Machine Learning
Decision Trees, Neural Networks and nonparametric Methods

Prof. Matthias Hein

Machine Learning Group
Department of Mathematics and Computer Science
Saarland University, Saarbrücken, Germany

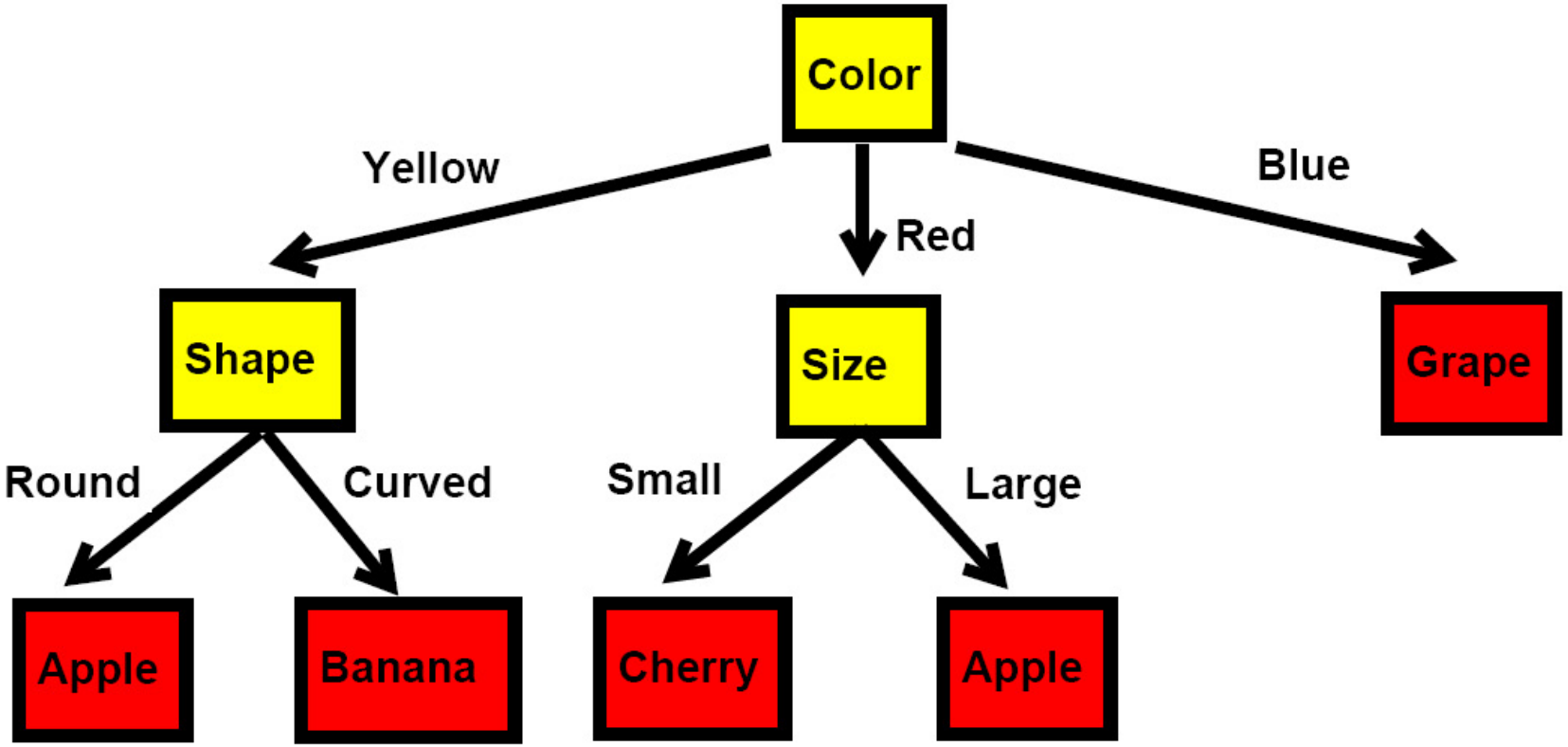
Lecture 18, 15.01.2014

Roadmap of today and wednesday

Classification methods:

- Boosting (how to get a good classifier from a set of simple ones)
- **Decision Trees**,
- **Neural Networks**,
- **Nearest Neighbor Methods, Parzen-Window**,

(Binary) Decision trees



Properties and questions:

- Designed for categorical features (but real-valued ones are possible),
- How to grow the tree, when to stop and how to prune the tree?

Decision trees II

Why binary trees ?

Any tree can be rewritten in terms of a binary tree.

What is the binary decision if we have a group of attributes ?

We partition the group of attributes into two sets.

What is the binary decision for real-valued features ?

A simple split of the coordinate.

How to classify a node ?

Let $p_N(k)$ be the fraction of training points at node N of class k.

$\Longrightarrow$ Classification by majority vote: $f(N) = \underset{k=1,\ldots,K}{\arg\max}\, p_N(k)$.

How to grow a decision tree ?

- Measures of **node impurity** $I(N)$ of node N,

$$\text{Entropy:} \qquad I(N) = -\sum_{k=1}^{K} p_N(k) \log p_N(k),$$

$$\text{Gini Index:} \qquad I(N) = \sum_{k \neq l} p_N(k) p_N(l),$$

$$\text{Zero one loss:} \qquad I(N) = 1 - \max_{k=1,\ldots,K} p_N(k).$$

- Determine for each feature the best split by minimizing

$$\frac{N_L}{N} I(N_L) + \frac{N_R}{N} I(N_R)$$

- Take the feature and the corresponding split with minimal impurity.

Decision trees IV

When to stop ?

- cross validation,
- minimal decrease in impurity or minimal number of training points in each node.

Alternative: grow the tree until each leaf is maximal pure, then prune it.

How to prune ?

- collapse successively the pair of leafs which leads to a minimal increase of the complexity criterion

$$\sum_{i=1}^{|T|} N_i \, I(N_i) + \alpha |T|.$$

- choose α by cross validation.

Many variants: ID3, C4.5, CART - works also for regression.

Decision trees V

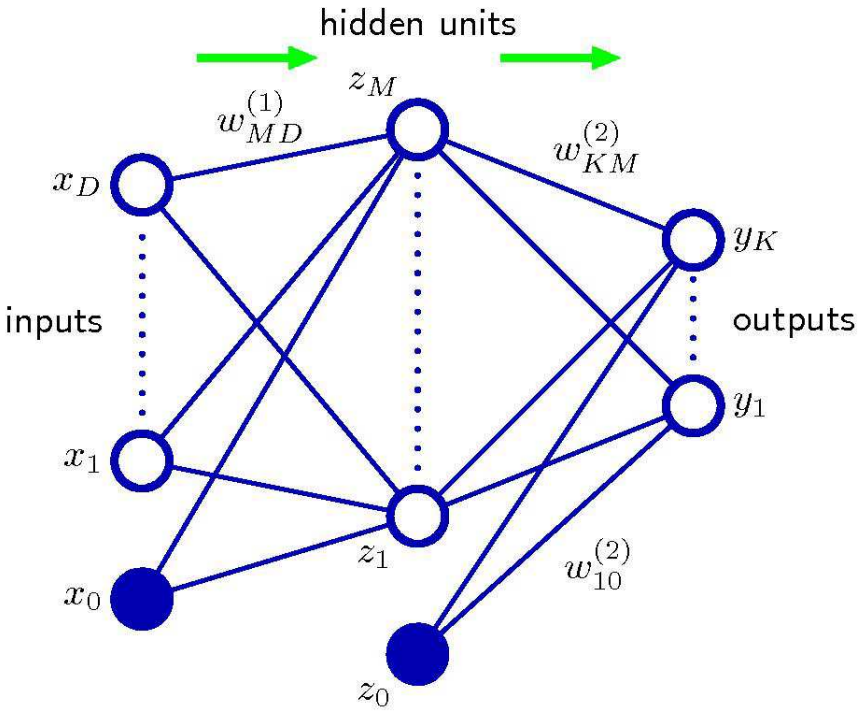
Pro

- if tree is small allows an easy interpretation (simple rules),
- very fast classifiers (real-time performance).

Contra

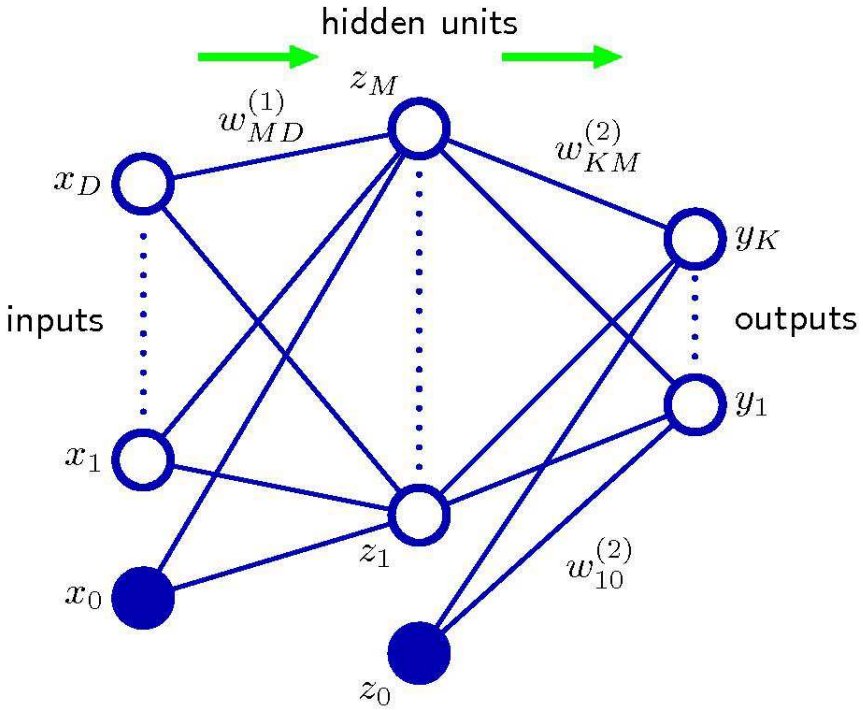
- often bad accuracy (the larger the tree (possibly better accuracy), the less interpretable),
- tree construction is quite unstable (greedy procedure),
- complex decision boundaries are difficult to model,
- forward/backward selection of features - no joint model.

What is a neural network ?



- **Input:** D features (real-valued),
- **Output:** K classes,

What is a neural network ? parameterized function model



- **Input:** D features (real-valued),
- **Output:** K classes,
- **Hidden layer:** M units.

$$f_k(x, w) = \sigma\Big(\sum_{j=1}^{M} w^{(2)}_{kj}\,\sigma\Big(\sum_{i=1}^{D} w^{(1)}_{ji} x_i + w^{(1)}_{j0}\Big) + w^{(2)}_{k0}\Big).$$

Parameters of a neural network

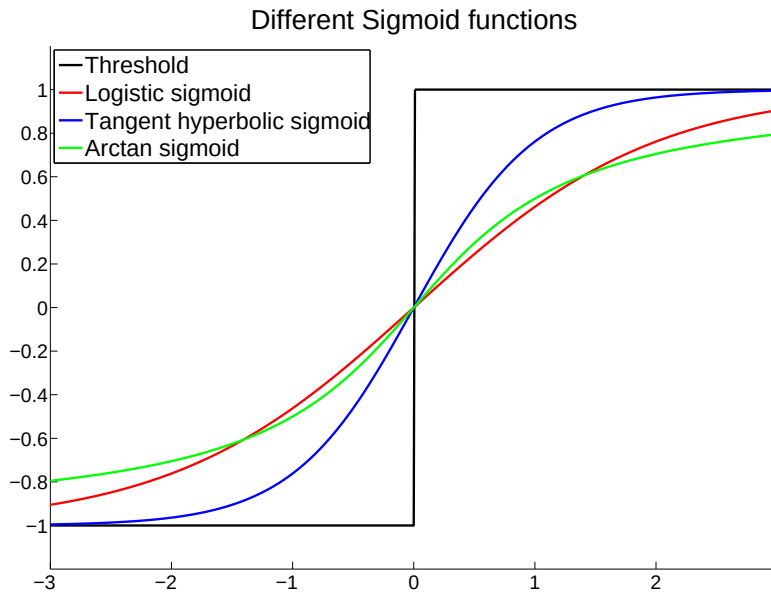
- Number of hidden layers L,
- Number of hidden units for each hidden layer M_l,
- choice of the sigmoid or activation function σ,

$$\text{threshold: } \sigma(x) = \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{if } x \leq 0. \end{cases}$$

$$\text{logistic sigmoid: } \quad \sigma(x) = \frac{1 - e^{-x}}{1 + e^{x}},$$

$$\text{tangent hyp.: } \quad \sigma(x) = \tanh(x),$$

$$\text{arctan sigmoid: } \quad \sigma(x) = \frac{2}{\pi} \arctan(x).$$



Objective function of a neural network

- **Common loss functions:** squared loss or so called cross-entropy,
- **Regularization:** penalty on the weights (often squared L_2-norm).

Objective $F(w)$ could look like this,

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} \|f(X_i, w) - Y_i\|^2 + \lambda \sum_{l=1}^{L}\sum_{j=1}^{M_l}\sum_{i=1}^{M_{l-1}} {w^{(l)}}_{ji}^2,$$

where the label of each training input Y_i is a k-dimensional vector,

$$Y_{ij} = 1, \text{ if } j \text{ is the true label of } X_i, \text{ and } Y_{ij} = -1 \text{ else.}$$

$\Longrightarrow$ **Complicated non-convex optimization problem !**

$$w_{ij}^l = w_{ij}^l + \gamma\, \Delta w_{ij}^l.$$

- gradient descent $\Longrightarrow$ backpropagation (chain rule to compute gradient),
- stepsize of descent step $\Longrightarrow$ learning rate γ.

Some theory for neural networks

Results for two classes: $K = 2$.

Theorem

Let $\mathcal{F}_k$ be the set of all functions which can be represented by a neural network with one hidden layer with k units and an arbitrary, monotonically increasing sigmoid function $\sigma(x)$. Then

$$\lim_{k \to \infty} \inf_{f \in \mathcal{F}_k} R(f) = R^*.$$

Conclusion:

- Neural networks can represent any function in the limit of infinite hidden units (same result holds for SVM with Gaussian kernel for $n \to \infty$),
- Is the function class used by neural networks better suited for problems in nature than kernel expansions ("deep versus shallow methods") ?
- Since the optimization is non-convex only convergence to local minima is guaranteed and theoretical analysis considers the global

Biological motivation of neural networks

Biology:

- Neurons cause other neurons to fire (**activation** of other neurons). All-or-none principle (either fires or not).
- Neural network tries to model sensory input, **but:** only V1 (visual cortex) is well understood, understanding of higher order activity is poor.
- Only in recent years multiple neurons are recorded in order to understand the interaction of neurons in the (monkey) brain.

Comments:

- It is very interesting to understand how the brain works !
- Neural networks are a very coarse approximation of the brain,
- Biological plausibility is o.k. if method has well-founded statistical and mathematical foundation.

$\Longrightarrow$ **Neural networks are nested parameterized function classes !**

Summary of neural networks

Pro

- heavily used in industry - currently again a hype topic
- has a well-founded theoretical underpinning independent of the biological motivation,
- dependent of the number of hidden layers and units can still be reasonably fast at test time.

Contra

- difficult to train (slow convergence and local minima) but recent improvements,
- design of network (number of hidden units, layers) requires expert knowledge,
- too many free parameters.
- generalization to non-linear input spaces seems difficult.

Nearest neighbor methods

What is a nearest neighbor method ?

Classify or estimate the function value of a test point based on the nearest neighbors in the training set.

Properties:

- one of the most simple and oldest classification method,
- despite its simplicity it often yields reasonable performance,
- no training required - testing is more expensive,
- well studied theory - many variants of such classifiers,
- very flexible - can be applied to any kind of data !

Nearest neighbor methods II

What is a nearest neighbor method ?

Let $X_{(1)}, \ldots, X_{(k)}$ be the k training points which have the smallest distance to the given test point x, $w(x)_{(i)}$ the associated positive weights and $Y_{(1)}, \ldots, Y_{(k)}$ their corresponding label.

- **Classification**

$$f(x) = \begin{cases} 1, & \text{if } \operatorname{sign}(\sum_{i=1}^{k} w(x)_i Y_i) > 0, \\ -1, & \text{else.} \end{cases}$$

Simple: $w(x)_i = 1 \Longrightarrow$ majority vote - use odd values of k to avoid ties.

- **Regression:**

$$f(x) = \frac{\sum_{i=1}^{k} w(x)_i Y_i}{\sum_{i=1}^{k} w(x)_i}.$$

$\Longrightarrow$ simple weighted average - but choice of the weights $w(x)_i$ and the number of neighbors can significantly influence the result.

Nearest neighbor method - Classification

Classification in Euclidean space

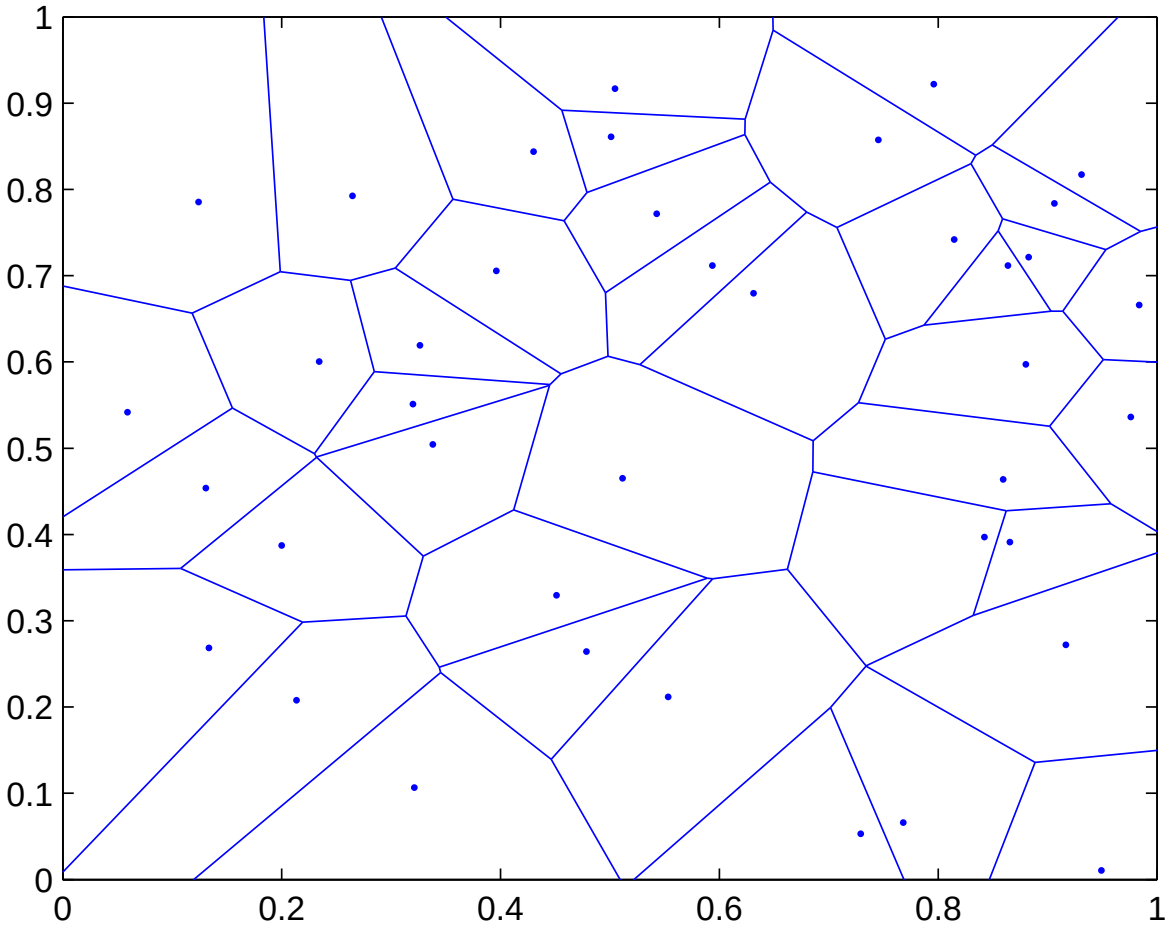
$$f(x) = \begin{cases} 1, & \text{if } \operatorname{sign}(\sum_{i=1}^{k} w(x)_i Y_i) > 0, \\ -1, & \text{else.} \end{cases}$$

Choices for the weights:

- Gaussian weights $w(x)_i = e^{-\lambda \left\| x - X_{(i)} \right\|^2} \Longrightarrow \lambda$ is determined by cross-validation (problems if high- and low density regions vary),
- Adaptive Gaussian weights $w(x)_i = e^{-\frac{\left\| x - X_{(i)} \right\|^2}{r_k^2}}$, where $r_k = \left\| x - X_{(k)} \right\|$ is the distance of the k-nearest neighbor.

Multi-class Extension: Classify test point by majority vote using the labels of the k nearest neighbors - break ties either randomly (no weights) or use weights for each point.

Nearest neighbor - Voronoi diagram



The Voronoi-diagram shows the influence region for each point corresponding to the nearest neighbor.

Theoretical results for nearest neighbor classification

Theorem

Let R_n be the classification error made by the k-nearest neighbor classifier in $\mathbb{R}^d$. Assume that X has a density with respect to the Lebesgue measure. If $k \to \infty$, $k/\log n \to \infty$ and $k/n \to 0$, then for every $\varepsilon > 0$ there exists an n_0 such that for $n \geq n_0$,

$$\mathrm{P}(R_n - R^* > \varepsilon) \leq 2\, e^{-\frac{n\varepsilon^2}{72\gamma_d^2}},$$

where γ_d is a constant depending only on the dimension.

Basic idea:

- as $k \to \infty$ and $k/n \to 0$, we have $r_k \to 0$.
- One averages over decreasing neighborhoods and since $k \to \infty$ the majority vote converges to $\arg\max_{m=1,\ldots,K} \mathrm{P}(Y = m | X = x)$.
- note that any rate for k faster than $\log n$ is sufficient to get convergence.

Theoretical results for nearest neighbor classification II

What happens when we keep k fixed and $n \to \infty$? One can compute the asymptotic error

$$R_{kNN} = \lim_{n \to \infty} R_n,$$

for a k nearest neighbor classifier with fixed k as

- $R_{1NN} = 2\,\mathbb{E}[\mathrm{P}(Y = 1|X)(1 - \mathrm{P}(Y = 1|X)],$
- $R_{3NN} = \mathbb{E}\Big[\mathrm{P}(Y = 1|X)\big[(1 - \mathrm{P}(Y = 1|X)) + 4(1 - \mathrm{P}(Y = 1|X))^2\big]\Big]$

Reminder: the Bayes error

$$R^* = \mathbb{E}_X[\min\{\mathrm{P}(Y = 1|X), \mathrm{P}(Y = -1|X)\}],$$

We have

$$\begin{aligned} R_{1NN} &= 2\,\mathbb{E}[\mathrm{P}(Y = 1|X)\mathrm{P}(Y = -1|X)] \\ &= 2\,\mathbb{E}[\min\{\mathrm{P}(Y = 1|X), \mathrm{P}(Y = -1|X)\}\max\{\mathrm{P}(Y = 1|X), \mathrm{P}(Y = -1| \\ &\leq 2\,\mathbb{E}[\min\{\mathrm{P}(Y = 1|X), 1 - \mathrm{P}(Y = 1|X)\}] = 2\,R^* \end{aligned}$$

Metric space

Metric spaces:

Definition

A **metric space** is a set $\mathcal{X}$ with a distance function $d : \mathcal{X} \times X \to \mathbb{R}$ such that:

- $d(x, y) \geq 0$,
- $d(x, y) = 0$ if and only if $x = y$,
- $d(x, y) = d(y, x)$, (symmetry)
- $d(x, y) \leq d(x, z) + d(z, y)$. (triangle inequality)

It is denoted as $(\mathcal{X}, d)$.

- We can define nearest neighbor classifier on any metric space !
- **More general:** we can define nearest neighbor classifier for any set with a similarity function (instead of nearest neighbors take **most similar points**).

Examples of distances

Examples of distances on $\mathbb{R}^d$:

- For $x, y \in \mathbb{R}^d$, use $d(x,y) = \|x-y\|_p = \left(\sum_{i=1}^d (x_i - y_i)^p\right)^{\frac{1}{p}}$, with the extreme case $p = \infty$,

$$d(x,y) = \|x-y\|_\infty = \max_{1 \leq i \leq d} |x_i - y_i|.$$

- Mahalanobis distance - a weighted Euclidean distance,

$$d(x,y) = \left(\sum_{i,j=1}^d A_{ij}(x_i - y_i)(x_j - y_j)\right)^{\frac{1}{2}} = \sqrt{\langle x-y, A(x-y)\rangle},$$

where A is a positive-definite matrix,

Distance on the sphere in $\mathbb{R}^d$:

$$d(x,y) = \arccos(\langle x, y\rangle).$$

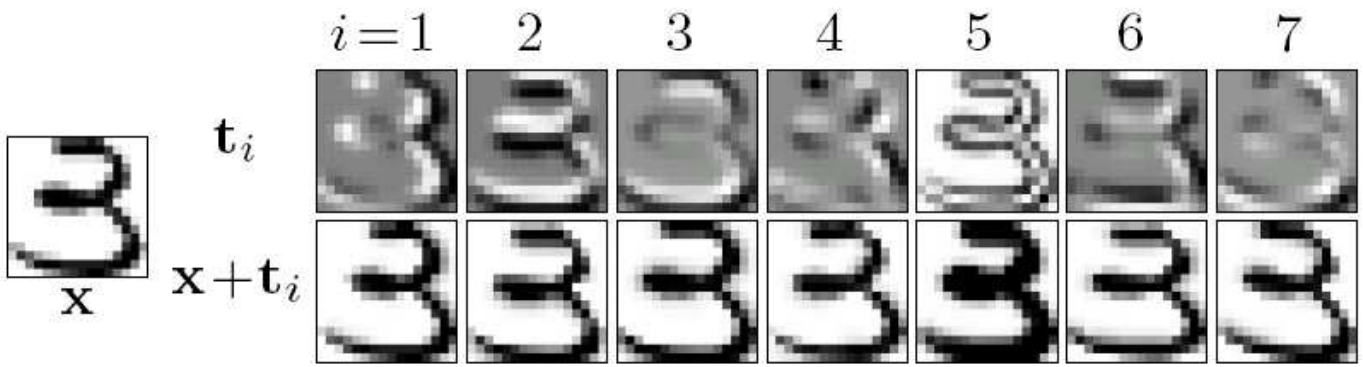
The famous cosine measure in text classification is a similarity measure !

Tangent distance

A dissimilarity measure designed for a particular application

small changes of digit images $\Longrightarrow$ label does not change !

but: Euclidean distance changes dramatically !



Degrees of freedom:

- **Geometric transformations:** 1+2) translation, 3) scaling, 4) rotation,
- **Application specific:** 5) line thickness, 6+7) shear.

Idea: build distance measure which is invariant under the transformations !

Tangent distance II

Definition of general tangent distance

Definition

Let x, y be two instances in $\mathcal{X}$ and $T(\alpha), T(\beta)$ be a group action G on $\mathcal{X}$,

$$T : \mathcal{X} \times \mathcal{G} \mapsto \mathcal{X}, \quad (x, \alpha) \to T(\alpha)x,$$

with which we want to be invariant. Then define the general tangent distance on $\mathcal{X}$ as,

$$d'(x, y) = \min_{\alpha,\beta \in G} d(T(\alpha)\, x, T(\beta)\, y),$$

where $d(x, y)$ is the original metric on $\mathcal{X}$.

- generally does not yield a metric (even if d is a metric !),
- the tangent distance minimizes usually only over group elements close to the identity (tangent elements),
- quite expensive to compute.

Nearest neighbor method - Regression

Regression:

$$f(x) = \frac{\sum_{i=1}^{k} w(x)_i Y_i}{\sum_{i=1}^{k} w(x)_i}.$$

For the specific choice of weights,

$$w(x)_i = k(\|x - X_i\| / h),$$

where $k : \mathbb{R}_+ \to \mathbb{R}$ satisfies

- $k(x)$ is monotonically decreasing,
- $k(x)$ is always positive,
- the number of neighbors k is equal to n.

then f is called the **Nadaraya-Watson estimator**,

$$f(x) = \frac{\sum_{i=1}^{n} k(\|x - X_i\| / h) Y_i}{\sum_{i=1}^{n} k(\|x - X_i\| / h)}.$$

Motivation of the Nadaraya-Watson estimator:

Proposition

The Nadaraya-Watson estimator $f(x)$ at x is the result of the following optimization problem,

$$f(x) = \arg\min_{c \in \mathbb{R}} \sum_{i=1}^{n} k(\|x - X_i\| / h)(Y_i - c)^2.$$

Proof: The Functional $F(c) = \sum_{i=1}^{n} k(\|x - X_i\| / h)(Y_i - c)^2$ is convex in c, and thus we find the minimizer by solving,

$$\frac{\partial F}{\partial c} = 2 \sum_{i=1}^{n} k(\|x - X_i\| / h)(Y_i - c) = 0.$$

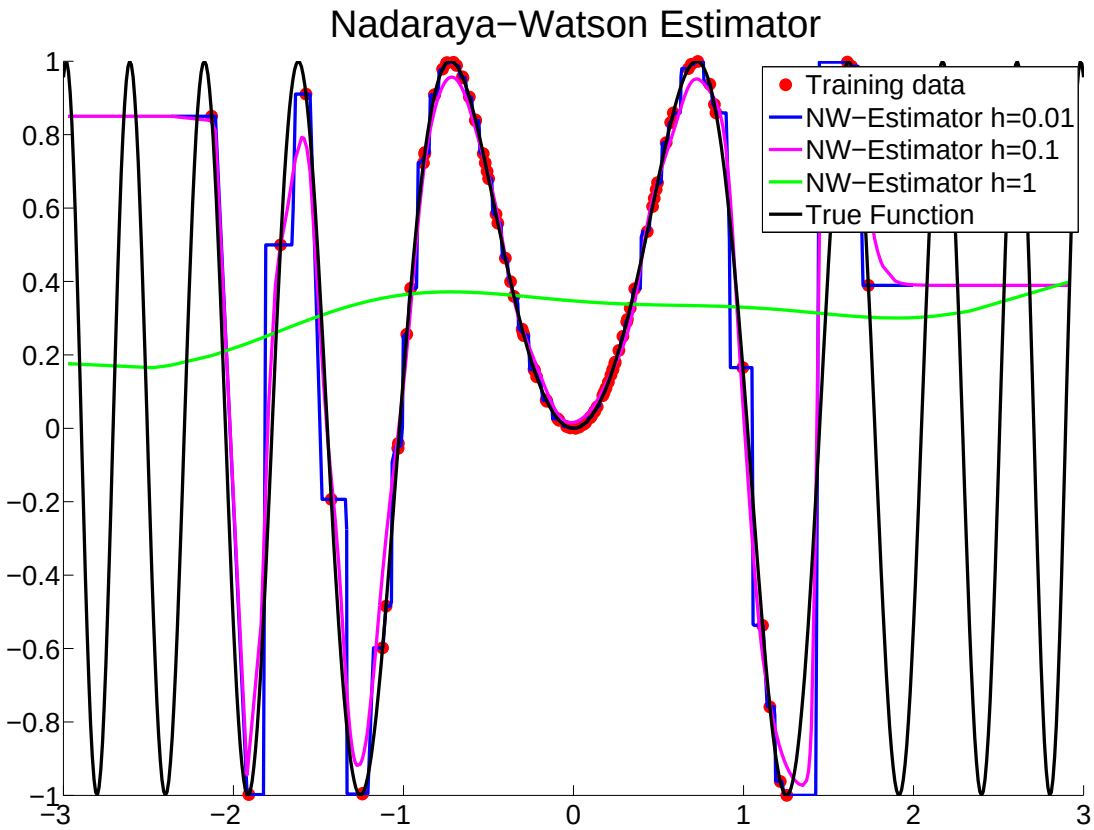
which yields,

$$c = \frac{\sum_{i=1}^{n} k(\|x - X_i\| / h) Y_i}{\sum_{i=1}^{n} k(\|x - X_i\| / h)}.$$

Parameters of the Nadaraya-Watson estimator:

h is the so called bandwidth and influences the smoothness of f,

$$f(x) = \frac{\sum_{i=1}^{n} k(\|x - X_i\| / h) Y_i}{\sum_{i=1}^{n} k(\|x - X_i\| / h)}.$$



Distances in high dimensions

Lemma

Let $x, y \in \mathbb{R}^d$ and $\epsilon_1, \epsilon_2 \sim N(0, \sigma^2)$ and define $X = x + \epsilon_1$ and $Y = y + \epsilon_2$, then

$$\mathbb{E}\,\|X - Y\|^2 = \|x - y\|^2 + 2\,d\,\sigma^2,$$
$$\mathrm{Var}\,\|X - Y\|^2 = 8\sigma^2\,\|x - y\|^2 + 8\,d\,\sigma^4.$$

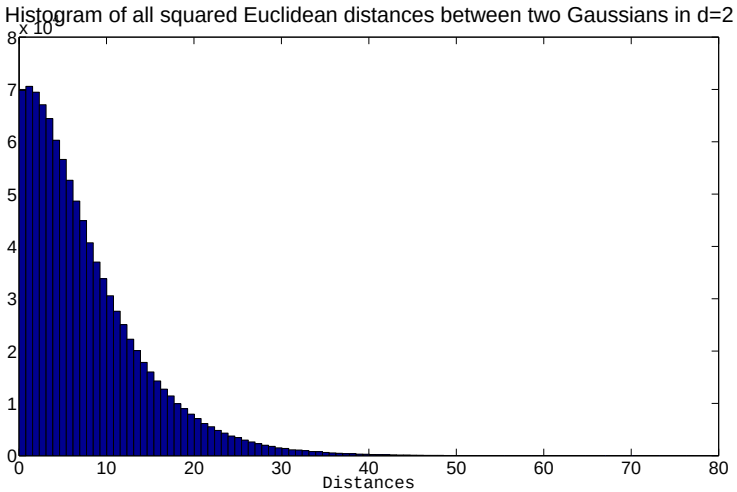
Distances in high dimensions

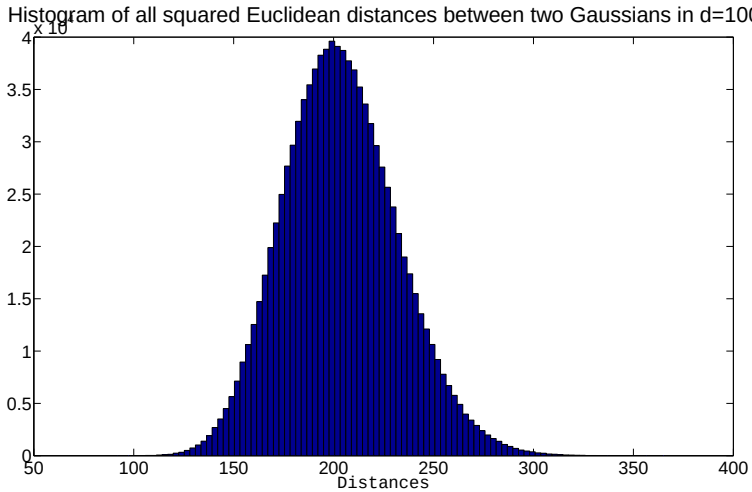
Lemma

Let $x, y \in \mathbb{R}^d$ *and* $\epsilon_1, \epsilon_2 \sim N(0, \sigma^2)$ *and define* $X = x + \epsilon_1$ *and* $Y = y + \epsilon_2$, *then*

$$\mathbb{E}\,\|X - Y\|^2 = \|x - y\|^2 + 2\,d\,\sigma^2,$$
$$\mathrm{Var}\,\|X - Y\|^2 = 8\sigma^2\,\|x - y\|^2 + 8\,d\,\sigma^4.$$

- Distances start to concentrate in high dimensions !
- All points have almost all the same distance.





Summary of nearest neighbor methods

Pro

- easy to understand,
- flexible, can be used with any user-specified similarity or distance,
- often competitive in performance,
- requires no training.

Contra

- Problems in high dimensions - distances are almost all equal,
- No interpretation,
- Slow at test time (but depends heavily on the dimension and the use of efficient data structures to compute the distances).