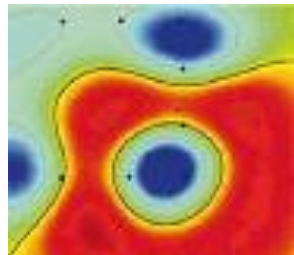




## Lecture 3

# Fisher Discriminant

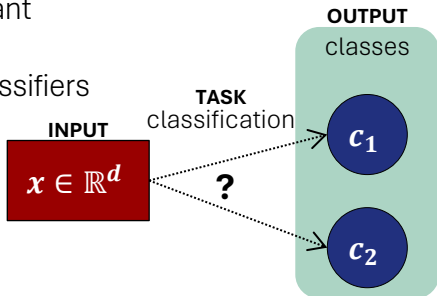
Hannah Marienwald



# Outline

---

- ▶ Recap
  - ▶ Bayes Optimal Classifier
  - ▶ Parameter Estimation
- ▶ Classification without Learning Distributions
  - ▶ Mean Separation
  - ▶ Fisher Discriminant
  - ▶ Perceptron
  - ▶ Large Margin Classifiers



# Recap: Bayes Optimal Classifier

## Recap:

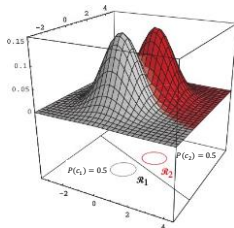
- Assume our data is generated for each class  $c_k$  according to the multivariate Gaussian distribution  $p(\mathbf{x}|c_k) = \mathcal{N}(\boldsymbol{\mu}_k|\Sigma)$  and with class priors  $P(c_k)$ . The Bayes optimal classifier is derived as

$$= \operatorname{argmax}_k \{P(c_k|\mathbf{x})\}$$

$$= \operatorname{argmax}_k \{\log p(\mathbf{x}|c_k) + \log P(c_k)\}$$

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

- 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.



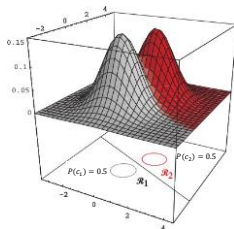
# Recap: Parameter Estimation

Example of estimator:

- ▶ Maximum likelihood estimator:

$$\hat{\mu} = \frac{1}{N} \sum_{n=1}^N x_n$$

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



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).

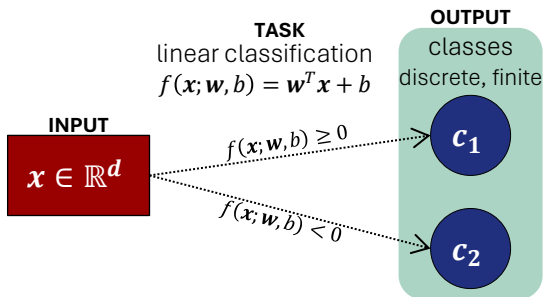
# Linear Classifiers

- Functions of the type

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

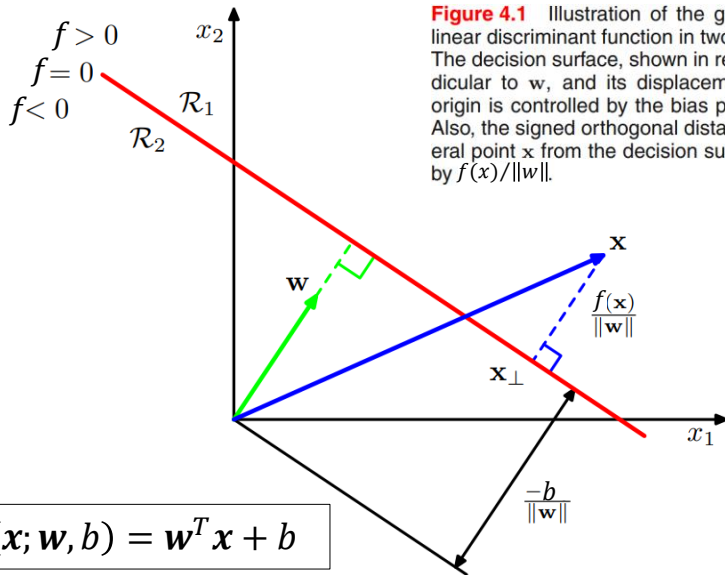
where  $\mathbf{w} \in \mathbb{R}^d$  and  $b \in \mathbb{R}$  are parameters to learn.

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



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

# Linear Classifiers

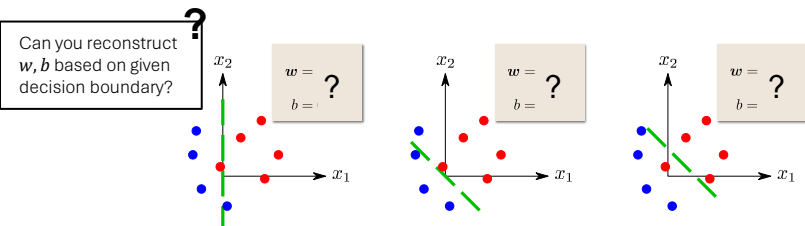
- Functions of the type

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

where  $\mathbf{w} \in \mathbb{R}^d$  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:



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



# Linear Classifiers

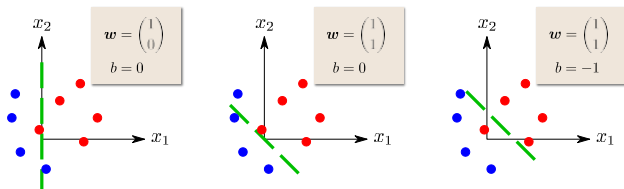
- Functions of the type

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

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

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

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



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

# Mean Separation Criterion

## Idea:

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

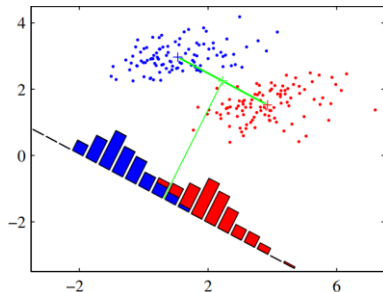
## Approach:

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

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{i \in \mathcal{C}_1} z_i \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{i \in \mathcal{C}_2} z_i$$

- ▶ Then we would like to find  $\mathbf{w}$  that maximizes the difference of means, i.e., we express the means as a function of  $\mathbf{w}$  and pose the optimization problem:

$$\operatorname{argmax}_{\mathbf{w}} |\mathbf{m}_2(\mathbf{w}) - \mathbf{m}_1(\mathbf{w})| \text{ s.t. } \|\mathbf{w}\| = 1$$

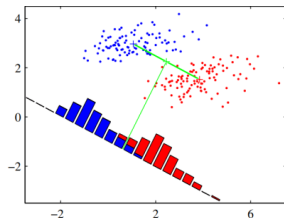


# Derivation of Mean Separation

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

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

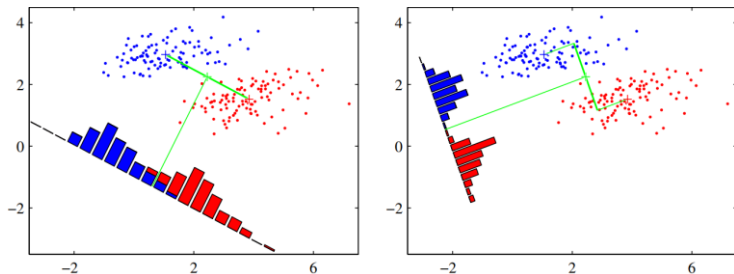
where  $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2$  are the emp. means in input space.



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

# 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.



# 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.

$$\begin{aligned} \mathbf{m}_1 &= \frac{1}{N_1} \sum_{i \in \mathcal{C}_1} \mathbf{z}_i & \mathbf{m}_2 &= \frac{1}{N_2} \sum_{i \in \mathcal{C}_2} \mathbf{z}_i \\ \mathbf{s}_1 &= \sum_{i \in \mathcal{C}_1} (\mathbf{z}_i - \mathbf{m}_1)^2 & \mathbf{s}_2 &= \sum_{i \in \mathcal{C}_2} (\mathbf{z}_i - \mathbf{m}_2)^2 \end{aligned}$$

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

$$\operatorname{argmax}_{\mathbf{w}} \frac{(\mathbf{m}_2(\mathbf{w}) - \mathbf{m}_1(\mathbf{w}))^2}{\mathbf{s}_2(\mathbf{w}) + \mathbf{s}_1(\mathbf{w})}$$

# Deriving the Fisher Discriminant (1)

---

The within-class variance can be expanded as:

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

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

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

as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

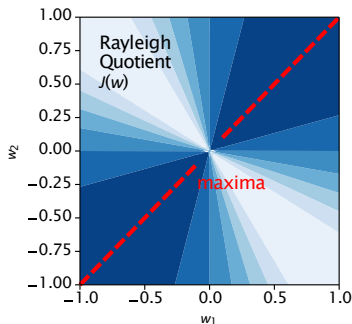
where

$$\mathbf{S}_B = (\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)(\hat{\boldsymbol{\mu}}_2 - \hat{\boldsymbol{\mu}}_1)^T$$

$$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$$

are the '**B**etween-class' and '**W**ithin-class' scatter matrices resp.

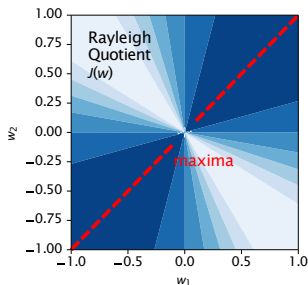
$J(\mathbf{w})$  is known as Rayleigh Quotient.



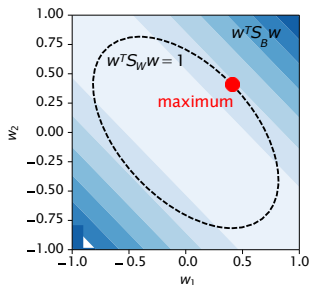
# Deriving the Fisher Discriminant (3)

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

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



Constrained  
formulation  
→





# Deriving the Fisher Discriminant (4)

We start with the constrained optimization problem:

$$\operatorname{argmax}_{\mathbf{w}} \mathbf{w}^T \mathbf{S}_B \mathbf{w} \quad \text{s.t.} \quad \mathbf{w}^T \mathbf{S}_W \mathbf{w} = 1$$

## Method of Lagrange Multipliers

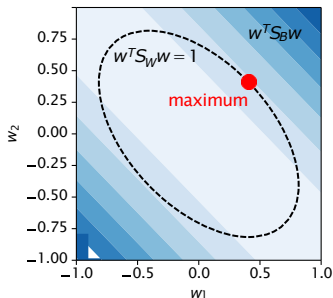
**Step 1:** We build the Lagrangian

$$\nabla \mathcal{L}(\mathbf{w}, \lambda) = \mathbf{w}^T \mathbf{S}_B \mathbf{w} + \lambda(1 - \mathbf{w}^T \mathbf{S}_W \mathbf{w})$$

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

$$\mathbf{S}_B \mathbf{w} = \lambda \mathbf{S}_W \mathbf{w}$$

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

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

- ▶ Expanding the term  $S_B$ , we get:

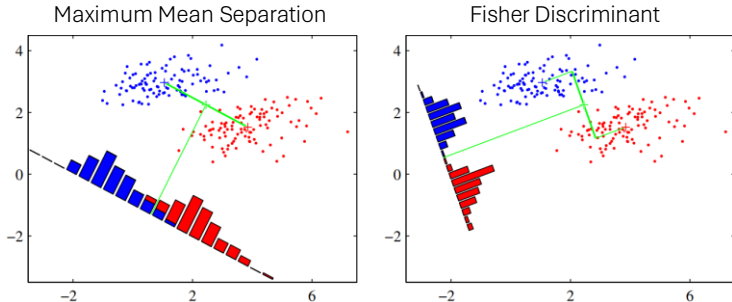
$$S_W^{-1} (\hat{\mu}_2 - \hat{\mu}_1) \underbrace{(\hat{\mu}_2 - \hat{\mu}_1)^T \mathbf{w}}_{\text{scalar}} = \lambda \mathbf{w}$$

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

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

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

# Mean Separation vs. 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).

# Decision Theory vs. Fisher Discriminant

---

## Bayesian decision theory:

Discriminant has the form

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

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

## Fisher discriminant:

Discriminant has the form

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

- ▶ No generative assumption. Classifier only derived from a criterion on mean and dispersion of the data in projected space.
- ▶ Offset  $\mathbf{b}$  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 perfectly separates training data (if the data is linearly separable).
- ▶ Trained using a simple and cheap iterative procedure.
- ▶ The perceptron gave rise to artificial neural networks.

# The Perceptron Algorithm

- Consider the linear model

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

and let  $t_i$  be  $+1$  or  $-1$  when the true class of  $\mathbf{x}_i$  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  $\mathbf{x}_i$  is correctly classified ( $y_i = t_i$ ), continue.
4.    If observation  $\mathbf{x}_i$  is wrongly classified ( $y_i \neq t_i$ ), do:  
         $\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \mathbf{x}_i t_i$   
         $b \leftarrow b + \eta \cdot t_i$   
        where  $\eta$  is the learning rate.

# The Perceptron Algorithm

$$\mathbf{w} \leftarrow \mathbf{w} + \eta \cdot \mathbf{x}_i t_i$$

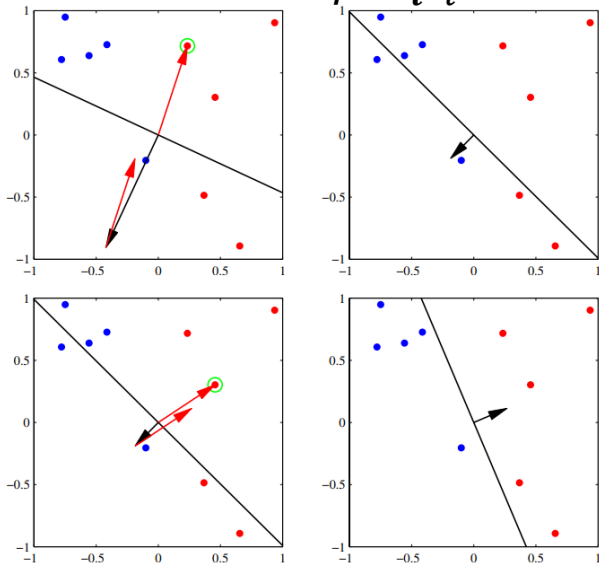


Figure taken from C.M. Bishop (2009), "Pattern Recognition and Machine Learning", 9ed.



# The Perceptron: Optimization View

---

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

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

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

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

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

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

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

# The Perceptron: Optimization View

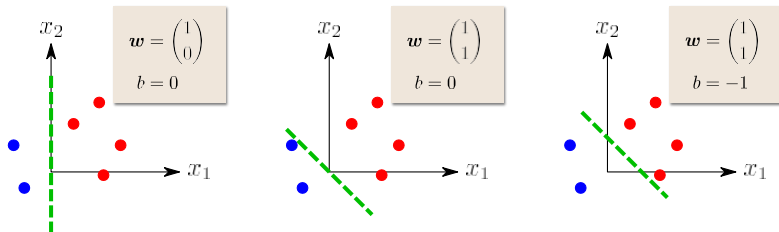
---

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

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

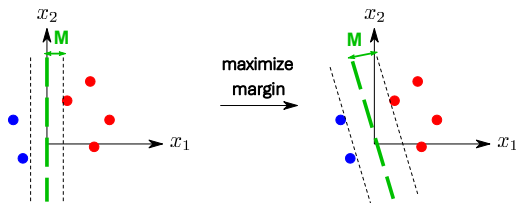
- ▶ This objective can be interpreted as measuring for each wrongly classified data point how far the data point is from the decision boundary and penalizing accordingly.
- ▶ In practice various strategies can be implemented to optimize this objective (e.g. gradient descent on  $E(\mathbf{w}, b)$  directly, 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  $E(\mathbf{w}, b) = 0$ .
- ▶ Some solutions are obviously better than others, e.g., those with a decision boundary separates the classes with a large margin.

# Large Margin Classifiers

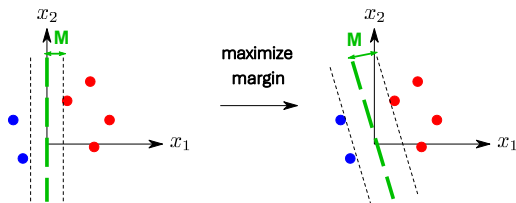


**Idea:** Induce large margin  $M$  by redefining the optimization problem as:

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

- ▶ (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:

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

(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

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

The same objective can be rewritten as:

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

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

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

Redefine  $\mathbf{w} \rightarrow \mathbf{w}/M$  and  $b \rightarrow b/M$

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

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

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



# 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.