Data analysis: Statistical principals and computational methods

Discriminative Learning

Dmitrij Schlesinger, Carsten Rother

SS2014, 09.07.2014

Outline

- The Idea of discriminative learning – parameterized families of classifiers, non-statistical learning
- Linear classifiers, the Perceptron algorithm
- Feature spaces, "generalized" linear classifiers
- MAP in MRF is a linear classifier !!!
- A Perceptron like algorithm for discriminative learning of MRF-s

Discriminative statistical models

There exists a joint probability distribution $p(x, k; \theta)$ (observation, class; parameter). The task is to learn θ

On the other side (see the "Bayesian Decision theory"),

$$R(d) = \sum_k p(k|x; \theta) \cdot C(d, k)$$

i.e. only the posterior $p(k|x; \theta)$ is relevant for the recognition.

The Idea: decompose the joint probability distribution into

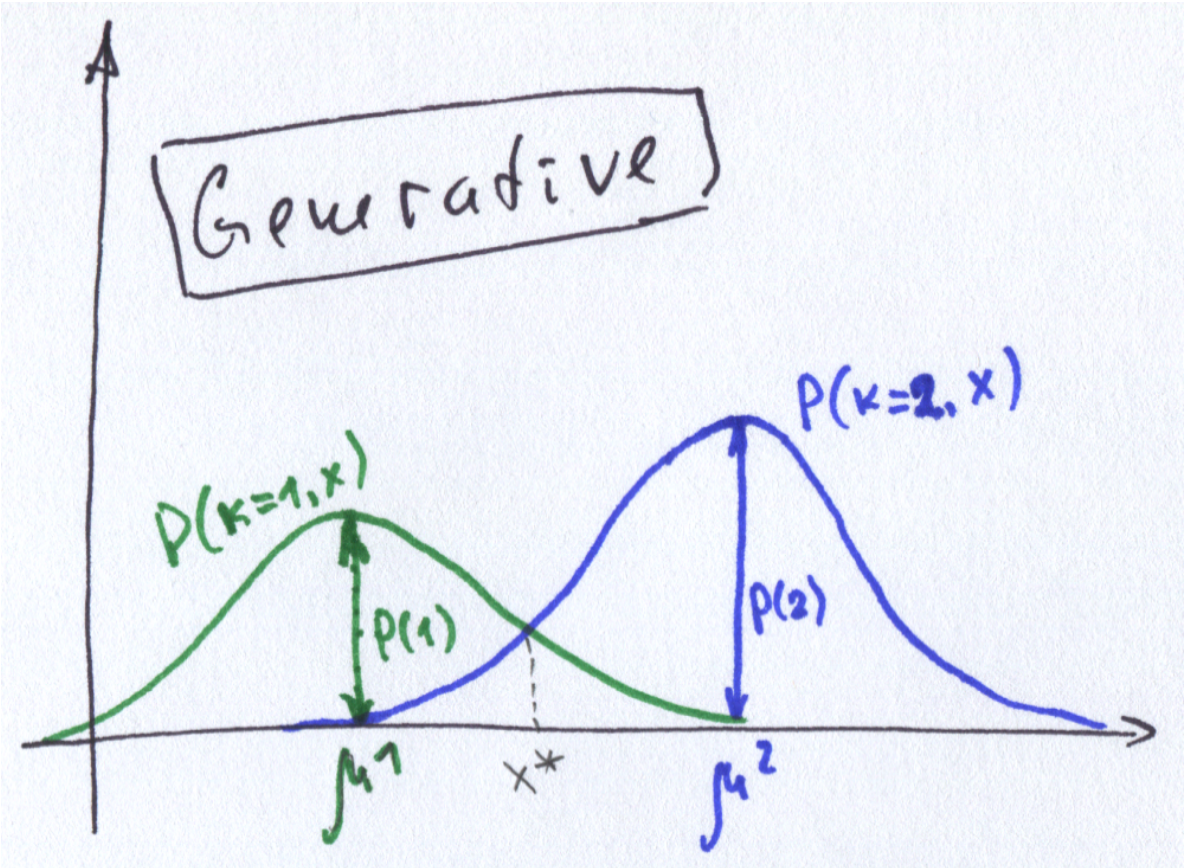
$$p(x, k; \theta) = p(x) \cdot p(k|x; \theta)$$

with an **arbitrary** $p(x)$ and a **parameterized posterior**.

$\rightarrow$ learn the parameters of the posterior p.d. directly

An example

Two Gaussians of equal variance, i.e. $k \in \{1, 2\}$, $x \in \mathbb{R}^n$,



$$p(x, k) = p(k) \cdot \frac{1}{(\sqrt{2\pi\sigma})^n} \exp\left[-\frac{\|x - \mu^k\|^2}{2\sigma^2}\right]$$

An example

Posterior:

$$p(k{=}1|x) = \frac{p(1)p(x|1)}{p(1)p(x|1) + p(2)p(x|2)} = \frac{1}{1 + \frac{p(2)p(x|2)}{p(1)p(x|1)}} =$$

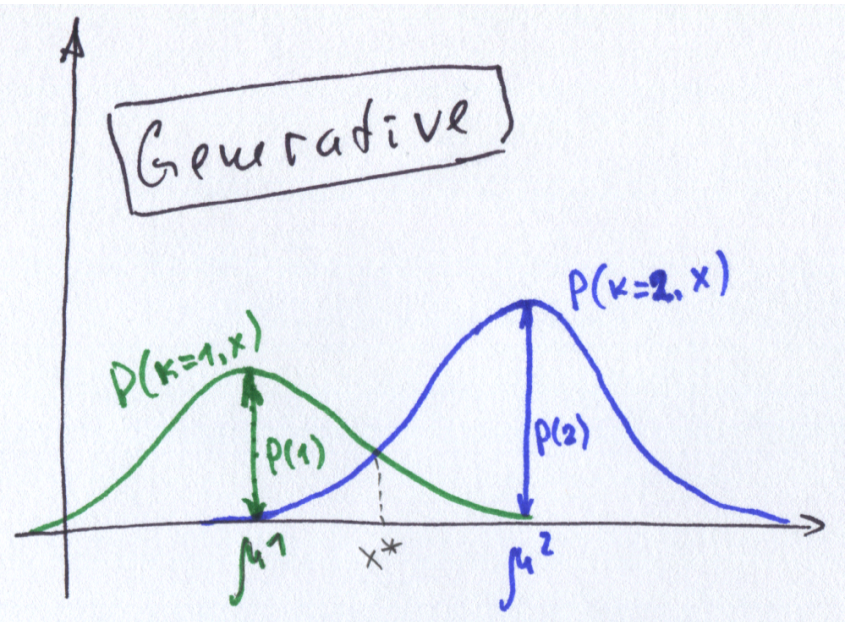
$$= \frac{1}{1 + \exp\left[-\frac{\|x-\mu^2\|^2}{2\sigma^2} + \frac{\|x-\mu^1\|^2}{2\sigma^2} + \ln p(2) - \ln p(1)\right]} =$$

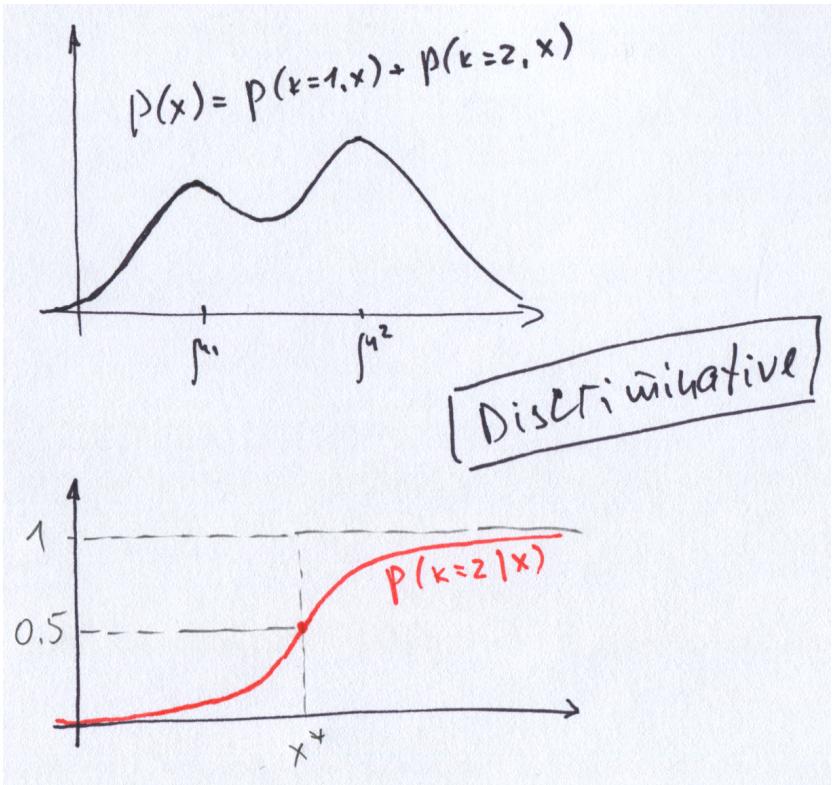
$$= \frac{1}{1 + \exp\Big(\langle x, w\rangle + b\Big)} \quad \text{with} \quad w = (\mu^2 - \mu^1)/\sigma^2$$

$$p(k{=}2|x) = 1 - p(k{=}1|x) = \frac{\exp\Big(\langle x, w\rangle + b\Big)}{1 + \exp\Big(\langle x, w\rangle + b\Big)}$$

Logistic regression model

An example





Discriminative statistical learning (statistical learning of discriminative models) – learn the posterior probability distribution from the training set directly

Generative vs. discriminative

Posterior p.d.-s have less free parameters as joint ones

Compare (for Gaussians):

- $2n + 2$ free parameters for the generative representation $p(k, x) = p(k) \cdot p(x|k)$, i.e. $p(1)$, σ, μ^1, μ^2
- $n + 1$ free parameters for the posterior $p(k|x)$, i.e. w and b

$\rightarrow$ one posterior corresponds to many joint p.d.-s

Gaussian example again:
centers μ^1 and μ^2 are not relevant, but their difference $\mu^2 - \mu^1$ (see the board for the explanation).

Discriminant functions

- Let a parameterized family of p.d.-s be given.
- If the loss-function is fixed, each p.d. leads to a classifier
- The final goal is the classification (applying the classifier)

Generative approach:

1. Learn the parameters of the p.d. (e.g. ML)
2. Derive the corresponding classifier (e.g. Bayes)
3. Apply the classifier for test data

Discriminative (non-statistical) approach:

1. Learn the unknown parameters of the classifier directly
2. Apply the classifier for test data

If the family of classifiers is "well parameterized", it is not necessary to consider the underlying p.d. at all !!!

Linear discriminant functions

As before: two Gaussians of the same variance, known prior
Now: let the loss function be δ so the decision strategy is MAP

Remember the posterior:

$$p(k{=}1|x) = \frac{1}{1 + \exp\Big(\langle x, w\rangle + b\Big)}$$

$\rightarrow$ the classifier is given by $\langle x, w\rangle \lessgtr b$

It defines a **hyperplane** orthogonal to w that is shifted from the origin by $b/||w||$

Note: for the classifier the variance σ is irrelevant
$\rightarrow$ the classifier has even less free parameters then the corresponding posterior

Classifiers vs. generative models

Families of classifiers are usually "simpler" compared to the corresponding families of probability distributions (lower dimensions, less restricted etc.)

Often it is not necessary to care about the model consistency (such as e.g. normalization) $\rightarrow$ algorithms become simpler.

It is possible to use more complex decision strategies, i.e. to reach better recognition results.

However:

Large classified training sets are usually necessary, unsupervised learning is not possible at all.

Worse generalization capabilities, overfitting.

Conclusion – a "hierarchy of abstraction"

1. **Generative models** (joint probability distributions) represent the **entire** "world". At the learning stage (ML) the **probability** of the training set is maximized.

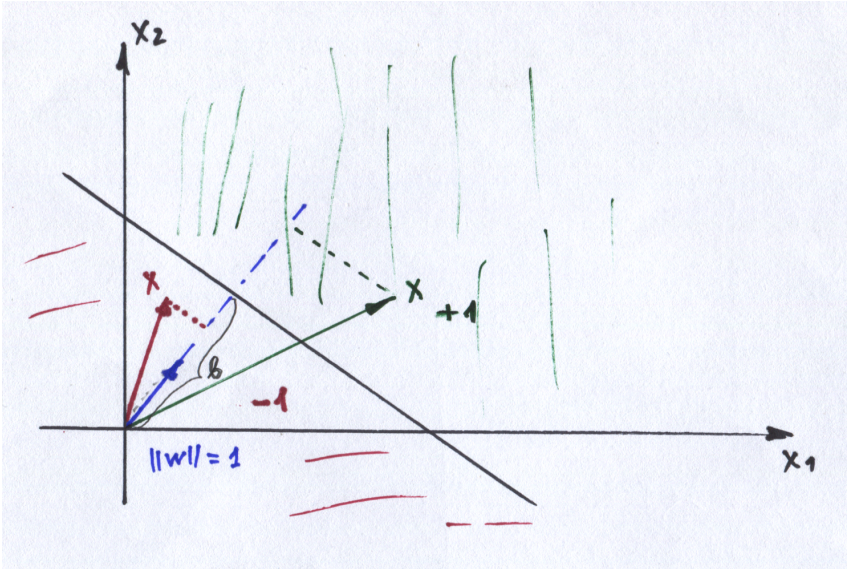
2. **Discriminative statistical models** represent posterior probability distributions, i.e. only what is needed for recognition. At the learning stage (ML) the **conditional likelihood** is maximized.

3. **Discriminant functions**: no probability distribution, decision strategy is learned directly.

   What should be optimized ?

Part 2: Linear classifiers

Building block for almost everything: a mapping $f : \mathbb{R}^n \to \{+1, -1\}$ – partitioning of the input space into two half-spaces that correspond to two classes



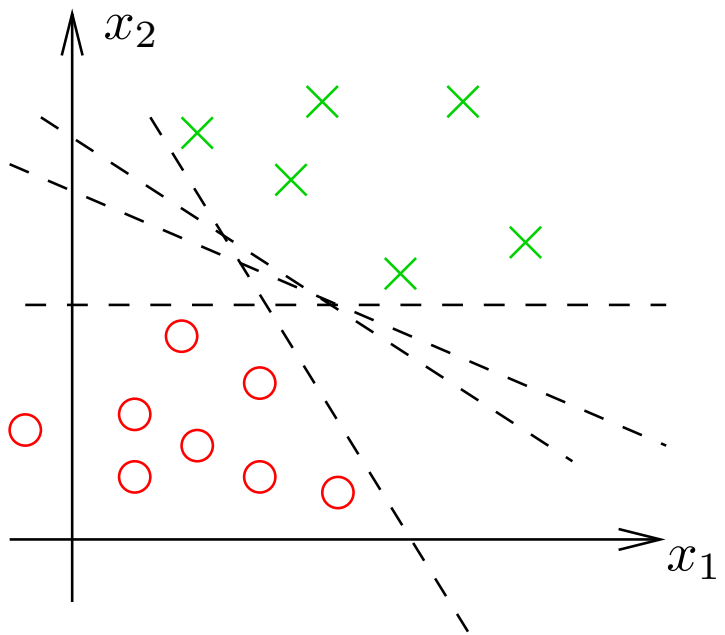
$$y = f(x) = \text{sgn}(\langle x, w \rangle - b)$$

with weights $w \in \mathbb{R}^n$ and a threshold $b \in \mathbb{R}$. Geometry: w is the normal of a hyperplane (given by $\langle x, w \rangle = b$) that separates the data. If $||w|| = 1$, the threshold b is the distance to the origin

The learning task

Let a training set $L = \Big((x^l, y^l) \ldots\Big)$ be given with
(i) data $x^l \in \mathbb{R}^n$ and (ii) classes $y^l \in \{-1, +1\}$
Find a hyperplane that separates data correctly, i.e.

$$y^l \cdot [\langle w, x^l \rangle + b] > 0 \quad \forall l$$



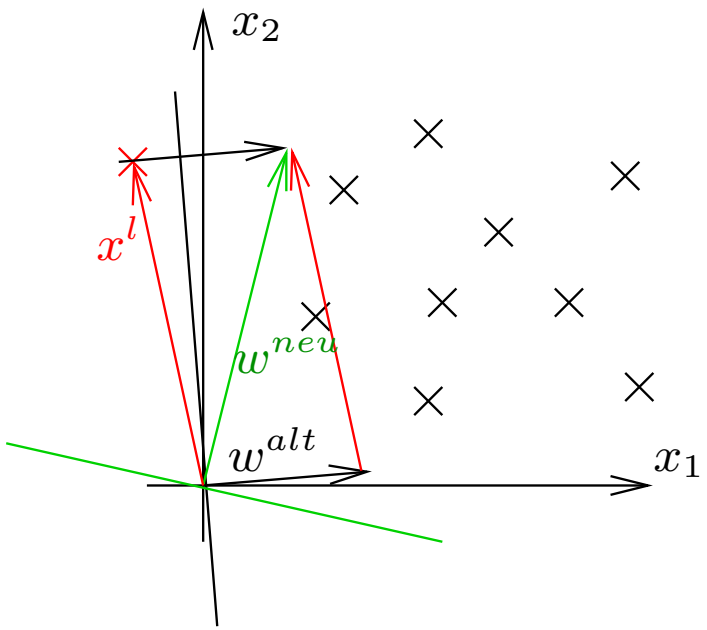
The task can be reduced to a system of linear inequalities:

$$\langle w, x^l \rangle > 0 \quad \forall l$$

The perceptron algorithm

1) Search for an inequality that is not satisfied, i.e. an l so that $\langle x^l, w \rangle \leq 0$ holds;
2) If not found – End,

   otherwise, update $w^{new} = w^{old} + x^l$, go to 1).



The algorithm terminates after a finite number of steps (!!!), if there exists a solution. Otherwise, it never finishes :-(

A generalization (example)

Consider the following classifier family for a scalar $x \in \mathbb{R}$

$$\begin{aligned} f(x) &= \text{sgn}(a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0) \\ &= \text{sgn}\Big(\sum_i a_i x^i\Big) \end{aligned}$$

i.e. a polynomial of n-th order.

The unknown coefficients a_i should be learned from a classified training set $\Big((x^l, y^l) \ldots\Big)$, $x^l \in \mathbb{R}$, $y^l \in \{-1, +1\}$.

Note: the classifier is not linear anymore.

Is it nevertheless possible to learn it in a perceptron-like fashion?

A generalization (example)

Key idea: although the decision rule is not linear with respect to the input x, it is still linear with respect to the unknowns a_i

$\rightarrow$ can be represented by the system of linear inequalities

$w = (a_n, a_{n-1}, \ldots, a_1, a_0)$
$\tilde{x} = (x^n, x^{n-1}, \ldots, x, 1)$ for each l (an $n+1$-dim. vector)

$\sum_i a_i x^i = \langle \tilde{x}, w \rangle \quad \Rightarrow$ perceptron task

In doing so the input space $\mathcal{X}$ is transformed into a feature (vector) space $\mathbb{R}^d$

A generalization

In general: many non-linear classifiers can be learned by the perceptron algorithm using an appropriate transformation of the input space (as e.g. in the example above, more examples at the exercises).

The "generalized" linear classifier:

$$f(x) = \text{sgn}\Big(\langle \phi(x), w \rangle\Big)$$

with an arbitrary (but fixed) mapping $\phi : \mathcal{X} \to \mathbb{R}^d$ and a parameter vector $w \in \mathbb{R}^d$

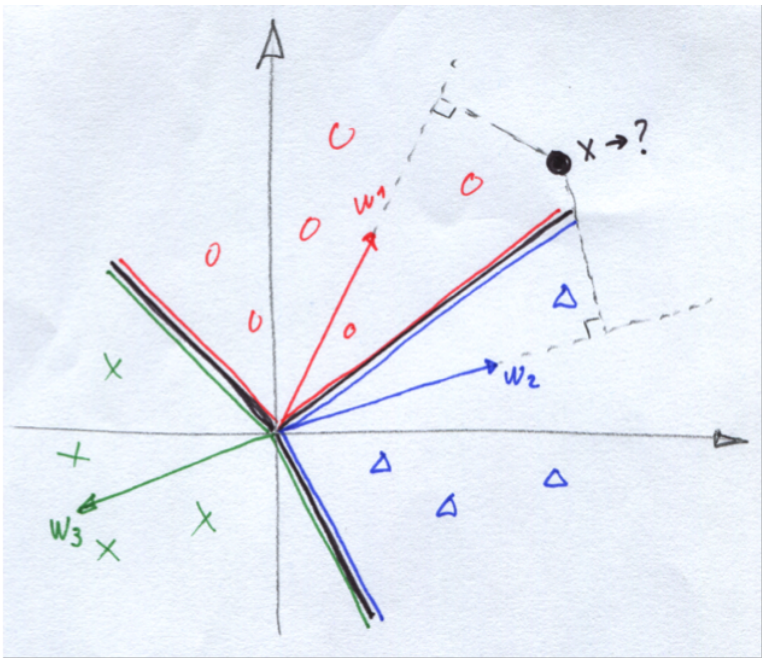
The parameter vector can be learnt by the perceptron algorithm, it the data are separable in the feature space

Note: in order to update the weights in the perceptron algorithm it is necessary to add $\phi(x)$ but not x

Multi-class perceptron

The problem: design a classifier family $f : \mathcal{X} \to \{1, 2 \dots K\}$ (i.e. for more than two classes)

Idea: in the binary case the output y is more likely to be "1" the greater is the scalar product $\langle x, w \rangle$



Fisher classifier:

$$y = f(x) = \arg\max_k \langle x, w_k \rangle$$

The input space is partitioned into the set of convex cones.

Multi-class perceptron, the learning task

Given: training set $\Big((x^l, k^l)\ldots\Big)$, $x \in \mathbb{R}^n$, $k \in \{1\ldots K\}$

To be learned: class specific vectors w_k.

They should be chosen in order to satisfy

$$y^l = \arg\max_k \langle x^l, w_k \rangle$$

It can be equivalently written as

$$\langle x^l, w_{y^l} \rangle > \langle x^l, w_k \rangle \quad \forall l, k \neq y^l$$

– system of linear inequalities

Multi-class perceptron algorithm

In fact it is the usual perceptron algorithm but again in an appropriately chosen feature space (exercise)

1) Search for an inequality that is not satisfied, i.e. a pair l, k so that $\langle x^l, w_{y^l} \rangle \leq \langle x^l, w_k \rangle$ holds;
2) If not found – End,

   otherwise, update

$$w_{y_l}^{new} = w_{y_l}^{old} + x^l$$
$$w_k^{new} = w_k^{old} - x^l$$

   go to 1).

All the stuff can be also done for "generalized" linear classifiers $\rightarrow$ "generalized" Fisher classifier

Part 3: Back to MRF-s

Remember the energy minimization, i.e. MAP for MRF-s:

$$y^* = \arg\min_y E(x, y) =$$

$$= \arg\min_y \left[\sum_{ij} \psi_{ij}(y_i, y_j) + \sum_i \psi_i(y_i, x_i) \right]$$

It can be understood as a "usual" classification task, where each labeling y is a class

---

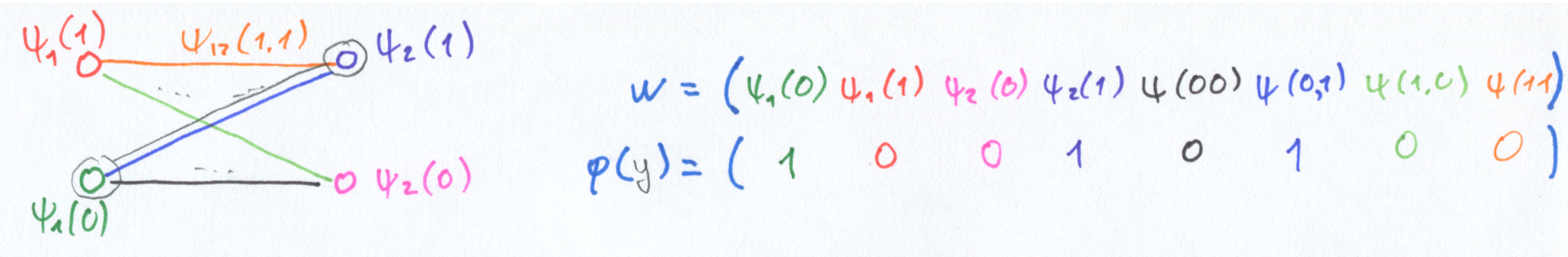
The learning task: given a training set $\left((x^l, y^l) \ldots\right)$, find the parameters ψ that satisfy

$$\arg\min_y \left[\sum_{ij} \psi_{ij}(y_i, y_j) + \sum_i \psi_i(y_i, x_i^l) \right] = y^l \quad \forall l$$

The goal is now to express the energy as a scalar product

MAP in MRF is a linear classifier

Example for non-hidden MRF (y only), for $E(x, y)$ – similar



In the parameter vector $w \in \mathbb{R}^d$ there is a component for each ψ-value of the task, i.e. tuples (i, k) or (i, j, k, k')

$\phi(y)$ is composed of "indicator" values that are 1 if the corresponding value of ψ "is contained" in the energy $E(y)$

$$\phi_{ijkk'}(y) = \begin{cases} 1 & \text{if } y_i = k, y_j = k' \\ 0 & \text{otherwise} \end{cases}$$

$\rightarrow$ the energy of a labeling can be written as scalar product

$$E(y; w) = \langle \phi(y), w \rangle$$

Multi-class perceptron + Energy Minimization

1. Search for an inequality that is not satisfied so far, i.e. for an example l and a labeling y so that

$$E(x^l, y^l) > E(x^l, y)$$

Note: it is not necessary to try all y in order to find it. It is enough to pick just one, for instance solving

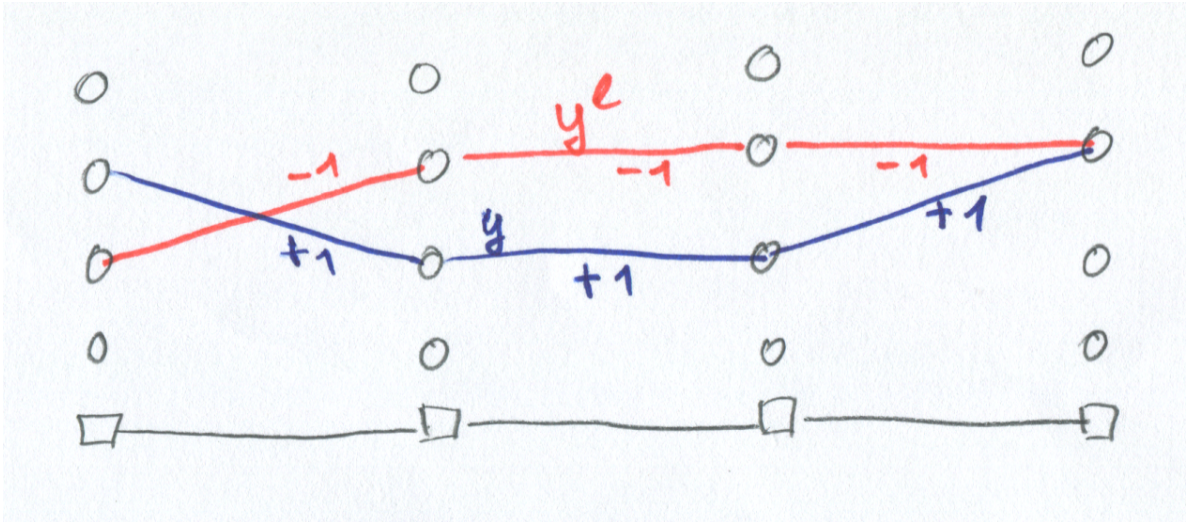
$$y = \arg\min_{y'} E(x^l, y')$$

2. If not found – End;

otherwise, update ψ by the corresponding ϕ-vectors

$$\psi_{ij}^{new}(k,k') = \begin{cases} \psi_{ij}^{old}(k,k') - 1 & \text{if } y_i^l = k, y_j^l = k' \\ \psi_{ij}^{old}(k,k') + 1 & \text{if } y_i = k, y_j = k' \\ \psi_{ij}^{old}(k,k') & \text{otherwise} \end{cases}$$

Multi-class perceptron + Energy Minimization



Further topics:

- How to obtain the "best" possible classifier ?

  $\rightarrow$ SVM
- What to do if the data are not separable ?

  $\rightarrow$ loss-based learning, empirical risk ...
- Technical issues ...