Learning and Linear Classifiers

lecturer: Jiří Matas, matas@cmp.felk.cvut.cz

authors: V. Hlaváč, J. Matas, O. Drbohlav

Czech Technical University, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
121 35 Praha 2, Karlovo nám. 13, Czech Republic

http://cmp.felk.cvut.cz

6/Nov/2015

Last update: 6/Nov/2015, 11am

LECTURE PLAN

- The problem of classifier design.
- Learning in pattern recognition.
- Linear classifiers.
- Perceptron algorithms.
- Optimal separating plane with the Kozinec algorithm.

Classifier Design (1)

The object of interest is characterised by observable properties $x \in X$ and its class membership (unobservable, hidden state) $k \in K$, where X is the space of observations and K the set of hidden states.

The objective of classifier design is to find a strategy $q^*\colon X \to K$ that has some optimal properties.

Bayesian decision theory solves the problem of minimisation of risk

$$R(q) = \sum_{x,k} W(q(x),k)\, p(x,k)$$

given the following quantities:

- $p(x,k), \forall x \in X, k \in K$ – the statistical model of the dependence of the observable properties (measurements) on class membership
- $W(q(x),k)$ the loss of decision $q(x)$ if the true class is k

Classifier Design (2)

Non-Bayesian decision theory solves the problem if $p(x|k), \forall x \in X, k \in K$ are known, but $p(k)$ are unknown (or do not exist). Constraints or preferences for different errors depend on the problem formulation.

However, in applications typically:

- none of the probabilities are known! The designer is only given a training multiset $T = \{(x_1, k_1) \ldots (x_L, k_L)\}$, where L is the length (size) of the training multiset.
- the desired properties of the classifier $q(x)$ are known

Classifier Design via Parameter Estimation

- Assume $p(x, k)$ have a particular form, e.g. Gaussian (mixture), piece-wise constant, etc., with a finite (i.e. small) number of parameters Θ_k.
- Estimate the parameters from the using training set T
- Solve the classifier design problem (e.g. risk minimisation), substituting the estimated $\hat{p}(x, k)$ for the true (and unknown) probabilities $p(x, k)$

? : What estimation principle should be used?

– : There is no direct relationship between known properties of estimated $\hat{p}(x, k)$ and the properties (typically the risk) of the obtained classifier $q'(x)$

– : If the true $p(x, k)$ is not of the assumed form, $q'(x)$ may be arbitrarily bad, even if the size of training set L approaches infinity!

\+ : Implementation is often straightforward, especially if parameters Θ_k for each class are assumed independent.

\+ : Performance on training data can be predicted by crossvalidation.

Learning in Statistical Pattern Recognition

- Choose a class Q of decision functions (classifiers) $q : X \to K$.
- Find $q^* \in Q$ minimising some criterion function on the training set that approximates the risk $R(q)$ (true risk is uknown).
- Objective functions:

  Empirical risk (training set error) minimization. True risk approximated by

$$R_{emp}(q_\Theta(x)) = \frac{1}{L} \sum_{i=1}^{L} W(q_\Theta(x_i), k_i) ,$$

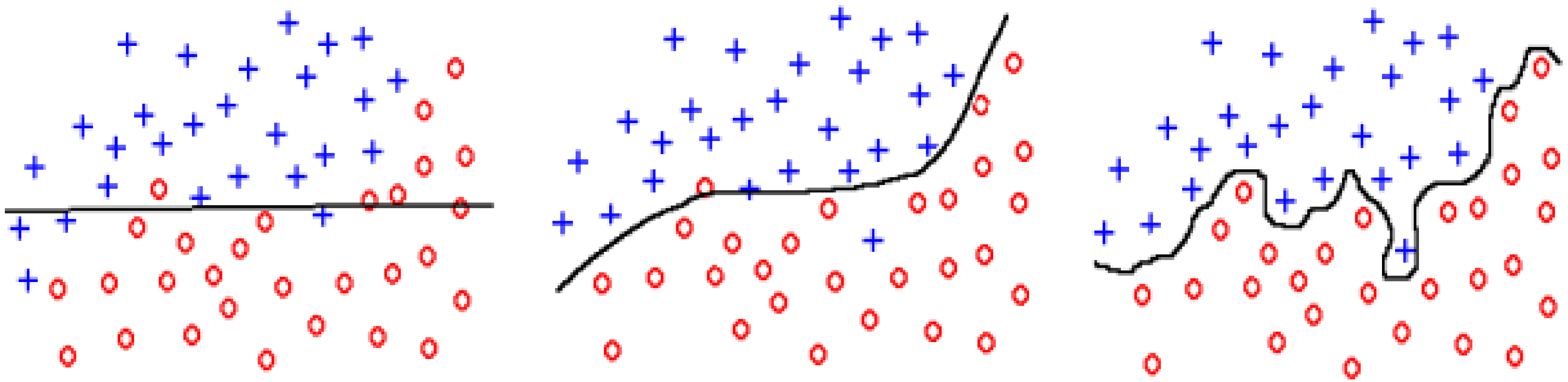
$$\Theta^* = \operatorname*{argmin}_{\Theta} R_{emp}(q_\Theta(x))$$

  Examples: Perceptron, Neural nets (Back-propagation), etc.

  Structural risk minimization.
  Example: SVM (Support Vector Machines).

Overfitting and Underfitting

- How rich class Q of classifiers $q_\Theta(x)$ should be used?
- The problem of generalization is a key problem of pattern recognition: a small empirical risk R_{emp} need not imply a small true expected risk R!



underfit fit overfit

As discussed previously, a suitable model can be selected e.g. using cross-validation.

Structural Risk Minimization Principle (1)

We would like to minimise the risk

$$R(q) = \sum_{x,k} W(q_\Theta(x), k)\, p(x,k)$$

but $p(x,k)$ is unknown.

Vapnik and Chervonenkis proved a remarkable inequality

$$R(q) \leq R_{emp}(q) + R_{str}\left(h, \frac{1}{L}\right) ,$$

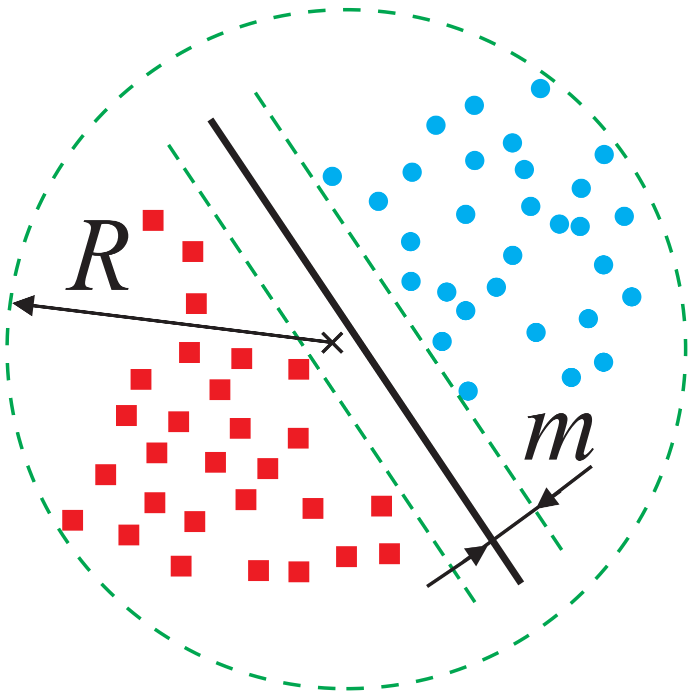
where h is VC dimension (capacity) of the class of strategies Q.

Notes:

+ R_{str} does not depend on the unknown $p(x,k)$
+ R_{str} known for some classes of Q, e.g. linear classifiers.

Structural Risk Minimization Principle (2)

- There are more types of upper bounds on R.
  E.g. for linear discriminant functions



VC dimension (capacity)

$$h \leq \frac{R^2}{m^2} + 1$$

- Examples of learning algorithms: SVM or ε-Kozinec.

$$(w^*, b^*) = \underset{w,b}{\operatorname{argmax}} \min \left(\min_{x \in X_1} \frac{\langle w, x \rangle + b}{|w|}, \min_{x \in X_2} \frac{\langle w, x \rangle + b}{|w|} \right) .$$

Empirical Risk Minimisation, Notes

Is then empirical risk minimisation = minimisation of training set error, e.g. neural networks with backpropagation, useless? No, because:

– R_{str} may be so large that the upper bound is useless.

+ Vapnik's theory justifies using empirical risk minimisation on classes of functions with VC dimension.

+ Vapnik suggests learning with progressively more complex classes Q.

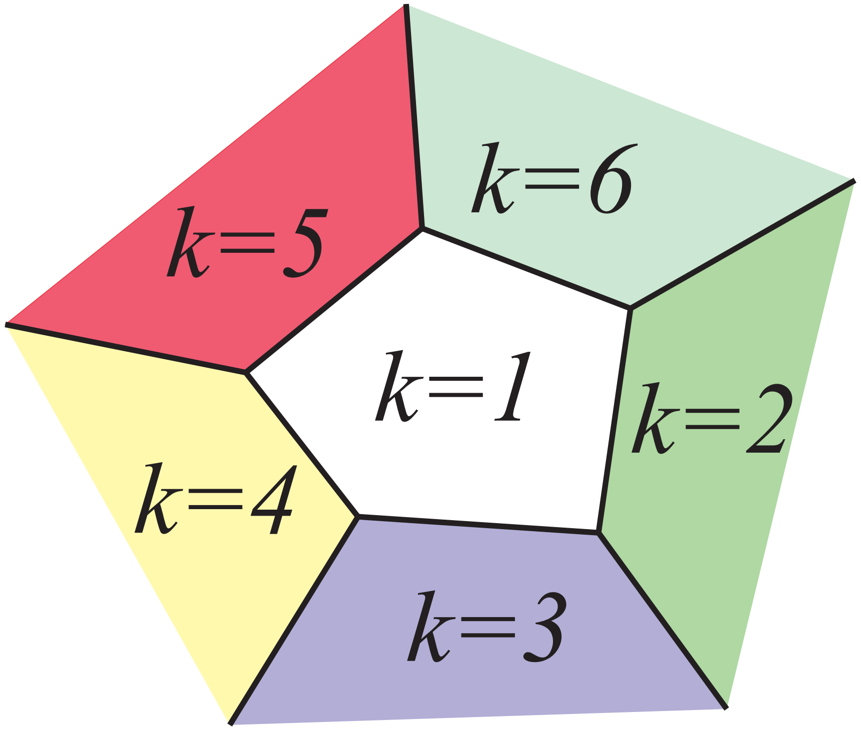
+ Empirical risk minimisation is computationally hard (impossible for large L). Most classes of decision functions Q where empirical risk minimisation (at least local) can be effeciently organised are often useful.

Linear Classifiers

- For some statistical models, the Bayesian or non-Bayesian strategy is implemented by a linear discriminant function.
- Capacity (VC dimension) of linear strategies in an n-dimensional space is $n + 2$. Thus, the learning task is well-posed, i.e., strategy tuned on a finite training multiset does not differ much from correct strategy found for a statistical model.
- There are efficient learning algorithms for linear classifiers.
- Some non-linear discriminant functions can be implemented as linear after the feature space transformation.

Linear Discriminant Function

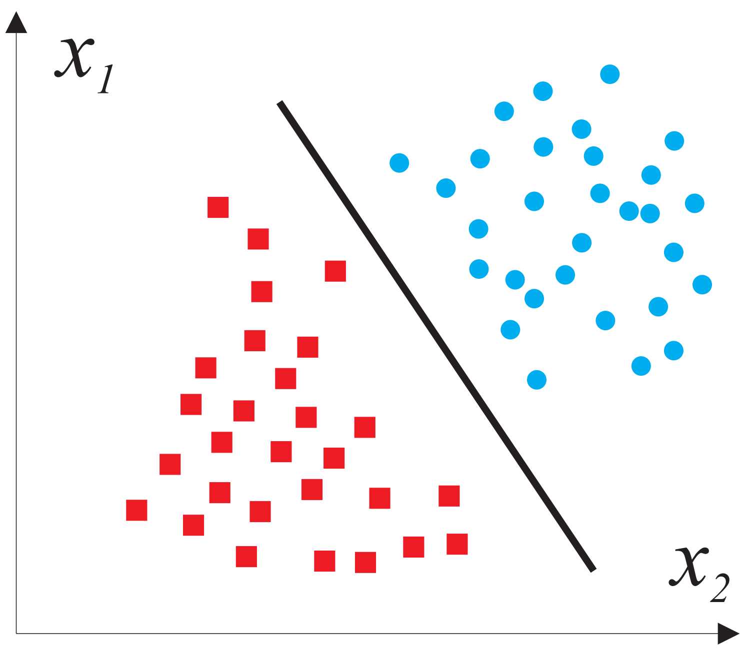
- $f_j(x) = \langle w_j, x \rangle + b_j$, where $\langle \ \rangle$ denotes a scalar product.
- A strategy $j = \operatorname*{argmax}_j f_j(x)$ divides X into $|K|$ convex regions.

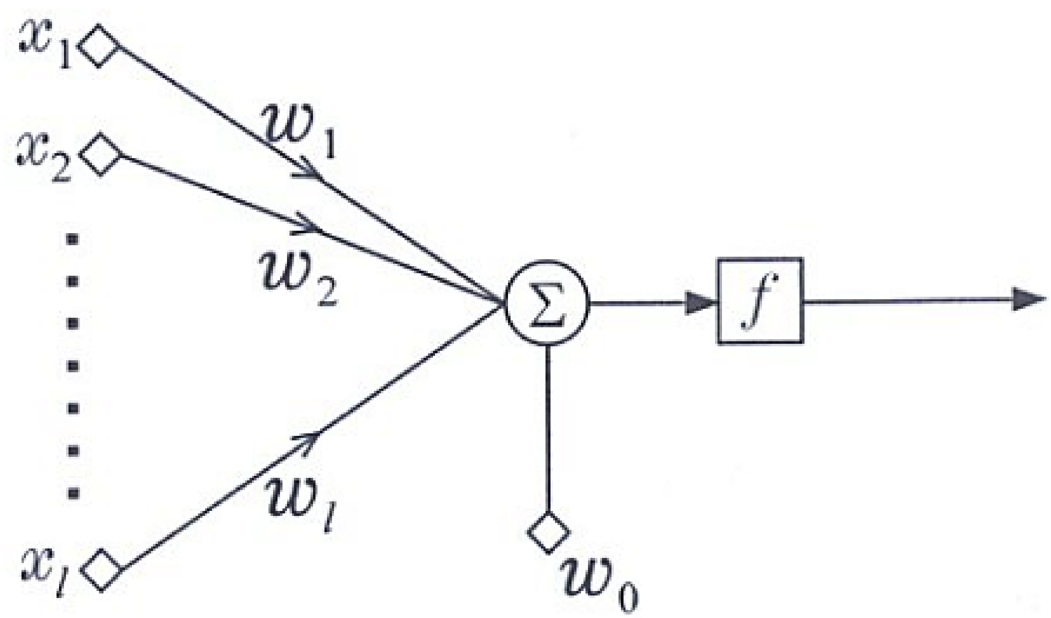


Dichotomy, Two Classes Only

$|K| = 2$, i.e. two hidden states (typically also classes)

$$q(x) = \begin{cases} k = 1\,, & \text{if} \quad \langle w, x \rangle + b \geq 0\,, \\ k = -1\,, & \text{if} \quad \langle w, x \rangle + b < 0\,. \end{cases}$$





Perceptron Classifier

Input: $T = \{(x_1, k_1) \dots (x_L, k_L)\}, k \in \{-1, 1\}$

Goal: Find a weight vector w and offset b such that :

$$\begin{aligned} \langle w, x_j \rangle + b > 0 \quad \text{if} \quad k_j = 1\,, \qquad (\forall j \in \{1, 2, ..., L\}) \\ \langle w, x_j \rangle + b < 0 \quad \text{if} \quad k_j = -1 \end{aligned} \tag{1}$$

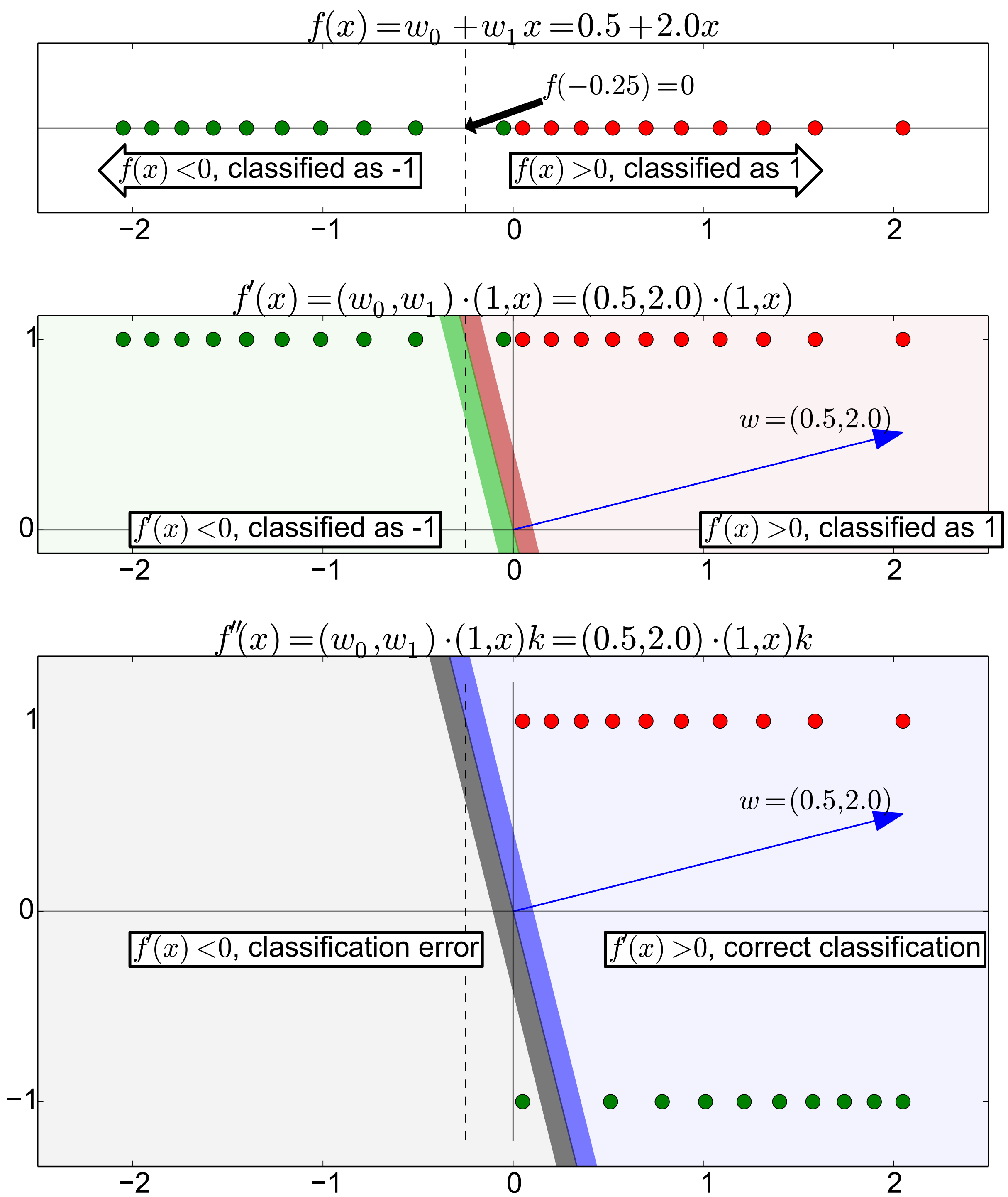
Equivalently, (as in the logistic regression lecture), with $x' = [1, x]$ and $w' = [b, w]$:

$$\begin{aligned} \langle w', x'_j \rangle > 0 \quad \text{if} \quad k_j = 1 \qquad (\forall j \in \{1, 2, ..., L\})\,, \\ \langle w', x'_j \rangle < 0 \quad \text{if} \quad k_j = -1\,, \end{aligned} \tag{2}$$

or, with $x''_j = k_j x'_j$,

$$\langle w', x''_j \rangle > 0\,, \qquad (\forall j \in \{1, 2, ..., L\}\,.) \tag{3}$$

Perceptron Classifier, Formulation, Example



- class 1
- class -1

Top: Training set, $x_j \in \mathbb{R}$

Middle: Augmenting by 1's, $x'_j \in \mathbb{R}^2$

Bottom: Multiplying by k_j, $k_j x''_j \in \mathbb{R}^2$

Perceptron Learning: Algorithm

We use the last representation ($x''_j = k_j[1, x_j]$, $w' = [b, w]$) and drop the dashes to reduce notation clatter.

Goal: Find a weight vector $w \in \mathbb{R}^{D+1}$ (original feature space dimensionality is D) such that:

$$\langle w, x \rangle > 0 \qquad (\forall j \in \{1, 2, ..., L\}) \tag{4}$$

Perceptron algorithm, (Rosenblat 1962):

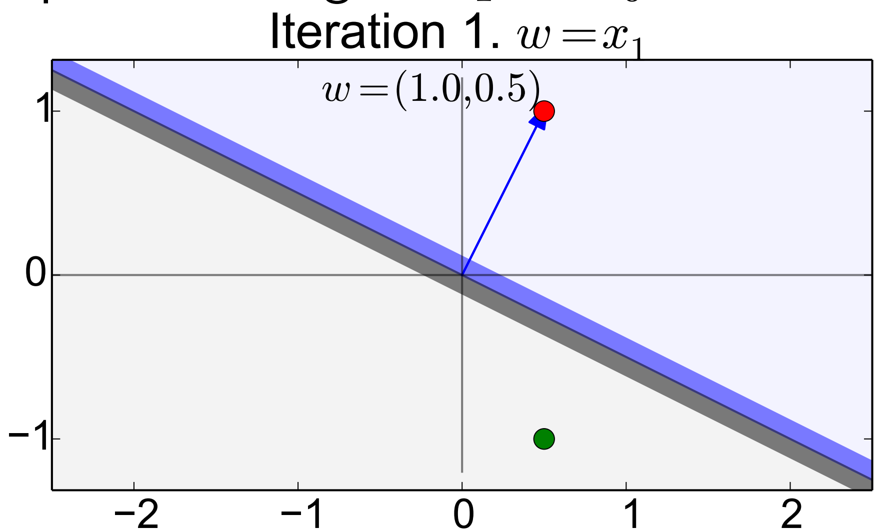
1. $w_{t=0} = 0$.
2. A wrongly classified observation x_j is sought, i.e., $\langle w_t, x_j \rangle < 0$, $j \in \{1, 2, ..., L\}$.
3. If there is no misclassified observation then the algorithm terminates otherwise

$$w_{t+1} = w_t + x_j \; .$$

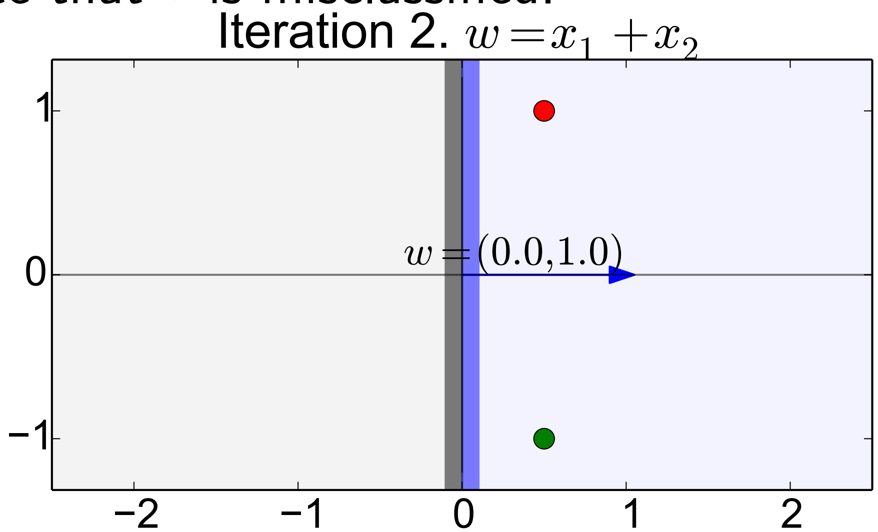
4. Goto 2.

Perceptron: Weight Update, Example

Consider this dataset with just 2 points. As $w_{t=0} = 0$, all points are misclassified. Order the points randomly and go over this dataset. Find the first misclassified point. It is $\color{red}{\bullet}$. Make the update of weight, $w_1 \leftarrow w_0 + x_{\color{red}{\bullet}}$.

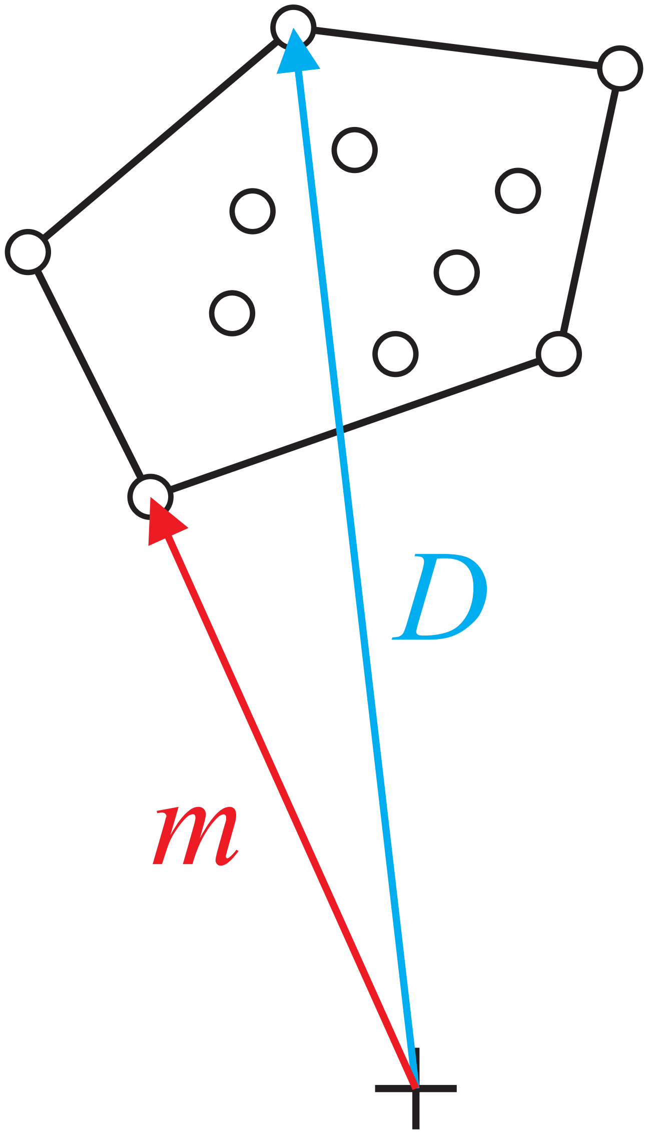


Note that $\color{green}{\bullet}$ is misclassified.



Whole dataset is correctly classified. Done.

Novikoff Theorem



If the data are linearly separable then there exists a number $t^* \leq \frac{D^2}{m^2}$, such that the vector w_{t^*} satisfies the inequality

$$\langle w_{t^*}, x^j \rangle > 0, \forall j \in \{1, 2, ..., L\}.$$

? What if the data is not separable?

? How to terminate perceptron learning?

Perceptron Learning: Non-Separable Case

Perceptron algorithm, batch version, handling non-separability:

Input: $T = \{x_1, \dots x_L\}$
Output: a weight vector w^*

1. $w_{t=0} = 0$, $E = |T| = L$, $w^* = 0$.
2. Find all mis-classified observations $X^- = \{x \in X : \langle w_t, x \rangle < 0\}$.
3. if $|X^-| < E$ then $E = |X^-|; w^* = w_t$
4. if $tc(w^*, t, t_{lu})$ then terminatate else $w_{t+1} = w_t + \eta_t \sum_{x \in X^-} x$
5. Goto 2.

- The algorithm converges with probability 1 to the optimal solution.
- Convergence rate not known.
- Termination condition $tc(.)$ is a complex function of the quality of the best solution, time since last update $t - t_{lu}$ and requirements on the solution.

Perceptron Learning as an Optimisation Problem (1)

Perceptron algorithm, batch version, handling non-separability, another perspective:

Input: $T = \{x_1, \ldots x_L\}$
Output: a weight vector w minimising

$$J(w) = |\{x \in X : \langle w_t, x \rangle < 0\}|$$

or, equivalently

$$J(w) = \sum_{x \in X : \langle w_t, x \rangle < 0} 1$$

What would the most common optimisation method, i.e. gradient descent, perform?

$$w_t = w - \eta \nabla J(w)$$

The gradient of $J(w)$ is either 0 or undefined. Gradient minimisation cannot proceed.

Perceptron Learning as an Optimisation Problem (2)

Let us redefine the cost function:

$$J_p(w) = - \sum_{x \in X : \langle w, x \rangle < 0} \langle w, x \rangle$$

$$\nabla J_p(w) = \frac{\partial J}{\partial w} = \sum_{x \in X : \langle w, x \rangle < 0} (-x)$$

- The Perceptron Algorithm is a gradient **descent** method for $J_p(w)$ (gradient for a single misclassified sample is $-x$, so the weight update is x)
- Learning and empirical risk minimisation is just and instance of an optimization problem.
- Either gradient minimisation (backpropagation in neural networks) or convex (quadratic) minimisation (in mathematical literature called convex programming) is used.

Optimal Separating Plane and The Closest Point To The Convex Hull

The problem of optimal separation by a hyperplane

$$(1) \qquad w^* = \operatorname*{argmax}_w \min_j \left\langle \frac{w}{|w|}, x_j \right\rangle$$

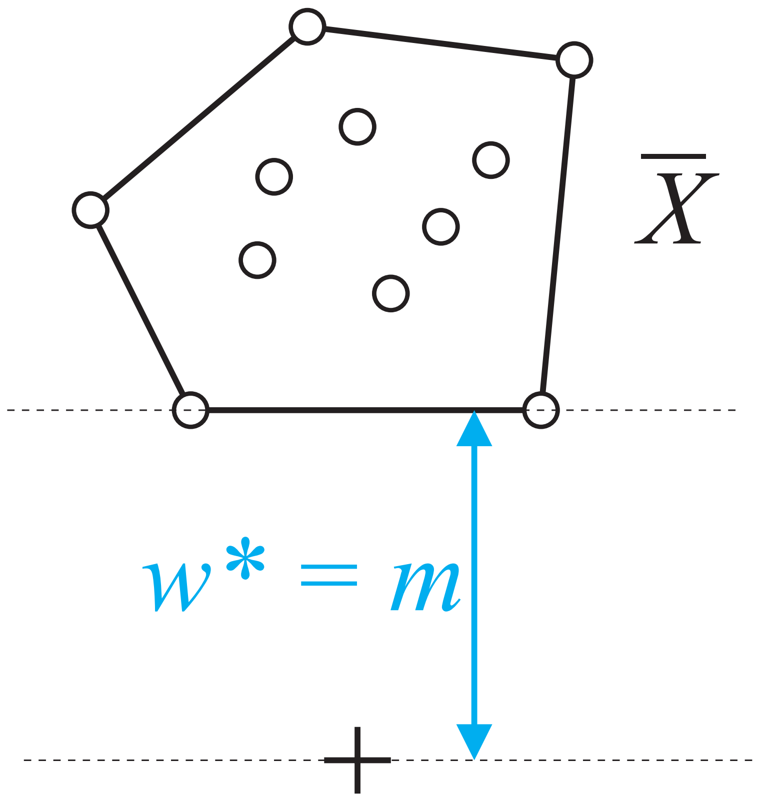
can be converted to seek for the closest point to a convex hull (denoted by the overline)

$$x^* = \operatorname*{argmin}_{x \in \overline{X}} |x|$$

There holds that x^* solves also the problem (1).

Recall that the classfier that maximises separation minimises the structural risk R_{str} (page 8)!

Convex Hull, Illustration



$$\min_j \left\langle \frac{w}{|w|}, x_j \right\rangle \leq \quad m \quad \leq |w| \, , \; w \in \overline{X}$$

lower bound   upper bound

ε-Solution

- The aim is to speed up the algorithm.
- The allowed uncertainty ε is introduced.

$$|w| - \min_j \left\langle \frac{w}{|w|},\ x_j \right\rangle \leq \varepsilon$$

Training Algorithm 2 – Kozinec (1973)

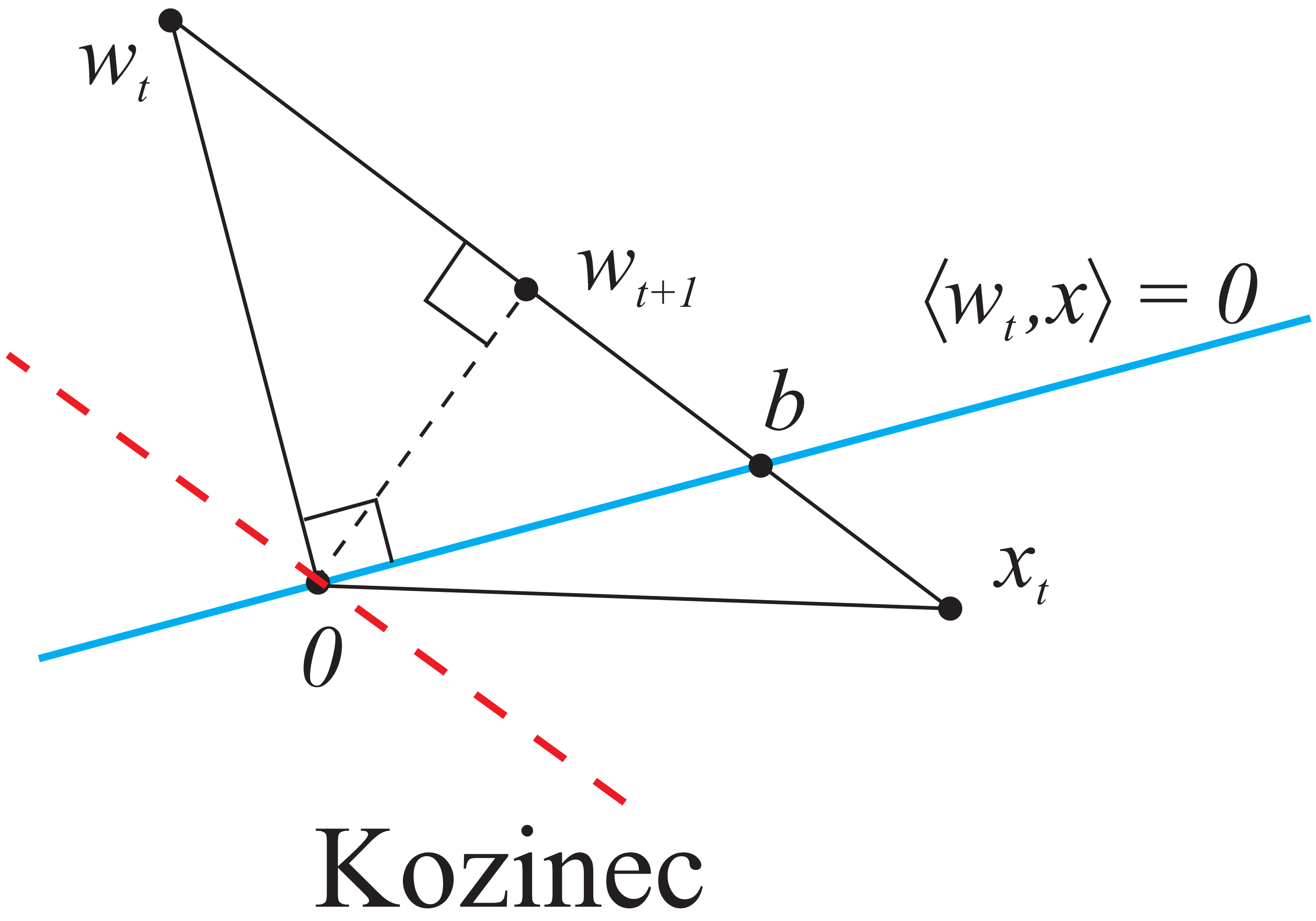
1. $w_{t=0} = x_j$, i.e. any observation.
2. A wrongly classified observation x_t is sought, i.e., $\langle w_t, x^j \rangle < b$, $j \in J$.
3. If there is no wrongly classified observation then the algorithm finishes otherwise

$$w_{t+1} = (1-k) \cdot w_t + x_t \cdot k \,, \qquad k \in \mathbb{R} \,.$$

   where $k = \operatorname*{argmin}_k |(1-k) \cdot w_t + x_t \cdot k|$.

4. Goto 2.

Kozinec, Pictorial Illustration

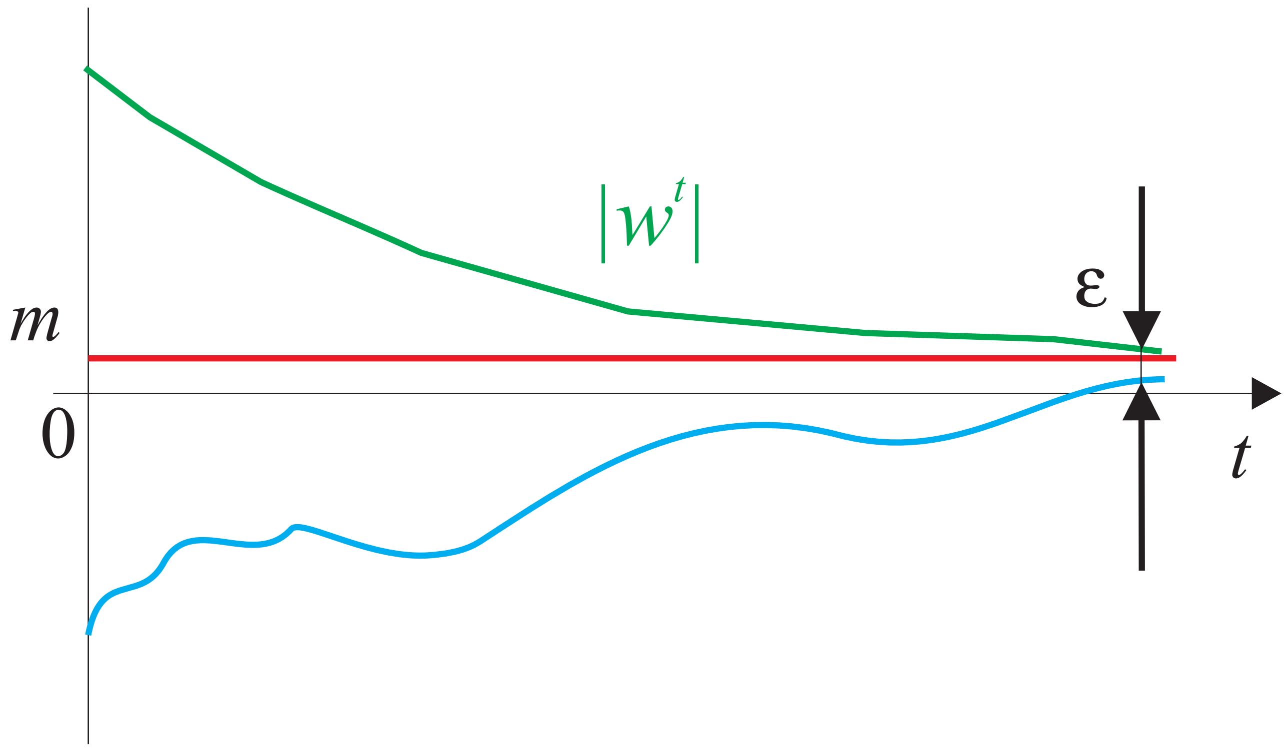


Kozinec and ε-Solution

The second step of Kozinec algorithm is modified to:

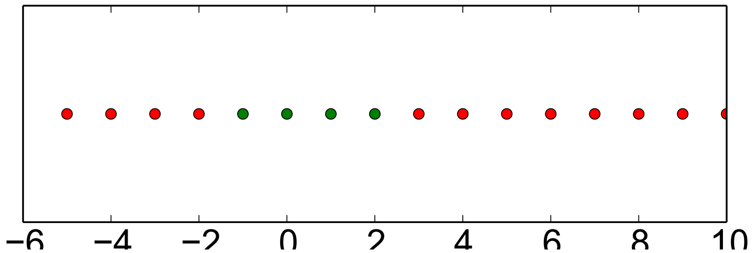
A wrongly classified observation x_t is sought, i.e.,

$$|w^t| - \min_j \left\langle \frac{w^t}{|w^t|}, x_t \right\rangle \geq \varepsilon$$



Note on dimension lifting

Original data, not linearly separable



Transformed data $x \leftarrow [x, x^2]$, linearly separable