# Lecture 6: Linear Classification

Isabel Valera

Machine Learning Group
Department of Mathematics and Computer Science
Saarland University, Saarbrücken, Germany

06.05.2023

# Outline

# Main references

- Bishop - Chapter 4
- ESL - Chapter 4

# Outline

# Classification setting

**Binary classification setting:** We consider problems where the output (target) variable takes values in $\mathcal{Y} = \{-1, +1\}$. Moreover, we assume we have access to training data $(\mathbf{x}_i, y_i)_{i=1}^n$, which is an i.i.d. sample from the probability measure $\mathrm{P}$ on $\mathcal{X} \times \mathcal{Y}$.

**Bayes classifier**, which decides according to $y^*(\mathbf{x}) = sign(\mathbb{E}[Y|X = \mathbf{x}])$, is optimal.

**Goal:** Learn a mapping function $\hat{y}(X)$ that minimizes the probability of error, i.e., $P(error) = R(\hat{y}) = \mathbb{E}\left[\mathbb{1}_{\hat{y}(X)Y \leq 0}\right]$. To this end, we often assume that $\hat{y}(X) = sign(f(X))$ and focus on learning (using training data) the discriminant function $f(X)$ that minimizes a surrogate loss that is convex and upperbounds the 0-1-loss (refer to Lecture 2).

**Linear classification** considers a family of functions $f \in \mathcal{F}$ that are linear, i.e., it takes the form $\left\{ \langle \mathbf{w}, \mathbf{x} \rangle + b, \text{ with } \mathbf{x}, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \right\}$.

# Linear discriminant function

Linear discriminant functions: $f(\mathbf{x}) \in \left\{ \langle \mathbf{w}, \mathbf{x} \rangle + b \right\}$

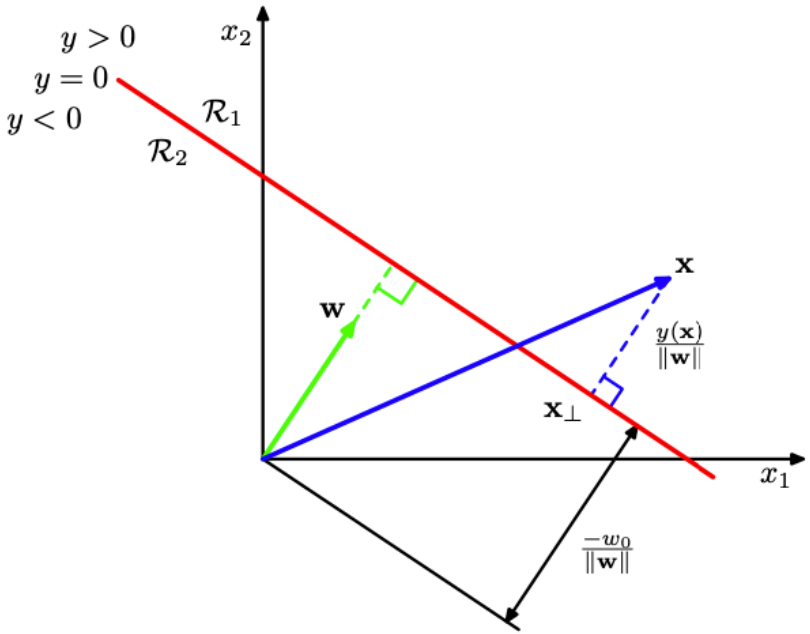



Figure: Example of linear discriminant function in $\mathcal{X} = \mathbb{R}^2$

- Any pair of points $\mathbf{x}_a$ and $\mathbf{x}_b$ lying on the decision surface satisfy that $f(\mathbf{x}_a) = f(\mathbf{x}_b) = 0$ and that $\langle \mathbf{w}, \mathbf{x}_a - \mathbf{x}_b \rangle = 0$, as $\mathbf{w}$ is orthonormal to any vector lying in the decision surface.
- $f(\mathbf{x})$ gives a signed measure of the perpendicular distance $r = \frac{f(\mathbf{x})}{\|\mathbf{w}\|}$ of the point $\mathbf{x}$ to the decision surface.
- Thus, we may use such a (hyper-)plane to divide $\Re^d$ into two regions $\mathcal{R}_{-1}$ and $\mathcal{R}_{+1}$ (in the image, $\mathcal{R}_2$ and $\mathcal{R}_1$, respectively) .

# Linear Classification

Let $\mathcal{X} = \mathbb{R}^d$ be the input space, then a linear binary classifier $\hat{y} : \mathbb{R}^d \to \{-1, 1\}$ has the form

$$\hat{y}(x) = \text{sign}(f(\mathbf{x})) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) = \begin{cases} 1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle + b > 0, \\ -1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle + b \leq 0. \end{cases}$$

**Separation of the input space $\mathbb{R}^d$ into two half spaces.**

*Observation:* From now on we will include the bias term directly in the weight vector $\mathbf{w}$.

# Decision boundary

## Decision boundary

**Definition**

Let $f : \mathcal{X} \to \mathbb{R}$ be a function and $\hat{y}(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$ be the resulting classifier with output in $\mathcal{Y} = \{-1, 1\}$, then we call the set

$$\{\mathbf{x} \in \mathcal{X} \,|\, f(\mathbf{x}) = 0\},$$

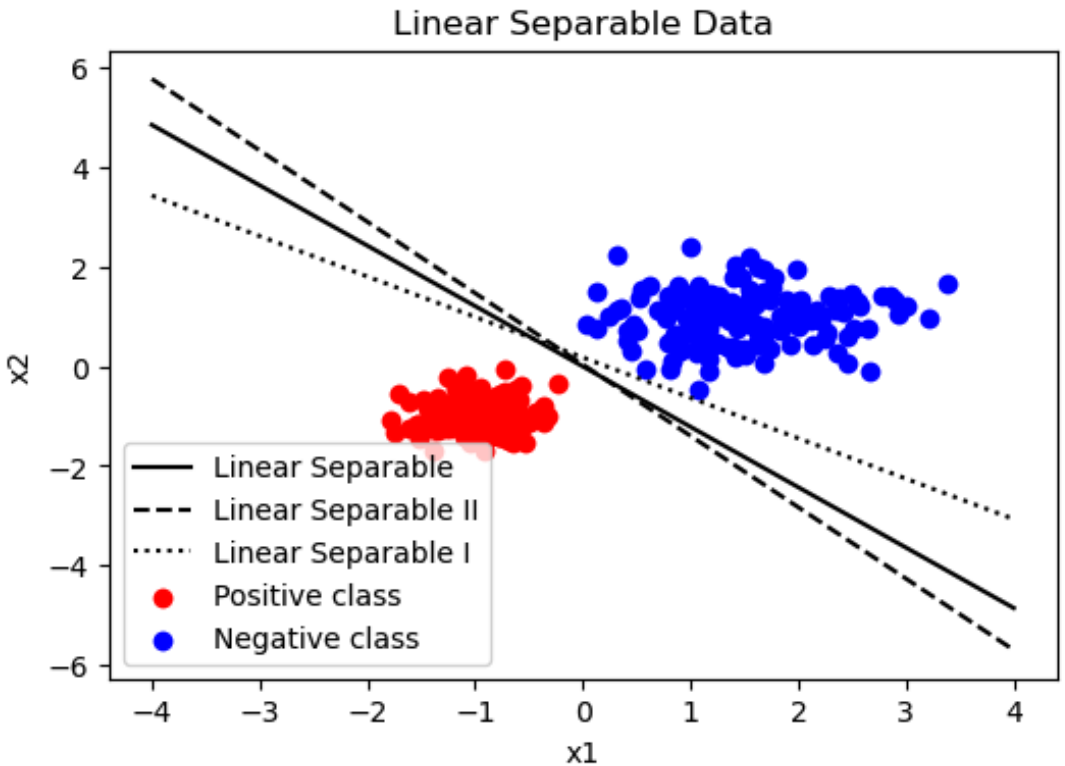
the **decision boundary** of the classifier $\hat{y}$.

A training set $D = (\mathbf{x}_i, y_i)_{i=1}^n$ is **linearly separable** if there exists a weight vector $\mathbf{w}$ such that,

$$y_i \, f(\mathbf{x}_i) = y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle\right) > 0, \qquad \forall i = 1, \ldots, n,$$

$\Rightarrow$ There exists a **hyperplane** $\{\mathbf{x} \in \mathbb{R}^d \,|\, \langle \mathbf{w}, \mathbf{x} \rangle = 0\}$ which separates the sets $\mathbf{X}_+ = \{\mathbf{x}_i \in D \,|\, y_i = 1\}$ and $\mathbf{X}_- = \{\mathbf{x}_i \in D \,|\, y_i = -1\}$.

# Example




A training sample of a two-class problem in $\mathbb{R}^2$. The two classes are linearly separable and three different decision hyperplanes are shown which separate the two classes.

# Basis functions

As in linear regression we may apply basis functions to map the feature vectors $\mathcal{X} = \mathbb{R}^d$ into a possibly larger **feature space** $\mathbb{R}^D$, i.e.,

$$\mathbf{x} \in \mathbb{R}^d \longrightarrow (\phi_1(\mathbf{x}), \ldots, \phi_D(\mathbf{x})),$$

such that we define the linear binary classifier as

$$\hat{y}(x) = \text{sign}(f(x)) = \text{sign}(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle)$$

**The discriminant function is linear in the parameters $\mathbf{w}$ but not necessarily linear in the input space!**

*Observation:* In the following we will make no distinction between using or not using basis functions $\Phi$.

# Examples of linear classification models/methods

**Three linear methods:** $\hat{y}(x) = \text{sign}(f(\mathbf{x})) = \text{sign}(\langle \mathbf{w}, \Phi(\mathbf{x}) \rangle)$.

- **Linear Discriminant Analysis**:
  - Loss: Squared loss, $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
  - Regularization: none
- **Logistic Regression**:
  - Loss: Logistic loss, $L(y, f(\mathbf{x})) = \log(1 + \exp(-y\, f(\mathbf{x})))$
  - Regularization: usually none, but there exist regularized versions.
- **Support Vector Machines** (Lecture 13).
  - Loss: hinge loss, $L(y, f(\mathbf{x})) = \max(0, 1 - y\, f(x))$
  - Regularization: L2-regularization, i.e., $\Omega(\mathbf{w}) = \|\mathbf{w}\|_2^2$

All three methods construct a **linear** classifier but all three have different **objectives**.

# Outline

# Linear Discriminant Analysis (LDA)

**Linear Discriminant Analysis (LDA):**

- is often called **Fisher Discriminant Analysis** named after its inventor Ronald A. Fisher, the "father" of parametric statistics.
- projects the data $\mathbf{x} \in \mathbb{R}^d$ into a lower dimensional space via the inner product with the weight vector, i.e., $\langle \mathbf{w}, \mathbf{x} \rangle$.
  - In binary classification, such a projection of the feature space $\mathbb{R}^D$ onto the line $L = \{\alpha \mathbf{w} \,|\, \alpha \in \mathbb{R}\}$.
  - Classification of the data by thresholding, i.e., $\hat{y}(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = \text{sign}(\langle \mathbf{w}^*, \mathbf{x} \rangle + b)$.

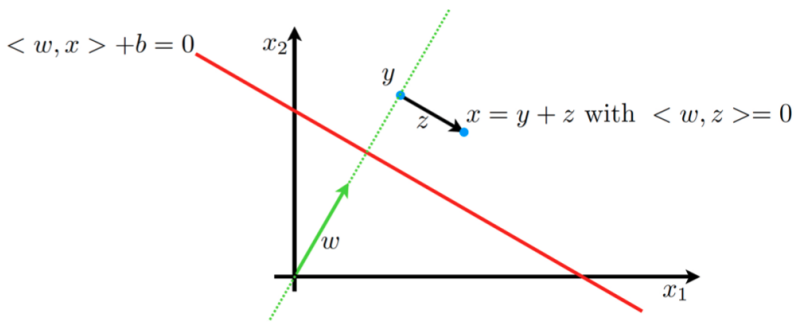



Figure: Projection onto the line $L = \{\alpha \mathbf{w} \,|\, \alpha \in \mathbb{R}\}$

- $x = y + z$ such that $y \in L = \{\alpha \mathbf{w}\}$ and $\langle \mathbf{w}, z \rangle = 0$
- For each $y \in L(\alpha \mathbf{w})$ there exists an $\alpha_0$ such that $y = \alpha_0 \frac{\mathbf{w}}{\|\mathbf{w}\|}$ and $\|y\| = \alpha_0$.

# LDA illustration

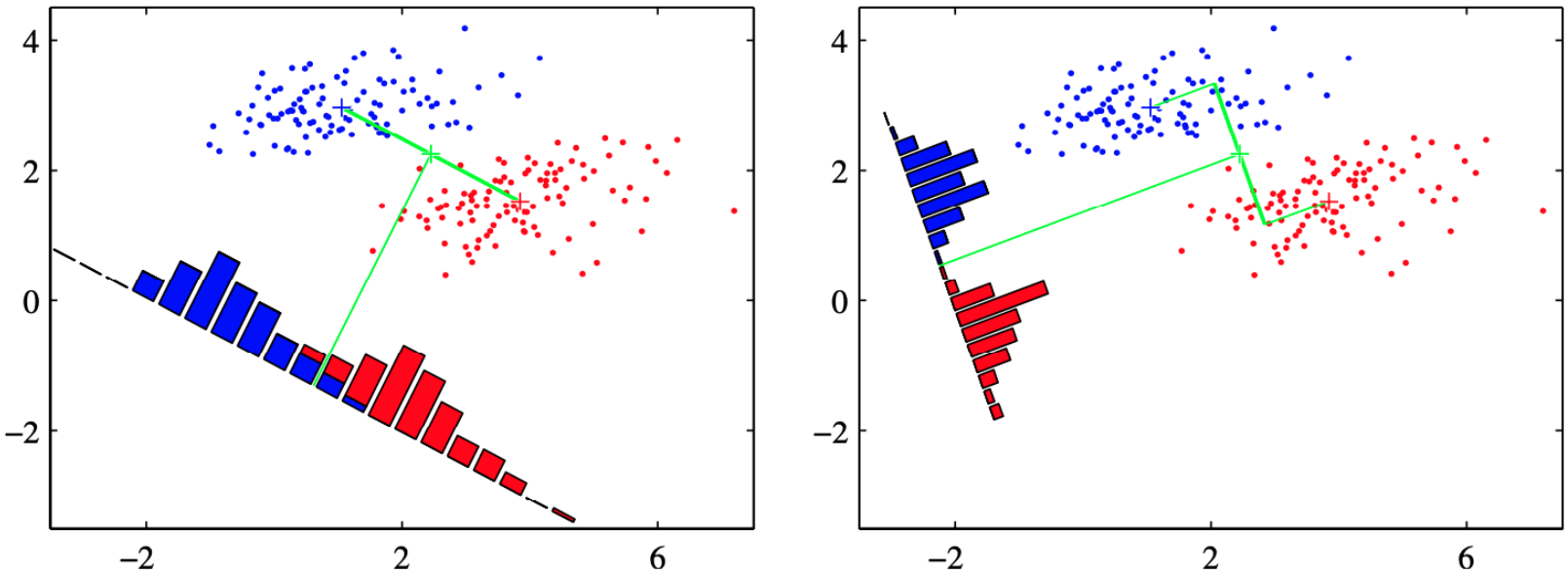



**Figure 4.6** The left plot shows samples from two classes (depicted in red and blue) along with the histograms resulting from projection onto the line joining the class means. Note that there is considerable class overlap in the projected space. The right plot shows the corresponding projection based on the Fisher linear discriminant, showing the greatly improved class separation.

Figure: Image from Bishop.

*Question:* **What is the best projection in the sense that it optimally separates the data?**

# Fisher criterion:

The **Fisher criterion** is defined as

$$J(\mathbf{w}) = \frac{\langle \mathbf{w}, \boldsymbol{\mu}_+ - \boldsymbol{\mu}_- \rangle^2}{\sigma^2_{\mathbf{w},+} + \sigma^2_{\mathbf{w},-}},$$

which aims for:

- **Large distance** of the projected class centroids $\langle \mathbf{w}, \boldsymbol{\mu}_+ \rangle$ and $\langle \mathbf{w}, \boldsymbol{\mu}_- \rangle$. The class **centroids** $\boldsymbol{\mu}_+$ and $\boldsymbol{\mu}_-$ of the positive and negative class are defined as:

$$\boldsymbol{\mu}_+ = \frac{1}{n_+} \sum_{\{i \,|\, y_i = 1\}} \mathbf{x}_i, \qquad \boldsymbol{\mu}_- = \frac{1}{n_-} \sum_{\{i \,|\, y_i = -1\}} \mathbf{x}_i,$$

- **Small variances** around the projected class centroids. The **within-class covariances** of the projections of the positive and negative class are given by:

$$\sigma^2_{\mathbf{w},+} = \sum_{\{i \,|\, y_i = 1\}} \Big( \langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}, \boldsymbol{\mu}_+ \rangle \Big)^2, \; \sigma^2_{\mathbf{w},-} = \sum_{\{i \,|\, y_i = -1\}} \Big( \langle \mathbf{w}, \mathbf{x}_i \rangle - \langle \mathbf{w}, \boldsymbol{\mu}_- \rangle \Big)^2.$$

# Fisher criterion in matrix formulation

The **between-class covariance** matrix $\boldsymbol{\Sigma}_B$ is defined as

$$\boldsymbol{\Sigma}_B = (\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-)^T,$$

and the total **within-class covariance** matrix $\boldsymbol{\Sigma}_W$ as

$$\boldsymbol{\Sigma}_W = \sum_{\{i \mid y_i = 1\}} (\mathbf{x}_i - \boldsymbol{\mu}_+)(\mathbf{x}_i - \boldsymbol{\mu}_+)^T + \sum_{\{i \mid y_i = -1\}} (\mathbf{x}_i - \boldsymbol{\mu}_-)(\mathbf{x}_i - \boldsymbol{\mu}_-)^T.$$

Then the **Fisher criterion** $J(\mathbf{w})$ can be written as

$$J(\mathbf{w}) = \frac{\langle \mathbf{w}, \boldsymbol{\Sigma}_B \mathbf{w} \rangle}{\langle \mathbf{w}, \boldsymbol{\Sigma}_W \mathbf{w} \rangle}.$$

# LDA solution

**Lemma**

*The* **optimal projection** $\mathbf{w}^* = \arg\max_{\mathbf{w}\in\mathbb{R}^d} J(\mathbf{w})$ *is given by*

$$\mathbf{w}^* = \boldsymbol{\Sigma}_W^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-).$$

**Proof:** We have

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2\frac{1}{\langle \mathbf{w}, \boldsymbol{\Sigma}_W \mathbf{w}\rangle}\boldsymbol{\Sigma}_B\mathbf{w} - 2\frac{\langle \mathbf{w}, \boldsymbol{\Sigma}_B \mathbf{w}\rangle}{\langle \mathbf{w}, \boldsymbol{\Sigma}_W \mathbf{w}\rangle^2}\boldsymbol{\Sigma}_W\mathbf{w}.$$

We solve $J(\mathbf{w}) = 0$ and get

$$\frac{\langle \mathbf{w}, \boldsymbol{\Sigma}_W \mathbf{w}\rangle}{\langle \mathbf{w}, \boldsymbol{\Sigma}_B \mathbf{w}\rangle}\boldsymbol{\Sigma}_B\mathbf{w} = \boldsymbol{\Sigma}_W\mathbf{w}.$$

Now, $\boldsymbol{\Sigma}_B\mathbf{w}$ is always proportional to $\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-$ and $\frac{\langle \mathbf{w}, \boldsymbol{\Sigma}_W \mathbf{w}\rangle}{\langle \mathbf{w}, \boldsymbol{\Sigma}_B \mathbf{w}\rangle}$ is just a scalar factor. Therefore,

$$\mathbf{w}^* \propto \boldsymbol{\Sigma}_W^{-1}(\boldsymbol{\mu}_+ - \boldsymbol{\mu}_-).$$

# LDA classification

- **Final classifier:**

$$\hat{y}(\mathbf{x}) = sign(f(\mathbf{x})) = \text{sign}(\langle \mathbf{w}^*, \mathbf{x} \rangle + b^*).$$

Determine **the threshold** $b^*$ by minimizing the training error.
**Exercise**:
Show that the optimal solution is $b^* = \frac{1}{2} \langle \mathbf{w}^*, (\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-) \rangle$

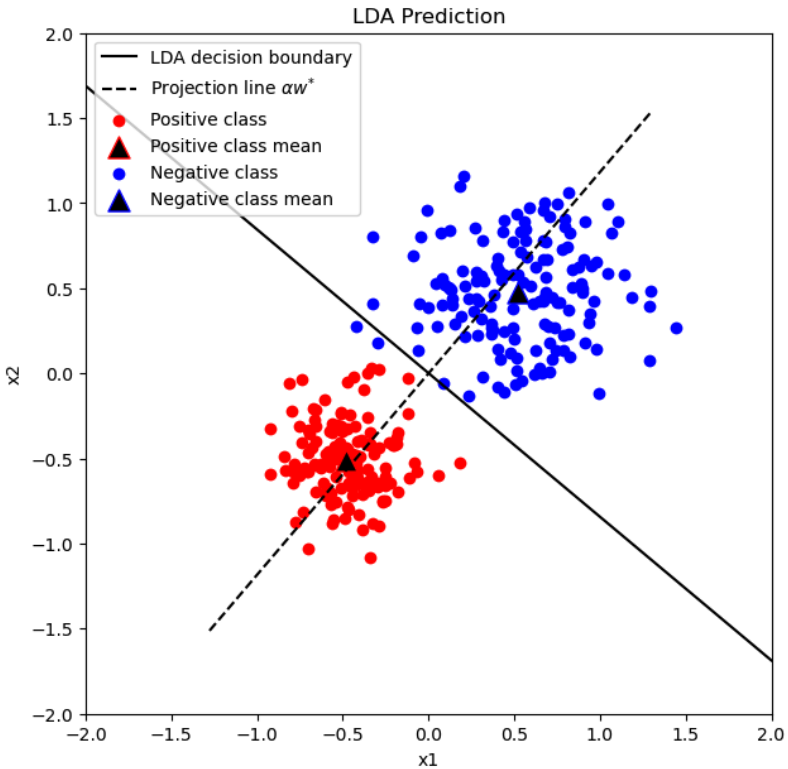
- Optimal projection can be also derived using **least squares**, as

$$(\mathbf{w}', w_0') = \underset{\mathbf{w} \in \mathbb{R}^D, w_0 \in \mathbb{R}}{\arg\min} \sum_{i=1}^{n} (y_i - \langle \mathbf{w}, \mathbf{x}_i \rangle - w_0)^2.$$

One can prove that (*exercise*!):

$$\mathbf{w}^* = \beta \mathbf{w}' \text{ (up to scaling the magnitude).}$$

# Illustration



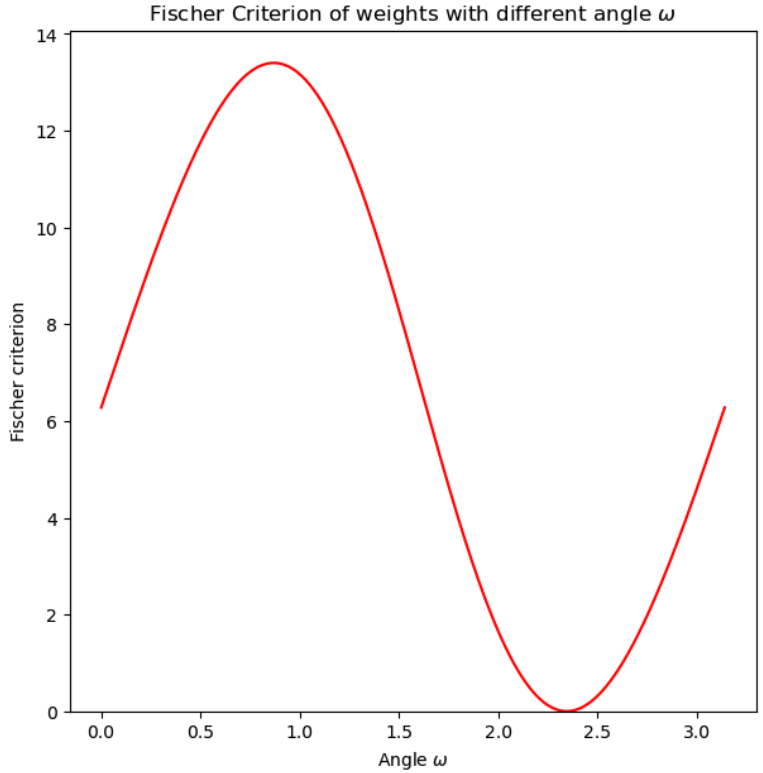




Figure: **Left:** Optimal projection line $\{\alpha \mathbf{w}^* + \frac{1}{2}(\boldsymbol{\mu}_+ + \boldsymbol{\mu}_-) \,|\, \alpha \in \mathbb{R}\}$. **Right:** The Fisher criterion as a function of the angle $\omega$ of all weighting vectors $\mathbf{w}$ in $\mathbb{R}^2$.

# Generalization to the multi-class case I

The mean of all feature vectors is denoted by $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$, and as before the centroid for each class is denoted as $\boldsymbol{\mu}_k$.
The **between-class covariance** matrix $C_B$ is defined as

$$\boldsymbol{\Sigma}_B = \sum_{k=1}^{K} n_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T.$$

and the total **within-class covariance** matrix $\boldsymbol{\Sigma}_W$ as

$$\boldsymbol{\Sigma}_W = \sum_{k=1}^{K} \sum_{\{i \mid y_i = k\}} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T,$$

The **Fisher criterion** $J(\mathbf{w})$ stays the same, i.e.,

$$J(\mathbf{w}) = \frac{\langle \mathbf{w}, \boldsymbol{\Sigma}_B \mathbf{w} \rangle}{\langle \mathbf{w}, \boldsymbol{\Sigma}_W \mathbf{w} \rangle}.$$

# LDA as Dimensionality Reduction

In general, we project the feature vectors $\mathbf{x} \in \mathbb{R}^d$ into a new space $\mathbb{R}^{d'}$, where we often assume $d' = K - 1$.
Thus, multi-class LDA can be seen as a 'supervised' approach for **dimensionality reduction**, where we seek for:

- a lower dimensional $d' \ll d$ representation of the data, which preserves the "interesting" properties of the data
- a lower dimensional representation of the data in which the classifier performs as well as on the original $d$-dimensional space.

# LDA as Dimensionality Reduction

**How can we get $d' > 1$ projections from the Fisher criterion?**

$$J(\mathbf{w}) = \frac{\langle \mathbf{w}, \mathbf{\Sigma}_B \mathbf{w} \rangle}{\langle \mathbf{w}, \mathbf{\Sigma}_W \mathbf{w} \rangle}.$$

The solution is given by the following **generalized eigenvalue problem:**

$$\mathbf{\Sigma}_B \mathbf{w} = \lambda \mathbf{\Sigma}_W \mathbf{w} \quad (\text{If } \mathbf{\Sigma}_W \text{ is invertible, then } \mathbf{\Sigma}_W^{-1} \mathbf{\Sigma}_B \mathbf{w} = \lambda \mathbf{w}).$$

$m$-dimensional projection is determined by the $m$ eigenvectors corresponding to the $m$ largest eigenvalues.

*Note:* $\mathbf{\Sigma}_B$ has rank $K - 1$ if class centroids are linearly independent. Thus, the (sorted) eigenvalues for $m > K - 1$ will be zero.
*Observation:* The above result can be explained by the generalized Rayleigh-Ritz principle (see the Appendix).
*Observation:* Further details in Chapter 4.3 of ESL book.

# Observations

- In general, one needs generally a $K-1$-dimensional subspace in order to separate $K$ classes!
- A linear projection to 1D of the data will in generally lead to a worse Bayes risk (in particular if the data is not linearly separable).
- However, in high dimension the problem might be very difficult to solve (curse of dimensionality) and the hope is that by doing dimensionality reduction we can at least find a relatively good solution in this low-dimensional subspace.
- LDA suffers from overfitting when the number of training observations is on the same order as the number of features.

# Outline

# Logistic Regression

**Idea**: (Linear) Logistic regression models the conditional likelihood using a sigmoid function, i.e.,

$$p = \mathrm{P}(Y = 1 | X = \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\langle \mathbf{w}, \phi(\mathbf{x}) \rangle}},$$

and finds the model parameters $\mathbf{w}$ using MLE.

**Question**: Why the sigmoid function? Because we can model the so-called **logistic function** (a.k.a. logit function) $\log(\frac{p}{1-p})$ using a linear function, i.e.,

$$\log\left(\frac{p}{1-p}\right) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle ,$$

*Note:* The ratio $(\frac{p}{1-p})$ is referred to as **odds**.
*Note:* We have included the bias term indirectly in the weight vector.

# Proof.

$$\log\left(\frac{p}{1-p}\right) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle \tag{1}$$

$$\left(\frac{p}{1-p}\right) = \exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle) \tag{2}$$

$$p = (1-p)\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle) \tag{3}$$

$$p = \frac{1}{(1+\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)}\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle) \tag{4}$$

$$p = \frac{\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)}{1+\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)} = \frac{1}{1+\exp(-\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)}, \tag{5}$$

where we used that

$$1 - p = 1 - \frac{\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)}{1+\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)} = \frac{1}{1+\exp(\langle \mathbf{w}, \phi(\mathbf{x}) \rangle)}.$$

# Illustration of sigmoid function

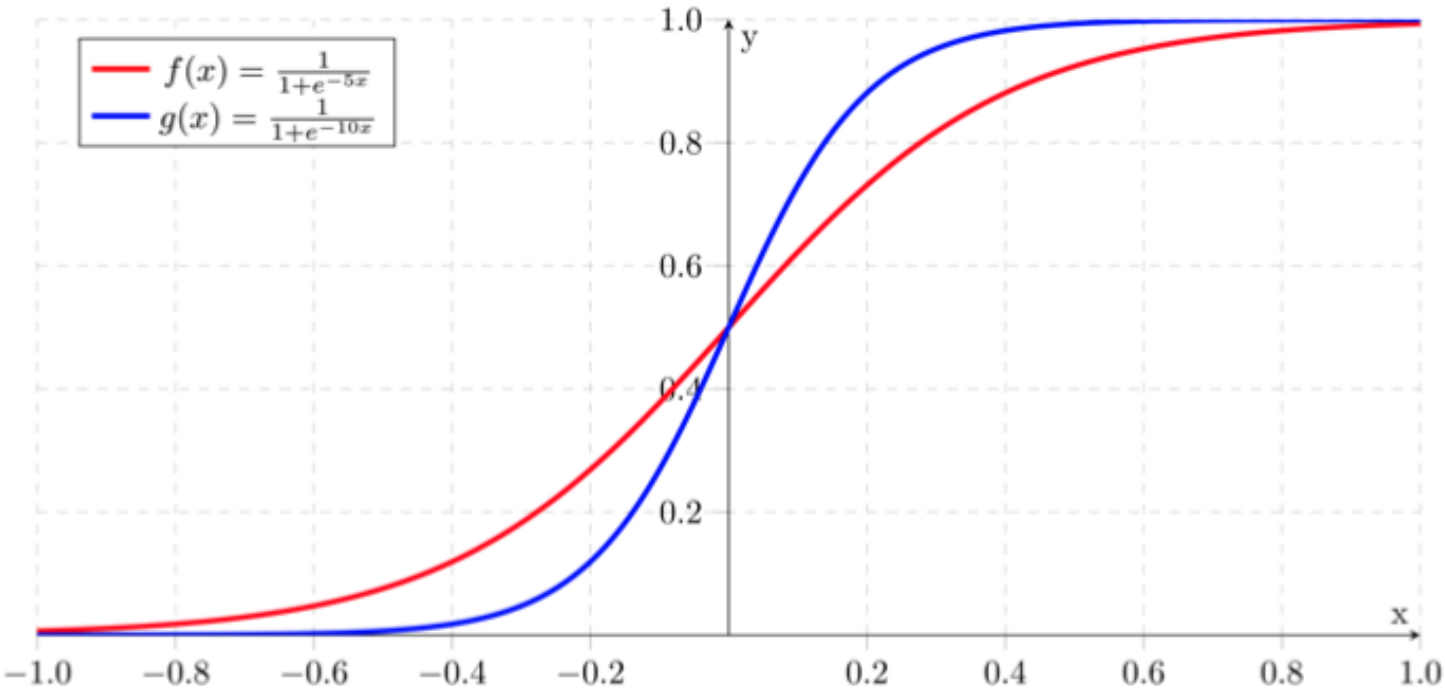



Figure: Illustration of sigmoid function (which is a special case of the logistic function).

# Logistic Regression

For a given dataset $D = (\mathbf{x}_i, y_i)_{i=1}^n$, we can obtain the optimal logistic regression parameters $\mathbf{w}$ by maximizing the likelihood:

$$\prod_{i=1}^{n} \mathrm{P}(Y = y_i | X = \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \frac{1}{1 + e^{-y_i \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle}}.$$

**Definition (Logistic regression)**

Given a training sample $D = (\mathbf{x}_i, y_i)_{i=1}^n$ with $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{-1, 1\}$ and the function space $\mathcal{F} = \{\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle \mid \mathbf{w} \in \mathbb{R}^d\}$ we define **logistic regression** as the mapping

$$D \mapsto f_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \log\Big(1 + \exp(-y_i \, f(\mathbf{x}_i))\Big). \tag{6}$$

*Observation:* Logistic regression corresponds to **Empirical risk minimization using the logistic loss, i.e.,**
$L(Y, f(X)) = \log(1 + \exp(-Y \, f(X)))$ where $f(X) = \langle \mathbf{w}, \Phi(X) \rangle$.

# Solving logistic regression

- Logistic regression has no analytical solution, but it can be written as a convex optimization problem w.r.t. the weights $\mathbf{w}$.
- Thus, we can use Newton-type gradient descent method.
- The gradient and the Hessian of the empirical risk:

$$R_{\text{emp}}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \log\Big(1 + \exp(-y_i \ \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle)\Big),$$

are given by:

$$\nabla_w R_{\text{emp}}(\mathbf{w}) = \frac{\partial R_{\text{emp}}}{\partial w_s}(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \, \phi_s(\mathbf{x}_i) \frac{\exp(-y_i \, \langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle)}{1 + \exp(-y_i \, \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle)},$$

$$H(R_{\text{emp}}) = \frac{\partial^2 R_{\text{emp}}}{\partial w_r \partial w_s}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} \phi_s(\mathbf{x}_i)\phi_r(\mathbf{x}_i) \frac{\exp(-y_i \, \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle)}{\Big(1 + \exp(-y_i \, \langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle)\Big)^2}.$$

# Newton-Raphson algorithm I

With stepsize fixed to 1, we can update the weights at each iteration of the algorithm as:

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \Big(H(R_{\text{emp}})\Big)^{-1} \nabla_{\mathbf{w}} R_{\text{emp}}(\mathbf{w}),$$

With the diagonal matrices $\mathbf{W}$ and $\mathbf{V}$ with diagonal entries

$$W_{ii} = \frac{\exp(-y_i \langle \mathbf{w}, \phi(\mathbf{x}_i) \rangle)}{(1 + \exp(-y_i \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle)^2}, \qquad V_{ii} = \frac{\exp(-y_i \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle)}{1 + \exp(-y_i \langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle)},$$

we can rewrite the gradient and Hessian $H(R_{\text{emp}})$ of $R_{\text{emp}}$ as

$$\nabla_w R_{\text{emp}}(\mathbf{w}) = -\frac{1}{n} \mathbf{\Phi}^T \mathbf{V} \mathbf{Y}, \qquad H(R_{\text{emp}})\big|_{\mathbf{w}} = \frac{1}{n} \mathbf{\Phi}^T \mathbf{W} \mathbf{\Phi}.$$

# Newton-Raphson algorithm II

Thus we can write the **Newton-Raphson update** as

$$\mathbf{w}^{\mathrm{t}+1} = \mathbf{w}^{\mathrm{t}} + \left(\mathbf{\Phi}^T \mathbf{W} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \mathbf{V}\mathbf{Y}$$

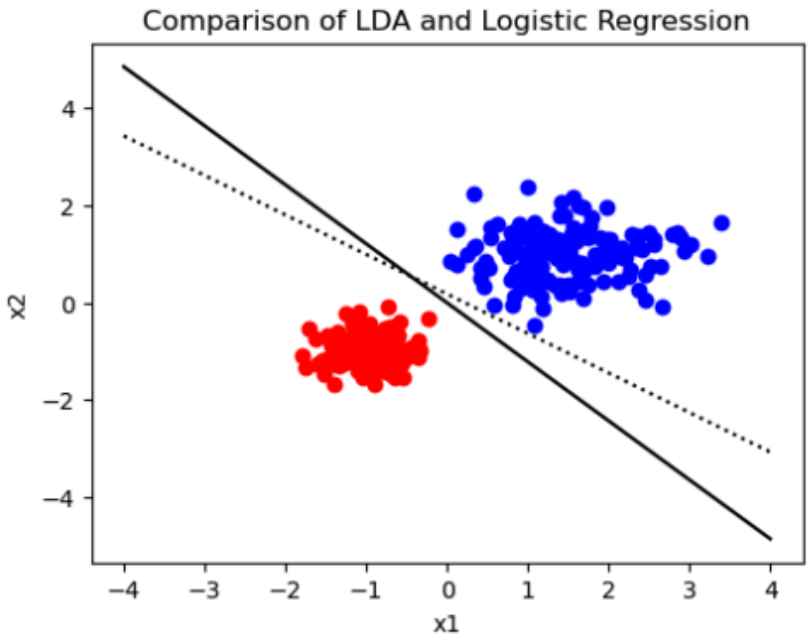
$$= \left(\mathbf{\Phi}^T \mathbf{W} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \mathbf{W}\left(\mathbf{\Phi}\mathbf{w}^{\mathrm{t}} + \mathbf{W}^{-1}\mathbf{V}\mathbf{Y}\right) = \left(\mathbf{\Phi}^T \mathbf{W} \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \mathbf{W}\mathbf{Z},$$

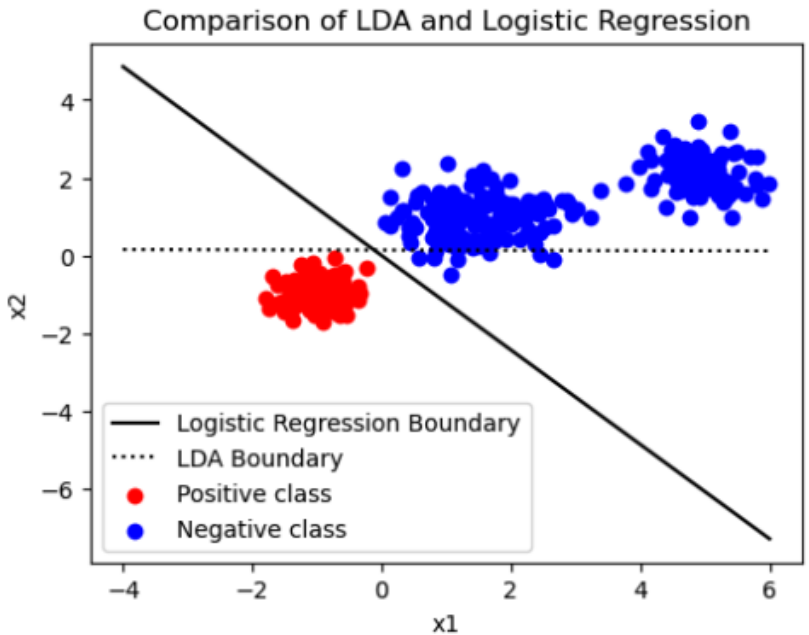
with $\mathbf{Z} = \mathbf{\Phi}\mathbf{w}^{\mathrm{t}+1} + \mathbf{W}^{-1}\mathbf{V}\mathbf{Y}$.

It can be seen as **iterative reweighted least squares problem**!

*Observation:* For further details, refer to Chapter 4.3.3 of Bishop.

# Comparison LDA vs Logistic Regression




**Left:** Original data, **Right:** Adding the second Gaussian cloud should not change the decision boundary. However, LDA changes its decision. completely.

# Regularized Logistic Regression

- As empirical risk minimization may suffer from overfitting, we can add a regularizer.
- For a linearly separable dataset the solution $\mathbf{w}^*$ is unbounded.
- Adding a regularizer on the weights, makes also the numerical solution more stable (Block III).

## Definition

Given a training sample $D = (\mathbf{x}_i, y_i)_{i=1}^n$ with $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \{-1, 1\}$ and the function space $\mathcal{F} = \{\langle \mathbf{w}, \Phi(\mathbf{x}_i) \rangle \mid \mathbf{w} \in \mathbb{R}^d\}$ we define $L_2$-**regularized logistic regression** as the mapping:

$$D \mapsto f_n = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp(-y_i\, f(\mathbf{x}_i))\right) + \lambda \, \|\mathbf{w}\|_2^2 \,,$$

where $\lambda$ is the regularization parameter.

# Multi-class Logistic Regression

We can directly use the **soft-max loss**:

$$\begin{aligned} L(y, f(\mathbf{x})) &= -\log p(Y = y | X = \mathbf{x}, \{\mathbf{w}_k\}_{k=1}^K\}) \\ &= -\log \left( \frac{\exp(\langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle)}{\sum_{k=1}^K \exp(\langle \mathbf{w}_k, \Phi(\mathbf{x}) \rangle)} \right), \end{aligned}$$

where now we have one weighting vector $\mathbf{w}_k$ for each class $k = 1, \ldots, K$.

**Classification:** Once we have trained the classifier, i.e., learned the weight vectors, we assign a new data point $\mathbf{x}$ to the class that maximizes $\langle \mathbf{w}_k, \Phi(\mathbf{x}) \rangle$, which is equivalent to maximize our estimate of the conditional probability $\log p(Y = k | X = \mathbf{x}, \{\mathbf{w}_k\}_{k=1}^K)$ w.r.t. $k$.

# Outline

# Summary

- **Linear Discriminant Analysis**:
  - Loss: Squared loss, $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
- **Logistic Regression**:
  - Loss: Logistic loss, $L(y, f(\mathbf{x})) = \log(1 + \exp(-y\, f(\mathbf{x})))$
- In general, logistic regression is much more broadly used than LDA, as it leads to more robust solutions with respect to data perturbations. Least squares loss is influenced heavily by training data which lies far away from the decision boundary, whereas the logistic loss quickly decays far away from the decision boundary and is therefore only marginally influenced by new training data far away from the decision boundary.
- In high dimensions, logistic regression often competes with more complex classification approaches (including non-linear classifiers, such as tree-based classifiers,[1] which are not covered in the course).
- **Support Vector Machines** will be introduced in Lecture 13.

[1] See Chapter 9 of ESL, if interested.

# Outline

# Rayleigh-Ritz principle

## Proposition (Rayleigh-Ritz principle)

*Let* $A \in \mathbb{R}^{d \times d}$ *be a* **symmetric matrix**, *then*

$$\lambda_{\max} = \max_{x \in \mathbb{R}^d} \frac{\langle x, Ax \rangle}{\langle x, x \rangle},$$

*is the largest eigenvalue of* $A$ *and the maximizing argument* $x_{\max}$ *is the corresponding eigenvector. Equivalently,*

$$\lambda_{\max} = \max_{x \in \mathbb{R}^d,\ \|x\|=1} \langle x, Ax \rangle .$$

*Other eigenvalues and eigenvectors can be found as follows. Denote by* $u_1, \ldots, u_r$ *the eigenvectors corresponding to the largest* $r$ *eigenvalues, then the* $r+1$ *largest eigenvalue can be found as,*

$$\lambda_{r+1} = \max_{x \in \mathbb{R}^d,\ \langle x, u_s \rangle = 0,\ s=1,\ldots,r} \frac{\langle x, Ax \rangle}{\langle x, x \rangle}.$$