m_j}$$

Where

- $q(a_j, c)$ is a fraction of records with class c and
- $r(c)$ is a fraction of records with class c
- α is a small value
- m_j is number of distinct values of j-th attribute.

Naïve Bayes Classifier

- Finally:

$$P(C = c|x_1 = a_1, \ldots, x_d = a_d)) \approx P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c)$$

- or

$$P(C = c|x_1 = a_1, \ldots, x_d = a_d)) \approx P(C = c) \prod_{j=1}^{d} \frac{q(a_j, c) + \alpha}{r(c) + \alpha \cdot m_j}$$

Naïve Bayes Classifier - Example

Name	Age	Salary	Donor?
Nancy	21	37,000	N
Jim	27	41,000	N
Allen	43	61,000	Y
Jane	38	55,000	N
Steve	44	30,000	N
Peter	51	56,000	Y
Sayani	53	70,000	Y
Lata	56	74,000	Y
Mary	59	25,000	N
Victor	61	68,000	Y
Dale	63	51,000	Y

- Assume a rule: Age>50 AND Salary>50
- $P(Donor = Yes) = 6/11$
  - $P(Age > 50|Donor = Yes) = 5/6$
  - $P(Salary > 50,000|Donor = Yes) = 6/6$
- $P(Donor = No) = 5/11$
  - $P(Age > 50|Donor = No) = 1/5$
  - $P(Salary > 50,000|Donor = No) = 1/5$
- $P(Donor = YES) = 6/11{*}5/6{*}1 = 6/11{*}5/6 = \mathbf{5/11}$
- $P(Donor = NO) = 5/11 * 1/5 * 1/5 = 5/11 * 1/25 = \mathbf{1/55}$

Naïve Bayes Classifier - Summary

- Numeric values
  - Discretization is possible but it may affects the precision.
  - Direct data-drives estimation of the probability distribution is more suitable.
  - A proper distribution have to be selected, usually Gaussian is taken.
  - A mean and variance is extracted from the data.
- The naïve assumption
  - The independence is usually not true in real data.
  - The more complex estimation of the probability are not precise when dimension increases.
  - The naïve approach is precise enough.

Support Vector Machines

Support Vector Machines

- Naturally defined binary classification of numeric data.
- Multi-class generalization possible using several different strategies.
- Categorical features may be binarized and used.
- The class labels are assumed to be from the set $\{-1, 1\}$.
- The separation hyperplanes are used as classification criterion as with all linear models.
- The hyperplane is determined using a notion of margin.

Support Vector Machines - Linearly separable case

- A hyperplane that clearly separate points that belongs to the two classes.
- An infinite number of possible ways of constructing a linear hyperplane between classes exists.
- A maximum margin between hyperplanes have to be set, e.g. the minimum perpendicular distance to data points have to be maximum.

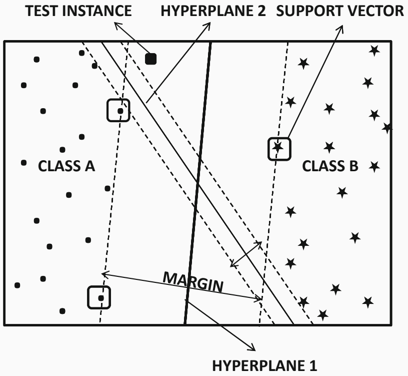


Figure 1: Linearly separable case

Support Vector Machines - Linearly separable case

- A hyperplane that cleanly separates two linearly separable classes exists.
- The margins of the hyperplanes is defined as he sum of its distances to the closest training points belonging to each of the two classes.
- The distance between the margin and the closest training points in either class is the same.
- A parallel hyperplanes may be constructed to the separating one that they touch the training points from either class and has no data points between them.
- The training points on these hyperplanes are referred to as the support vectors.
- The distance between the support vectors is the margin.
- The separating plane is precisely in the middle of these two hyperplanes in order to achieve the most accurate classification.

Support Vector Machines - Linearly separable case

Determination of the maximum margin hyperplane

- By setting up of the non-linear programming optimization formulation that maximizes the margin by expressing it as a functions of the coefficients of the hyperplane.
- The optimal coefficients can be determined by solving this optimization problem.

Support Vector Machines - Linearly separable case

Definition

- The n is the number of data points in the training set D
- The i-th data points is denoted as (X_i, y_i), where X_i is a d-dimensional row vector, and $y_i \in \{-1, +1\}$ is the binary class variable.

$$\overline{W} \cdot \overline{X} + b = 0$$

- $\overline{W} = (w_1, \ldots, w_d)$ is the d-dimensional row vector representing the direction of the normal of the hyperplane.
- b is a scalar, also know as bias.

Problem

Learning of the $(d+1)$ coefficients corresponding to the $\overline{W}$ and b from the training data that maximizes the margin.

Support Vector Machines - Linearly separable case

- The points from either class have to lie on the opposite sides of the hyperplane.

$$\overline{W} \cdot \overline{X_i} + b \geq 0 \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \leq 0 \quad \forall i : y_i = -1$$

- By introducing the margin parameter and its normalization and transformation we may get

$$y_i \left(\overline{W} \cdot \overline{X_i} + b \right) \geq +1 \quad \forall i \tag{1}$$

- The goal is to maximize the distance between two parallel hyperplanes.

$$\frac{2}{\|\overline{W}\|} = \frac{2}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

- Instead of the maximization of the above term we may minimize the following

$$\frac{\|\overline{W}\|^2}{2}$$

Support Vector Machines - Linearly separable case

- Minimization of the $\frac{\|\overline{W}\|^2}{2}$ is a complex quadratic programming problem because the parameter is minimized subject to a set of linear constraints, see eq. 1.
- Each data points leads to a constraint, therefore the SVM is computationally complex.
- One of the possible method that is able to solve such problem is a Lagrangian relaxations.
- It brings an set of non-negative multipliers $\overline{\lambda} = (\lambda_1, \ldots, \lambda_n)$ associated to each constraint.
- The constraints are then relaxed and the objective function is augmented by incorporating a Lagrangian penalty for constraints violation.

$$L_P = \frac{\|\overline{W}\|^2}{2} - \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X}_i + b \right) - 1 \right]$$

Support Vector Machines - Linearly separable case

- Conversion of the L_P into strictly pure maximization problem by eliminating the minimization part.
- The variables $\overline{W}$ and b are converted by gradient-based condition and set to zero.

$$\nabla L_P = \nabla \frac{\|\overline{W}\|^2}{2} - \nabla \sum_{i=1}^{n} \lambda_i \left[y_i \left(\overline{W} \cdot \overline{X}_i + b \right) - 1 \right] = 0$$

$$\overline{W} - \sum_{i=1}^{n} \lambda_i y_i \overline{X}_i = 0$$

- The expression of $\overline{W}$ is then derived directly

$$\overline{W} = \sum_{i=1}^{n} \lambda_i y_i \overline{X}_i$$

- The similar approach with variable b then generate

n

Support Vector Machines - Linearly separable case

- The final Lagrangian dual is as follows:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \overline{X_i} \cdot \overline{X_j}$$

- The class label for the test instance $\overline{Z}$ defined by the decision boundary may be computed as

$$F\left(\overline{Z}\right) = sign\left\{\overline{W} \cdot \overline{Z} + b\right\} = sign\left\{\left(\sum_{i=1}^{n} \lambda_i y_i \overline{X_i} \cdot \overline{Z}\right) + b\right\}$$

- The solving of the L_D is done using gradient ascent according the parameter vector $\overline{\lambda}$.

Support Vector Machines - Soft margin for Linearly Non-separable Data

- The margin is defined as soft with penalization of the margin violation constraints.
- The definition of the Lagrangian is very similar to the Lagrangian for the separable case, with induction new constraints on the hyperplanes.

$$\begin{aligned} \overline{W} \cdot \overline{X_i} + b &\geq +1 - \xi_i & \forall i : y_i = +1 \\ \overline{W} \cdot \overline{X_i} + b &\leq -1 + \xi_i & \forall i : y_i = -1 \\ & & \forall i : \xi_i \geq 0 \end{aligned}$$

- Objective function O is then defined as

$$O = \frac{\|\overline{W}\|^2}{} + C \sum^{n} \xi$$

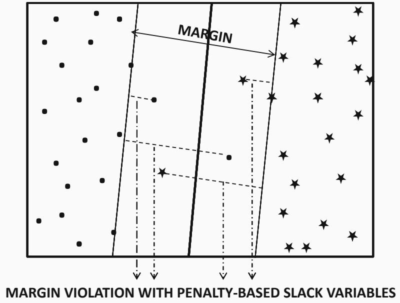
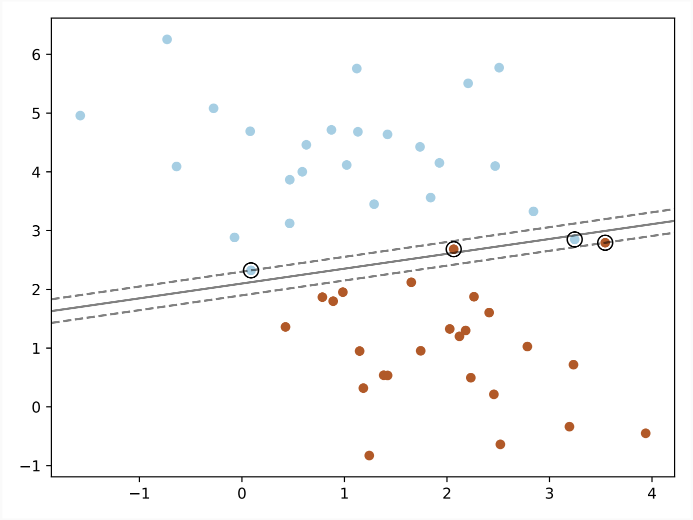


Figure 2: Soft margin for Non-separable Data

The C affects the allowed error during training (smaller C larger error)

Linear separation: Examples

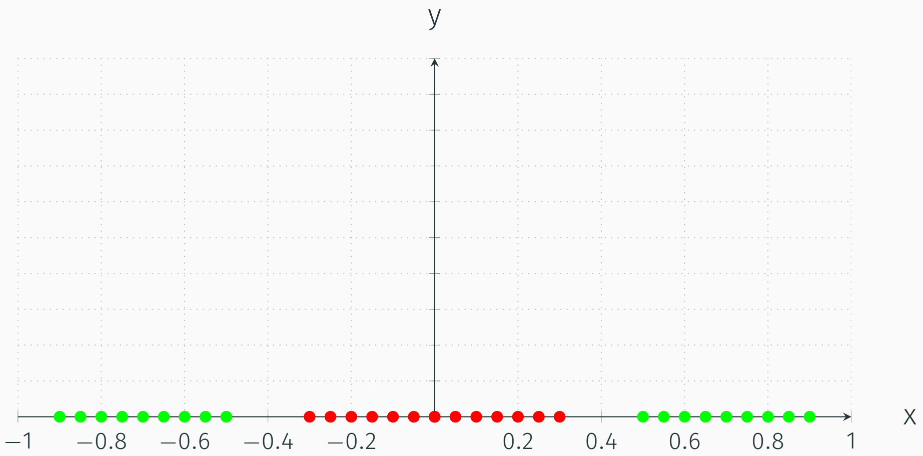


Support Vector Machines

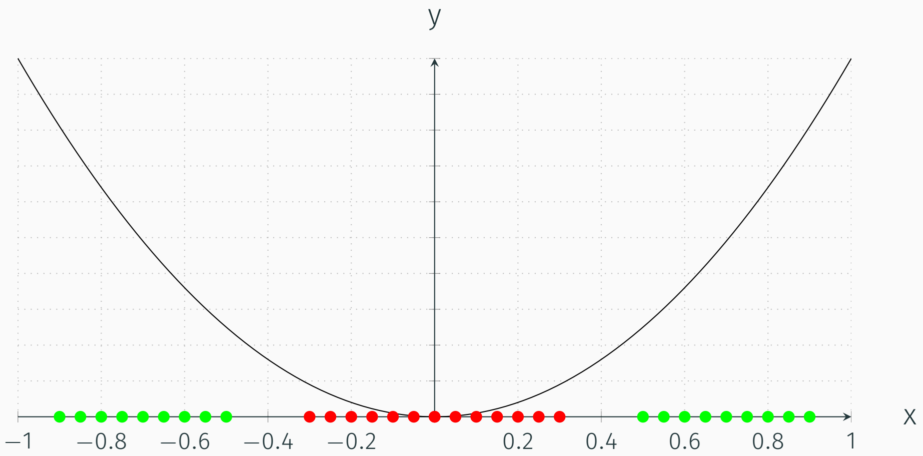
Non-linear decision boundary

- In real cases, the decision boundary is not linear.
- The points may be transformed into higher dimensions to enable linear decision boundary.

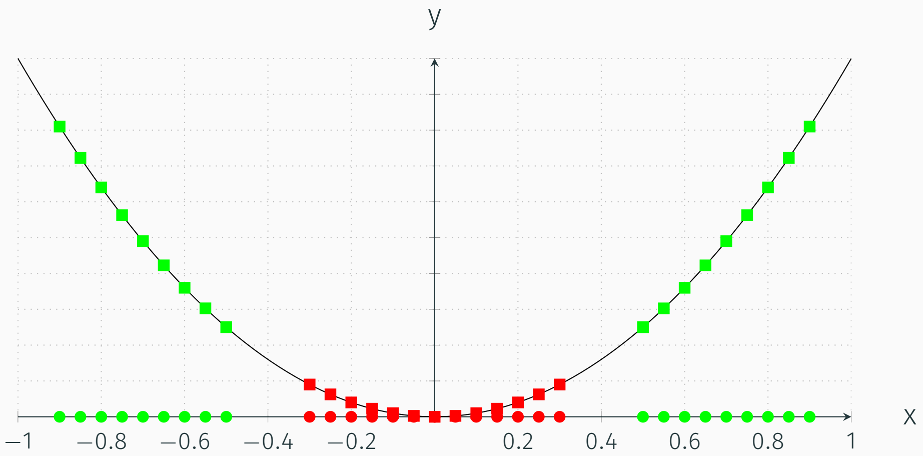
Support Vector Machines - Non-linear decision boundary



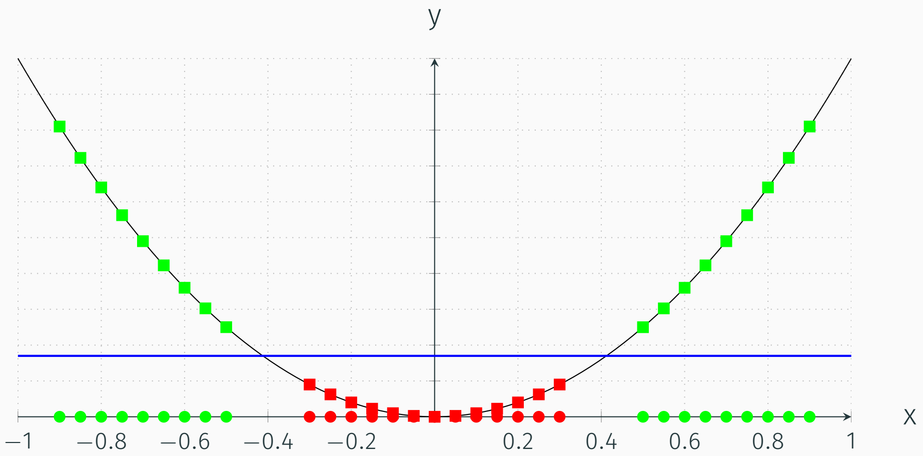
Support Vector Machines - Non-linear decision boundary



Support Vector Machines - Non-linear decision boundary



Support Vector Machines - Non-linear decision boundary



Support Vector Machines

The Kernel Trick

- It leverages the important observation that the SVM formulation can be fully solved in the terms of dot products (or similarities) between pairs of data points.
- The feature values itself are not important or needed.
- The key is to define the pairwise dot products (similarity function) directly in the d'-dimensional transformed representation $\Phi(\overline{X})$ such as:

$$K\left(\overline{X_i}, \overline{X_j}\right) = \Phi\left(\overline{X_i}\right) \cdot \Phi\left(\overline{X_j}\right)$$

- Only the dot product is required, therefore there is no need to compute transformed feature values $\Phi(X)$.

Support Vector Machines - The Kernel Trick

- The final Lagrangian dual with the substitution is defined as follows:

$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j K\left(\overline{X_i}, \overline{X_j}\right)$$

- The class label for the test instance $\overline{Z}$ defined by the decision boundary may be computed as follows:

$$F\left(\overline{Z}\right) = sign\left\{\overline{W} \cdot \overline{Z} + b\right\} = sign\left\{\left(\sum_{i=1}^{n} \lambda_i y_i K\left(\overline{X_i}, \overline{Z}\right)\right) + b\right\}$$

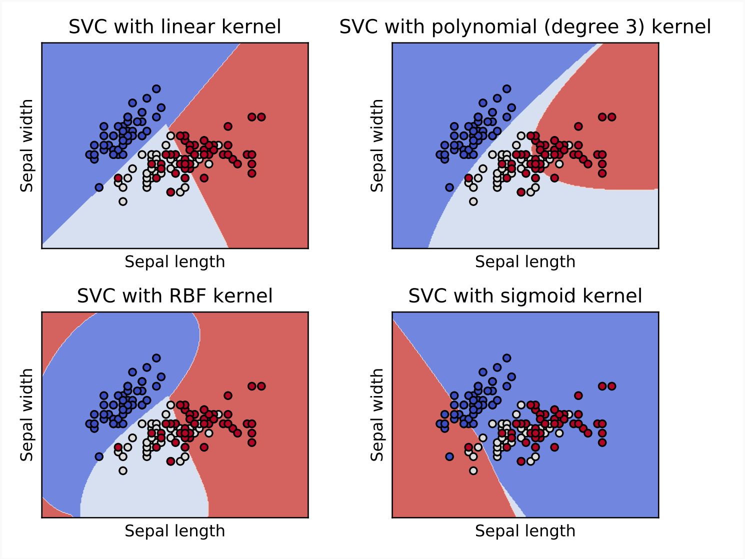
Support Vector Machines - The Kernel Trick

- All computations are performed in the original space.
- The actual transformation $\Phi(\cdot)$ does not to be known as long as the kernel similarity function $K(\cdot)$ is known.
- The kernel function have to be chosen carefully.
- The kernel function have to satisfy Mercer's theorem to be considered valid.
- This theorem ensures that the $n \times n$ kernel matrix (called Gramm matrix) $S = \left[K\left(\overline{X_i}, \overline{X_j}\right)\right]$ is symmetric, positive semidefinite.

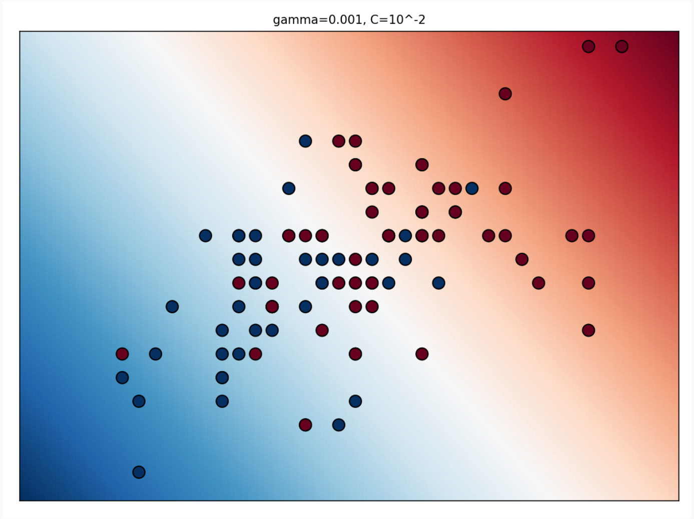
Support Vector Machines - The Kernel Trick

Function	Form
Linear kernel	$K(\overline{X_i}, \overline{X_j}) = \overline{X_i} \cdot \overline{X_j} + c$
Polynomial kernel	$K(\overline{X_i}, \overline{X_j}) = (\alpha \overline{X_i} \cdot \overline{X_j} + c)^h$
Gaussian Radial Basis Function (RBF)	$K(\overline{X_i}, \overline{X_j}) = \exp\left(-\frac{\lVert \overline{X_i} - \overline{X_j} \rVert^2}{2\sigma^2}\right)$
Sigmoid kernel	$K(\overline{X_i}, \overline{X_j}) = \tanh(\kappa \overline{X_i} \cdot \overline{X_j} - \delta)$
Exponential kernel	$K(\overline{X_i}, \overline{X_j}) = \exp\left(-\frac{\lVert \overline{X_i} - \overline{X_j} \rVert}{2\sigma^2}\right)$
Laplacian kernel	$K(\overline{X_i}, \overline{X_j}) = \exp\left(-\frac{\lVert \overline{X_i} - \overline{X_j} \rVert}{\sigma}\right)$
Rational Quadratic Kernel	$K(\overline{X_i}, \overline{X_j}) = 1 - \frac{\lVert \overline{X_i} - \overline{X_j} \rVert^2}{\lVert \overline{X_i} - \overline{X_j} \rVert^2 + c}$

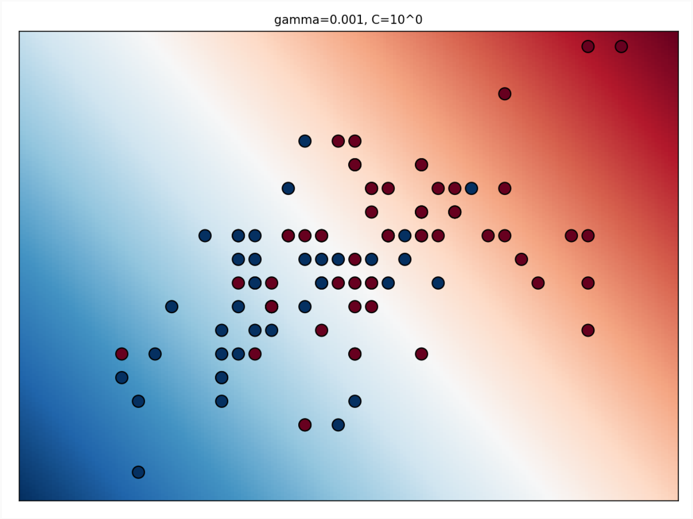
Support Vector Machines - Kernel examples



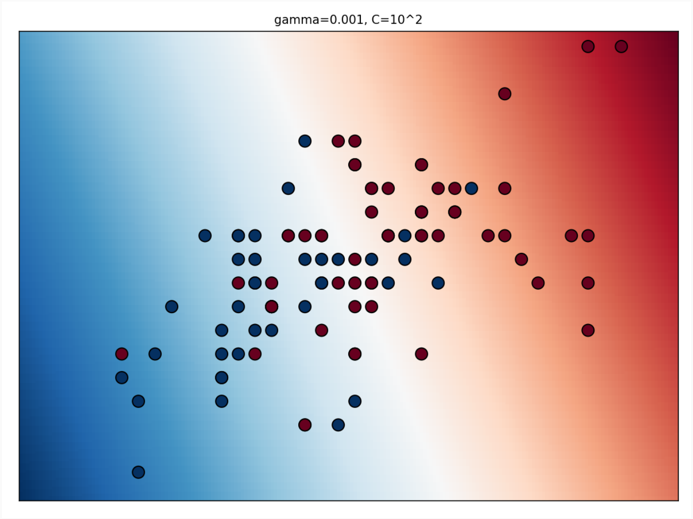
Support Vector Machines - Kernel parameters examples, C=0.01



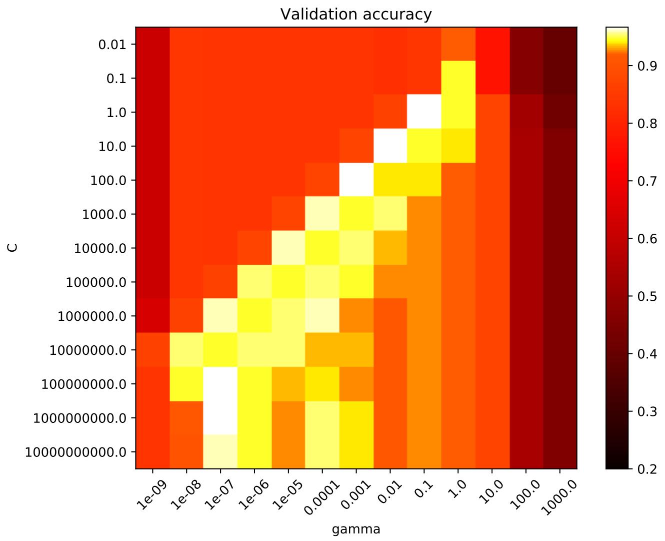
Support Vector Machines - Kernel parameters examples, C=1



Support Vector Machines - Kernel parameters examples, C=100



Support Vector Machines - Kernel parameter search



Other application of the Kernel Methods

Other application of the Kernel Methods

- Kernel K-means

$$\left\|\overline{X} - \overline{\mu}\right\|^2 = \left\|\overline{X} - \frac{\sum_{\overline{X_j}\in C} \overline{X_j}}{|C|}\right\|^2 = \overline{X}\cdot\overline{X} - 2\frac{\sum_{\overline{X_i}\in C} \overline{X}\cdot\overline{X_i}}{|C|} + \frac{\sum_{\overline{X_i},\overline{X_j}\in C} \overline{X_i}\cdot\overline{X_j}}{|C|^2}$$

  - The μ is the centroid of cluster C.
  - the cluster is assigned to the data points according the minimal kernel-based distance.
- Kernel PCA
  - Replacement of the dot products in the mean-centered data matrix.
- Kernel fisher Discriminant
- Kernel Linear Discriminant Analysis.
- ...

Classification Assesment

Classification Assessment

- How do we quantify the accuracy of the given classification model?
- These methods has several applications: evaluation of the classification effectiveness, comparing different models, selecting the best model for a particular data set, parameter tunning and advanced meta-algorithms (ensemble).
- The issues related to this task may be divided into two categories:
  1. Methodological issues
     - The proper division of the labeled dataset into training and testing part.
     - This choice has direct impact on the evaluation process (overestimation or underestimation).
     - Several approaches are possible (holdout, bootstrap, cross-validation).
  2. Quantification issues
     - These methods are associated with the providing numerical measure for the quality with respect ot the methodological issues.
     - Several methods output direct measure.
     - Other methods quantify the relative performance of classifiers.

Classification Assessment

Methodological Issues

- These methods defines the partitioning of the ground-truth data for classification evaluation.
- The using of the same data for training and testing is not possible due to over-fitting and overestimation.
- In practice, the input data shoudl be divided into three parts:
    - the model=building part of the labeled data
    - validation part of the labeled data
    - testing data.
- The validation part is used for parameter tunning or model solution.
- When the parameter tunning is done, the model is reconstructed on the whole dataset.
- The knowledge from the testing dataset should not be used in parameter tunning.

Classification Assessment - Methodological Issues

Holdout

- The labeled data is randomly divided into two disjoint sets (training and testing).
- Typically 60% to 75% is used for training set.
- This partition may be repeated several times to get he final estimation.
- The over-presented samples in the training set are under-presented in the testing sets.
- Due to not using of the whole data set for training the estimation are pessimistic.
- By repeating the process over *b* different holdout samples the mean and the variance of the error estimates may be determined.
- These information may be used for building the confidence intervals on the error.
- In case of imbalanced data, an independent sampling (for each class

Classification Assessment - Methodological Issues

Cross-Validation

- The data is divided into m disjoint subsets of equal size n/m.
- A typical choice for m is around 10.
- One segment is used as a testing set the the remaining $m-1$ as a training set.
- This process is repeated by selection each of the m subsets as a testing sets.
- The average accuracy over the m different test sets is reported.
- The size of the training set is $(m-1) * n/m$.
- When m is chosen large, the training set size is close to the whole dataset and the reported prediction is very close to the whole data set.
- The estimate of the accuracy tends to be highly representative but pessimistic.
- A special case is when $m = n$, this is called a *leave-one-out* cross-validation.
- *Stratified cross-validation* uses proportional representation of each class in the different folds and usually provides less pessimistic results

Classification Assessment - Methodological Issues

Bootstrap

- The labeled data are sampled uniformly with replacement to create a training set that may contain a duplicates.
- The labeled data of size n is sampled n times with replacement.
- The probability that a particular data point is not included in a sample is given by $(1 - 1/n)$
- The probability that the point is not included in n samples is then $(1 - 1/n)^n$.
- For large values of n the expression is approximately $1/e$.
- The fraction of labeled points included included at least once in the dataset is $1 - 1/e = 0.632$.
- The training model is constructed on the bootstrapped sample with duplicates.
- The overall accuracy is computed using the whole dataset.
- The estimate is highly optimistic due to large overlap between training and

Classification Assessment

Quantification Issues

- When the output of the classifier is in the form of a class label the prediction value is compared to the ground-truth.
- When the output of the classifier is in the form of a numerical score for each labeling possibility the label with highest score imply greater likelihood to a particular class.

Classification Assessment - Quantification Issues

Output as Class Labels

- *Accuracy* - the fraction of test instances in which the predicted value matched the ground-truth value.
- *Cost-sensitive accuracy*
    - Not all cases are equally important in all scenarios while comparing the accuracy, e.g. Imbalanced data, ill vs. healthy patients, etc.
    - This is frequently quantified by imposting different costs $c_1, \ldots, c_k$ on the misclassification on the different classes.
    - Let $n_1, \ldots, n_k$ be the number of test instances belonging to each class.
    - Let $a_1, \ldots, a_k$ be the accuracies (expressed as a fraction) on the subset of test instances belonging to each class.
    - The overall accuracy A can be computed as a weighted combination of the accuracies over the individual labels:

$$A = \frac{\sum_{i=1}^{k} c_i n_i a_i}{\sum_{i=1}^{k} c_i n_i}$$

Classification Assessment - Quantification Issues

Output as Numerical Score

- For simplicity, we will consider the two class classification problem.
- The numerical score provides more flexibility in evaluating the overall trade-off between labeling a varying number of data points as positives.
- The different setting of the threshold leads to different models.
- When the threshold is set too aggressive, the algorithm will miss the true-positives and false negatives.
- When threshold is too relaxed the algorithm produces many false-positives (false negatives).
- The correct threshold in not known a priori, but depends on the data.

Classification Assessment - Quantification Issues - Output as Numerical Score

- For any given threshold t on the predicted positive-class score the declared positive class set is denoted by $S(t)$.
- The size of the $S(t)$ changes with the changes of the t.
- The G represents the true set (ground-truth) of positive instances.
- The *Precision* is defined as the percentage of reported positives that truly turn out to be positive

$$Precision(t) = 100 * \frac{|S(t) \cap G|}{|S(t)|}$$

- The value of *Precision(t)* is not necessarily monotonic in t because both numerator and denominator may change with t differently.

Classification Assessment - Quantification Issues - Output as Numerical Score

- The *recall* is correspondingly defined as the percentage of ground-truth positives that have been reported as positive at threshold t.

$$Recall(t) = 100 * \frac{|S(t) \cap G|}{|G|}$$

- The natural trade-off between precision and recall exists, but it is not necessarily monotonic.
- The F_1-measure summarizes both precision and the recall.

$$F_1(t) = \frac{2 \cdot Precision(t) \cdot Recall(t)}{Precision(t) + Recall(t)}$$

- The F_1-measure provides better quantification that precision or recall, but it is still depends on the t.
- The entire trade-off between recall and precision may be investigated by the plotting these values with respect to the threshold t.

Classification Assessment - Quantification Issues - Output as Numerical Score

ROC curve

- ROC curve is a different method for evaluating the trade-off which is more intuitive.
- The *true-positive rate* is the same as the recall:

$$TPR(t) = Recall(t) = 100 * \frac{|S(t) \cap G|}{|G|}$$

- The false positive rate is the percentage of the falsely reported positives out of the ground-truth negatives.

$$FPR(t) = 100 * \frac{|S(t) - G|}{|D - G|}$$

- The ROC curve the define by plotting the *FPR(t)* on the x-axis and *TPR(t)* on the y-axis for varying values of t.

Classification Assessment - Quantification Issues - Output as Numerical Score

ROC curve cont.

- The ROC curve has always points $(0, 0)$ and $(100, 100)$.
- The random classifier is expected to exhibit performance along the diagonal.
- The *lift* above the diagonal provides the idea about accuracy of the approach.
- The area below the ROC curve provides a concrete quantitative evaluation of the effectiveness of the particular method.

Questions?