Machine Learning

Volker Roth

Department of Mathematics & Computer Science
University of Basel

Chapter 3: Classification

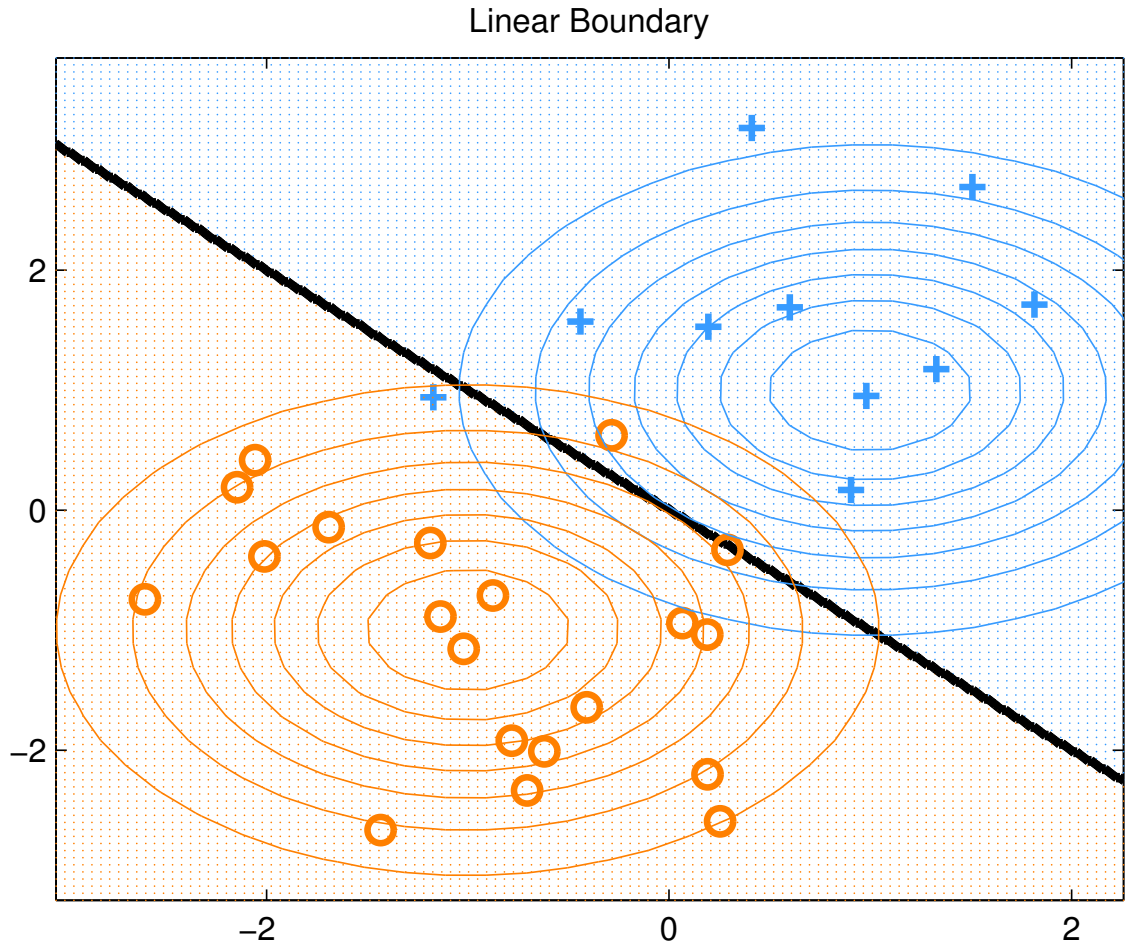


Fig 4.5 in K. Murphy: Machine Learning. MIT Press 2012

Bayesian Decision Theory

- Assign observed $\boldsymbol{x} \in \mathbb{R}^d$ into one of k classes. A **classifier** is a mapping that assigns labels to observations

$$f_\alpha : \boldsymbol{x} \to \{1, \ldots, k\}.$$

- For any observation $\boldsymbol{x}$ there exists a set of k **possible actions** α_i, i.e. k different assignments of labels.
- The **loss** L incurred for taking action α_i when the true label is j is denoted by a loss matrix $L_{ij} = L(\alpha_i | c = j)$.
- "Natural" $0-1$ **loss function** $\rightsquigarrow$ **counting misclassifications:**

$$L_{ij} = 1 - \delta_{ij}, \text{ where } \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

Bayesian Decision Theory (cont'd)

- A classifier is trained on a set of **observed pairs** $\{(\boldsymbol{x}_1, c_1), \ldots, (\boldsymbol{x}_n, c_n)\} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}, c) = p(c|\boldsymbol{x})p(\boldsymbol{x})$
- The probability that a given $\boldsymbol{x}$ is member of class c_j , i.e. the posterior probability of membership in class j, is obtained via the **Bayes rule:**

$$P(c_j|\boldsymbol{x}) = \frac{\overbrace{p(\boldsymbol{x}|c=j)}^{\text{Given the label, observation is generated}}}{p(\boldsymbol{x})} \overbrace{P(c=j)}^{\text{Nature picks a label first}},$$

where

$$p(\boldsymbol{x}) = \sum_{j=1}^{k} p(\boldsymbol{x}|c=j)P(c=j).$$

- Given an observation $\boldsymbol{x}$, the expected loss associated with choosing action α_i (the **conditional risk** or **posterior expected loss**) is

$$R(f_{\alpha_i}|\boldsymbol{x}) = \sum_{j=1}^{k} L_{ij} P(c_j|\boldsymbol{x}) \overset{(\text{if } L_{ij}=1-\delta_{ij})}{=} \sum_{j \neq i} P(c_j|\boldsymbol{x}) = 1 - P(c_i|\boldsymbol{x}).$$

Bayesian Decision Theory (cont'd)

- **Goal:** minimize the **overall risk** of the classifier f_α:

$$R(f_\alpha) = \int_{\boldsymbol{R}^d} R(f_\alpha(\boldsymbol{x})|\boldsymbol{x}) p(\boldsymbol{x})\, d\boldsymbol{x}.$$

- If $f_\alpha(\boldsymbol{x})$ minimizes the conditional risk $R(f_\alpha(\boldsymbol{x})|\boldsymbol{x})$ for every $\boldsymbol{x}$, the overall risk will be minimized as well.
- This is achieved by the **Bayes optimal classifier** which chooses the mapping

$$f(\boldsymbol{x}) = \operatorname*{argmin}_i \sum_{j=1}^{k} L_{ij}\, P(c = j|\boldsymbol{x}).$$

- For $0-1$ loss, this reduces to classifying $\boldsymbol{x}$ to the class **with highest posterior probability:**

$$f(\boldsymbol{x}) = \operatorname*{argmax}_i P(c = i|\boldsymbol{x}).$$

Bayesian Decision Theory (cont'd)

- **Simplification:** only 2 classes: c is **Bernoulli RV.**
- Bayes optimal classifier is defined by the **zero crossings** of the **Bayes optimal discriminant function**

$$G(\boldsymbol{x}) = P(c_1|\boldsymbol{x}) - P(c_2|\boldsymbol{x}), \text{ or } g(\boldsymbol{x}) = \log \frac{P(c_1|\boldsymbol{x})}{P(c_2|\boldsymbol{x})}.$$

- Problem: direct approximation of G would require the knowledge of the Bayes optimal discriminant.
- Define a **parametrized family of classifiers** $\mathcal{F}_{\boldsymbol{w}}$ from which we can choose one (or more) function(s) by some **inference mechanism**.
- One such family: linear discriminant functions $g(\boldsymbol{x}; \boldsymbol{w}) = w_0 + \boldsymbol{w}^t \boldsymbol{x}$.
- Two-category case: Decide c_1 if $g(\boldsymbol{x}; \boldsymbol{w}) > 0$ and c_2 if $g(\boldsymbol{x}; \boldsymbol{w}) < 0$. Eq. $g(\boldsymbol{x}; \boldsymbol{w}) = 0$ defines the **decision surface**.
- The hyperplane divides the feature space into **half-spaces** $\mathcal{R}_1$ ("positive side") and $\mathcal{R}_2$ ("negative side").

Decision Hyperplanes

- $g(\boldsymbol{x}; \boldsymbol{w})$ defines distance r from $\boldsymbol{x}$ to the hyperplane: $\boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$.
- $g(\boldsymbol{x}_p) = 0 \Rightarrow g(\boldsymbol{x}) = r\|\boldsymbol{w}\| \quad \Leftrightarrow \quad r = g(\boldsymbol{x})/\|\boldsymbol{w}\|$.

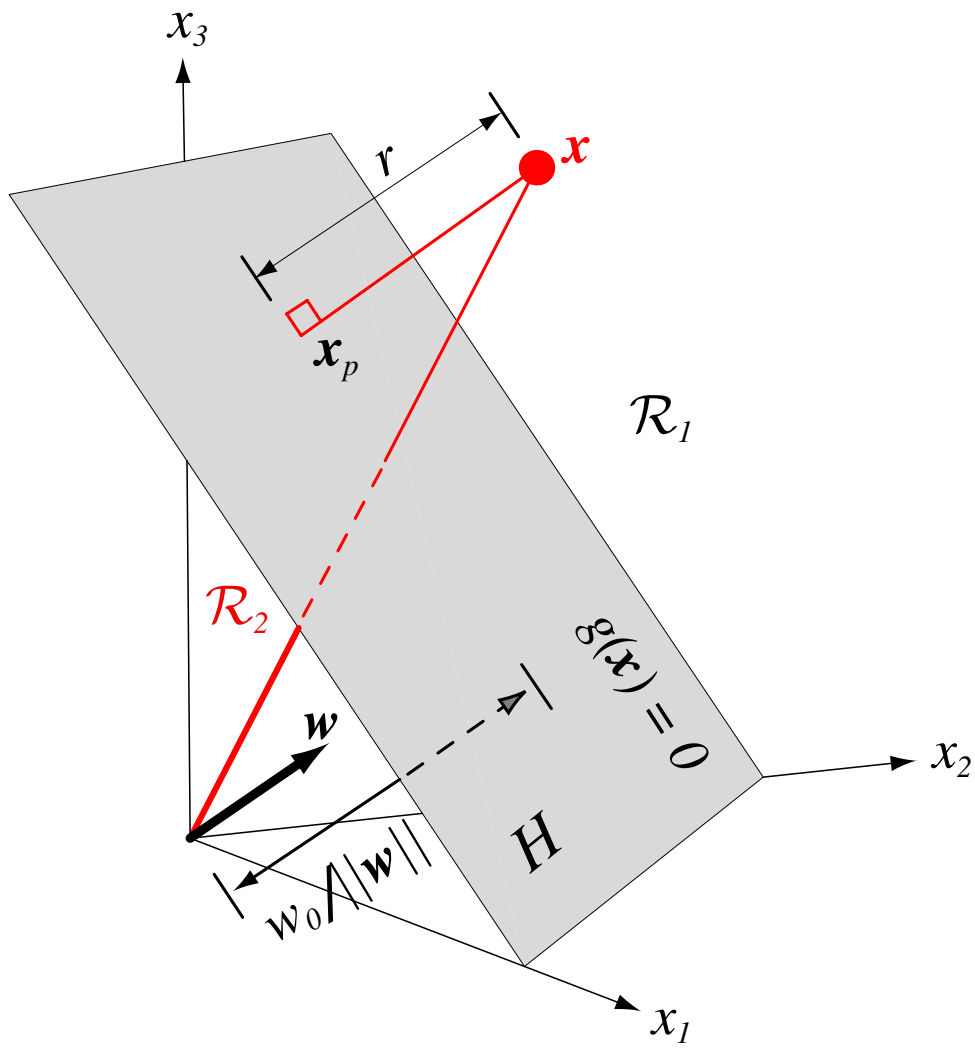


Fig 5.2 in Duda, Hart & Stork: Pattern Classification. Wiley 2001.

Generalized Linear Discriminant Functions

Use basis functions $\{b_1(\boldsymbol{x}), \ldots, b_m(\boldsymbol{x})\}$, where each $b_i(\boldsymbol{x}) : \mathbb{R}^d \mapsto \mathbb{R}$, and $g(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 b_1(\boldsymbol{x}) + \cdots + w_m b_m(\boldsymbol{x}) =: \boldsymbol{w}^t \boldsymbol{y}$ (note that we have redefined $\boldsymbol{y}$ here in order to be consistent with the following figure)

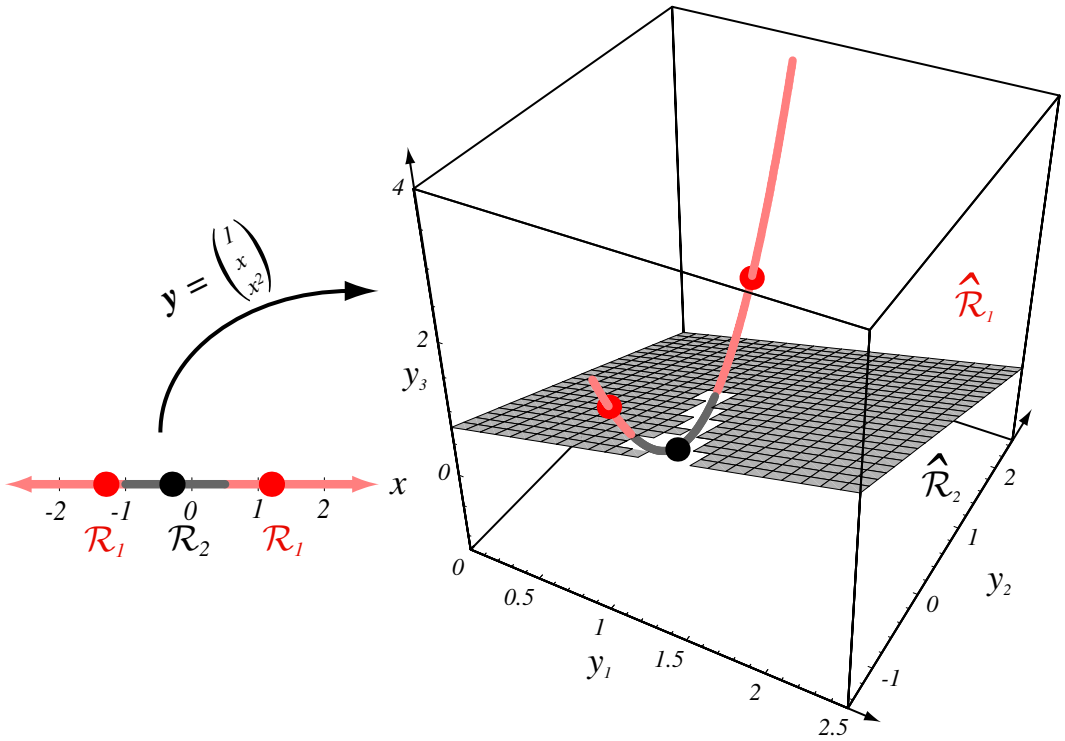


Fig 5.5 in Duda, Hart & Stork: Pattern Classification. Wiley 2001.

Generalized Linear Discriminant Functions

Use basis functions $\{b_1(\boldsymbol{x}), \ldots, b_m(\boldsymbol{x})\}$, where each $b_i(\boldsymbol{x}) : \mathbb{R}^d \mapsto \mathbb{R}$, and

$$g(\boldsymbol{x}; \boldsymbol{w}) = w_0 + w_1 b_1(\boldsymbol{x}) + \cdots + w_m b_m(\boldsymbol{x}) =: \boldsymbol{w}^t \boldsymbol{y}.$$

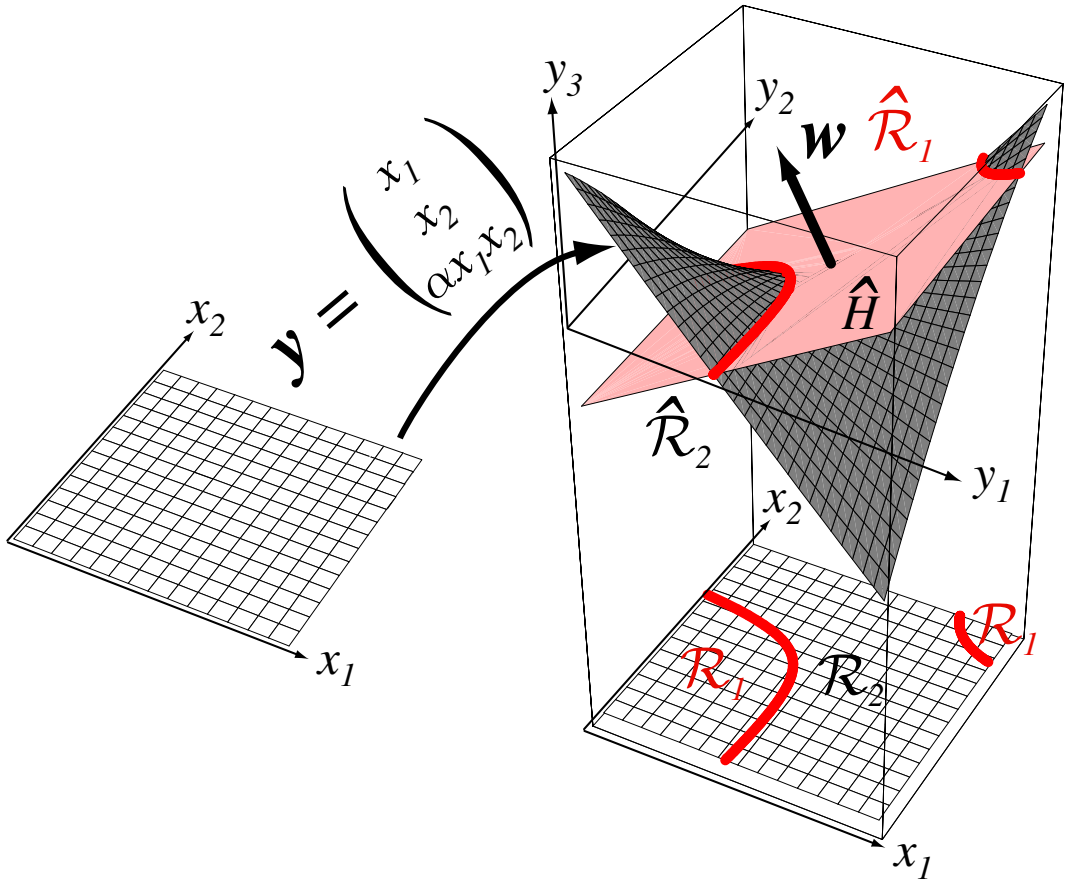


Fig 5.6 in Duda, Hart & Stork: Pattern Classification. Wiley 2001.

Separable Case

- Consider sample $\{\boldsymbol{y}_i, c_i\}_{i=1}^n$. If there exists $f(\boldsymbol{y}; \boldsymbol{w}) = \boldsymbol{y}^t \boldsymbol{w}$ which is positive for all examples in class 1 and negative for all examples in class 2, we say that the sample is **linearly separable.**
- **Normalization:** replace all samples labeled c_2 by their negatives
  $\rightsquigarrow$ simply write $\boldsymbol{y}^t \boldsymbol{w} > 0$ for all samples.
- Each sample places a constraint on the possible location of $\boldsymbol{w}$
  $\rightsquigarrow$ **solution region.**

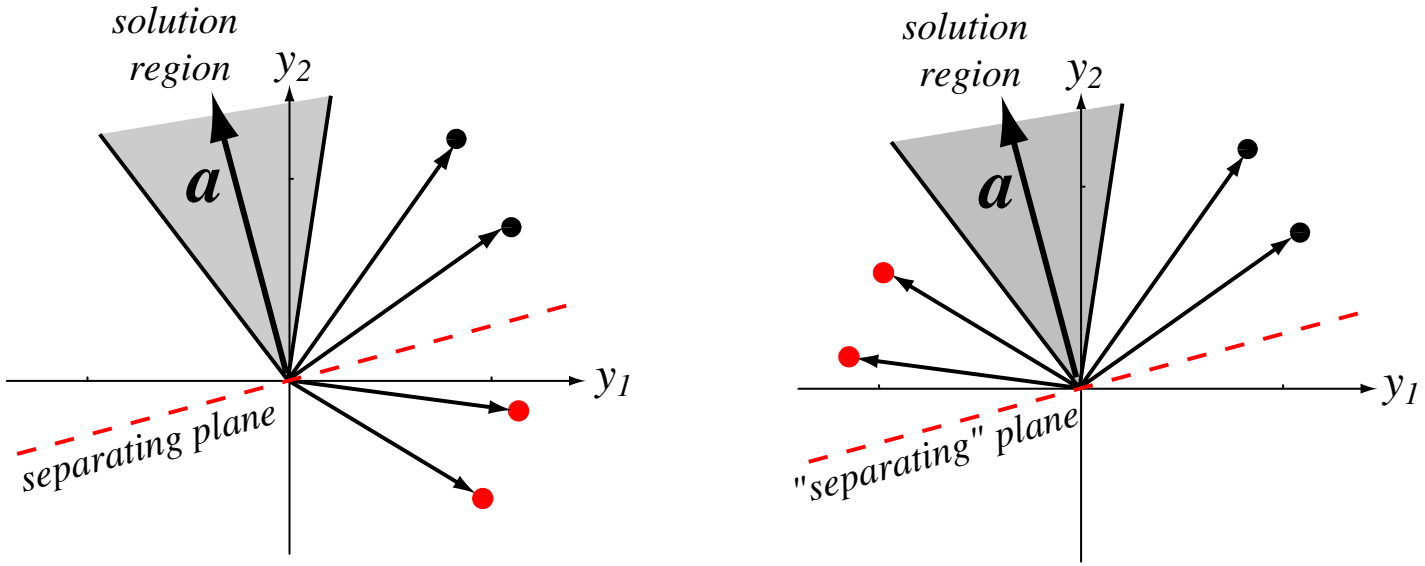


Fig 5.8 in Duda, Hart & Stork: Pattern Classification. Wiley 2001.

Separable Case: Margin

- Different solution vectors may have different **classification margins** b: $\boldsymbol{y}^t \boldsymbol{w} \geq b > 0$.
- Intuitively, **large margins are good.** We will formalize this in the chapter on **Statistical Learning Theory.**

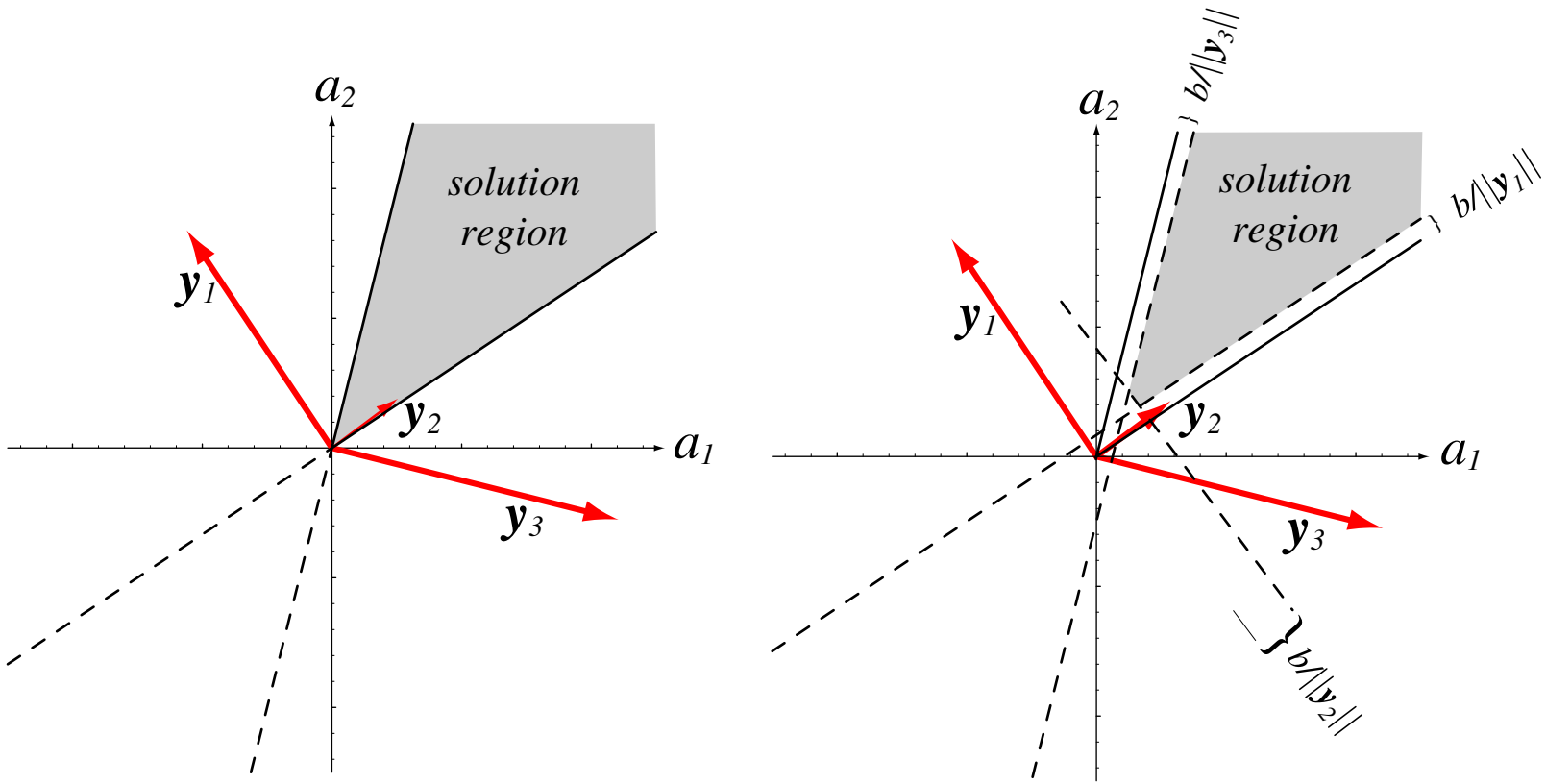


Fig 5.9 in in Duda, Hart & Stork: Pattern Classification. Wiley 2001.

Gradient Descent

- Solve $\boldsymbol{y}^t\boldsymbol{w} > 0$ by defining $J(\boldsymbol{w})$ such that a minimizer of J is a solution.
- Most obvious choice for J: #(misclassifications), **not differentiable.**
- Alternative choice: $J_p(\boldsymbol{w}) = \sum_{\boldsymbol{y}\in\mathcal{M}} -\boldsymbol{y}^t\boldsymbol{w}$, where $\mathcal{M}(\boldsymbol{w})$ is the set of samples **misclassified** by $\boldsymbol{w}$. **$J_p(\boldsymbol{w})$ is differentiable.**
- Note that $\boldsymbol{y}^t\boldsymbol{w} < 0\ \forall \boldsymbol{y} \in \mathcal{M}$
  $\rightsquigarrow$ **J_p is non-negative, and zero only if $\boldsymbol{w}$ is a solution.**
- **Gradient descent:** Start with initial $\boldsymbol{w}^{(1)}$, choose next move in the direction of the negative gradient: $\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \eta^{(k)}\nabla J(\boldsymbol{w}^{(k)})$.
- Gradient: $\nabla J(\boldsymbol{w}) = -\sum_{\boldsymbol{y}\in\mathcal{M}} \boldsymbol{y}$. **Gradient descent:**

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta^{(k)} \sum_{\boldsymbol{y}\in\mathcal{M}} \boldsymbol{y}.$$

  This defines the **Batch Perceptron algorithm.**

Fixed-Increment Single Sample Perceptron

- Fix learning rate $\eta = 1$.
- Gradient step is **sum of individual steps** in the direction of **single misclassified samples.**
- **Sequential single-sample updates:** use superscripts $\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots$ for misclassified samples $\boldsymbol{y} \in \mathcal{M}$. Ordering is irrelevant.
- Simple algorithm:

$$\boldsymbol{w}^{(1)} \text{ arbitrary}$$
$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \boldsymbol{y}^k, \; k \geq 1$$

Perceptron Convergence Theorem

If the samples are linearly separable, the sequence of weight vectors given by the Fixed-Increment Single Sample Perceptron algorithm will terminate at a solution vector.

Proof: exercises.

Minimizing the Perceptron Criterion (2)

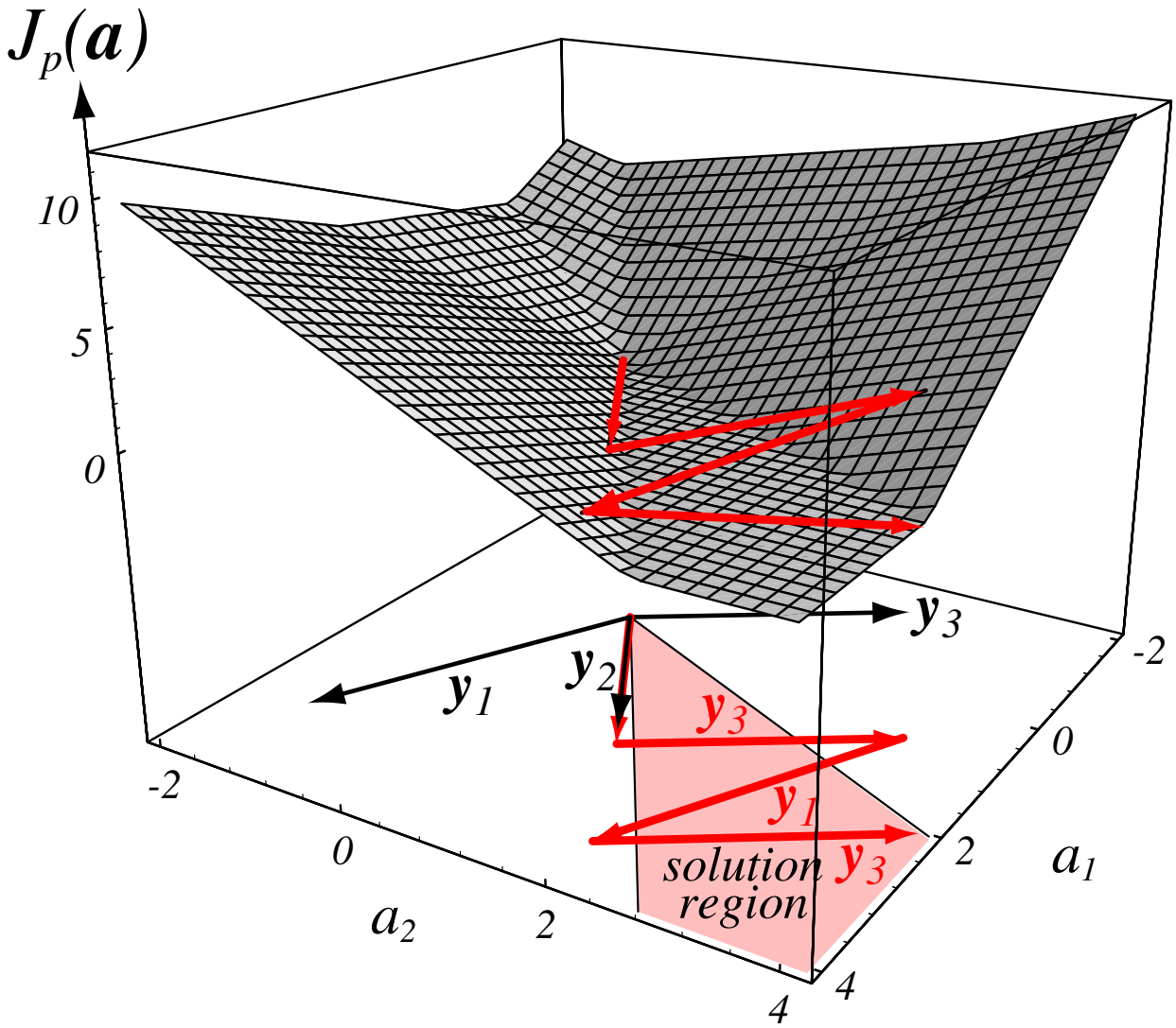


Fig 5.12 in Duda, Hart & Stork: Pattern Classification. Wiley 2001.

Issues

A number of problems with the perceptron algorithm:

- When the data are separable, there are **many solutions**, and which one is found **depends on the starting values**.
- In particular, no **separation margin** can be guaranteed (however, there exist modified versions...)
- The number of steps can be **very** large.
- When the data are **not separable**, the algorithm will **not necessarily converge**, and **cycles** may occur.
  The cycles can be **long** and therefore **hard to detect.**
- Method "technical" in nature, no (obvious) probabilistic interpretation (but we will see that there is one).

But the perceptron algorithm is **historically important** (1957, one of the first ML algorithms!), was even implemented in analog hardware(!)

Generative (or Informative) vs Discriminative

- Notation: For the following discussion it is more convenient to go back to the original $\boldsymbol{x}$-vectors (potentially after some basis expansion) instead of using the "normalized" representation $\boldsymbol{y}$.
- Two main strategies:
  - **Generative:** Generative classifiers specify how to generate data using the **class densities.**
    Likelihood/posterior of each class is examined and classification is usually done by assigning to the most likely class.
  - **Discriminative:** These classifiers focus on modeling the **class boundaries** or the class membership probabilities directly.
    No attempt is made to model the underlying class conditional densities.

Generative Classifiers

- Central idea: model the **conditional class densities $p(\boldsymbol{x}|c)$.**
- Assuming a parametrized family $p_{\boldsymbol{w}_j}(\boldsymbol{x}|c=j)$ and collecting all model parameters in a vector $\boldsymbol{w}$, a typical **Frequentist** approach now proceeds by maximizing the log likelihood:

$$\hat{\boldsymbol{w}}_{MLE} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \sum_{i=1}^{n} \log p_{\boldsymbol{w}}(\boldsymbol{x}_i|c_i)$$

- (Approximate) **Bayesian** interpretation: $\hat{\boldsymbol{w}}_{MLE}$ might then be plugged into Bayes rule to compute the class assignment probabilities

$$P(c_j|\boldsymbol{x}) = \frac{p_{\hat{\boldsymbol{w}}_{MLE}}(\boldsymbol{x}|c=j)}{p(\boldsymbol{x})} P(c=j).$$

In **Linear Discriminant Analysis** (LDA), a Gaussian model is used where all classes share a common covariance matrix Σ:

$$p_{\boldsymbol{w}}(\boldsymbol{x}|c=j) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_j, \Sigma).$$

Common continuous distributions: Multivariate Normal

- The multivariate normal distribution of a k-dimensional random vector $\boldsymbol{X} = (X_1, \ldots, X_k)^t$ can be written as: $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, with k-dimensional **mean vector**

  $$\boldsymbol{\mu} = \mathrm{E}[\mathbf{X}] = [\mathrm{E}[X_1], \mathrm{E}[X_2], \ldots, \mathrm{E}[X_k]]^{\mathrm{t}}$$

  and $k \times k$ **covariance matrix**

  $$\Sigma =: \mathrm{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^{\mathrm{t}}] = [\mathrm{Cov}[X_i, X_j]; 1 \leq i, j \leq k],$$

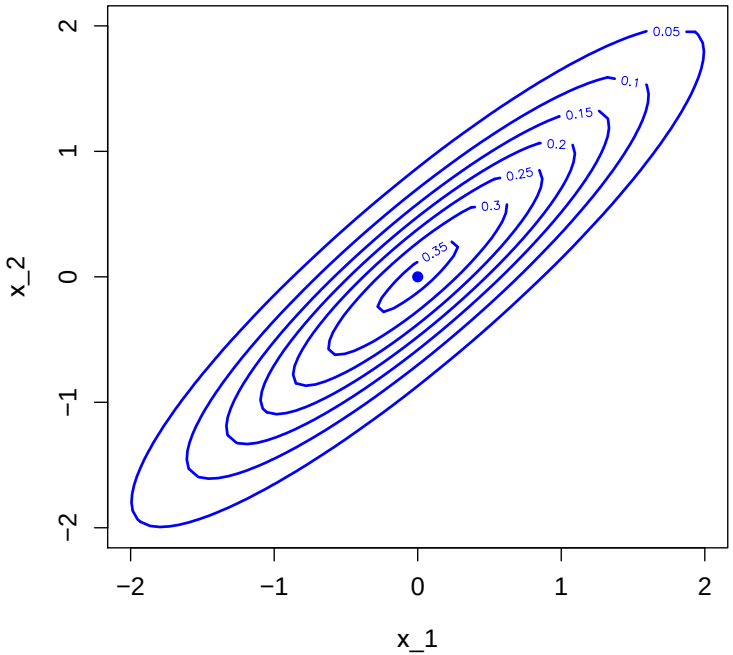
  where

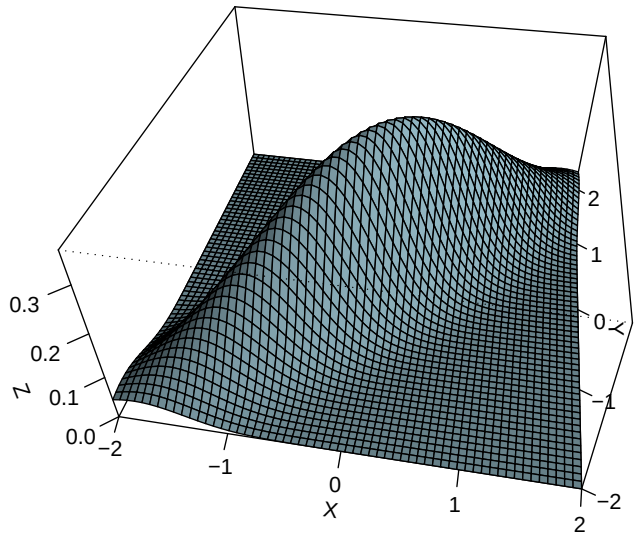
  $$\mathrm{Cov}[X_i, X_j] = \mathrm{E}[(X_i - \mu_i)(X_j - \mu_j)].$$

- The inverse of the covariance matrix is called **precision matrix**
- The pdf of the multivariate normal distribution is

$$p(x_1, \ldots, x_k | \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{t}} \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

The multivariate Normal distribution





Affine transformations:

If $\mathbf{Y} = \boldsymbol{c} + B\mathbf{X}$ is an affine transformation of $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then $\mathbf{Y} \sim \mathcal{N}\left(\boldsymbol{c} + B\boldsymbol{\mu}, B\Sigma B^{\mathrm{t}}\right)$. Why?

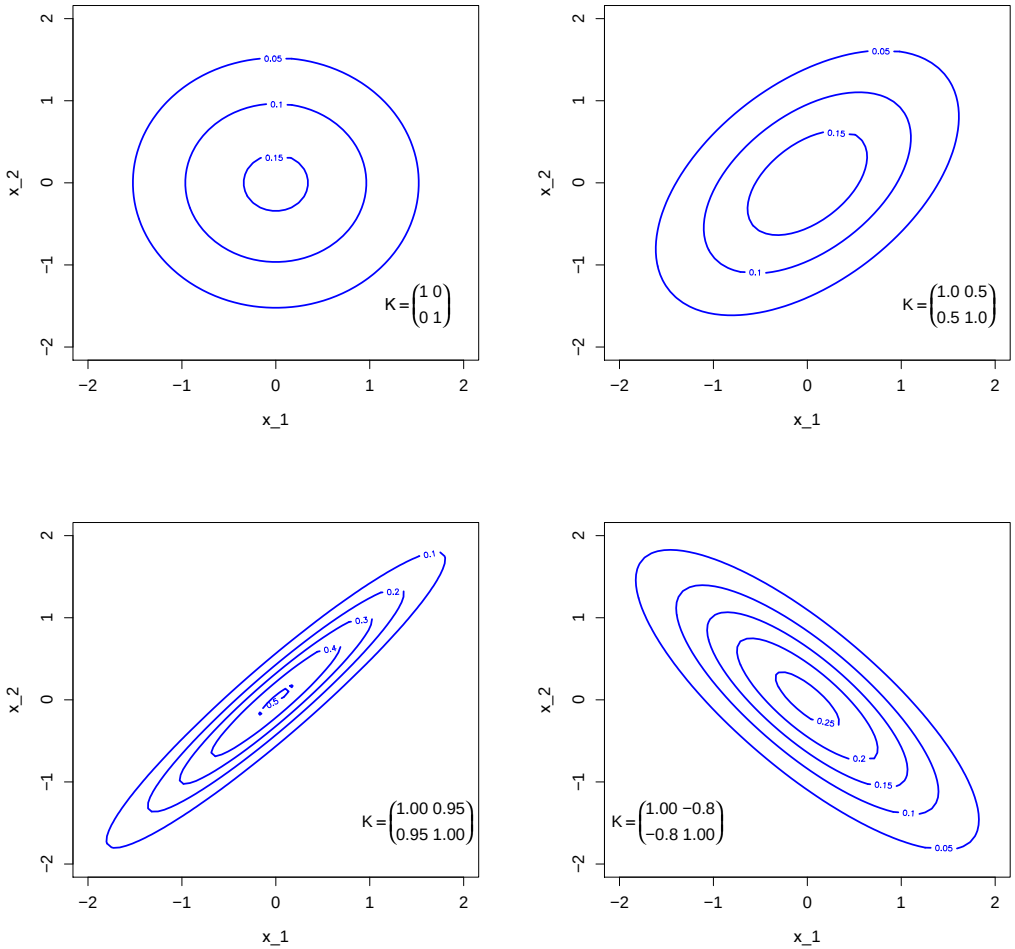
$$\begin{aligned}
\boldsymbol{\mu}_{\mathbf{Y}} &= \mathrm{E}[\boldsymbol{c} + B\mathbf{X}] = \boldsymbol{c} + B\,\mathrm{E}[\mathbf{X}] = \boldsymbol{c} + B\boldsymbol{\mu} \\
\Sigma_{\mathbf{Y}} &= \mathrm{E}[(\boldsymbol{c} + B\mathbf{X} - \boldsymbol{c} + B\boldsymbol{\mu})(\boldsymbol{c} + B\mathbf{X} - \boldsymbol{c} + B\boldsymbol{\mu})^t] \\
&= \mathrm{E}[B(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^t B^t] = B\Sigma B^t
\end{aligned}$$

The multivariate normal distribution

2D Gaussian: $p(\boldsymbol{x}|\boldsymbol{\mu} = \mathbf{0}, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp(-\frac{1}{2}\boldsymbol{x}^t\Sigma^{-1}\boldsymbol{x})$

Covariance
(also written "co-variance")
is a measure of how much **two random variables vary together**:

- **positive**: positive linear coherence,
- **negative**: negative linear coherence,
- **0**: no linear coherence.



Generative Classifiers: LDA

- In **Linear Discriminant Analysis**, a Gaussian model is used where all classes share a common covariance matrix Σ:

$$p_{\boldsymbol{w}}(\boldsymbol{x}|c=j) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_j, \Sigma).$$

- The resulting **discriminant functions are linear:**

$$\begin{aligned} g(\boldsymbol{x}) &= \log \frac{P(c_1|\boldsymbol{x})}{P(c_2|\boldsymbol{x})} = \log \frac{P(c_1)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \Sigma)}{P(c_2)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_2, \Sigma)} \\ &= \underbrace{\log \frac{P(c_1)}{P(c_2)} - \frac{1}{2}\,(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^t\,\Sigma^{-1}\,(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)}_{w_0} \\ &\quad + \underbrace{(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^t\,\Sigma^{-1}\boldsymbol{x}}_{\boldsymbol{w}^t\boldsymbol{x}} \\ &= w_0 + \boldsymbol{w}^t\boldsymbol{x}, \quad \text{with } \boldsymbol{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2). \end{aligned}$$

LDA algorithm

- Let $\hat{\Sigma}$ be an estimate of the shared covariance matrix Σ:

$$\Sigma_c = \frac{1}{n_c} \sum_{\boldsymbol{x} \in \mathcal{X}_c} (\boldsymbol{x} - \boldsymbol{m}_c)(\boldsymbol{x} - \boldsymbol{m}_c)^t, \quad c \in \{c_1, c_2\}$$

$$\hat{\Sigma} = \frac{1}{2}(\Sigma_1 + \Sigma_2).$$

- Let $\boldsymbol{m}_j$ an estimate of $\boldsymbol{\mu}_j$:

$$m_c = \frac{1}{n_c} \sum_{\boldsymbol{x} \in \mathcal{X}_c} \boldsymbol{x}, \quad n_c = |\mathcal{X}_c|.$$

- LDA finds the weight vector

$$\boldsymbol{w}^{\text{LDA}} = \hat{\Sigma}^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2).$$

Law of large numbers: as $n \to \infty$, $\hat{\Sigma} \to \Sigma$ and $\boldsymbol{m}_j \to \boldsymbol{\mu}_j$

$\rightsquigarrow$ This classifier **is asymptotically Bayes-optimal,**

if the Gaussian model with shared covariances is correct.

LDA

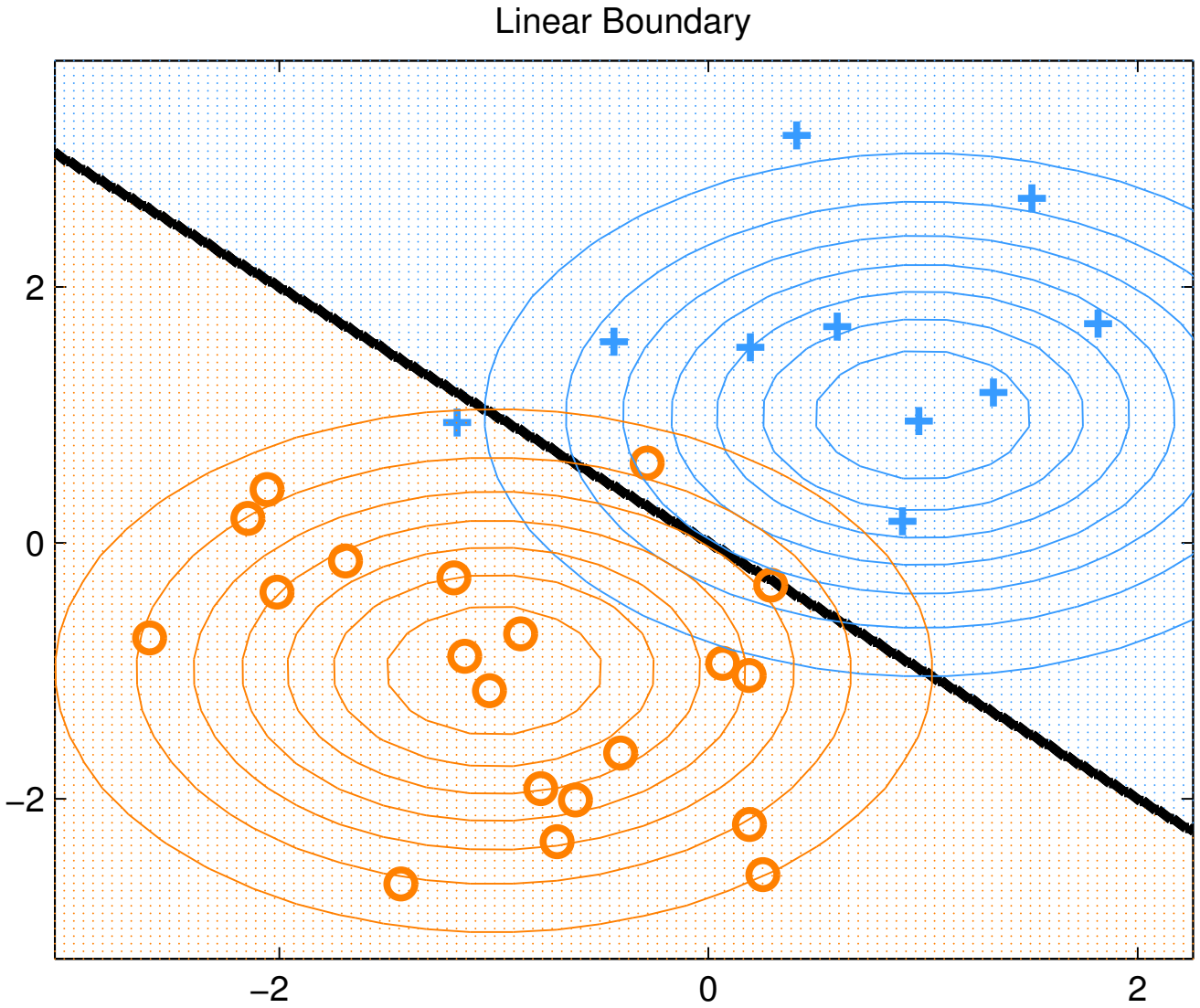


Fig 4.5 in K. Murphy: Machine Learning. MIT Press 2012

Discriminative classifiers

- Discriminative classifiers focus directly on the **discriminant function.**
- In general, they are more flexible with regard to the class conditional densities they are capable of modeling.
- Notation: Can use any class encoding scheme. Here: $c \in \{0, 1\}$.
- Bayes formula:

$$\begin{aligned} g(\boldsymbol{x}) &= \log \frac{P(c=1|\boldsymbol{x})}{P(c=0|\boldsymbol{x})} \\ &= \log \frac{p(\boldsymbol{x}|c=1)P(c=1)}{p(\boldsymbol{x}|c=0)P(c=0)}, \end{aligned}$$

- Can model any conditional probabilities that are exponential "tilts" of each other:

$$p(\boldsymbol{x}|c=1) = e^{g(\boldsymbol{x})} p(\boldsymbol{x}|c=0) \frac{P(c=0)}{P(c=1)}$$

Logistic Regression (LOGREG)

- **Logistic regression** uses a **linear discriminant function**, i.e. $g(\boldsymbol{x}) = \boldsymbol{w}^t \boldsymbol{x} + w_0$.
- For the special case $p(\boldsymbol{x}|c) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_{0,1}, \Sigma)$, same as LDA:

$$p(\boldsymbol{x}|c=1) = \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu}_1, \Sigma) = e^{g(\boldsymbol{x})} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_0, \Sigma) \frac{P(c=0)}{P(c=1)}$$

$$\Rightarrow g(\boldsymbol{x}) = w_0 + \boldsymbol{w}^t \boldsymbol{x} = \log \frac{P(c=1)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_1, \Sigma)}{P(c=0)\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_0, \Sigma)}$$

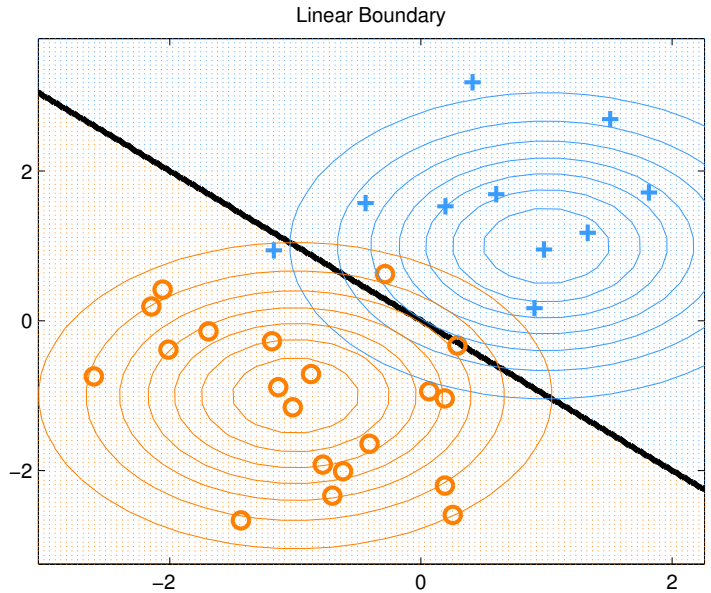


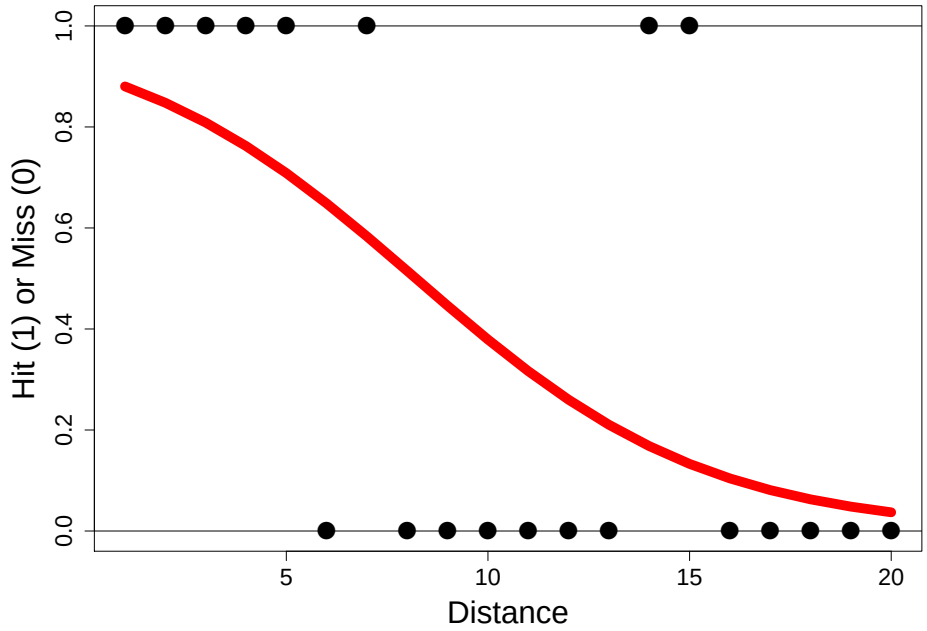
Fig 4.5 in K. Murphy: Machine Learning. MIT Press 2012

Logistic Regression (LOGREG)

- Two-class problem with Bernoulli RV c taking values in $\{0, 1\}$
  $\rightsquigarrow$ sufficient to represent $P(1|\boldsymbol{x})$, since $P(0|\boldsymbol{x}) = 1 - P(1|\boldsymbol{x})$.
- "Success probability" of the Bernoulli RV: $\pi(\boldsymbol{x}) := P(1|\boldsymbol{x})$.
- Probability of miss ($c = 0$) or hit ($c = 1$) as a function of $\boldsymbol{x}$:

$$p(c|\boldsymbol{x}) = \pi(\boldsymbol{x})^c (1 - \pi(\boldsymbol{x}))^{1-c}, \quad \pi(\boldsymbol{x}) = P(c = 1|\boldsymbol{x}).$$

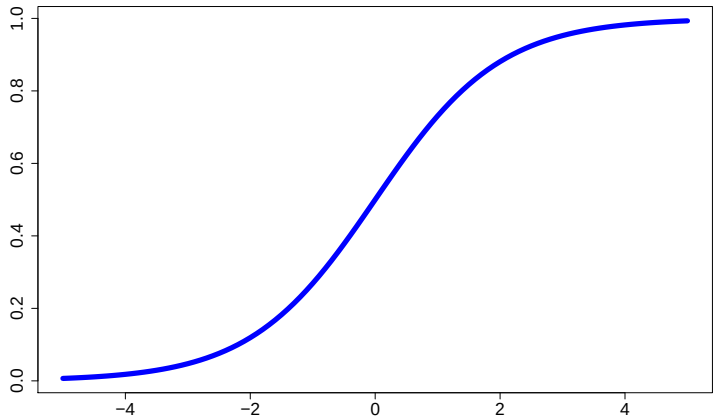
- Basketball example:



Adapted from Fig. 7.5.1 in B. Flury: A first course in multivariate statistics. Springer 1997.

Logistic Regression (LOGREG)

- LOGREG: $g(\boldsymbol{x}) = \boldsymbol{w}^t\boldsymbol{x} + w_0 = \log \frac{P(c=1|\boldsymbol{x})}{P(c=0|\boldsymbol{x})} = \log \frac{\pi(\boldsymbol{x})}{1-\pi(\boldsymbol{x})}$
- This implies $\frac{\pi(\boldsymbol{x})}{1-\pi(\boldsymbol{x})} = \exp\{g(\boldsymbol{x})\}$
  $\Rightarrow \ \pi(\boldsymbol{x}) = P(c=1|\boldsymbol{x}) = \frac{\exp\{g(\boldsymbol{x})\}}{1+\exp\{g(\boldsymbol{x})\}} =: \sigma(g(\boldsymbol{x}))$.
- **Sigmoid** or **logistic** "squashing function" $\sigma(z) = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}}$ turns linear predictions into probabilities



- Simple extension for K classes: the **softmax** function:
  $P(c=k|\boldsymbol{x}) = \frac{\exp\{g_k(\boldsymbol{x})\}}{\sum_{m=1}^{K}\exp\{g_m(\boldsymbol{x})\}}$.

Logistic Regression (LOGREG)

- Assume that w_0 is "absorbed" in $\boldsymbol{w}$ using $\boldsymbol{x} \leftarrow (1, \boldsymbol{x})$. Estimate $\boldsymbol{w}$ by maximizing the conditional likelihood

$$\hat{\boldsymbol{w}}_{DISCR} = \operatorname*{argmax}_{\boldsymbol{w}} \prod_{i=1}^{n} \left(\pi(\boldsymbol{x}_i; \boldsymbol{w})\right)^{c_i} \left(1 - \pi(\boldsymbol{x}_i; \boldsymbol{w})\right)^{1-c_i},$$

  or by **minimizing the negative log likelihood:**

$$-l(\boldsymbol{w}) = -\sum_{i=1}^{n} \left[c_i \log \pi(\boldsymbol{x}_i; \boldsymbol{w}) + (1 - c_i) \log(1 - \pi(\boldsymbol{x}_i; \boldsymbol{w}))\right].$$

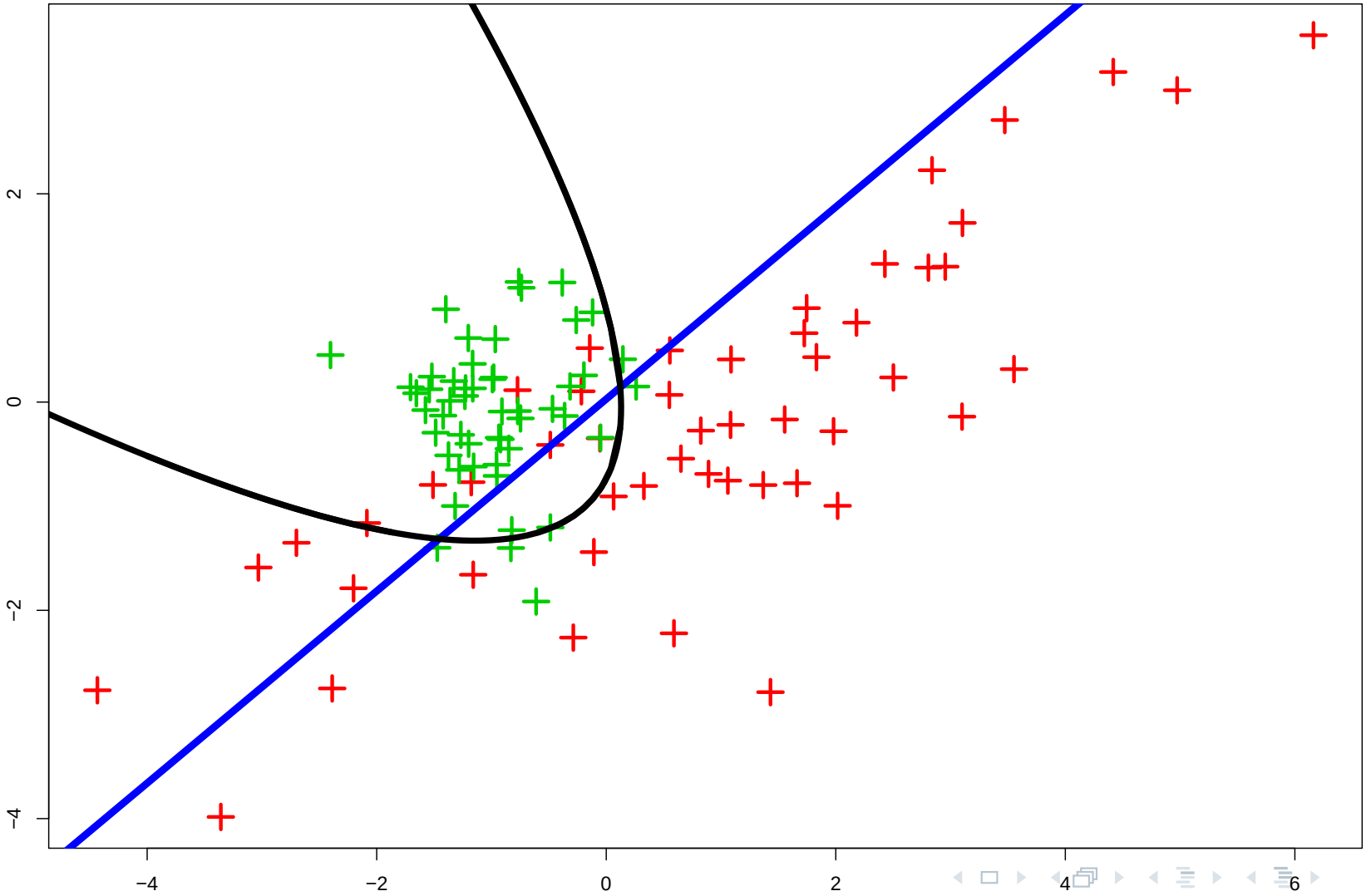
- The the **gradient of l** is:

$$\nabla_{\boldsymbol{w}} l = \frac{\partial}{\partial \boldsymbol{w}} l(\boldsymbol{w}) = \sum_{i=1}^{n} \boldsymbol{x}_i (c_i - \pi_i).$$

- The π_i depend non-linearly on $\boldsymbol{w}$
  $\rightsquigarrow$ equation system $\nabla_{\boldsymbol{w}} l = \mathbf{0}$ cannot be solved analytically
  $\rightsquigarrow$ iterative techniques needed (e.g. gradient descent).

Logistic Regression (LOGREG)

Simple binary classification problem in $\mathbb{R}^2$. Solved with LOGREG using polynomial basis functions.



LOGREG and Perceptron

- Gradient of log-likelihood (at step k):

$$\nabla_{\boldsymbol{w}^{(k)}} l = \frac{\partial}{\partial \boldsymbol{w}} l(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}^{(k)}} = \sum_{i=1}^{n} \boldsymbol{x}_i (c_i - \pi_i^{(k)}).$$

- Gradient descent (for negative log.l.): $\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta \nabla_{\boldsymbol{w}^{(k)}}$.
- Assume stream of data $\rightsquigarrow$ online update for new observation $\boldsymbol{x}_i$:

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta(c_i - \pi_i^{(k)})\boldsymbol{x}_i, \quad \text{with } \pi_i^{(k)} = P(c = 1|\boldsymbol{x}_i, \boldsymbol{w}^{(k)}).$$

- Now consider approximation: define **most probable label** $\hat{c}_i = \arg\max_{c \in \{0,1\}} P(c|\boldsymbol{x}_i, \boldsymbol{w}^{(k)})$ and replace π_i with $\hat{c}_i$.
- If we predicted correctly, then $\hat{c}_i = c_i \rightsquigarrow$ approximate gradient is zero $\rightsquigarrow$ update has no effect.
- If $\hat{c}_i = 0$ but $c_i = 1$: $\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta(c_i - \hat{c}_i)\boldsymbol{x}_i = \boldsymbol{w}^{(k)} + \eta \boldsymbol{x}_i$.
- Note that **this is again the perceptron algorithm.**
- Solution to most problems of the classical perceptron: use exact gradient instead of approximation based on most probable labels. **We do this in modern Multi Layer Perceptrons (MLP).**

Loss function

- LOGREG minimizes the negative log likelihood

$$-l(\boldsymbol{w}) = -\sum_{i=1}^{n} \left[c_i \log \pi(\boldsymbol{x}_i; \boldsymbol{w}) + (1 - c_i) \log(1 - \pi(\boldsymbol{x}_i; \boldsymbol{w})) \right],$$

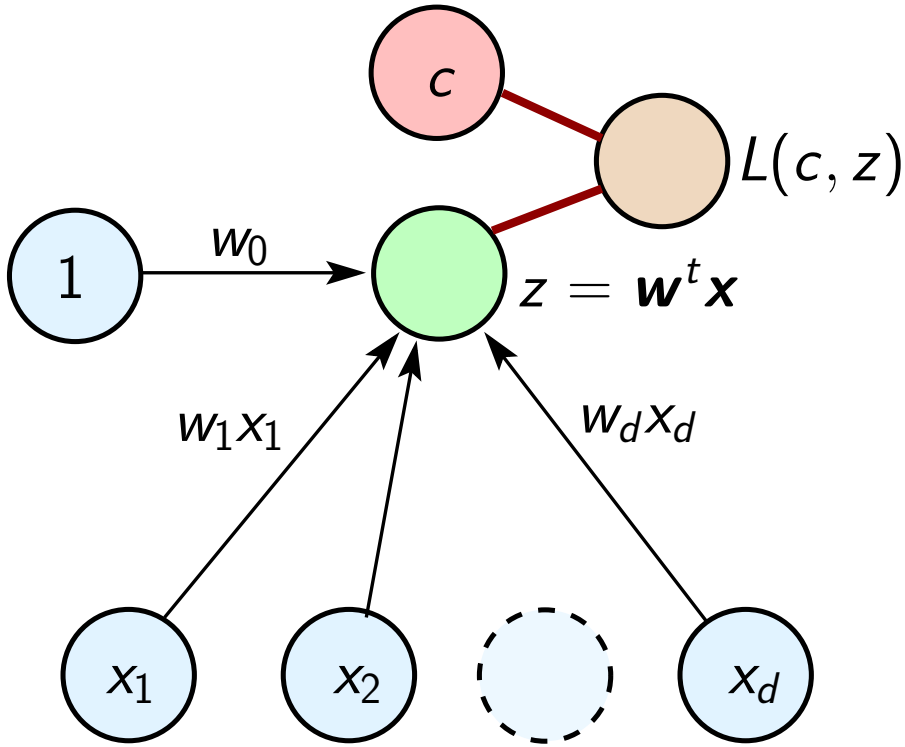
  where $z = \boldsymbol{w}^t \boldsymbol{x}$, $\pi = \frac{1}{1+e^{-z}}$, $1 - \pi = \frac{e^{-z}}{1+e^{-z}}$.

- Introducing a **loss function**, we can write this as minimizing the average loss

$$\frac{1}{n} \sum_{i=1}^{n} \text{Loss}(c_i, z_i).$$

The perceptron / LOGREG

Can be viewed as an output layer in a neural network:



Adding additional layers $\rightsquigarrow$ **Multi-Layer Perceptrons (MLP).**