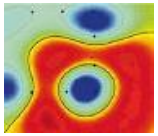
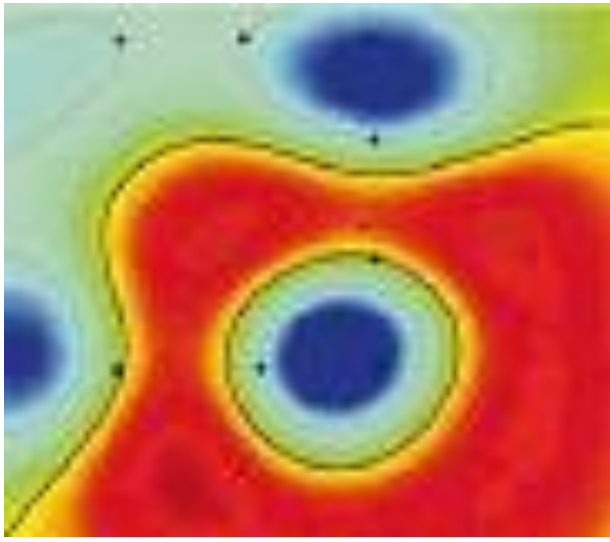


Technische Universität Berlin



Lecture 3

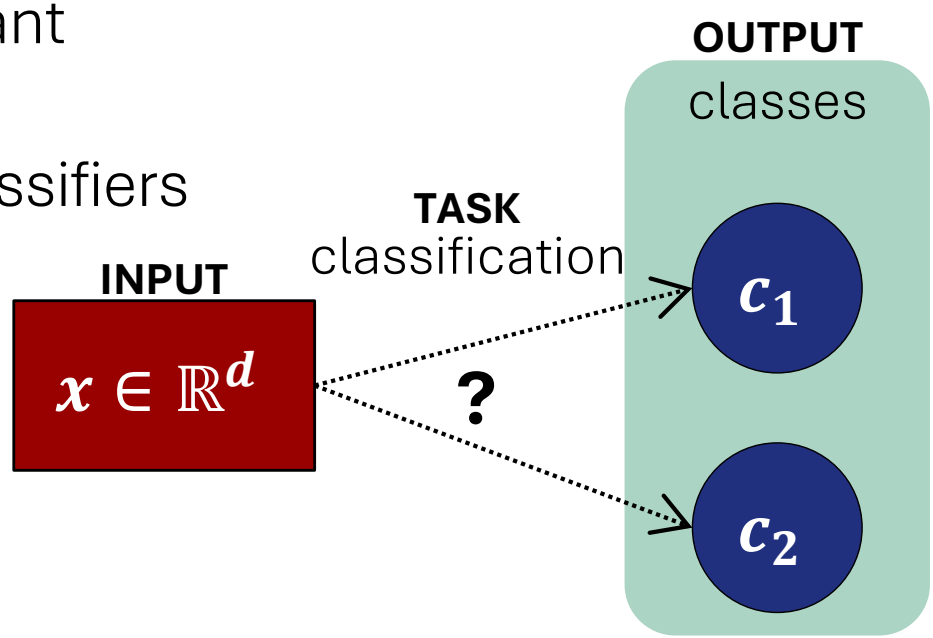
Fisher Discriminant

Hannah Marienwald

Outline

- Recap
  - Bayes Optimal Classifier
  - Parameter Estimation

- Classification without Learning Distributions
  - Mean Separation
  - Fisher Discriminant
  - Perceptron
  - Large Margin Classifiers



Recap: Bayes Optimal Classifier

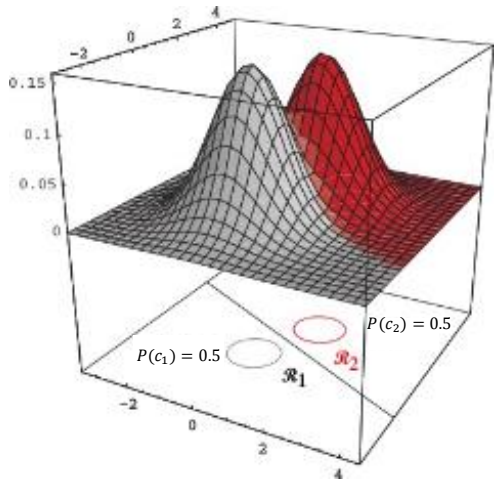
Recap:

- Assume our data is generated for each class c_k according to the multivariate Gaussian distribution $p(\boldsymbol{x}|c_k) = \mathcal{N}(\boldsymbol{\mu}_k|\Sigma)$ and with class priors $P(c_k)$. The Bayes optimal classifier is derived as

$$= \underset{k}{\operatorname{argmax}}\{P(c_k|\boldsymbol{x})\}$$

$$= \underset{k}{\operatorname{argmax}}\{\log p(\boldsymbol{x}|c_k) + \log P(c_k)\}$$

$$= \underset{k}{\operatorname{argmax}}\left\{\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{\mu}_k^T - \frac{1}{2}\boldsymbol{\mu}_k\Sigma^{-1}\boldsymbol{\mu}_k + \log P(c_k)\right\}$$



- Given our generative assumptions, there is no better classifier than the one above.
- However, in practice, we don't know these distributions and only have the data.

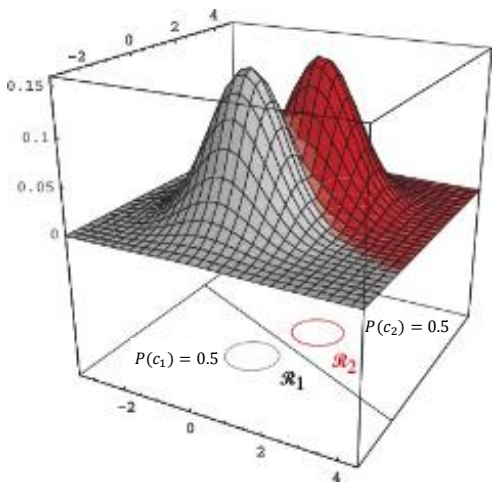
Recap: Parameter Estimation

Example of estimator:

- Maximum likelihood estimator:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}})^T$$



Problem:

- The covariance matrix (and its inverse) may be difficult to estimate.
- We make an assumption about the data (e.g. Gaussian-distributed) which may not correspond to reality.

Distribution-Free Approaches

> *When solving a problem of interest, do not solve a more general problem as an intermediate step.*
> – V. Vapnik

Interpretation in our setting:

- Don't take the intermediate step of learning distributions to build the classifier. Instead, build the classifier directly.
- That is, rather than assuming a set of distributions (e.g., Gaussian), assume a set of models (e.g., linear), and find the parameters of the model that optimize some classification objective (e.g., based on the statistics of the data in projected space).

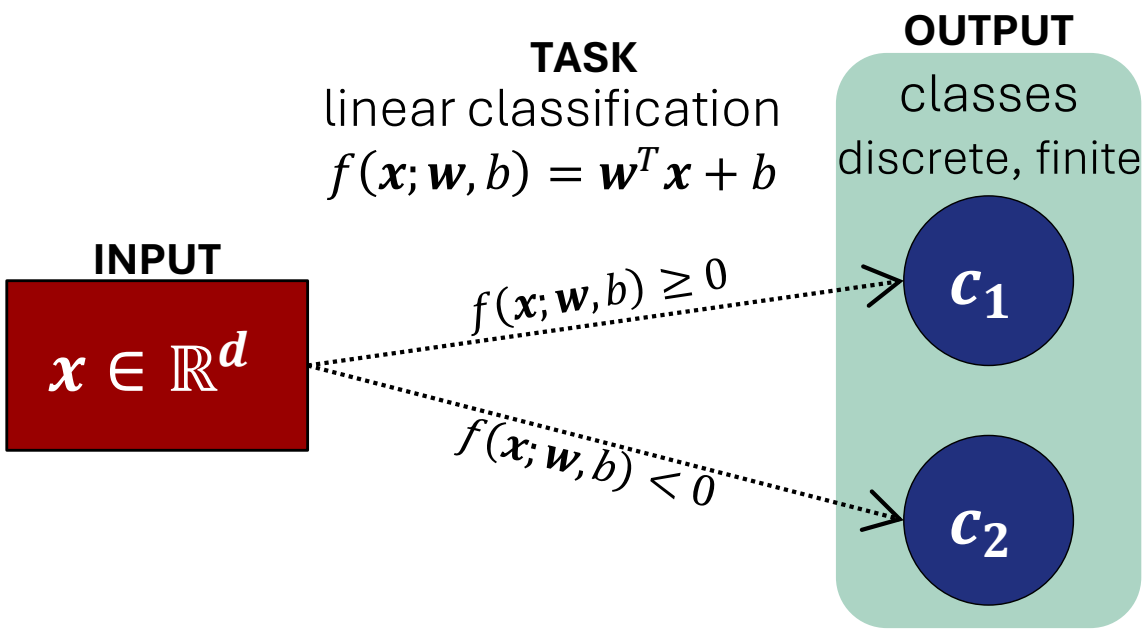
Linear Classifiers

- Functions of the type

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^T \boldsymbol{x} + b$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are parameters to learn.

We decide for class c_1 if $f(x) >= 0$ and for class c_2 if $f(x) < 0$.



Based on the training data $\mathcal{D} = \{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$
find $\boldsymbol{w}, b$ of $y_n = \text{sign}\big(f(\boldsymbol{x}_n; \boldsymbol{w}, b)\big)$
such that $y_n \approx t_n$.

Geometry of Linear Classifiers

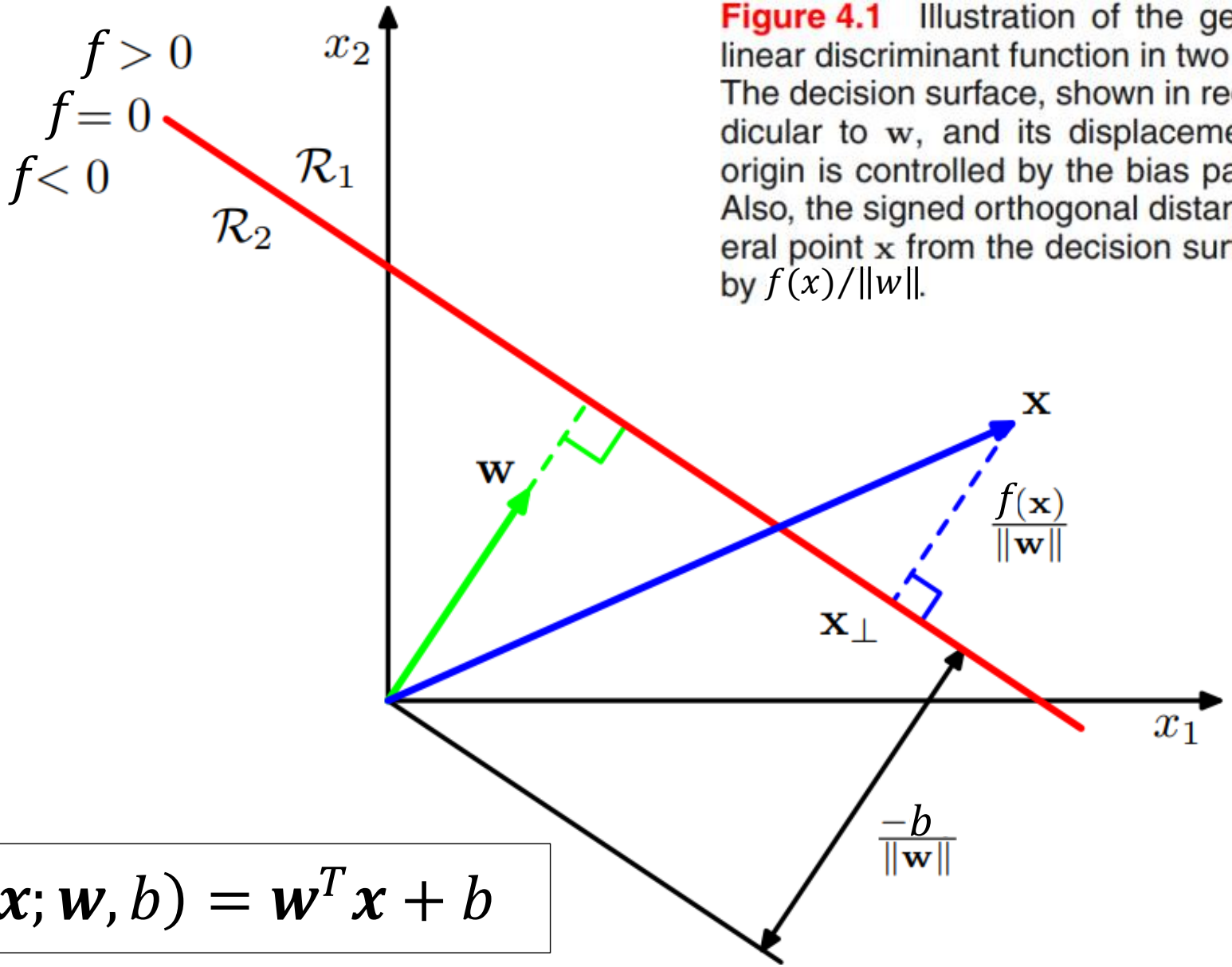


Figure 4.1 Illustration of the geometry of a linear discriminant function in two dimensions. The decision surface, shown in red, is perpendicular to $\mathbf{w}$, and its displacement from the origin is controlled by the bias parameter b. Also, the signed orthogonal distance of a general point $\mathbf{x}$ from the decision surface is given by $f(x)/\|w\|$.

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^T \boldsymbol{x} + b$$

Figure taken from C.M. Bishop (2009), "Pattern Recognition and Machine Learning", 9ed. Notation adapted.

Linear Classifiers

- Functions of the type

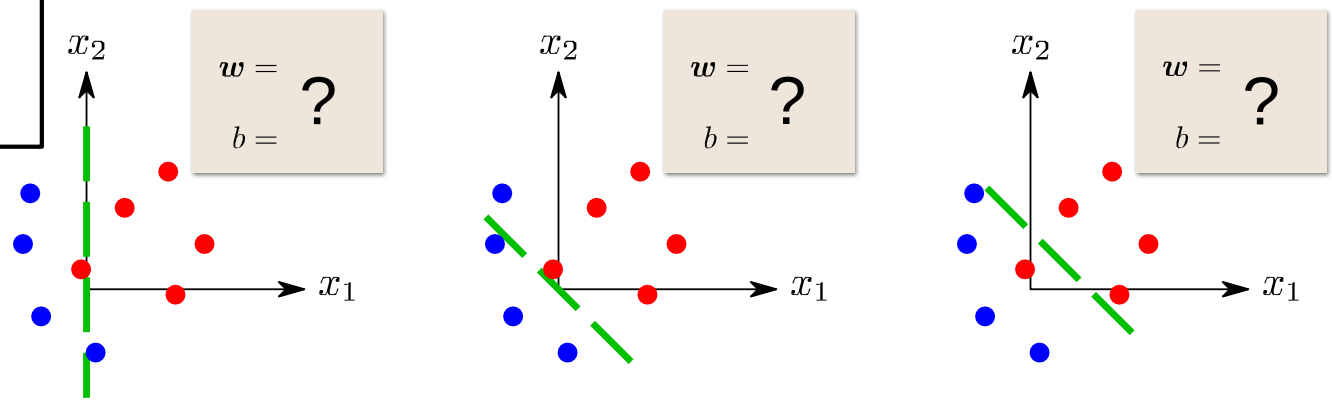
$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^T \boldsymbol{x} + b$$

where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are parameters to learn.

We decide for class c_1 if $f(\mathbf{x}) > 0$ and for class c_2 if $f(\mathbf{x}) < 0$.

- Examples of linear classifiers on a simple 2d example:

Can you reconstruct $\boldsymbol{w}, b$ based on given decision boundary?



- **Question:** Based on what criterion do we choose the parameters $\boldsymbol{w}, b$?

Linear Classifiers

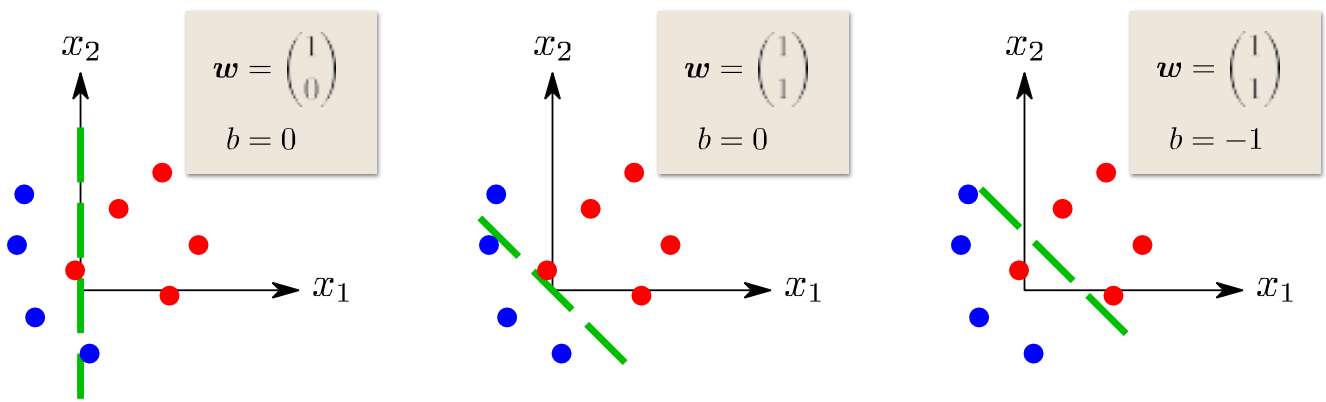
- Functions of the type

$$f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^T \boldsymbol{x} + b$$

where $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are parameters to learn.

We decide for class c_1 if $f(\boldsymbol{x}; \boldsymbol{w}, b) \geq 0$ and for class c_2 if $f(\boldsymbol{x}; \boldsymbol{w}, b) < 0$.

- Examples of linear classifiers on a simple 2d example:

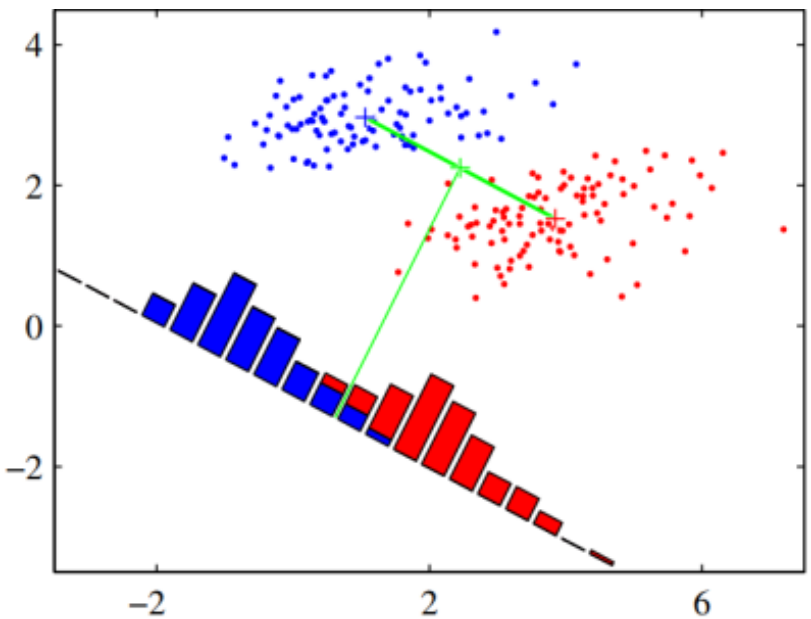


- **Question:** Based on what criterion do we choose the parameters $\boldsymbol{w}, b$?

Mean Separation Criterion

Idea:

- Build a projection the data $z = \boldsymbol{w}^T \boldsymbol{x}$ with $\|\boldsymbol{w}\| = 1$ such that the means of classes in projected space are as distant as possible.



Approach:

- First, we compute the means in projected space for the two classes

$$m_1 = \frac{1}{N_1} \sum_{i \in \mathcal{C}_1} z_i \qquad m_2 = \frac{1}{N_2} \sum_{i \in \mathcal{C}_2} z_i$$

- Then we would like to find w that maximizes the difference between the projected means, i.e., we express the means as a function of w and pose the optimization problem:

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} |m_2(\boldsymbol{w}) - m_1(\boldsymbol{w})| \;\; s.t. \;\; \|\boldsymbol{w}\| = 1$$

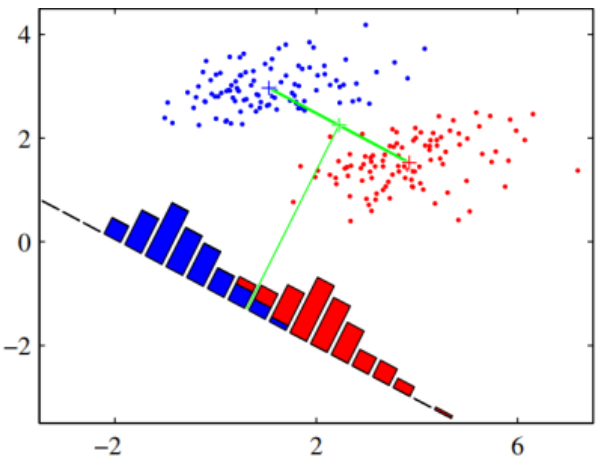
Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

Derivation of Mean Separation

- The constrained optimization problem (subject to the constraint $\|\boldsymbol{w}\| = 1$) can be developed as:

$$\begin{aligned}
&\underset{\boldsymbol{w}}{\text{argmax}} |m_2(w) - m_1(w)| \\
&= \underset{\boldsymbol{w}}{\text{argmax}} \left| \frac{1}{N_1} \sum_{i \in \mathcal{C}_2} z_i - \frac{1}{N_1} \sum_{i \in \mathcal{C}_1} z_i \right| \\
&= \underset{\boldsymbol{w}}{\text{argmax}} \left| \frac{1}{N_2} \sum_{i \in \mathcal{C}_2} \boldsymbol{w}^T \boldsymbol{x}_i - \frac{1}{N_1} \sum_{i \in \mathcal{C}_1} \boldsymbol{w}^T \boldsymbol{x}_i \right| \\
&= \underset{\boldsymbol{w}}{\text{argmax}} \left| \boldsymbol{w}^T \left(\frac{1}{N_2} \sum_{i \in \mathcal{C}_2} \boldsymbol{x}_i \right) - \boldsymbol{w}^T \left(\frac{1}{N_1} \sum_{i \in \mathcal{C}_1} \boldsymbol{x}_i \right) \right| \\
&= \underset{\boldsymbol{w}}{\text{argmax}} |\boldsymbol{w}^T \, \hat{\boldsymbol{\mu}}_2 - \boldsymbol{w}^T \, \hat{\boldsymbol{\mu}}_1|
\end{aligned}$$

where $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2$ are the emp. means in input space.



- The best vector w is the one that aligns with $(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)$, i.e., $\boldsymbol{w} = (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)/\|\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1\|$

Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

Limitations of Mean Separation

- There is a significant class overlap in projected space.
- A better classifier seems achievable if we rotate the projection a few degrees clockwise.
- Making means distant may not be sufficient to induce class separability in projected space.

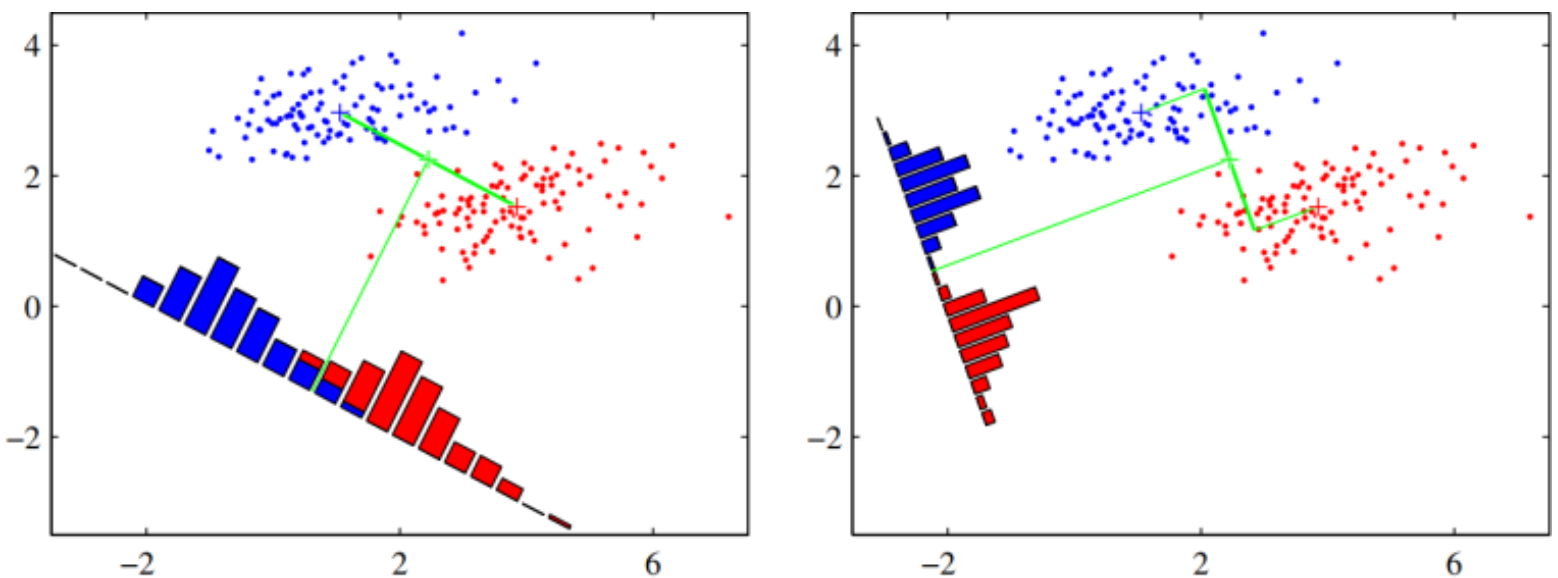


Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

Fisher Discriminant


R.A. Fisher (1890 - 1962)

Idea:

- In addition to maximizing the separation between class means in projected space, also consider to reduce the within-class variance.

$$m_1 = \frac{1}{N_1}\sum_{i \in \mathcal{C}_1} z_i \qquad m_2 = \frac{1}{N_2}\sum_{i \in \mathcal{C}_2} z_i$$

$$s_1 = \sum_{i \in \mathcal{C}_1} (z_i - m_1)^2 \qquad s_2 = \sum_{i \in \mathcal{C}_2} (z_i - m_2)^2$$

- Maximizing distance between means while minimizing within-class variance can be formulated as:

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \frac{\left(m_2(\boldsymbol{w}) - m_1(\boldsymbol{w})\right)^2}{s_2(\boldsymbol{w}) + s_1(\boldsymbol{w})}$$

Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

Deriving the Fisher Discriminant (1)

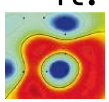
The within-class variance can be expanded as:

$$
\begin{aligned}
s_k(\boldsymbol{w}) &= \sum_{i \in \mathcal{C}_k} (z_i - m_k)^2 \\
&= \sum_{i \in \mathcal{C}_k} (\boldsymbol{w}^T \boldsymbol{x}_i - \boldsymbol{w}^T \hat{\boldsymbol{\mu}}_k)^2 \\
&= \boldsymbol{w}^T \underbrace{\left(\sum_{i \in \mathcal{C}_k} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_k)^T \right)}_{= S_k} \boldsymbol{w}
\end{aligned}
$$

where S_k is a scatter matrix (unnormalized emp. covariance matrix) for the data of class k.

Observations:

- Similar structure as the PCA objective (but for each class separately).
- Unlike PCA, we want to minimize the variance rather than maximizing it.



Deriving the Fisher Discriminant (2)

Making use of the results of Slide 10 and 13, we can rewrite the Fisher objective

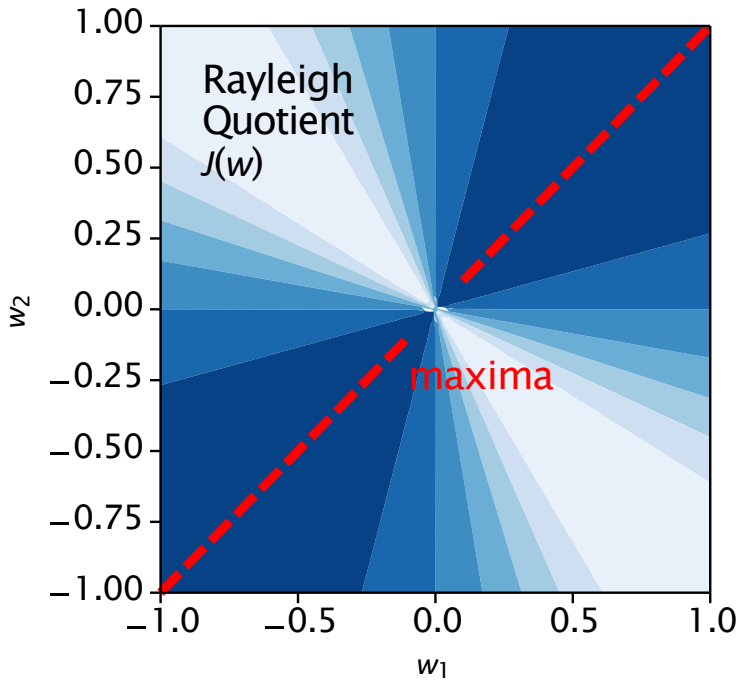
$$J(\boldsymbol{w}) = \frac{\left(m_2(\boldsymbol{w}) - m_1(\boldsymbol{w})\right)^2}{s_2(\boldsymbol{w}) + s_1(\boldsymbol{w})}$$

as

$$\boxed{J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T S_B \boldsymbol{w}}{\boldsymbol{w}^T S_W \boldsymbol{w}}}$$

where

$$S_B = (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)^T$$
$$S_W = S_1 + S_2$$

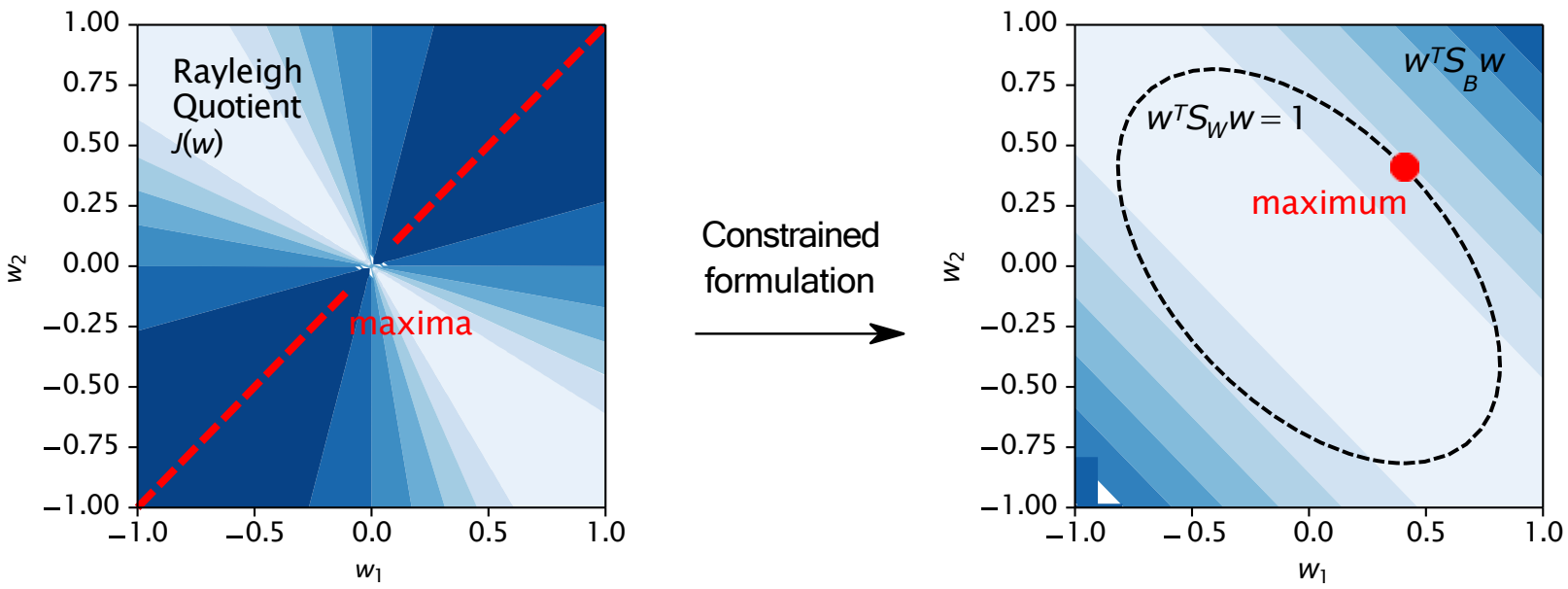


are the '*B*etween-class' and '*W*ithin-class' scatter matrices resp.
$J(w)$ is known as Rayleigh Quotient.

Deriving the Fisher Discriminant (3)

- A solution that maximizes the Rayleigh quotient $J(\boldsymbol{w})$ can be obtained by first observing that $\forall \alpha \neq 0: J(\alpha \boldsymbol{w}) = J(\boldsymbol{w})$ and searching for the particular solution for which the denominator is exactly one. This can be stated as the constrained optimization problem

$$\underset{\boldsymbol{w}}{\operatorname{argmax}} \ \boldsymbol{w}^T S_B \boldsymbol{w} \quad s.t. \quad \boldsymbol{w}^T S_W \boldsymbol{w} = 1$$



Deriving the Fisher Discriminant (4)

We start with the constrained optimization problem:

$$\underset{\boldsymbol{w}}{\text{argmax}} \quad \boldsymbol{w}^T S_B \boldsymbol{w} \quad s.t. \quad \boldsymbol{w}^T S_W \boldsymbol{w} = 1$$

Method of Lagrange Multipliers

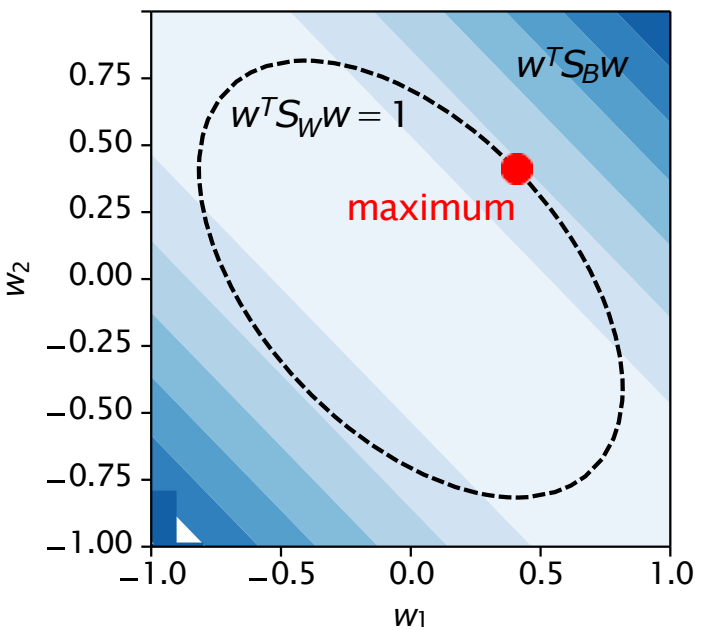
Step 1: We build the Lagrangian

$$\mathcal{L}(\boldsymbol{w}, \lambda) = \boldsymbol{w}^T S_B \boldsymbol{w} + \lambda(1 - \boldsymbol{w}^T S_W \boldsymbol{w})$$

Step 2: We look for potential solutions by posing $\nabla\mathcal{L}(\boldsymbol{w}, \lambda) = 0$, which leads to the equation:

$$\boxed{S_B \boldsymbol{w} = \lambda S_W \boldsymbol{w}}$$

This is a generalized eigenvalue problem.



Deriving the Fisher Discriminant (5)

Further steps:

- If S_W is invertible, it can be restated as the standard eigenvalue problem

$$S_W^{-1} S_B \boldsymbol{w} = \lambda \boldsymbol{w}$$

- Expanding the term S_B, we get:

$$S_W^{-1} (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1) \underbrace{(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)^T \boldsymbol{w}}_{\text{scalar}} = \lambda \boldsymbol{w}$$

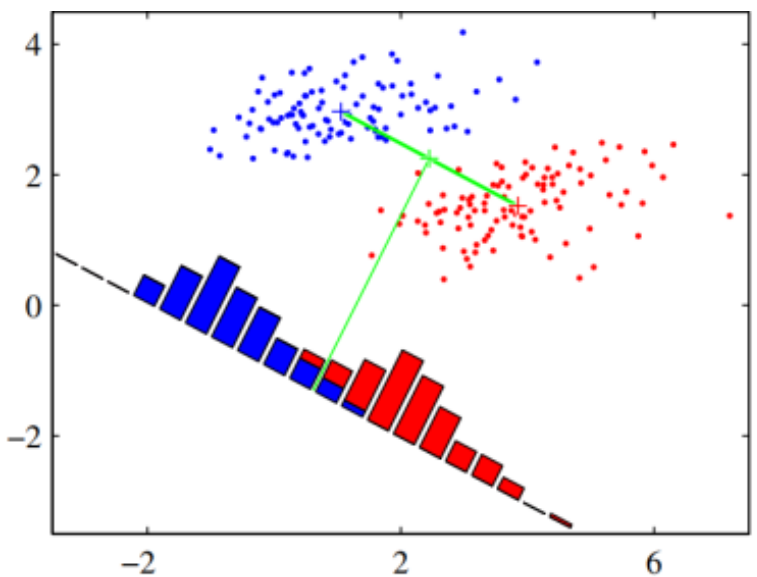
- Therefore, one possible solution for $\boldsymbol{w}$ is given by

$$S_W^{-1} (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1) = \boldsymbol{w}$$

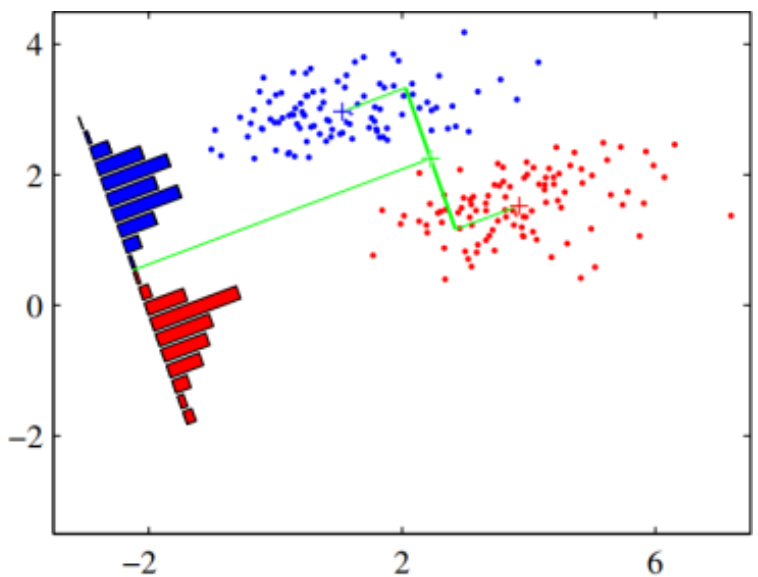
(Reminder, in our original Rayleigh quotient formulation, $J(\alpha \boldsymbol{w}) = J(\boldsymbol{w})$, i.e., the optimum is defined up to a scaling factor.)

Mean Separation vs. Fisher Discriminant

Maximum Mean Separation



Fisher Discriminant



- Fisher Discriminant leads (in general) to better class separability, and therefore, better classification accuracy.
- Fisher Discriminant requires inversion of a covariance matrix (only tractable for low-dimensional data).

Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

Decision Theory vs. Fisher Discriminant

Bayesian decision theory:

Discriminant has the form

$$\boldsymbol{w} = \Sigma^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$$

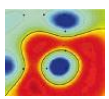
- under the assumption that the data-generating distributions are Gaussian with means $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2$ and the same covariance matrix Σ.
- Offset b is also given by the analysis (cf. slide 2)

Fisher discriminant:

Discriminant has the form

$$\boldsymbol{w} = \mathrm{S}_\mathrm{W}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$$

- No generative assumption. Classifier only derived from a criterion on mean and dispersion of the data in projected space.
- Offset b is not provided by the analysis but it can be fitted in a second step.



Fisher Discriminant – Strengths/Limitations

Strengths:

- Accurate when the means/covariances describe the data well, and close to optimal when the data is Gaussian of fixed covariance.
- The Fisher discriminant is given in closed form and is fast to compute.

Limitations:

- Although applicable to non-Gaussian distributions, the resulting decision boundary might become strongly suboptimal.
- In particular, like principal component analysis, Fisher Discriminant is not robust to outliers.

Idea:

- To overcome these limitations, we will discuss other learning algorithms that more specifically focus on modeling the decision boundary between the two classes.

The Perceptron

F. Rosenblatt (1928–1971)

- Proposed by F. Rosenblatt in 1958.
- Classifier that prefectly separates training data (if the data is linearly separable).
- Trained using an simple and cheap iterative procedure.
- The perceptron gave rise to artificial neural networks.

Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

The Perceptron Algorithm

- Consider the linear model

$$z_i = \boldsymbol{w}^T \boldsymbol{x}_i + b \qquad y_i = sign(z_i) = \begin{cases} -1, & z_i < 0 \\ +1, & z_i \geq 0 \end{cases}$$

and let t_i be $+1$ or -1 when the true class of $\boldsymbol{x}_i$ is c_1 or c_2 respectively.

Algorithm

1. Repeat until all samples are correctly classified:
2. Iterate over all data samples $i = 1, \dots, N$ (multiple times)
3. If observation $\boldsymbol{x}_i$ is correctly classified ($y_i = t_i$), continue.
4. If observation $\boldsymbol{x}_i$ is wrongly classified ($y_i \neq t_i$), do:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot \boldsymbol{x_i} t_i$$
$$b \leftarrow b + \eta \cdot t_i$$

where η is the learning rate.

The Perceptron Algorithm

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot \boldsymbol{x_i} t_i$$

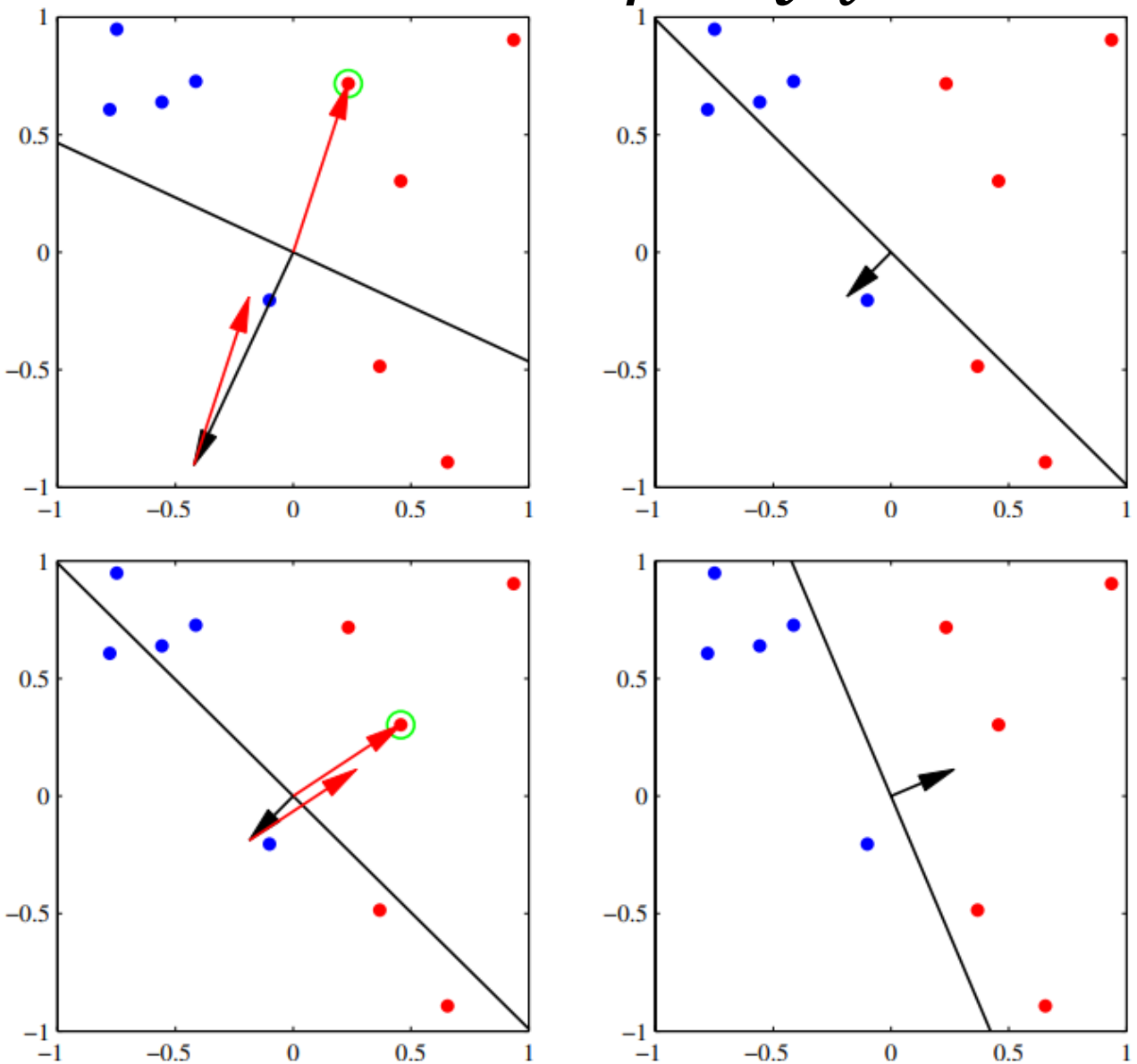


Figure taken from C.M. Bishop (2009), “Pattern Recognition and Machine Learning”, 9ed.

The Perceptron: Optimization View

- The Perceptron can be seen as the minimization of the error function

$$E(\boldsymbol{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \underbrace{\max\{0, -z_i\, t_i\}}_{E_i(\boldsymbol{w}, b)}$$

recall $z_i = \boldsymbol{w}^T \boldsymbol{x}_i + b$

- **Proof**: Computing the gradient of E_i gives

$$\begin{aligned} \nabla_{\boldsymbol{w}} E_i(\boldsymbol{w}, b) &= \mathbf{1}\{-z_i t_i > 0\} \cdot (-\boldsymbol{x}_i t_i) \\ &= \mathbf{1}\{y_i \neq t_i\} \cdot (-\boldsymbol{x}_i t_i) \\ &= \begin{cases} 0 & y_i = t_i \\ -\boldsymbol{x}_i t_i & y_i \neq t_i \end{cases} \end{aligned}$$

which results in the update rule $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta \cdot \boldsymbol{x}_i t_i$.

- A similar result can be obtained for the offset b.

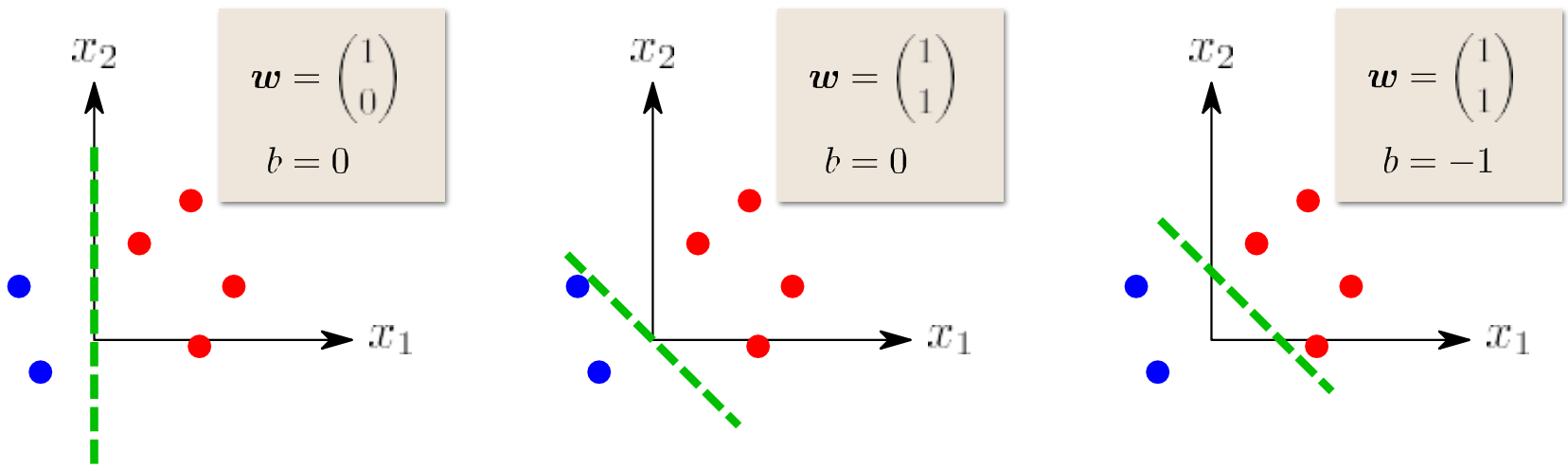
The Perceptron: Optimization View

- Recall: The objective function corresponding to the perceptron algorithm is given by:

$$E(\boldsymbol{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \max\{0, -z_i \, t_i\}$$

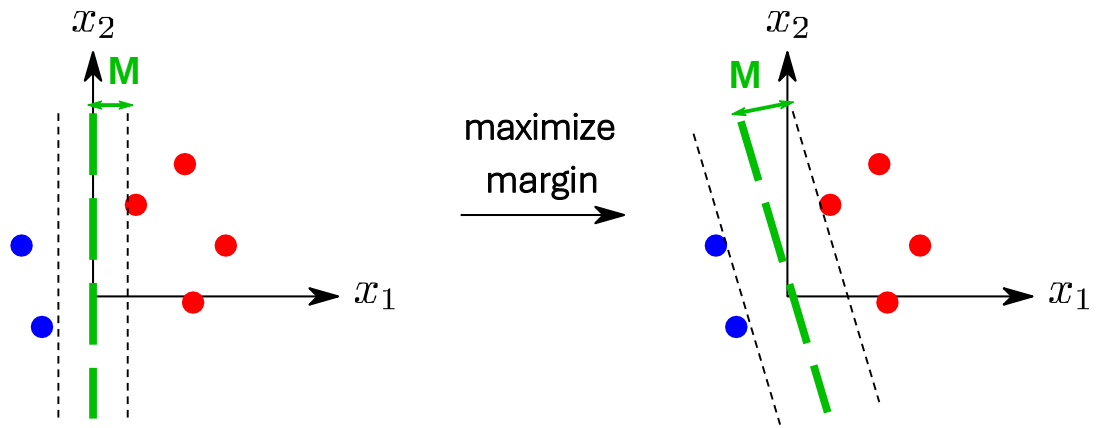
- This objective can be interpreted as measuring for each wrongly classified data point how far the data point is from the decision boundary and penalizing accordingly.
- In practice various strategies can be implemented to optimize this objective (e.g. gradient descent on $E(\boldsymbol{w}, b)$ directly, or adding momentum to the gradient descent).
- Optimization of the objective also works for data that is not linearly separable.

A Problem of the Perceptron



- All these solutions have an error $E(\boldsymbol{w}, b) = 0$.
- Some solutions are obviously better than others, e.g., those with a decision boundary separates the classes with a large margin.

Large Margin Classifiers

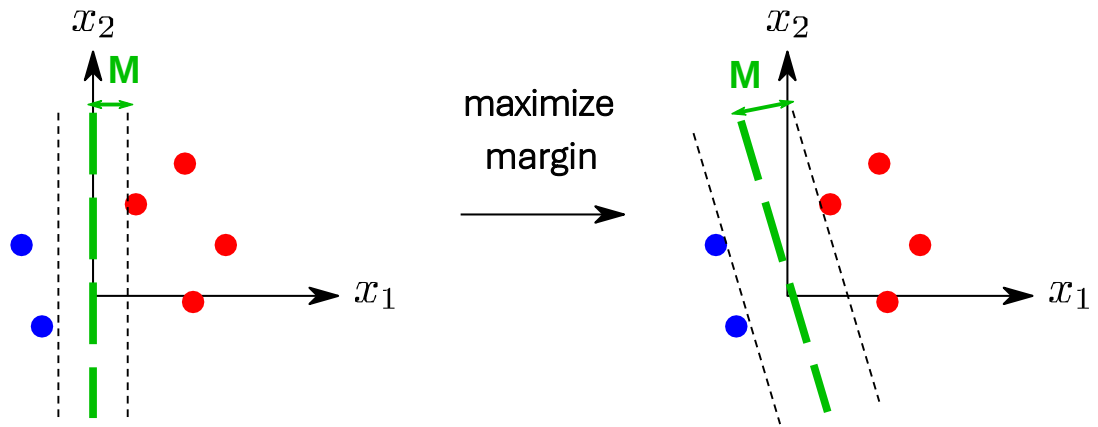


Idea: Induce large margin M by redefining the optimization problem as:

$$\underbrace{\underset{\boldsymbol{w},b,M}{\operatorname{argmin}}}_{(1)}\underbrace{\left(\frac{1}{M}}_{(2)}+\frac{1}{N}\sum_{i=1}^{N}\underbrace{\max\{0, M-(\boldsymbol{w}^T\boldsymbol{x}_i+b)t_i\}}_{(3)}\right) \quad \text{s.t.} \quad \underbrace{\|w\|=1}_{(4)}$$

- (1) actively optimize the margin
- (2) apply a penalty if the margin is too small
- (3) apply a penalty if some points violate the margin and
- (4) constrain the projection so that M is interpretable as a margin.

Large Margin Classifiers



- An equivalent (and more standard) formulation of the optimization problem is given by:

$$\underset{\boldsymbol{w},b,}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \max\{0, 1 - (\boldsymbol{w}^T \boldsymbol{x}_i + b) t_i\} + \|w\|^2 \right)$$

  (proof on next slide).

- Large Margin Classifiers are typically solved using (stochastic) gradient descent or quadratic programming.
- More will be said about these model during the lecture on support vector machines.

Equivalence of Problem Formulations

$$\underset{\boldsymbol{w},b,M}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \max\{0, M - (\boldsymbol{w}^T \boldsymbol{x}_i + b) t_i\} + \frac{1}{M} \right) \quad \text{s.t.} \quad \|w\| = 1$$

The same objective can be rewritten as:

$$\underset{\boldsymbol{w},b,M}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \max\{0, M - (\boldsymbol{w}^T \boldsymbol{x}_i + b) t_i\} + \frac{\|\boldsymbol{w}\|^2}{M} \right) \quad \text{s.t.} \quad \|w\| = 1$$

Divide by M which does not affect the solution of the optimization problem

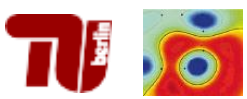
$$\underset{\boldsymbol{w},b,M}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \max\{0,1 - ((\boldsymbol{w}/M)^T \boldsymbol{x}_i + b/M) t_i\} + \frac{\|\boldsymbol{w}\|^2}{M^2} \right) \quad \text{s.t.} \quad \|w\|/M = 1/M$$

Redefine $\boldsymbol{w} \to \boldsymbol{w}/M$ and $b \to b/M$

$$\underset{\boldsymbol{w},b,M}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \max\{0,1 - (\boldsymbol{w}^T \boldsymbol{x}_i + b) t_i\} + \|\boldsymbol{w}\|^2 \right) \quad \text{s.t.} \quad \|w\| = 1/M$$

Because optimization w.r.t. M is now trivial, we can further simplify to

$$\underset{\boldsymbol{w},b,}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} \max\{0,1 - (\boldsymbol{w}^T \boldsymbol{x}_i + b) t_i\} + \|\boldsymbol{w}\|^2 \right)$$



Advanced Classification Topics

- **Nonlinear Classification**, e.g., kernel SVM, artificial neural networks, decision trees, random forests, boosting
  → covered in ML1
- Incorporating **prior knowledge** such as invariances into a classifier, e.g., convolutional neural network
  → covered in ML2
- Classification in **high dimensions**: Unintuitively, methods that promote within-class variance instead of reducing it tend to perform better in this setting.

Summary

- In practice, it is preferable to **train a classifier directly** rather than learning the class distributions in the first place.
- The **maximum mean separation** and **Fisher discriminant** are two such instances, where one only needs to estimate the mean and the covariance of the data, without having to estimate full distributions.
- The **perceptron** (and its **large-margin** extension) more specifically focus on the decision boundary, which typically leads to higher classification accuracy on general tasks.