Machine Learning 2020

Volker Roth

Department of Mathematics & Computer Science
University of Basel

16th March 2020

Section 3

Classification

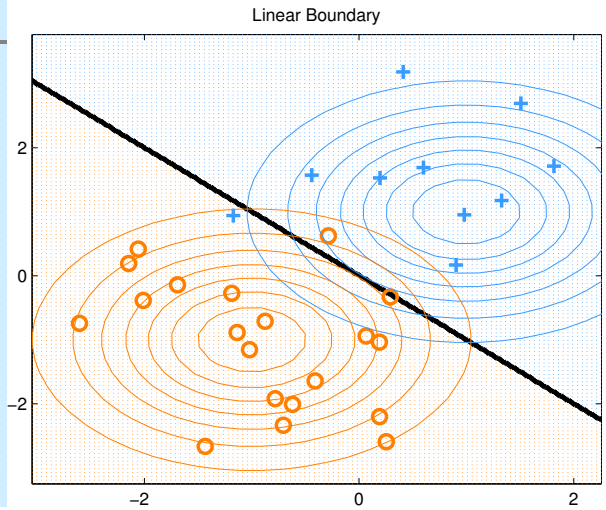
Classification

Example

Sorting fish according to species using optical sensing

Features:

- Length
- Brightness
- Width
- Shape of fins



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 can be defined by simply **counting misclassifications:** $L_{ij} = 1 - \delta_{ij}$, 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**

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

- Link to regression: use **encoding** $\{+1, -1\}$ for the two possible states $c_{1,2}$ of c. The conditional expectation of $c|\boldsymbol{x}$ equals the Bayes discriminant function:

$$E[c|\boldsymbol{x}] = \sum_{c \in \{+1,-1\}} cP(c|\boldsymbol{x}) = P(c_1|\boldsymbol{x}) - P(c_2|\boldsymbol{x}) = G(\boldsymbol{x}).$$

- Classification can be viewed as a (local) approximation of $G(\boldsymbol{x}) = E[c|\boldsymbol{x}]$ near its zero crossings.

Linear Discriminant Functions

- Problem: direct approximation of G would require the knowledge of the Bayes optimal discriminant.
- One approach: 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 is the set of 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 $g(\boldsymbol{x}; \boldsymbol{w}) < 0$.
- Equation $g(\boldsymbol{x}; \boldsymbol{w}) = 0$ defines the **decision surface**.
  Linearity of $g(\boldsymbol{x}; \boldsymbol{w}) \rightsquigarrow$ **hyperplane** $\rightsquigarrow$ $\boldsymbol{w}$ is orthogonal to any vector lying in the plane.
- 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: $\boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$.
- $g(\boldsymbol{x}_p) = 0 \Rightarrow g(\boldsymbol{x}) = r\|\boldsymbol{w}\| \quad \Leftrightarrow \quad r = g(\boldsymbol{x})/\|\boldsymbol{w}\|$.

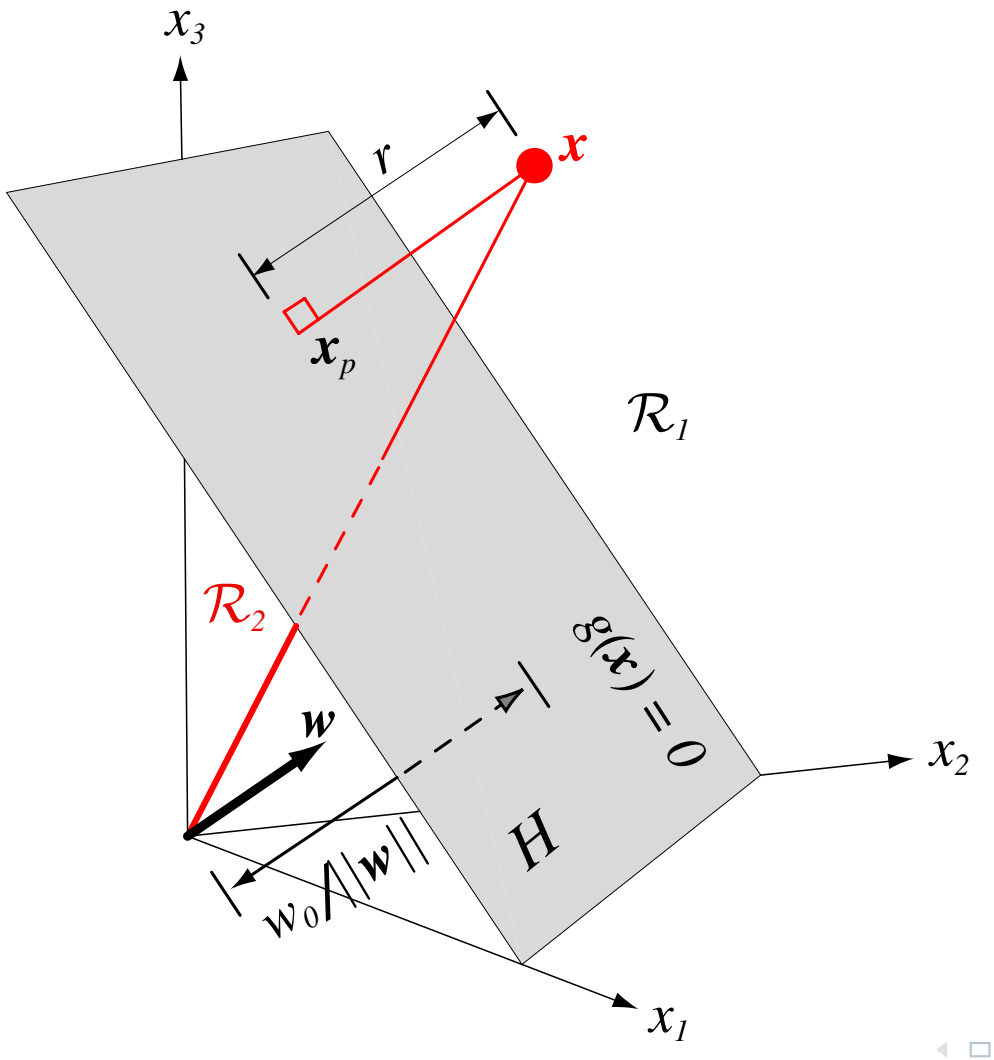


Fig 5.2 in (Duda& Hart)

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)

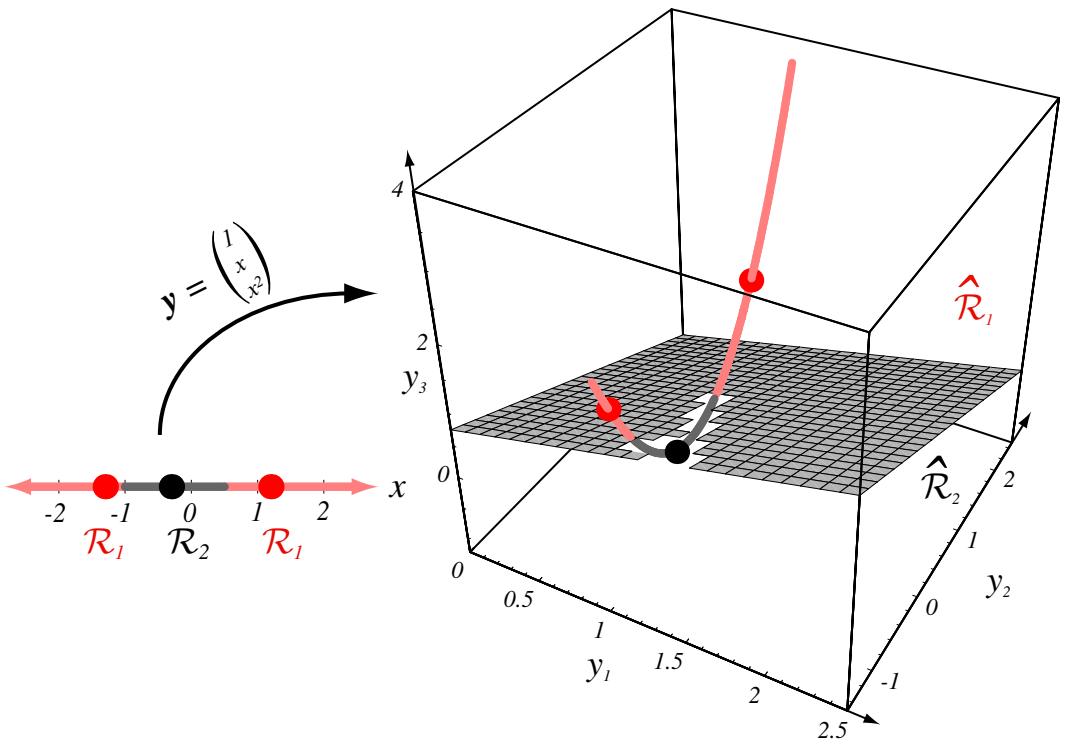


Fig 5.5 in (Duda& Hart)

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

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

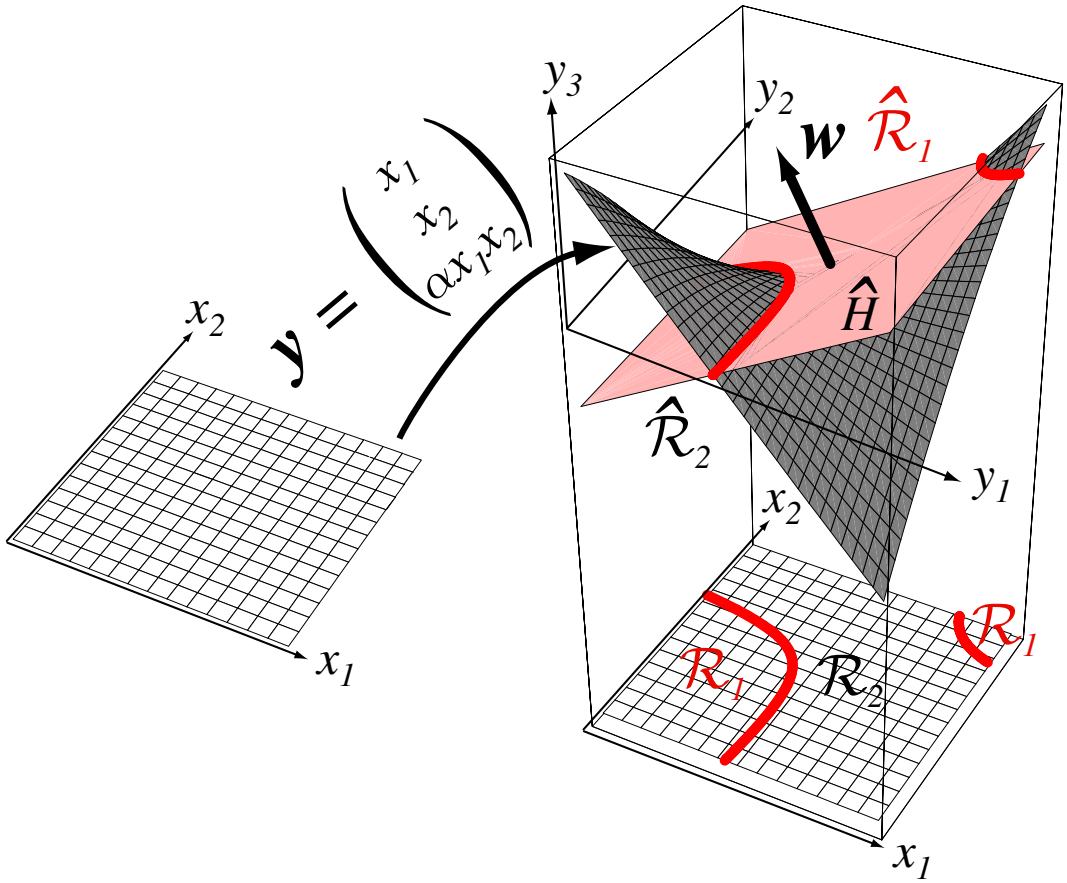
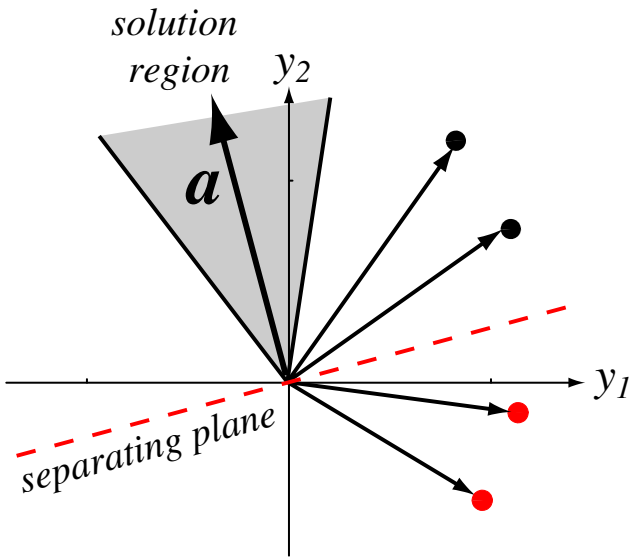


Fig 5.6 in (Duda& Hart)

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**.



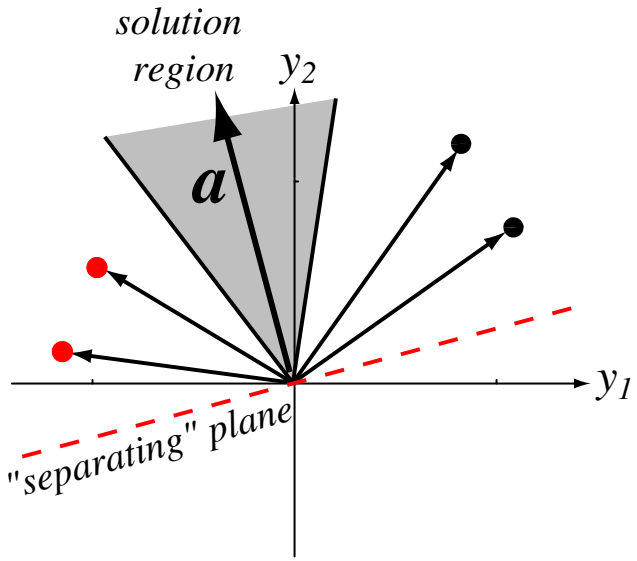


Fig 5.8 in (Duda& Hart)

Separable Case: margin

- Different solution vectors may have different **margins** b : $\boldsymbol{y}^t \boldsymbol{w} \geq b > 0$.
- Intuitively, **large margins are good.**

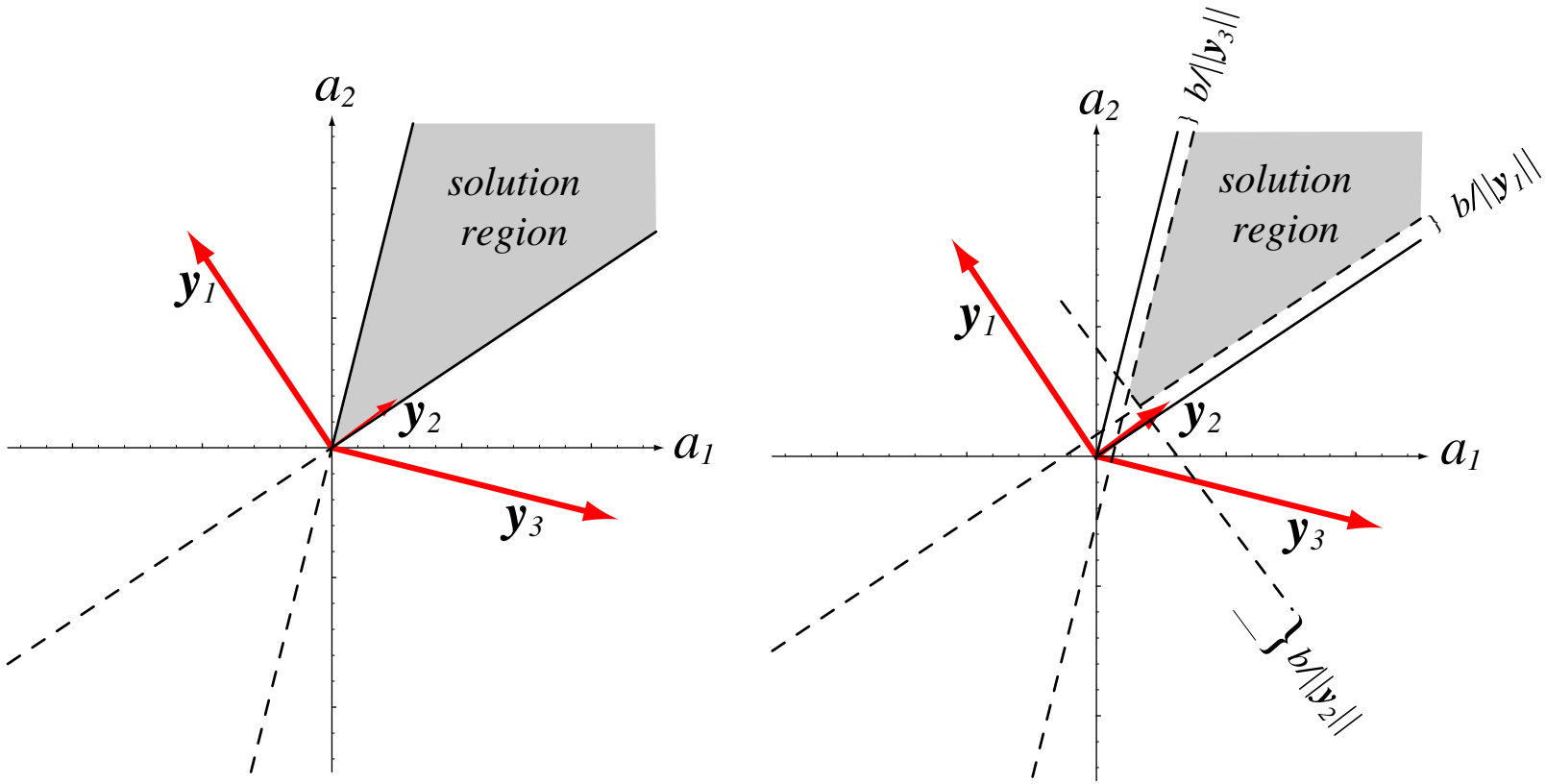


Fig 5.9 in (Duda& Hart)

Gradient Descent

- Solve $\boldsymbol{y}^t \boldsymbol{w} > 0$ by defining $J(\boldsymbol{w})$ such that a minimizer of J is a solution.
- Start with initial $\boldsymbol{w}(1)$, and choose next value by moving in the direction of steepest gradient: $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \eta(k) \nabla J(\boldsymbol{w}(k))$.
- Alternatively, use second order expansion (Newton): $\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - H^{-1} \nabla J(\boldsymbol{w}(k))$.

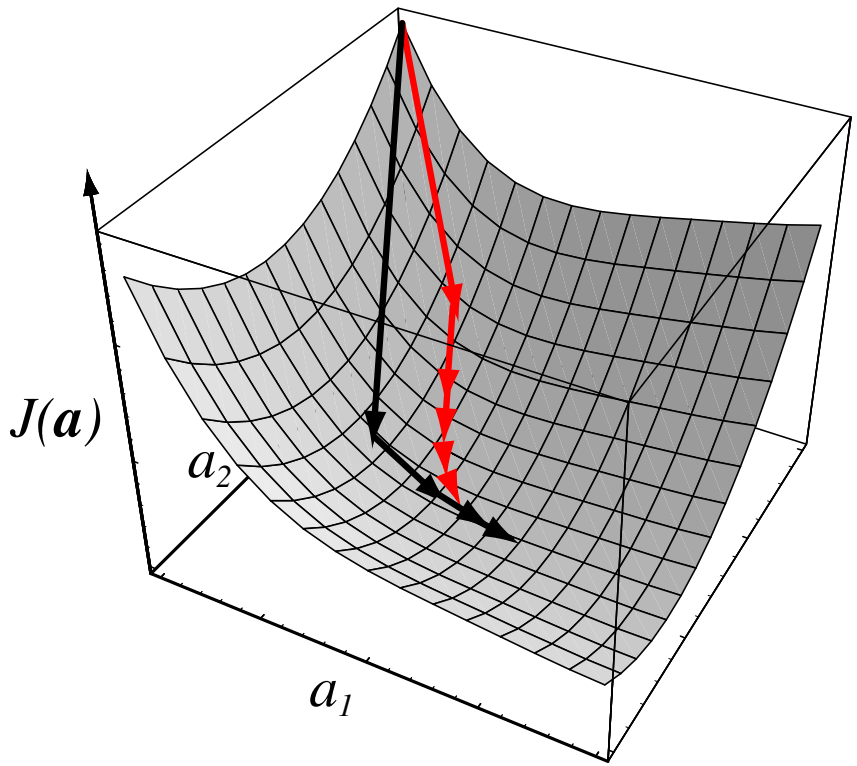


Fig 5.10 in (Duda& Hart)

Minimizing the Perceptron Criterion

- Solve $\boldsymbol{y}^t \boldsymbol{w} > 0$ by defining $J(\boldsymbol{w})$ such that a minimizer of J is a solution.
- Most obvious: number of misclassifications, but **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 $\boldsymbol{w}$.
- Since $\boldsymbol{y}^t \boldsymbol{w} < 0 \; \forall \boldsymbol{y} \in \mathcal{M}$, J_p is non-negative, and zero only if $\boldsymbol{w}$ is a solution.
- Gradient: $\nabla J(\boldsymbol{w}) = -\sum_{\boldsymbol{y} \in \mathcal{M}} \boldsymbol{y}$

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

This defines the **Batch Perceptron algorithm.**

Minimizing the Perceptron Criterion (2)

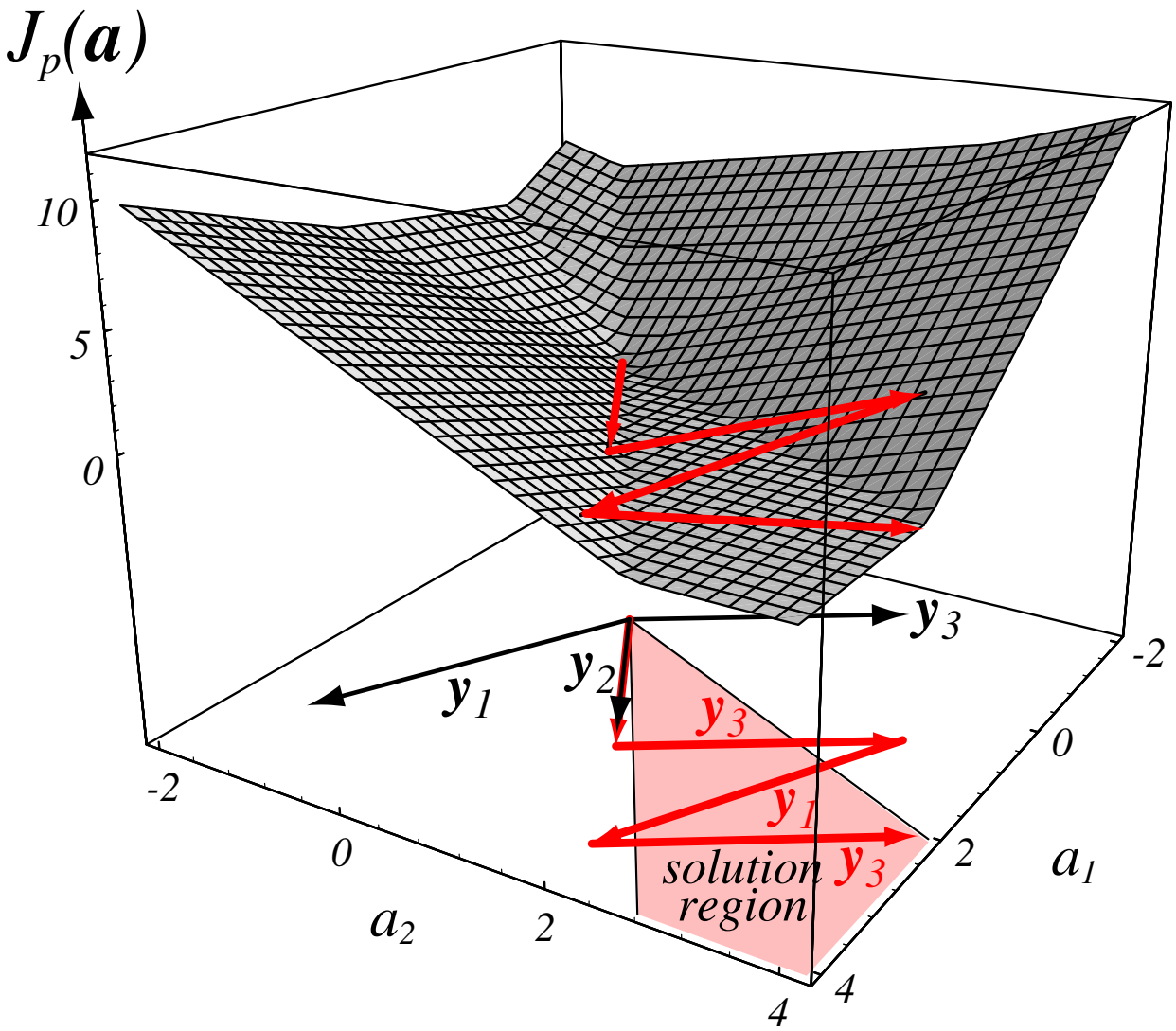


Fig 5.12 in (Duda& Hart)

Fixed-Increment Single Sample Perceptron

- Fix learning rate $\eta(k) = 1$.
- 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.

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 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 class conditional density $p_{\boldsymbol{w}_j}(\boldsymbol{x}|c=j)$ and collecting all model parameters in a vector $\boldsymbol{w}$, a typical (Frequentist) approach now proceeds by maximizing the log likelihood

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

- The resulting estimate $\hat{\boldsymbol{w}}_{MLE}$ might then be plugged into Bayes rule to compute the posteriors:

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

Generative Classifiers: LDA

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

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

- 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 $\boldsymbol{\mu}_j$:

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

- Fisher's LDA finds the weight vector

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

This $\boldsymbol{w}^F$ asymptotically coincides with the Bayes-optimal $\boldsymbol{w}$ **if Gaussian model is correct**.

LDA

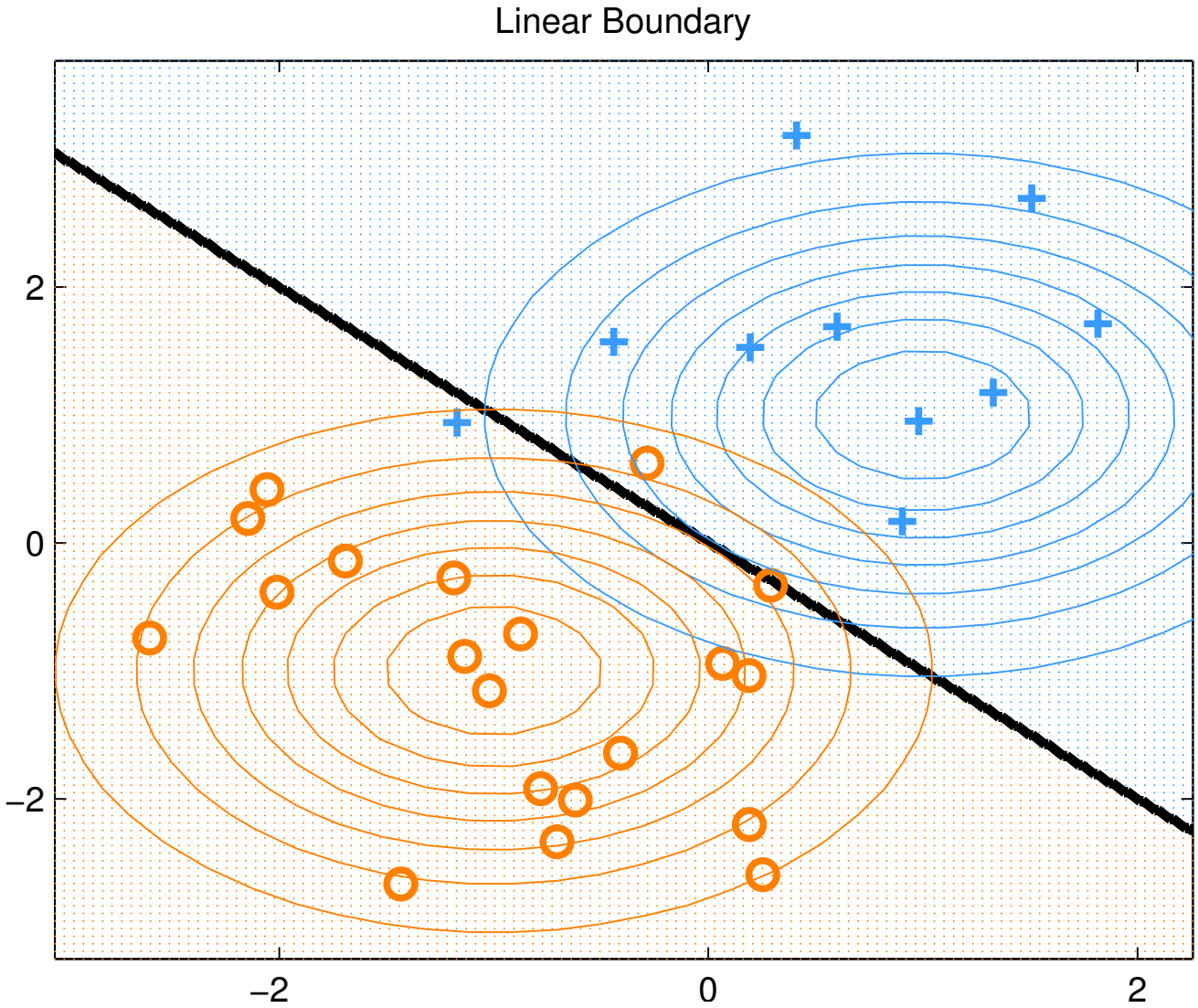


Fig 4.5 in K. Murphy

Fishers discriminant and least squares

Remark: The Fisher vector $\hat{\boldsymbol{w}}^F = \Sigma_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2)$ coincides with the solution of the LS problem $\hat{\boldsymbol{w}}^{LS} = \arg\min_{\boldsymbol{w}} \|A\boldsymbol{w} - \boldsymbol{b}\|^2$ if

$$
\begin{aligned}
n_1 &= \#\text{ samples in class } \mathbf{1} \\
n_2 &= \#\text{ samples in class } \mathbf{2} \\
\boldsymbol{b} &= \begin{pmatrix} +1/n_1 \\ \cdot \\ +1/n_1 \\ -1/n_2 \\ \cdot \\ -1/n_2 \end{pmatrix}, \qquad A = \begin{pmatrix} \boldsymbol{x}_1^t \\ \cdot \\ \boldsymbol{x}_{n_1}^t \\ \boldsymbol{x}_{n_1+1}^t \\ \cdot \\ \boldsymbol{x}_{n_1+n_2}^t \end{pmatrix},
\end{aligned}
$$

with $\boldsymbol{m} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i = \mathbf{0}$ (i.e. origin in sample mean).

Fishers discriminant and least squares (cont'd)

Proofsketch:

- Shared covariance matrix also called "within class covariance"
  $\Sigma_W \propto \sum_{\boldsymbol{x} \in \mathcal{X}_c} (\boldsymbol{x} - \boldsymbol{m}_c)(\boldsymbol{x} - \boldsymbol{m}_c)^t, \quad c = c_1, \text{or } c = c_2.$
- Its counterpart is the "between class covariance"
  $\Sigma_B \propto (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^t$
- The sum of both is the "total covariance" $\Sigma_B + \Sigma_W = \Sigma_T$
  $\Sigma_T \propto \sum_i (\boldsymbol{x}_i - \boldsymbol{m})(\boldsymbol{x}_i - \boldsymbol{m})^t = A^t A.$
- We know that $\boldsymbol{w}^F \propto \Sigma_W^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) \rightsquigarrow \Sigma_W \boldsymbol{w}^F \propto (\boldsymbol{m}_1 - \boldsymbol{m}_2).$
- Now $\Sigma_B \boldsymbol{w}^F = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^t \boldsymbol{w}^F \rightsquigarrow \Sigma_B \boldsymbol{w}^F \propto (\boldsymbol{m}_1 - \boldsymbol{m}_2).$
- $\Sigma_T \boldsymbol{w}^F = (\Sigma_B + \Sigma_W)\boldsymbol{w}^F \rightsquigarrow \Sigma_T \boldsymbol{w}^F \propto (\boldsymbol{m}_1 - \boldsymbol{m}_2).$
- With $A^t A = \Sigma_T$, $A^t \boldsymbol{b} = \boldsymbol{m}_1 - \boldsymbol{m}_2$, we arrive at
  $\boldsymbol{w}^F \propto \Sigma_T^{-1}(\boldsymbol{m}_1 - \boldsymbol{m}_2) = \Sigma_T^{-1} A^t \boldsymbol{b} = (A^t A)^{-1} A^t \boldsymbol{b} = \boldsymbol{w}^{LS}.$

Fishers discriminant and least squares (cont'd)

- Focus on last equation. For notational simplicity, denote the least-squares estimate $\boldsymbol{w}^{LS}$ by $\boldsymbol{w}$.
- Introducing the "residual sum of squares" as the least-squares cost function, the equation follows from:

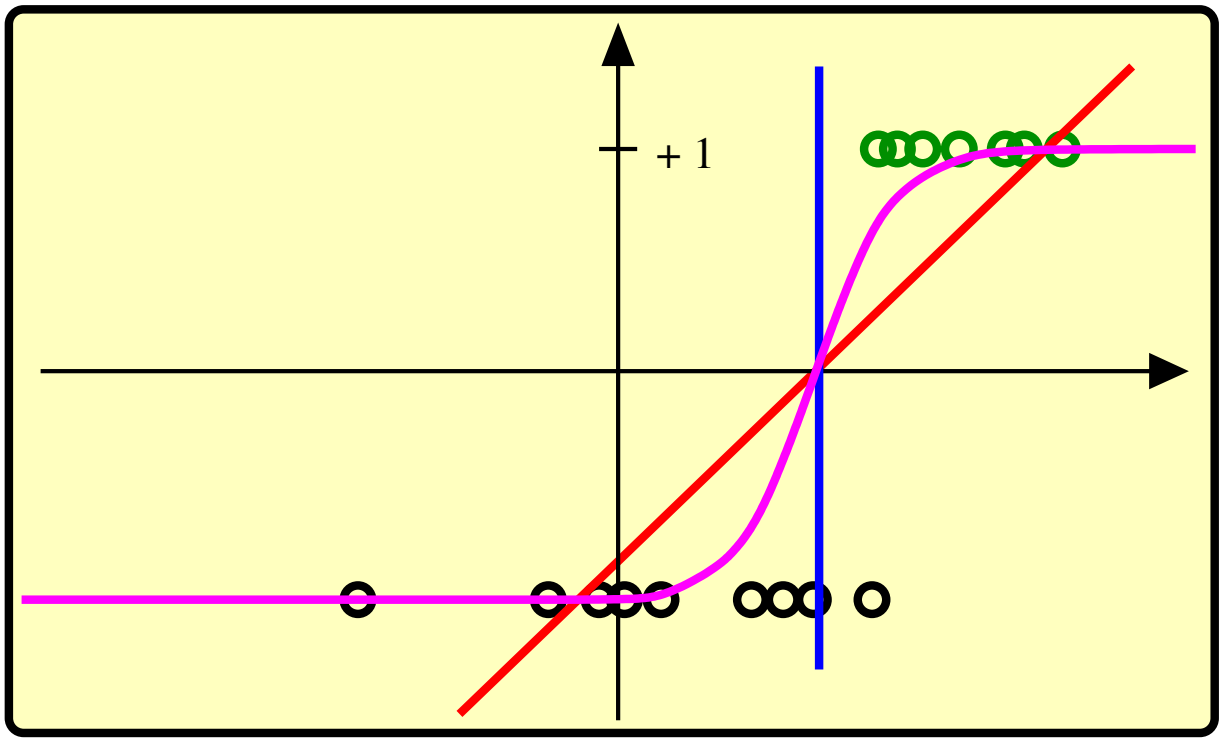
$$\begin{aligned} RSS(\boldsymbol{w}) &= \sum_{i=1}^{n} [b_i - \boldsymbol{w}^t \boldsymbol{x}_i]^2 \\ \frac{\partial RSS(\boldsymbol{w})}{\partial \boldsymbol{w}} &= \frac{\partial}{\partial \boldsymbol{w}} \left[\boldsymbol{b}^t \boldsymbol{b} - 2\boldsymbol{b}^t A \boldsymbol{w} + \boldsymbol{w}^t A^t A \boldsymbol{w}\right] \\ &= -2A^t \boldsymbol{b} + 2A^t A \boldsymbol{w} \stackrel{!}{=} \boldsymbol{0} \Rightarrow \boldsymbol{w} = (A^t A)^{-1} A^t \boldsymbol{b}. \end{aligned}$$

  $A^t \boldsymbol{b} = A^t A \boldsymbol{w}$ are called the **normal equations.**

- We have used the following results from matrix calculus:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{y}^t \boldsymbol{x} &= \boldsymbol{y} \\ \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^t M \boldsymbol{x} &= 2M\boldsymbol{x}, \text{ if } M \text{ is symmetric} \end{aligned}$$

Fishers discriminant and least squares (cont'd)



- Two-class LDA solution viewed as indicator regression.
- Magenta curve: Bayes-optimal discriminant function $G(\boldsymbol{x}) = P(c = +1|\boldsymbol{x}) - P(c = -1|\boldsymbol{x})$
- Red line: Regression fit $\rightsquigarrow$ zero crossing determines the separating hyperplane (vertical blue line).

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:

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

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

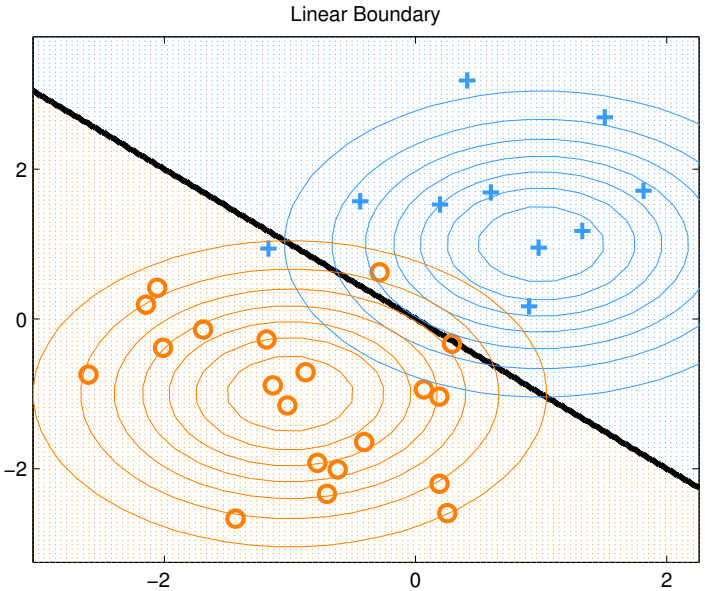


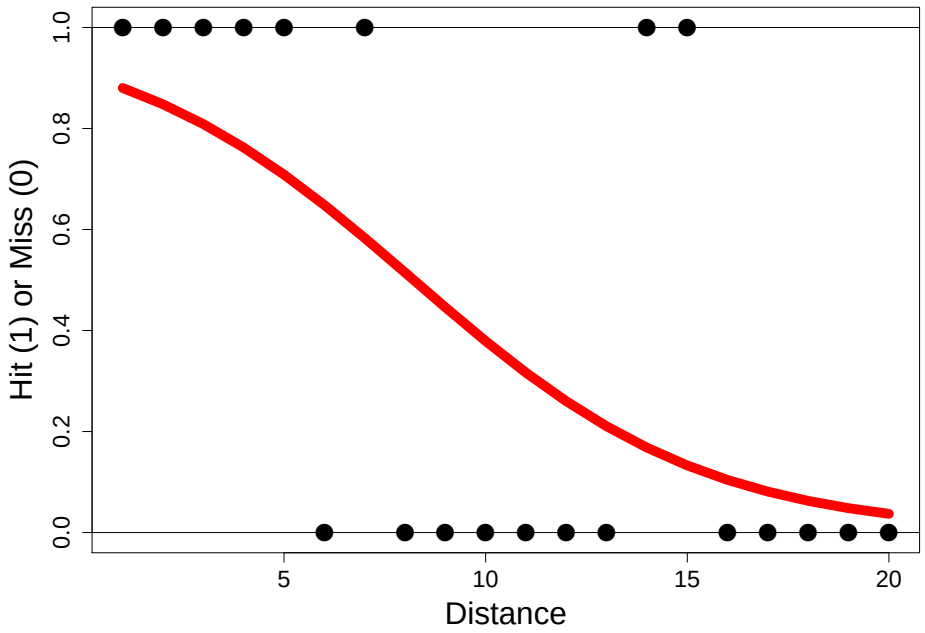
Fig 4.5 in K. Murphy

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 $\boldsymbol{x}$:

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

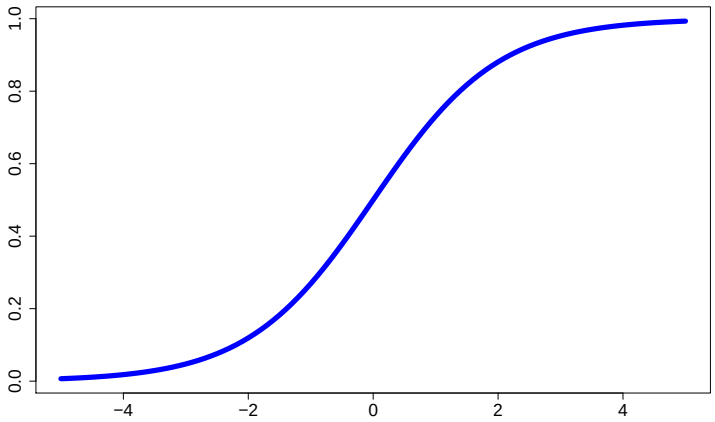
- Basketball example:



Adapted from Fig. 7.5.1 in (B. Flury)

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

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

  or by maximizing the corresponding log likelihood l:

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

- The **score functions** are defined as the gradient of l:

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

Logistic Regression (LOGREG)

- π_i depends non-linearly on $\boldsymbol{w}$
  $\rightsquigarrow$ equation system $\boldsymbol{s}(\boldsymbol{w}) = \boldsymbol{0}$ cannot be solved analytically
  $\rightsquigarrow$ iterative techniques needed.
- Newton's method: Update $\boldsymbol{w}$ at the r-th step as

$$\boldsymbol{w}^{(r+1)} = \boldsymbol{w}^{(r)} + \{H^{(r)}\}^{-1}\boldsymbol{s}^{(r)},$$

where $H^{(r)}$ is the Hessian of l, evaluated at $\boldsymbol{w}^{(r)}$:

$$\begin{aligned} H^{(r)} &= \left(\frac{\partial^2 l}{\partial \boldsymbol{w}\, \partial \boldsymbol{w}^t}\right)\Bigg|_{\boldsymbol{w}=\boldsymbol{w}^{(r)}} \\ &= \sum_{i=1}^{n} \pi_i^{(r)}(1-\pi_i^{(r)})\boldsymbol{x}_i\boldsymbol{x}_i^t. \end{aligned}$$

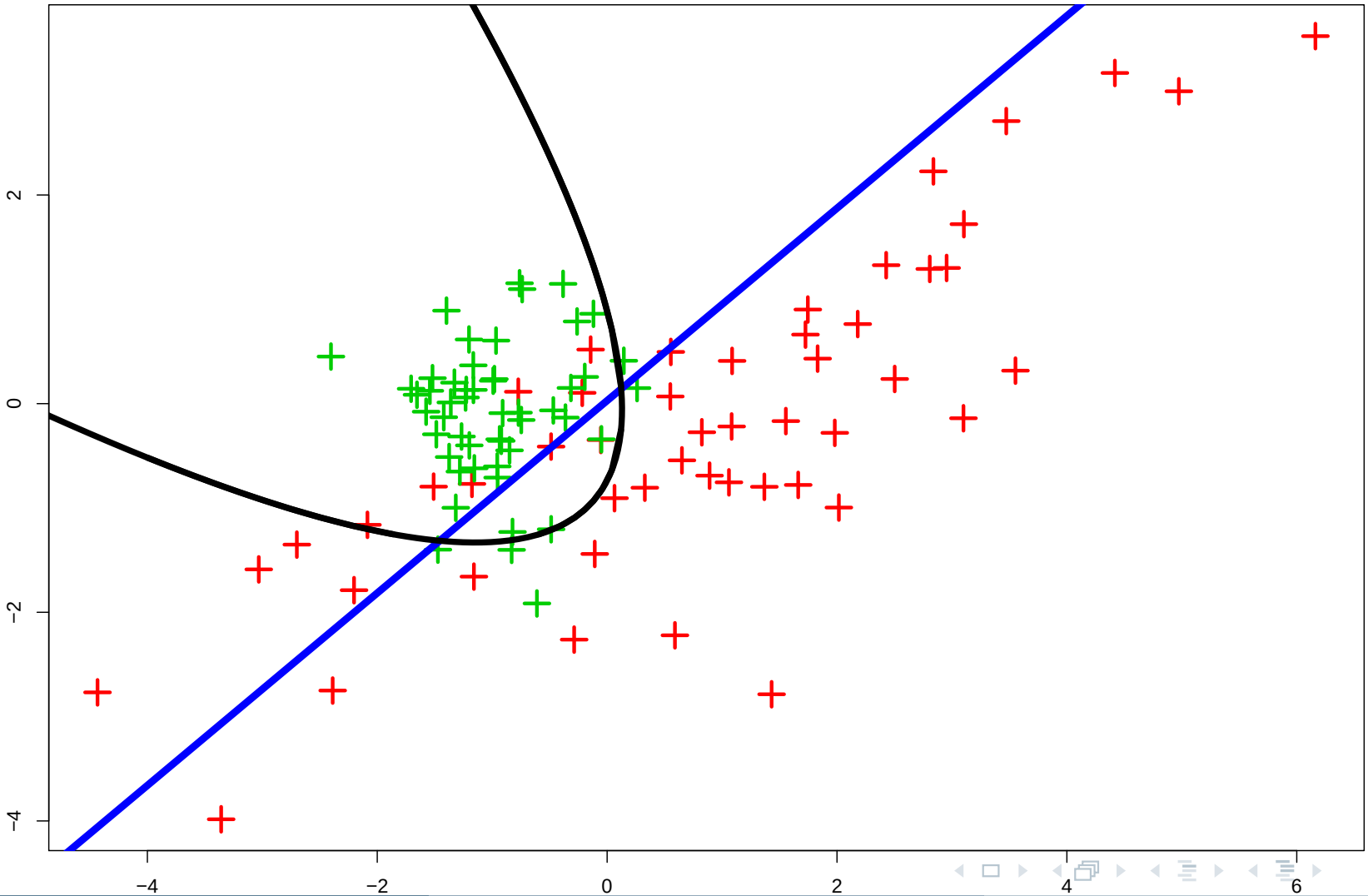
Logistic Regression (LOGREG)

Newton updates $\rightsquigarrow$ **Iterated Re-weighted Least Squares (IRLS):**

- The **Hessian** $H^{(r)}$ is equal to $(X^t W^{(r)} X)$, with
$$W = \text{diag}\,\{\pi_1(1-\pi_1), \ldots, \pi_n(1-\pi_n)\}.$$
- **Score functions**: $\boldsymbol{s}^{(r)} = X^t W^{(r)} \boldsymbol{e}^{(r)}$, where $\boldsymbol{e}$ is a vector with entries
$$e_j = (c_j - \pi_j)/W_{jj}.$$
- With $\boldsymbol{q}^{(r)} := X\boldsymbol{w}^{(r)} + \boldsymbol{e}^{(r)}$, the updates read
$$\begin{aligned} H^{(r)}\boldsymbol{w}^{(r+1)} &= H^{(r)}\boldsymbol{w}^{(r)} + s^{(r)} \\ (X^t W^{(r)} X)\boldsymbol{w}^{(r+1)} &= X^t W^{(r)} \boldsymbol{q}^{(r)}. \end{aligned}$$
- These are the **normal equations of a LS problem** $\|A\boldsymbol{w} - \boldsymbol{b}\|^2$ with input matrix $A = (W^{(r)})^{1/2} X$ and r.h.s. $\boldsymbol{b} = (W^{(r)})^{1/2} \boldsymbol{q}^{(r)}$.
- The values W_{ii} are functions of $\boldsymbol{w}$ $\rightsquigarrow$ iteration is needed.

Logistic Regression (LOGREG)

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



Loss functions

- LOGREG maximizes log likelihood

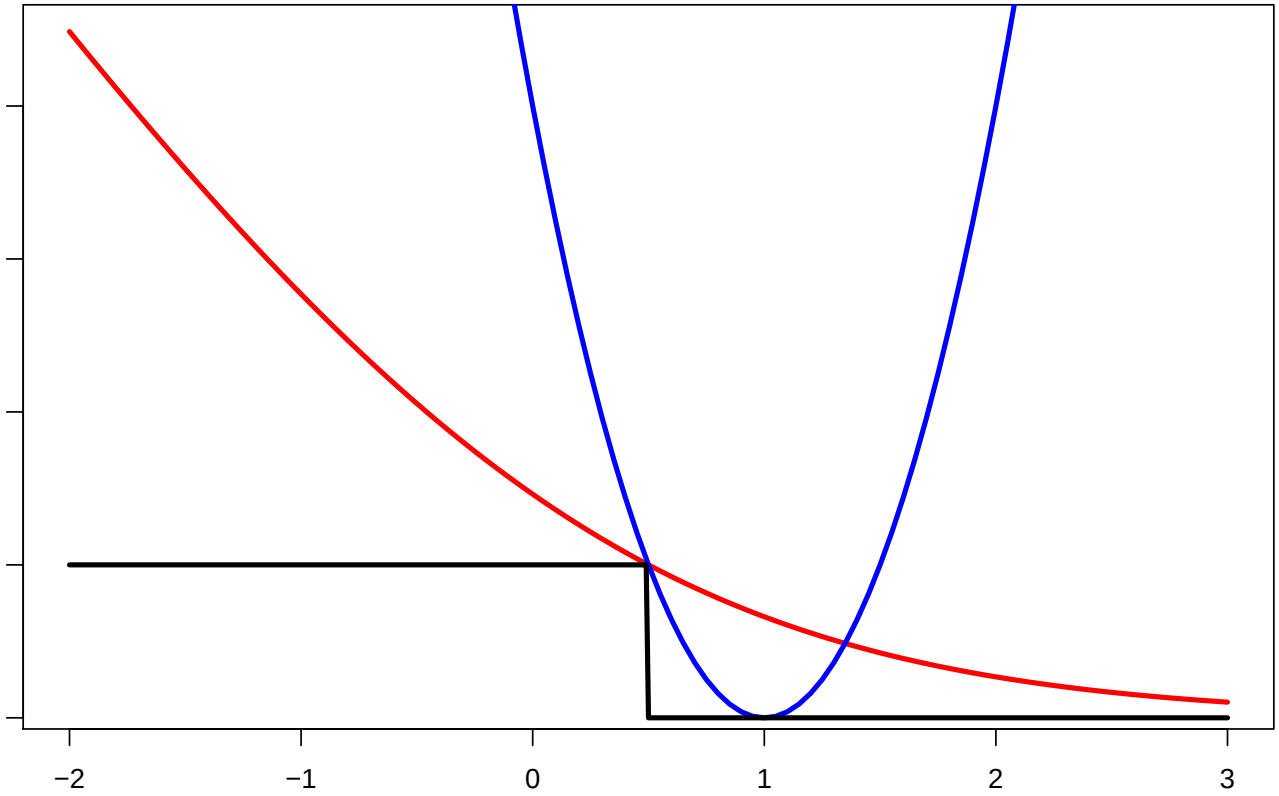
$$l(\boldsymbol{w}) = \sum_{i=1}^{n} \left[c_i \log \pi(\boldsymbol{x}_i; \boldsymbol{w}) + (1 - c_i) \log(1 - \pi(\boldsymbol{x}_i; \boldsymbol{w})) \right],$$

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

- This is the same as minimizing

$$\begin{aligned} -l(\boldsymbol{w}) &= \sum_{i=1}^{n} \left[-c_i \log \pi - (1 - c_i) \log(1 - \pi) \right] \\ &=: \sum_{i=1}^{n} \text{Loss}(c_i, z_i). \end{aligned}$$

Loss functions



Using $\{0, 1\}$ encoding of the two classes, and approximating a target with $c = +1$. Black: 0/1-loss, red: logistic loss, blue: quadratic loss (LDA).

LOGREG and Perceptron

- Gradient of negative log-likelihood:

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

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

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

- Now consider approximation: define **most probable label** $\hat{c}_i = \arg\max_{c \in \{0,1\}} P(c|\boldsymbol{x}_i, \boldsymbol{w}^{(r)})$ 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 $c_i = 1$: $\boldsymbol{w}^{(r+1)} = \boldsymbol{w}^{(r)} - \eta(\hat{c}_i - c_i)\boldsymbol{x}_i = \boldsymbol{w}^{(r)} + \eta \boldsymbol{x}_i$.
- Note that this is again the **perceptron algorithm.**
- Simple solution to most problems of the perceptron: use exact gradient instead of approximation based on most probable labels.