

Intro to Classification: Math aspects

Evgeny Burnaev, Alexey Zaytsev

Skoltech, Moscow, Russia



Skolkovo Institute of Science and Technology

- 1 Main concepts
- 2 Learning a Classifier
- 3 Logistic Regression
- 4 k-Nearest Neighbour Classifier
- 5 Classification and Regression Trees
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier

- 1 Main concepts**
- 2 Learning a Classifier**
- 3 Logistic Regression**
- 4 k-Nearest Neighbour Classifier**
- 5 Classification and Regression Trees**
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier**

- **Training data:** sample drawn i.i.d. w.r.t. D on $X \subseteq \mathbb{R}^d$

$$S = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\} \in \{X \times \{-1, +1\}\}^m$$

- **Problem:** find hypothesis $f : X \rightarrow \{-1, +1\}$ in F (classifier) with small generalization error $R(f)$

- **Object:** customer at some moment of time
- **Label:** $-1/ +1$ (not churn/churn)
- **Features:** sex, address, subscription plan, already purchased services, frequency and duration of calls, etc.
- **Challenges:**
 - estimate churn probability
 - huge samples
 - laborious feature engineering

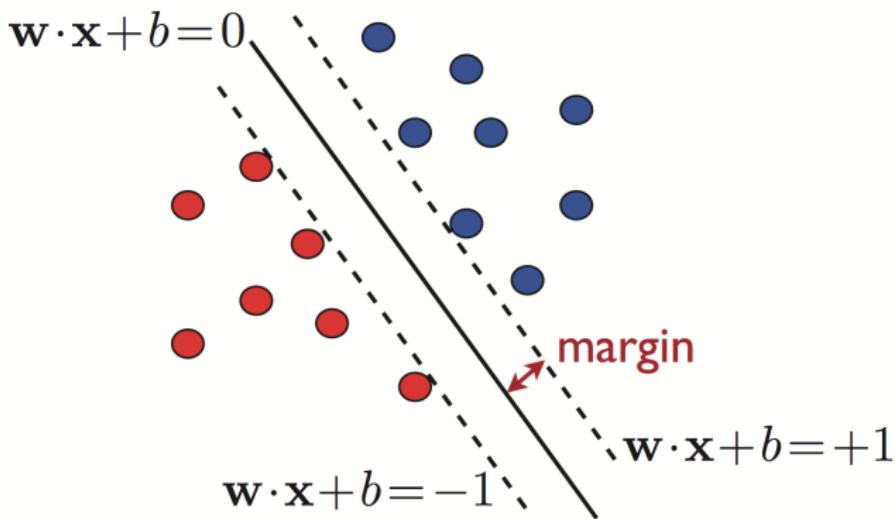


Figure – Linear classifier [Mohri]

- Example of linear classification (hyperplane)
- Set of classifiers: $F = \{x \rightarrow \text{sign}(w \cdot x + b), w \in \mathbb{R}^d, b \in \mathbb{R}^1\}$
- Geometric margin: $\rho = \min_{i \in [1, m]} \frac{|w \cdot x_i + b|}{\|w\|}$

- Binary classification, $y_i \in \{-1, +1\}$
- Classifier $f(\mathbf{x}_i) \in \{-1, +1\}$

Error type	Classifier	True label
TP, True Positive	$f(\mathbf{x}_i) = +1$	$y_i = +1$
TN, True Negative	$f(\mathbf{x}_i) = -1$	$y_i = -1$
FP, False Positive	$f(\mathbf{x}_i) = +1$	$y_i = -1$
FN, False Negative	$f(\mathbf{x}_i) = -1$	$y_i = +1$

- Accuracy = 1 – Proportion of classification errors, i.e.

$$\text{Accuracy} = 1 - \frac{1}{m} \sum_{i=1}^m \mathbb{1}_{\{f(\mathbf{x}_i) \neq y_i\}} = 1 - \frac{FP + FN}{FP + FN + TP + TN}$$

- **Shortcomings:** Accuracy does not take into account neither disbalance of classes nor different error costs for different classes

- Binary classification: $y_i \in \{-1, +1\}$
- Classifier: $f(\mathbf{x}; \mathbf{w}, w_0) = \text{sign}(h(\mathbf{x}; \mathbf{w}) - w_0)$
- Here $h(\mathbf{x}; \mathbf{w})$ can be considered as a kind of probability (score) of the positive class
- The bigger w_0 is the bigger the number of \mathbf{x}_i for which $f(\mathbf{x}_i) = -1$
- Let us denote by λ_y a penalty when object from class y is misclassified
- Loss function

$$L(f(\mathbf{x}_i), y_i) = \lambda_{y_i} 1_{\{f(\mathbf{x}_i; \mathbf{w}, w_0) \neq y_i\}} = \lambda_{y_i} 1_{\{(h(\mathbf{x}_i; \mathbf{w}) - w_0)y_i < 0\}}$$

- Let us define ROC (receiver operating characteristic) curve
- Each point of the curve corresponds to some

$$f(\mathbf{x}; \mathbf{w}, w_0) = \text{sign}(h(\mathbf{x}; \mathbf{w}) - w_0)$$

- Abscissa:** we depict values of FPR (false positive rate) as a function of w_0

$$FPR(w_0) = \frac{\sum_{i=1}^m 1_{\{y_i=-1\}} 1_{\{f(\mathbf{x}_i; \mathbf{w}, w_0) = +1\}}}{\sum_{i=1}^m 1_{\{y_i=-1\}}}$$

- Ordinate:** we depict values of TPR (true positive rate):

$$TPR(w_0) = \frac{\sum_{i=1}^m 1_{\{y_i=+1\}} 1_{\{f(\mathbf{x}_i; \mathbf{w}, w_0) = +1\}}}{\sum_{i=1}^m 1_{\{y_i=+1\}}}$$

- $(1 - FPR)$ is called specificity, TPR is called sensitivity

Example of ROC-curve

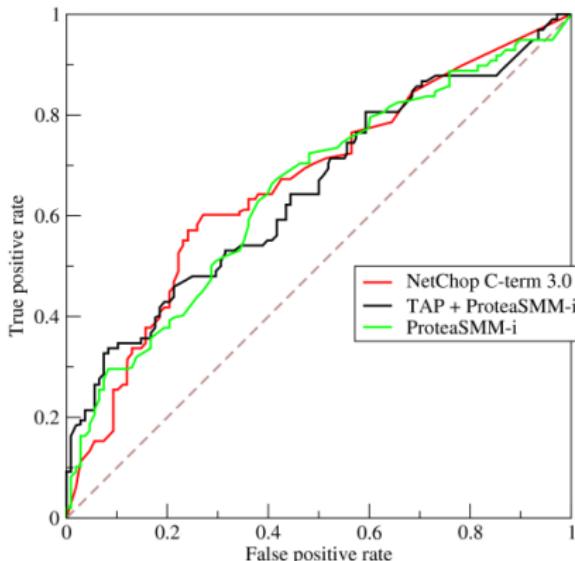


Figure – ROC-curve [Wikipedia]

- AUC is an Area Under ROC-curve, used to characterize classification accuracy ($AUC \geq 0.5$)
- Dashed line — the worst accuracy (random guessing)

Precision and Recall in case of Binary Classification

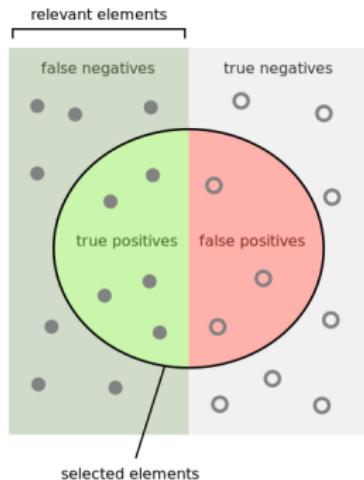
- TP — true positives
- FP — false positives
- FN — false negatives
- Precision and Recall

$$\text{Precision} : P = \frac{TP}{TP + FP},$$

$$\text{Recall} : R = \frac{TP}{TP + FN}$$

In information retrieval

- P is a fraction of relevant objects among retrieved
- R is a fraction of retrieved objects among relevant



How many selected items are relevant?



How many relevant items are selected?



Figure – Graphical illustration [Wikipedia]

Precision and Recall in case of Multi-Class Classification

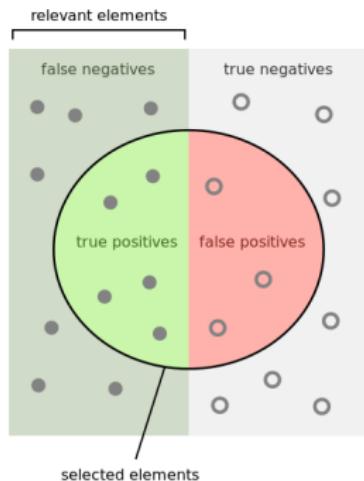
For each class $y \in Y$

- TP_y — true positives
- FP_y — false positives
- FN_y — false negatives
- Precision and Recall with micro-averaging

$$\text{Precision} : P = \frac{\sum_y TP_y}{\sum_y (TP_y + FP_y)},$$

$$\text{Recall} : R = \frac{\sum_y TP_y}{\sum_y (TP_y + FN_y)}$$

Micro-averaging is not sensitive to errors for rare classes



How many selected items are relevant?



How many relevant items are selected?



Figure – Graphical illustration [Wikipedia]

Precision and Recall in case of Multi-Class Classification

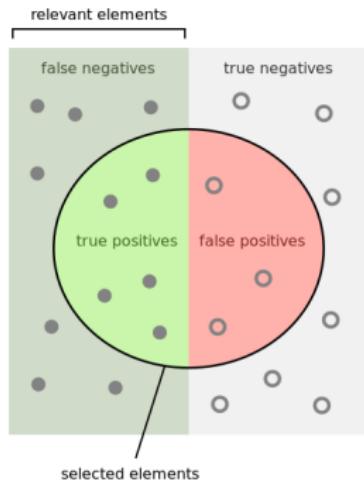
For each class $y \in Y$

- TP_y — true positives
- FP_y — false positives
- FN_y — false negatives
- Precision and Recall with macro-averaging

$$\text{Precision} : P = \frac{1}{|Y|} \sum_y \frac{TP_y}{TP_y + FP_y},$$

$$\text{Recall} : R = \frac{1}{|Y|} \sum_y \frac{TP_y}{TP_y + FN_y}$$

Macro-averaging is sensitive to errors for rare classes



How many selected items are relevant?



How many relevant items are selected?



Figure – Graphical illustration [Wikipedia]

Example of Precision-Recall curve

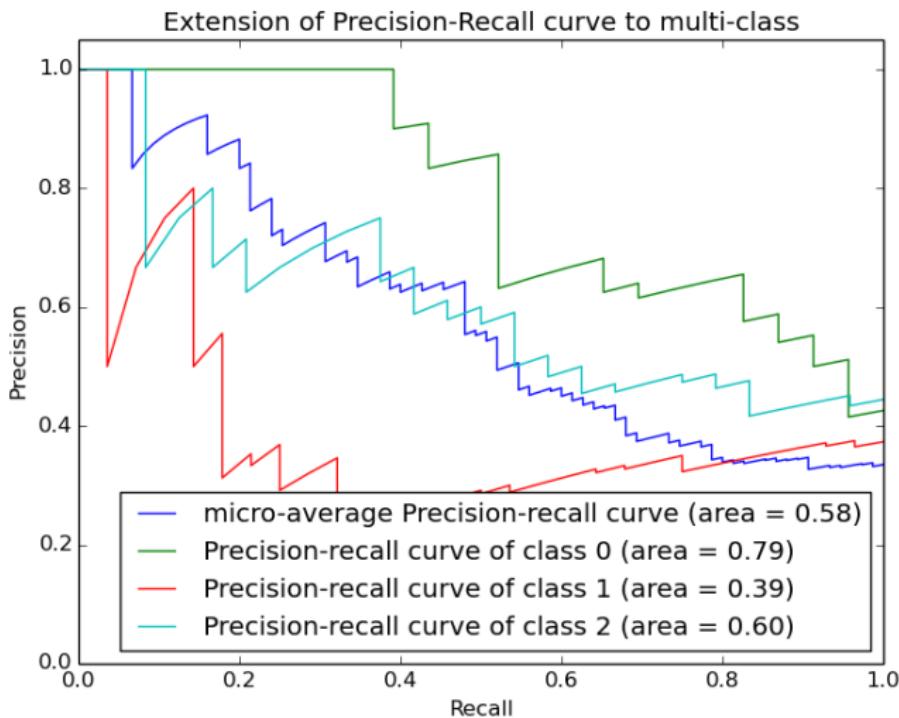


Figure – Precision-Recall curve [scikit-learn manual]

- Sensitivity and specificity are better suited for problems with unbalanced classes
- Precision and Recall are better suited for problems when a class of interest is rare
- Aggregated errors:
 - AUC is better suited when ratio of errors costs for different classes is not fixed
 - AUC-PR — area under precision-recall curve
 - F -measure $F_1 = \frac{2PR}{P+R}$
 - F_β -measure $F_\beta = \frac{(1+\beta^2)PR}{\beta^2P+R}$ (the bigger β is the more R is important)

- 1 Main concepts
- 2 Learning a Classifier
- 3 Logistic Regression
- 4 k-Nearest Neighbour Classifier
- 5 Classification and Regression Trees
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier

- Learning sample $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- Classifier

$$f(\mathbf{x}; \mathbf{w}) = \text{sign}(g(\mathbf{x}; \mathbf{w}))$$

- Example of Linear classifier: $g(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ (we assume x_1 to account for a constant $-w_0$)
- Loss function

$$\hat{R}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m 1_{\{g(\mathbf{x}_i; \mathbf{w}) y_i < 0\}}$$

can not be efficiently optimized w.r.t. \mathbf{w}

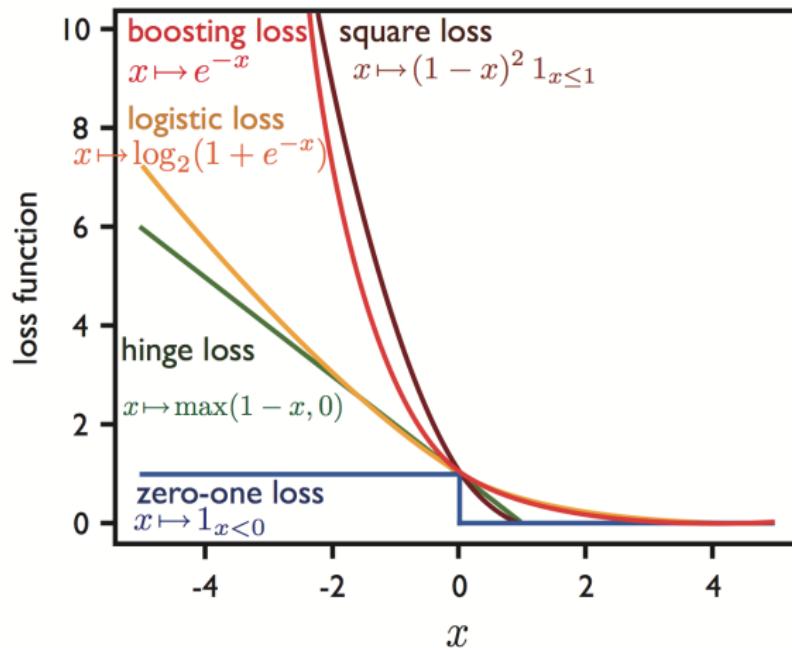
Surrogate Loss Functions

Convex upper bound for an indicator function

$$1_{\{x < 0\}} \leq L(x),$$

i.e.

$$1_{\{g(\mathbf{x}_i; \mathbf{w})y_i < 0\}} \leq L(g(\mathbf{x}_i; \mathbf{w})y_i),$$



- Learning sample $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$
- Classifier

$$f(\mathbf{x}; \mathbf{w}) = \text{sign}(g(\mathbf{x}; \mathbf{w}))$$

- Example of Linear classifier: $g(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \mathbf{x}$ (we assume x_1 to account for a constant $-w_0$)
- Binary Loss function and its convex upper bound

$$1_{\{g(\mathbf{x}_i; \mathbf{w})y_i < 0\}} \leq L(g(\mathbf{x}_i; \mathbf{w})y_i)$$

- Learning \equiv ERM

$$\widehat{R}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m 1_{\{g(\mathbf{x}_i; \mathbf{w})y_i < 0\}} \leq \frac{1}{m} \sum_{i=1}^m L(g(\mathbf{x}_i; \mathbf{w})y_i) \rightarrow \min_{\mathbf{w}}$$

- Estimate a test error using a separate test sample
 $S'_{m'} = \{(\mathbf{x}'_j, y'_j)\}_{j=1}^{m'}$

$$\frac{1}{m'} \sum_{j=1}^{m'} 1_{\{g(\mathbf{x}'_j; \mathbf{w})y'_j < 0\}}$$

- Causes of overfitting
 - too small number of examples
 - too big number of features
 - linear dependence between features (multicollinearity)
- Symptoms of Overfitting
 - too big absolute values of weights $|w_j|$ and different signs of w_j
 - $\hat{R}(\mathbf{w}^*) \ll \hat{R}'(\mathbf{w}^*)$ (test error is \gg than train error)
- Regularization is typically used to prevent overfitting

- We impose additional penalty for high absolute values of weights

$$\bar{L}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m L(g(\mathbf{x}_i; \mathbf{w})y_i) + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- In order to tune regularization coefficient λ we can use
 - cross-validation
 - Bayesian inference

- Probabilistic classifier model: a parametric model for

$$\mathbb{P}(y|\mathbf{x}) = p(y|\mathbf{x}; \mathbf{w})$$

- MLE for \mathbf{w}

$$\prod_{i=1}^m p(y_i|\mathbf{x}_i; \mathbf{w}) \rightarrow \max_{\mathbf{w}}$$

- Log-likelihood

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^m \log p(y_i|\mathbf{x}_i; \mathbf{w}) \rightarrow \max_{\mathbf{w}}$$

- If we set

$$p(y|\mathbf{x}; \mathbf{w}) = e^{-L(g(\mathbf{x}; \mathbf{w})y)},$$

then we get the surrogate loss ERM problem

$$\sum_{i=1}^m L(g(\mathbf{x}_i; \mathbf{w})y_i) \rightarrow \min_{\mathbf{w}},$$

i.e. the surrogate loss function $L(\cdot)$ and $g(\mathbf{x}; \mathbf{w})$ define the probabilistic classifier model

- 1 Main concepts
- 2 Learning a Classifier
- 3 Logistic Regression
- 4 k-Nearest Neighbour Classifier
- 5 Classification and Regression Trees
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier

- Linear Classifier in case of $Y = \{-1, +1\}$

$$f(\mathbf{x}; \mathbf{w}) = \text{sign}(g(\mathbf{x}; \mathbf{w})) = \text{sign}(\mathbf{w}^\top \mathbf{x})$$

- Logarithmic loss function

$$L(t) = \log(1 + e^{-t})$$

- A model for conditional probability

$$p(y|\mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}}}$$

- Regularized logistic regression

$$\overline{L}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-(\mathbf{w}^\top \mathbf{x}_i)y_i)) + \frac{\lambda}{2} \|\mathbf{w}\|^2 \rightarrow \min_{\mathbf{w}}$$

- Linear Classifier in a multi-class case, i.e. $|Y| > 1$
- Probability of an object to belong to some class y is equal to

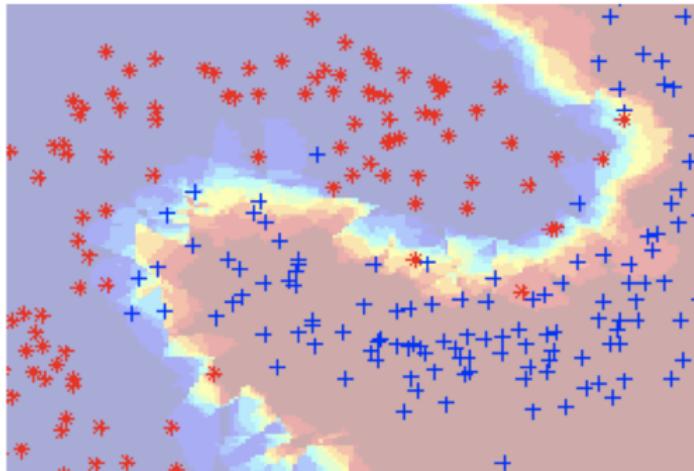
$$p(y|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_y^\top \mathbf{x})}{\sum_{z \in Y} \exp(\mathbf{w}_z^\top \mathbf{x})} = \text{SoftMax}_{y \in Y}(\mathbf{w}_y^\top \mathbf{x})$$

- Regularized log-likelihood for the logistic regression

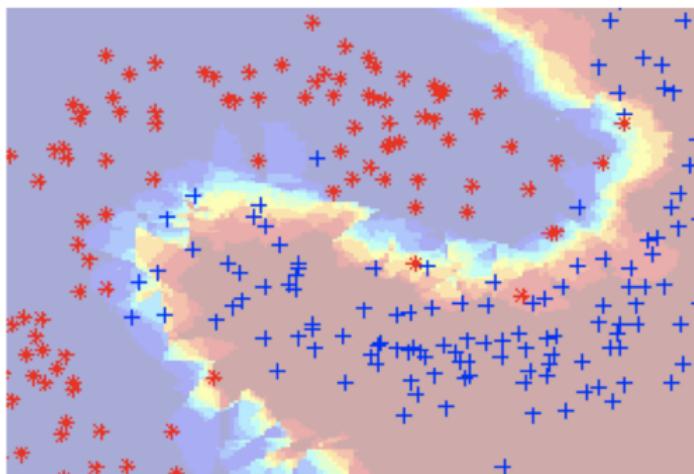
$$\bar{\mathcal{L}}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m \log p(y_i | \mathbf{x}_i, \mathbf{w}) - \frac{\lambda}{2} \sum_{y \in Y} \|\mathbf{w}_y\|^2 \rightarrow \max_{\mathbf{w}}$$

- 1 Main concepts
- 2 Learning a Classifier
- 3 Logistic Regression
- 4 k-Nearest Neighbour Classifier
- 5 Classification and Regression Trees
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier

- Compactness hypothesis (classification): as a rule similar objects belongs to the same class
- Continuity hypotheses (regression): as a rule similar characteristics correspond to similar objects



- Classification: objects as points in feature space belong to the same class if their feature vectors are close
- Regression: objects as points in feature space have the same characteristics if their feature vectors are close
- What is “close objects”?



Distance functions

- We define some distance $\rho : X \times X \rightarrow [0, \infty)$
- E.g. Euclidian or weighted Euclidian distance:

$$\rho(\mathbf{x}, \mathbf{x}_i) = \left(\sum_{j=1}^p |x_j - x_{i,j}|^2 \right)^{1/2}, \quad \rho(\mathbf{x}, \mathbf{x}_i) = \left(\sum_{j=1}^p \omega_j^2 |x_j - x_{i,j}|^2 \right)^{1/2}$$

- E.g. Manhattan distance

$$\rho(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^p |x_j - x'_j|$$

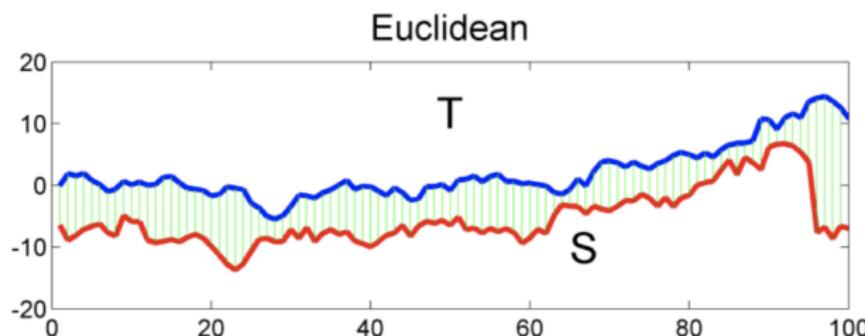
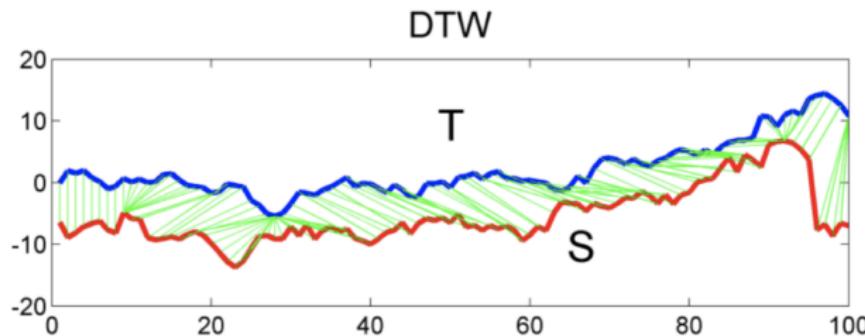


- Distance between strings
- Levenshtein distance is a string metric for measuring the difference between two sequences
- Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other

CTGGGCTA**AAA**GGT**C**CCTTAGCC..TTTAG**AAAAAA**.GGGCCATTAGG**AAA**TTGC
CTGGGACT**AAA**....CCTTAGCCT**T**TTTA**C**AAAAA**T**GGGCCATTAGG...TTGC

Distance functions

- Distance between time-series
- E.g. total Euclidean distance
- E.g. Dynamic Time Warping and others



- Let us consider classification problem
- Fix the number k of k-nearest neighbours
- Sample: $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- Train: memorize sample S_m

- For any new $\mathbf{x} \in X$ we rank objects $\mathbf{x}_1, \dots, \mathbf{x}_m$ such that

$$\rho(\mathbf{x}, \mathbf{x}_{(1)}) \leq \rho(\mathbf{x}, \mathbf{x}_{(2)}) \leq \dots \leq \rho(\mathbf{x}, \mathbf{x}_{(m)}),$$

i.e. e.g. $\mathbf{x}_{(1)}$ is the closest object to \mathbf{x} .

- So let

- $\mathbf{x}_{(i)}$ be the i th closest object to \mathbf{x} among $\mathbf{x}_1, \dots, \mathbf{x}_m$
- $y_{(i)}$ be the label of $\mathbf{x}_{(i)}$

- We define k-Nearest Neighbour Classifier prediction as the class, which is the most popular among the k nearest neighbours

$$\hat{f}(\mathbf{x}) = \arg \max_{y \in Y} \# \{ i, i = 1, 2, \dots, k : y_{(i)} = y \}$$

- Pros:** simple realization, interpretable, case-based reasoning
- Cons:** not-stable enough (for small k) to outliers and noise, not very accurate, the whole sample should be stored, k should be selected, **curse of dimensionality**

Iris flower data set [Fisher, 1936]

- Quantify the morphologic variation of Iris flowers of three related species (*Iris setosa*, *Iris virginica* and *Iris versicolor*) by four features: the length and the width of the sepals and petals, in centimetres
- $d = 4$ features, $K = 3$ classes (mono-label case), $m = 150$

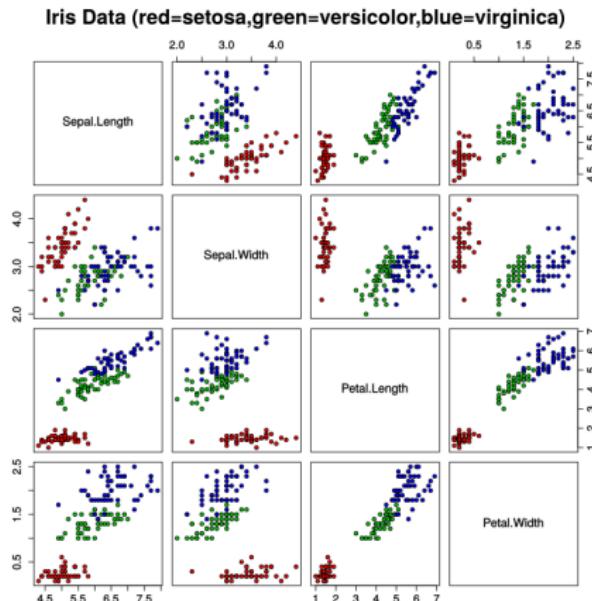
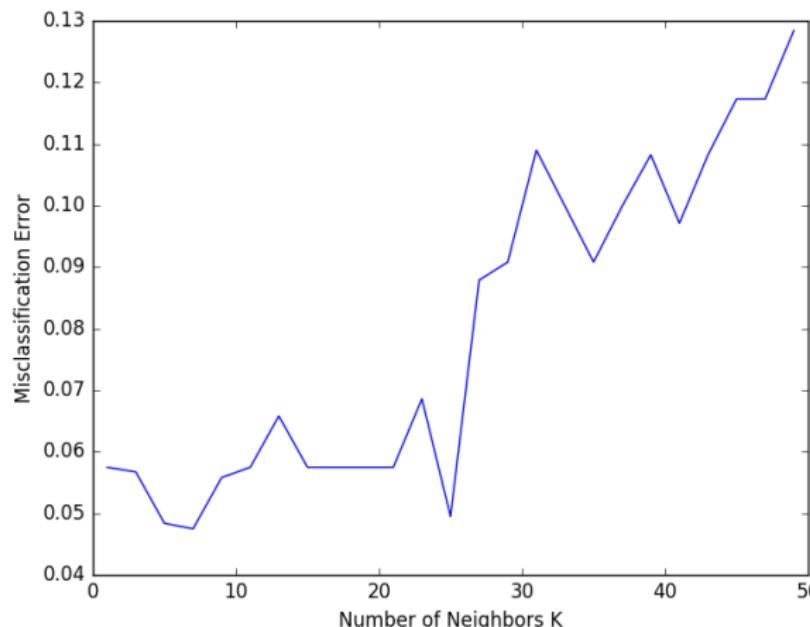


Figure – Scatterplot of the data set [Wikipedia]

Tuning of k

- Iris Flower Dataset [Fisher]
- 10-fold cross-validation to estimate classification error
- In practice usually $k \in \{1, \dots, 5\}$



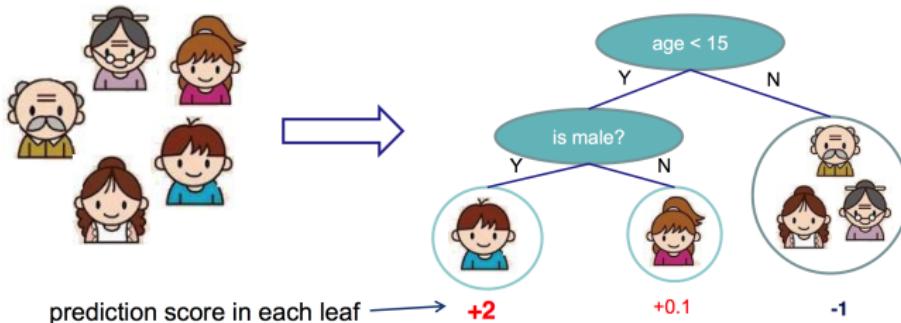
- 1 Main concepts
- 2 Learning a Classifier
- 3 Logistic Regression
- 4 k-Nearest Neighbour Classifier
- 5 Classification and Regression Trees
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier

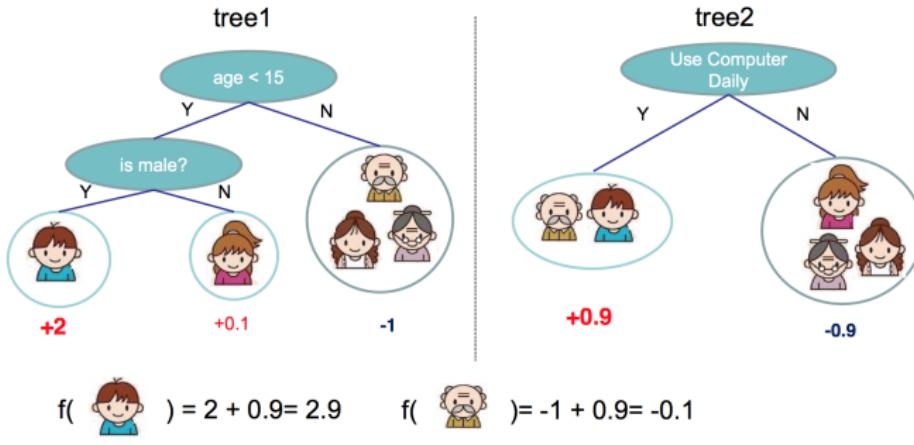
Classification and Regression Trees (CART)

- Classification and Regression Trees:

- Decision rules
- Contains one score in each leaf value

Input: age, gender, occupation,... \Rightarrow Does the person like computer games?





Prediction is a sum of scores predicted by each of the tree

- Model: we have T trees

$$\hat{f}_T(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^T f_t(\mathbf{x}), \quad f_t(\mathbf{x}) \in F,$$

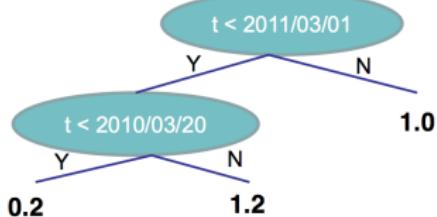
where F is a space of functions, containing all regression trees

- Parameters: structure of each tree, and the score in the leaf

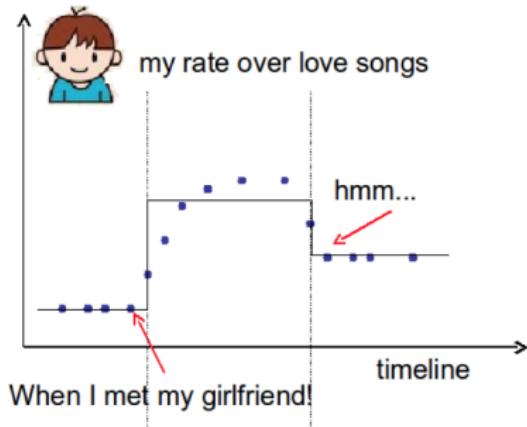
- Very widely used, look for GBM, random forest...
 - Almost half of data mining competitions are won by using some variants of tree ensemble methods
- Invariant to scaling of inputs, so you do not need to do careful features normalization
- Learn higher order interaction between features
- Can be scalable, and are used in Industry

- How can we learn functions?
- Define objective (loss, regularization), and optimize it
- Example:
 - Consider regression tree on single input t (time)
 - We want to predict whether a person like romantic music at time t

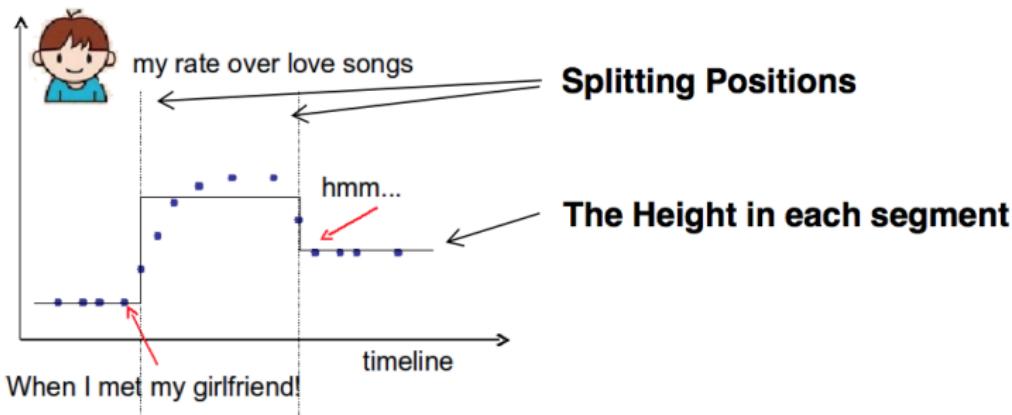
The model is a regression tree that splits on time



Piecewise step function over time

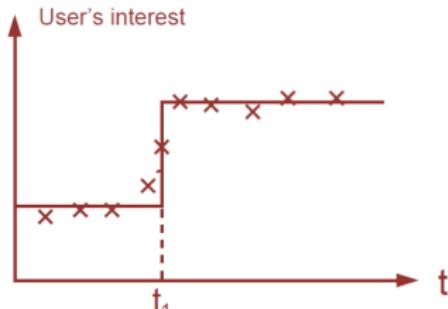
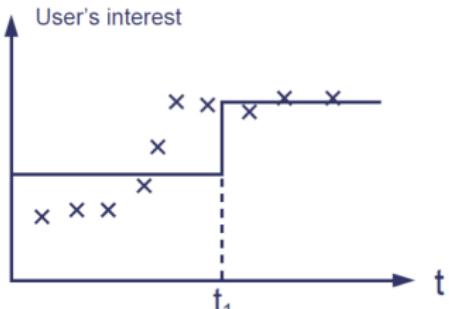
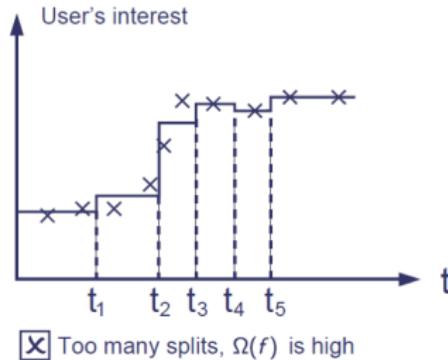
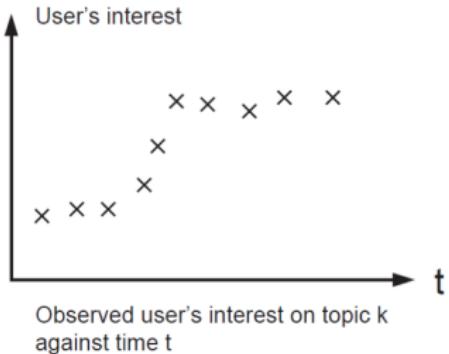


- Things we need to learn



- Objective for single variable regression tree (step functions)
 - Training Loss: How will the function fit on the points?
 - Regularization: How do we define complexity of the function?
E.g. number of splitting points, L_2 -norm of the height in each segment, ...

Learning a step function



- Assume that we construct T trees

$$\hat{f}_T(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^T f_t(\mathbf{x}), \quad f_t \in F$$

- Objective

$$R(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m l(y_i, \hat{y}_i) + \sum_{t=1}^T \lambda_t \Omega(f_t),$$

where e.g. $l(y, \hat{y})$ can be

- squared loss (regression)
- log-loss (classification)

- Possible ways to define Ω

- Number of nodes in the tree, depth
- L_2 -norm of the leaf weight
- ...

- Decision Trees algorithms contain a lot of heuristics:
 - Split by information gain
 - Prune the tree
 - Maximum depth
 - Smooth leaf values
- Most heuristics map well to objectives
 - Information gain → training loss
 - Pruning → regularization defined by number of nodes
 - Max depth → constraint on the function space
 - Smoothing leaf values → L_2 -norm regularization on leaf weights
- Decision Trees can be used for Classification, Regression, Ranking, ...
We just need to define appropriate objective function

- Different classification metrics to assess different applied requirements
- Many classification models with their own pros and cons
- Widely used classifiers: logistic regression and ensembles of decision trees

- 1 Main concepts
- 2 Learning a Classifier
- 3 Logistic Regression
- 4 k-Nearest Neighbour Classifier
- 5 Classification and Regression Trees
- 6 Appendix. Materials for Additional Study: The Newton-Raphson Method for logistic regression; Optimal Bayesian Classifier

- Surrogate Loss function for a binary logistic regression

$$R(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m L((\mathbf{w}^\top \mathbf{x}_i) y_i)$$

- Steps:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - r_t(R''(\mathbf{w}^t))^{-1} R'(\mathbf{w}^t)$$

- Components of a gradient

$$\frac{\partial R(\mathbf{w})}{\partial w_j} = -\frac{1}{m} \sum_{i=1}^m (1 - \sigma_i) y_i x_{i,j}, \quad j = 1, \dots, d$$

- Hessian

$$\frac{\partial^2 R(\mathbf{w})}{\partial w_j \partial w_k} = \frac{1}{m} \sum_{i=1}^m (1 - \sigma_i) \sigma_i x_{i,j} x_{i,k}, \quad j, k = 1, \dots, d,$$

where $\sigma_i = \sigma(\mathbf{w}^\top \mathbf{x}_i y_i)$, $\sigma(t) = \frac{1}{1+e^{-t}}$

- $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m \in \mathbb{R}^{m \times d}$ is a matrix of objects features
- $\Gamma = \text{diag}\left(\sqrt{(1 - \sigma_i)\sigma_i}\right) \in \mathbb{R}^{m \times m}$ is a diagonal matrix of weights
- $\tilde{\mathbf{X}} = \Gamma \mathbf{X}$ is a weighted matrix of features
- $\tilde{y}_i = y_i \sqrt{(1 - \sigma_i)/\sigma_i}$, $\tilde{\mathbf{y}} = \{\tilde{y}_i\}_{i=1}^m$ is a weighted vector of labels
- Then we get that

$$(R''(\mathbf{w}))^{-1} R'(\mathbf{w}) = -\left(\mathbf{X}^\top \Gamma^2 \mathbf{X}\right)^{-1} \mathbf{X}^\top \Gamma \tilde{\mathbf{y}} = -\left(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}} \tilde{\mathbf{y}}$$

This coincides with a solution of a weighted least-squares problem

$$R(\mathbf{w}) = \left\| \tilde{\mathbf{X}} \mathbf{w} - \tilde{\mathbf{y}} \right\|^2 = \frac{1}{m} \sum_{i=1}^m (1 - \sigma_i) \sigma_i \left(\mathbf{w}^\top \mathbf{x}_i - \frac{y_i}{\sigma_i} \right)^2 \rightarrow \min_{\mathbf{w}}$$

- On each step of the Newton-Raphson method we construct a weighted least-squares regression

$$R(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (1 - \sigma_i) \sigma_i \left(\mathbf{w}^\top \mathbf{x}_i - \frac{y_i}{\sigma_i} \right)^2 \rightarrow \min_{\mathbf{w}}$$

- Here
 - $\sigma_i = p(y_i | \mathbf{x}_i)$ is a probability to correctly classify \mathbf{x}_i
 - the closer \mathbf{x}_i to the boundary, the bigger the weight $(1 - \sigma_i)\sigma_i$
 - the bigger the probability of an error, the bigger the value of y_i/σ_i

Thus on each iteration we tune \mathbf{w} to perform better on more difficult examples

- **Input:** \mathbf{X} , \mathbf{y} , i.e. a matrix and a vector of input features and corresponding labels
- **Output:** estimate of \mathbf{w}
- For $t = 1, 2, \dots$
 - $\sigma_i = \sigma(\mathbf{w}^\top \mathbf{x}_i y_i)$, $i = 1, \dots, m$
 - $\gamma_i = \sqrt{(1 - \sigma_i)\sigma_i}$, $i = 1, \dots, m$
 - $\tilde{\mathbf{X}} = \text{diag}(\gamma_1, \dots, \gamma_m) \mathbf{X}$
 - $\tilde{y}_i = y_i \sqrt{(1 - \sigma_i)/\sigma_i}$, $i = 1, \dots, m$
 - select a gradient step r_t and calculate
$$\mathbf{w} \leftarrow \mathbf{w} + r_t \left(\tilde{\mathbf{F}}^\top \tilde{\mathbf{F}} \right)^{-1} \tilde{\mathbf{F}}^\top \mathbf{y}$$
 - if σ_i changes not significantly, then stop iterations

- Let
 - X be a feature space
 - Y be a space of labels, e.g. $Y = \{0, 1\}$
 - $p(\mathbf{x}, y)$ be a joint distribution on $X \times Y$
 - $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ be an i.i.d. sample
- We want to construct an optimal classifier $f : X \rightarrow Y$

- We assume that we know joint density

$$p(\mathbf{x}, y) = p(\mathbf{x})p(y|\mathbf{x}) = p(y)p(\mathbf{x}|y)$$

Here

- $p(y)$ is a prior distribution on Y
- $p(\mathbf{x}|y)$ is a likelihood of a class y
- $p(y|\mathbf{x})$ is a posterior probability of a class y

- Classifier maximizing posterior probability

$$f(\mathbf{x}) = \arg \max_{y \in Y} p(y|\mathbf{x}) = \arg \max_{y \in Y} p(\mathbf{x}|y)p(y)$$

- Classifier $f(\mathbf{x})$ divides X into disjoint domains

$$H_y = \{\mathbf{x} \in X | f(\mathbf{x}) = y\}, y \in Y$$

- We get error for (\mathbf{x}, y) if $\mathbf{x} \in H_z, z \neq y$
- Probability of Error: $P(H_z, y) = \int_{H_z} p(\mathbf{x}, y) d\mathbf{x}$
- Losses: λ_{yz} for all $(y, z) \in Y \times Y$
- Average Risk

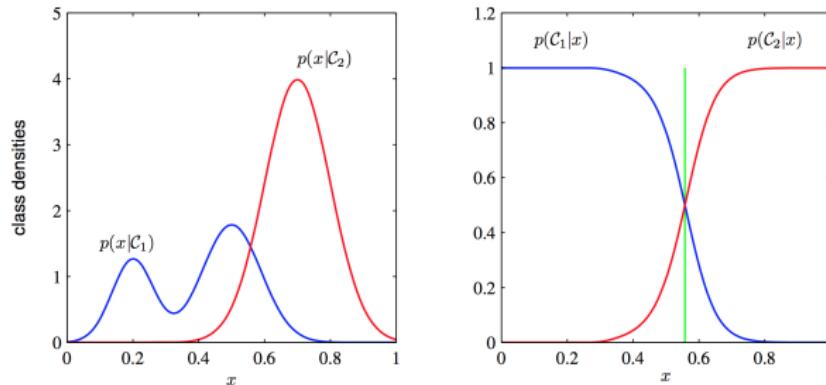
$$R(f) = \sum_{y \in Y} \sum_{z \in Y} \lambda_{yz} P(H_z, y)$$

- **Theorem:** Optimal Bayesian Classifier $f(\mathbf{x})$, minimizing average risk $R(f)$, has the form

$$f_{\text{opt}}(\mathbf{x}) = \arg \min_{z \in Y} \sum_{y \in Y} \lambda_{yz} p(y) p(\mathbf{x}|y)$$

- **Corollary:** If $\lambda_{yy} = 0$ and $\lambda_{yz} = \lambda_y$ for all $y, z \in Y$, then

$$f_{\text{opt}}(\mathbf{x}) = \arg \max_{y \in Y} \lambda_y p(\mathbf{x}|y) p(y)$$



- **Theoretical setup:**

- Assumption: we know probabilities $p(y)$ and $p(\mathbf{x}|y)$, $y \in Y$
 - We now how to construct a classifier $f(\mathbf{x})$, minimizing average risk $R(f)$

- **Applied setup:**

- Assumption: training set $S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$
 - Estimate probabilities $\hat{p}(y)$ and $\hat{p}(\mathbf{x}|y)$, $y \in Y$ to calculate a Bayesian classifier
- We loose optimality when using empirical probability estimates
- Usually it is more difficult to estimate probability density function than to construct efficient classifier
- Naive Bayesian Classifier:
 - assume that $p(\mathbf{x}|y) = \prod_{j=1}^d p(x_j|y)$, i.e. components of \mathbf{x} are independent
 - simplifying assumption, but it is easier to estimate d one-dimensional distributions $p(x_j|y)$, $j = 1, \dots, d$
 - although crude approximation, still it sometimes works in practice

- Let us consider Multidimensional Gaussian Distribution
- We assume that $X = \mathbb{R}^d$ and

$$p(\mathbf{x}|y) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y) = \frac{1}{\sqrt{(2\pi)^N \det \boldsymbol{\Sigma}_y}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1} (\mathbf{x}-\boldsymbol{\mu}_y)},$$

where $\boldsymbol{\mu}_y \in \mathbb{R}^d$, $\boldsymbol{\Sigma}_y \in \mathbb{R}^{d \times d}$, $y \in Y$

- Optimal Separating Boundary

$$B = \{\mathbf{x} \in X : \lambda_y p(y) p(\mathbf{x}|y) = \lambda_z p(z) p(\mathbf{x}|z)\},$$

where $y, z \in Y$, $y \neq z$

- In case when $\Sigma_y = \Sigma$ for all $y \in Y$ we get a Linear Discriminant function
- E.g. when $Y = \{0, 1\}$ we get the decision boundary

$$B = \left\{ \mathbf{x} \in X : [\Sigma^{-1}(\mu_1 - \mu_0)]^\top \mathbf{x} + \frac{1}{2} \mu_1^\top \Sigma^{-1} \mu_1 - \frac{1}{2} \mu_2^\top \Sigma^{-1} \mu_2 + \log \frac{p(1)}{p(0)} = 0 \right\}$$

- MLE for μ_y and Σ_y are

$$\hat{\mu}_y = \frac{1}{m_y} \sum_{i:y_i=y} \mathbf{x}_i, \quad \hat{\Sigma}_y = \frac{1}{m} \sum_{y \in Y} \sum_{i:y_i=y} (\mathbf{x}_i - \hat{\mu}_y)(\mathbf{x}_i - \hat{\mu}_y)^\top$$

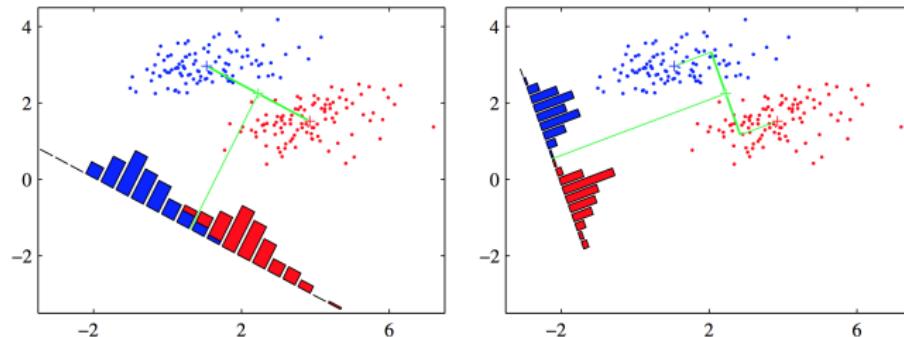
- Regularization of a covariance matrix estimate
 - Use $\hat{\Sigma} + \tau \mathbf{I}$ instead of $\hat{\Sigma}$
 - Select τ using cross-validation
- Before constructing a discriminator perform outlier detection and censoring of data

- We consider a two-class classification problem, i.e. $Y = \{0, 1\}$
- We consider a separating hyperplane

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

and we classify $\mathbf{x} : f(\mathbf{x}) \geq -w_0$ for some w_0 as class C_0 and otherwise as class C_1

- We want to select such projection direction that maximizes the class separation



- Let us denote by m_0 and m_1 the number of points, belonging to classes C_1 and C_2 correspondingly
- We define mean vectors of the two classes as

$$\mathbf{m}_0 = \frac{1}{m_0} \sum_{i:y_i=0} \mathbf{x}_i, \quad \mathbf{m}_1 = \frac{1}{m_1} \sum_{i:y_i=1} \mathbf{x}_i$$

- The simplest measure of separation = separation of the projected class means, i.e.

$$m_{1,\mathbf{w}} - m_{0,\mathbf{w}} = \mathbf{w}^\top (\mathbf{m}_1 - \mathbf{m}_0)$$

- The within-class variance of the transformed data from class C_k is given by

$$s_k^2 = \sum_{i:y_i=k} (z_i - m_{k,\mathbf{w}})^2, \quad k \in \{0, 1\}$$

where $z_i = \mathbf{w}^\top \mathbf{x}_i$

- The Fisher criterion is

$$J(\mathbf{w}) = \frac{(m_{1,\mathbf{w}} - m_{0,\mathbf{w}})^2}{s_1^2 + s_0^2}$$

- In vector form

$$J(\mathbf{w}) = \frac{\mathbf{w} \mathbf{S}_B \mathbf{w}}{\mathbf{w} \mathbf{S}_W \mathbf{w}},$$

where

$$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_0)(\mathbf{m}_1 - \mathbf{m}_0)^\top$$

is the between-class covariance matrix, and

$$\mathbf{S}_W = \sum_{i:y_i=0} (\mathbf{x}_i - \mathbf{m}_0)(\mathbf{x}_i - \mathbf{m}_0)^\top + \sum_{i:y_i=1} (\mathbf{x}_i - \mathbf{m}_1)(\mathbf{x}_i - \mathbf{m}_1)^\top$$

is the within-class covariance matrix

- $J(\mathbf{w})$ is maximized for

$$\mathbf{w} \sim \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_0)$$

- We consider $K > 2$ classes ($d > K$)
- The weight vectors $\{\mathbf{w}_k\}$ can be considered as columns of a matrix \mathbf{W} , s.t.

$$\mathbf{y} = \mathbf{W}\mathbf{x},$$

i.e. $f_k(\mathbf{x}) = \mathbf{w}_k^\top \mathbf{x}$ and we assign a point \mathbf{x} to class C_k if $f_k(\mathbf{x}) > f_j(\mathbf{x})$ for all $j \neq k$

- The generalization of the within-class covariance is $\mathbf{S}_W = \sum_{k=1}^K \mathbf{S}_k$, where for $k \in \{0, 1, \dots, K - 1\}$

$$\mathbf{S}_k = \sum_{i:y_i=k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^\top, \quad \mathbf{m}_k = \frac{1}{m_k} \sum_{i:y_i=k} \mathbf{x}_i,$$

- The total covariance matrix

$$\mathbf{S}_T = \sum_{i=1}^m (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^\top,$$

where the mean of the total data set $\mathbf{m} = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$, and

$$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B, \mathbf{S}_B = \sum_{k=0}^{K-1} m_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^\top$$

- Let us consider *projected* features $\mathbf{z} = \mathbf{W}\mathbf{x}$. In this space we can analogously defined matrices

$$\mathbf{s}_W = \sum_{k=0}^{K-1} \sum_{i:y_i=k} (\mathbf{z}_i - \mu_k)(\mathbf{z}_i - \mu_k)^\top, \mathbf{s}_B = \sum_{k=0}^{K-1} m_k (\mu_k - \mu)(\mu_k - \mu)^\top,$$

where $\mu_k = \frac{1}{m_k} \sum_{i:y_i=k} \mathbf{z}_i$, $\mu = \frac{1}{m} \sum_{k=0}^{K-1} m_k \mu_k$

- We want to construct a scalar that is large when the between-class covariance is large, and when the within-class covariance is small
- One example is given by

$$J(\mathbf{W}) = \text{Tr} \left\{ \mathbf{s}_W^{-1} \mathbf{s}_B \right\} = \text{Tr} \left\{ (\mathbf{W} \mathbf{S}_W \mathbf{W}^\top)^{-1} (\mathbf{W} \mathbf{S}_B \mathbf{W})^\top \right\}$$

- The weight vectors are determined by those eigenvectors of $\mathbf{S}_W^{-1} \mathbf{S}_B$ that correspond to the largest eigenvalues