Pattern Recognition

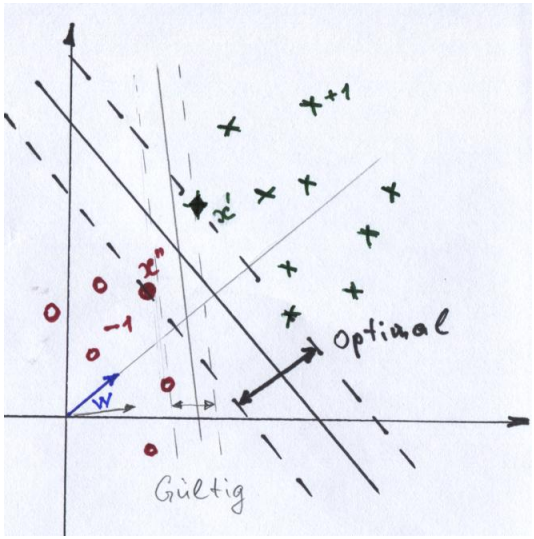
Hinge Loss

Recap – tasks considered before

Let a training dataset $X = \big((x_i, y_i) \ldots\big)$ be given with
(i) data $x_i \in \mathbb{R}^n$ and (ii) classes $y_i \in \{-1, +1\}$
The goal is to find a hyper plane that separates the data

$$y_i \cdot [\langle w, x_i \rangle + b] \geq 0 \quad \forall i$$

→ Perceptron algorithm



The goal is to find a "corridor" (stripe) of the maximal width that separates the data

→ **Large margin learning**, linear SVM

$$\frac{1}{2}||w||^2 \rightarrow \min_w$$
$$\text{s.t.} \quad y_i[\langle x_l, w \rangle + b] \geq 1$$

In both cases the training set is assumed to be **separable**.
What if **not**?

Empirical risk minimization

Let (in addition to the training data) a loss function $C(y, y')$ be given that penalizes deviations between the true class and the estimated one (the same as the cost function in the Bayesian decision theory).
The **Empirical Risk** of a decision strategy is the total loss:

$$R(e) = \sum_l C\big(y_l, e(x_l)\big) \to \min_e$$

It should be minimized with respect to the decision strategy e.

Special case (today):

- the set of decisions is $\{-1, +1\}$, i.e. the set of classes
- the loss is the (simplest) delta-function $C(y, y') = \delta(y \neq y')$
- the decision strategy can be expressed in the form

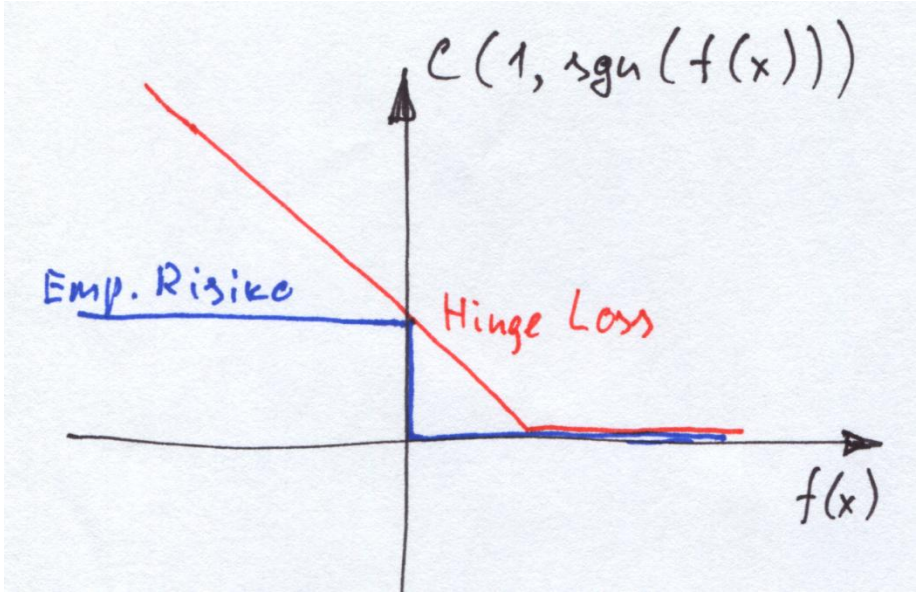
  $$e(x) = \operatorname{sign}\big(f(x)\big)$$

  with an **evaluation function** $f : X \to \mathbb{R}$
  Example: $f(x) = \langle x, w \rangle - b$ is a linear classifier.

Hinge Loss

The problem: the subject is not convex

The way out: replace the real loss by its **convex upper bound**



← example for $y = 1$

(for $y = -1$ it should be flipped)

$$\delta\Big(y \neq \operatorname{sign}\big(f(x)\big)\Big) \leq \max\big(0, 1 - y \cdot f(x)\big)$$

It is called **Hinge Loss**

Sub-gradient algorithm

Let the evaluation function be parameterized, i.e. $f(x;\theta)$, for example $\theta = (w, b)$ for linear classifiers.

The optimization problem is

$$H(\theta) = \sum_l \max\bigl(0, 1 - y_l \cdot f(x_l;\theta)\bigr) \to \min_\theta$$

It is convex with respect to f but **non-differentiable**.

Solution by the **sub-gradient** (descent) algorithm:

1. Compute the sub-gradient (later)
2. Apply it with a step size that is **decreasing in time**

$$\theta^{(t+1)} = \theta^{(t)} - \gamma(t) \cdot \frac{\partial H}{\partial \theta}$$

with $\lim_{t\to\infty} \gamma(t) = 0$ and $\sum_{t=1}^{\infty} \gamma(t) = \infty$ (e.g. $\gamma(t) = 1/t$)

Sub-gradient algorithm

Remember on the task of interest:

$$H(\theta) = \sum_l \max\big(0, 1 - y_l \cdot f(x_l; \theta)\big) \to \min_\theta$$

Computation of the sub-gradient for the Hinge Loss:

1. Estimate data points for which the Hinge Loss grater zero

$$L' = \{l : y_l \cdot f(x_l) < 1\}$$

2. The sub-gradient is

$$\frac{\partial H}{\partial \theta} = -\sum_{l \in L'} y_l \cdot \frac{\partial f(x_l; \theta)}{\partial \theta}$$

In particular, for linear classifiers

$$\frac{\partial H}{\partial w} = -\sum_{l \in L'} y_l \cdot x_l$$

i.e. some data points are added (weighted) to the parameter vector

→ it reminds on the Perceptron algorithm

Kernelization

Remember: the evaluation function can be expressed as

$$f(x;\alpha) = \langle \Phi(x), w \rangle = \langle \Phi(x), \sum_i \alpha_i y_i \Phi(x_i) \rangle =$$

$$\sum_i \alpha_i y_i \langle \Phi(x), \Phi(x_i) \rangle = \sum_i \alpha_i y_i \kappa(x, x_i)$$

The subject to be minimized is

$$H(\alpha) = \sum_l \max\bigl(0, 1 - y_l \sum_i \alpha_i y_i \kappa(x_l, x_i)\bigr) \to \min_\alpha$$

and the sub-gradient is

$$\frac{\partial H}{\partial \alpha_i} = -y_i \sum_{l \in L'} y_l \kappa(x_l, x_i)$$

As usual for Kernels, neither the feature space nor the mapping $\Phi(x)$ are necessary in order to estimate α, if the kernel $\kappa(x, x')$ is given.

Maximum margin vs. minimum loss

- Linear SVM – maximum margin learning, separable data
- Non-separable data – Empirical Risk Minimization, Hinge Loss
- “Kernelization” can be performed for both variants

Does it have sense to minimize the loss defined in the feature space?

It is indeed always possible to make the training set separable by choosing a suitable kernel.

Interesting – both formulations are equivalent in certain circumstances.

Maximum margin vs. minimum loss

In Machine Learning it is a common technique to enhance an objective function (e.g. the average loss) by a **regularizer**

A "unified" formulation:

$$\mathcal{R}(w) + \frac{C}{|L|} \sum_l \ell(x_l, y_l, w) \to \min_w$$

with

- parameter vector w
- loss $\ell(x_l, y_l, w)$ – e.g. delta, hinge, metric, additive etc.
- regularizer $\mathcal{R}(w)$ – e.g. $||w||^2$, $|w|_1$ etc.
- balancing factor C

Maximum margin vs. minimum loss

$$\mathcal{R}(w) + \frac{C}{|L|} \sum_l \ell(x_l, y_l, w) \to \min_w$$

Assumption: the training set is separable, i.e. the average loss is zero

Set C to a very high value, the above formulation can be written as

$$\mathcal{R}(w) \to \min_w$$
$$\text{s.t.} \quad \ell(x_l, y_l, w) = 0 \quad \forall l$$

Set $\mathcal{R}(w) = 1/2||w||^2$ and ℓ to the Hinge loss for linear classifiers, i.e.

$$\ell(x_l, y_l, w) = \max(0, 1 - y_l \langle x_l, w \rangle)$$

We obtain just the maximum margin learning

$$\frac{1}{2}||w||^2 \to \min_w$$
$$\text{s.t.} \quad y_l \langle x_l, w \rangle \geq 1$$

The last slide ☺

Recommended reading:

Sebastian Nowozin and Christoph H. Lampert,
"Structured Prediction and Learning in Computer Vision“,
Foundations and Trends in Computer Graphics and Vision,
Volume 6, Number 3-4

http://www.nowozin.net/sebastian/cvpr2012tutorial/

Next time: AdaBoost – how to combine simple (bad, weak) classifiers in order to obtain a complex (good, strong) one.