CSE 575
Statistical Machine Learning

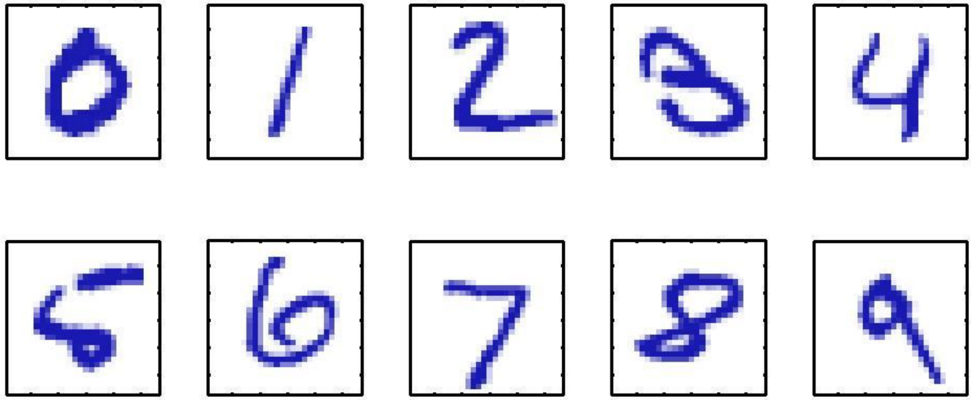
Lecture 7

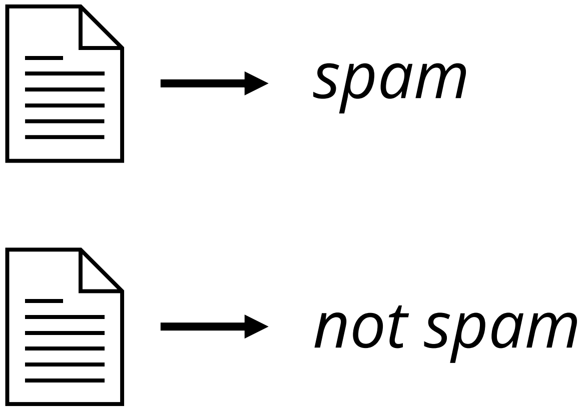
YooJung Choi

Fall 2022

Classification

- Assign each input $\mathbf{x}$ to one of K discrete classes $\mathcal{C}_1, \ldots, \mathcal{C}_K$





Classification

- Assign each input $\mathbf{x}$ to one of K discrete classes $\mathcal{C}_1, \ldots, \mathcal{C}_K$

1. *Discriminant functions*: directly map $\mathbf{x} \mapsto \mathcal{C}_k$
2. *Probabilistic generative models*: model $p(\mathbf{x}, \mathcal{C}_k)$ to compute $p(\mathcal{C}_k|\mathbf{x})$
3. *Probabilistic discriminative models*: model $p(\mathcal{C}_k|\mathbf{x})$ only

Linear models for classification

- Functions of the form:

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$$

- *Generalized linear models* (given some assumptions)
- $f(\cdot)$ is a fixed non-linear "activation" function:
    - Example: $f(u) = \begin{cases} 1 \text{ if } u \geq 0 \\ 0 \text{ otherwise} \end{cases}$
- *Linear* decision boundary
- Note: can also apply non-linear basis functions $\phi_j(\mathbf{x})$

Linear discriminant: binary case

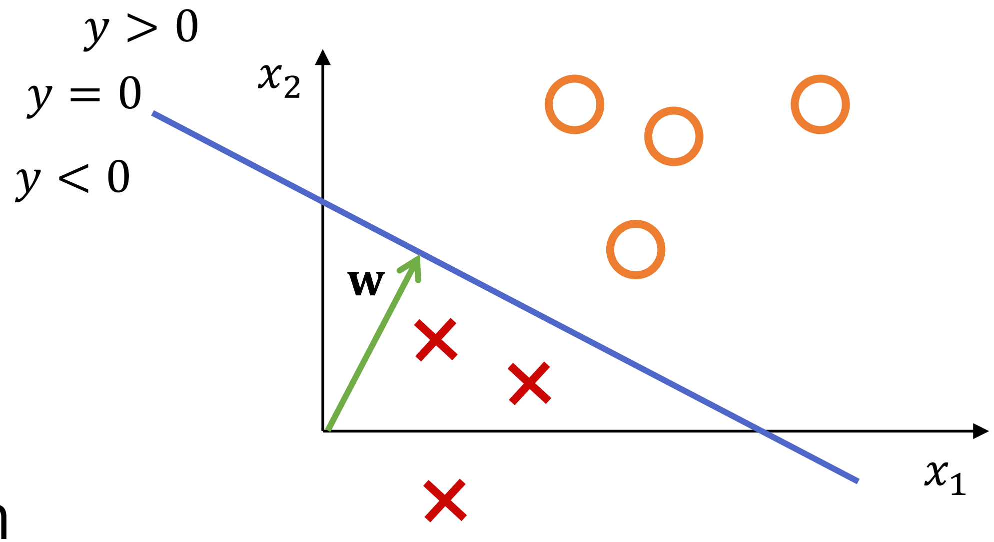
- Start with binary class: $t \in \{0,1\}$
- Simplest form of discriminant function:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

again, assume a dummy feature $x_0 = 1$.

- Apply threshold function to get classification
- Terminology: *decision boundary:* $\{\mathbf{x}|y(\mathbf{x}) = 0\}$

decision regions: $\{\mathbf{x}|y(\mathbf{x}) \geq 0\}, \{\mathbf{x}|y(\mathbf{x}) < 0\}$



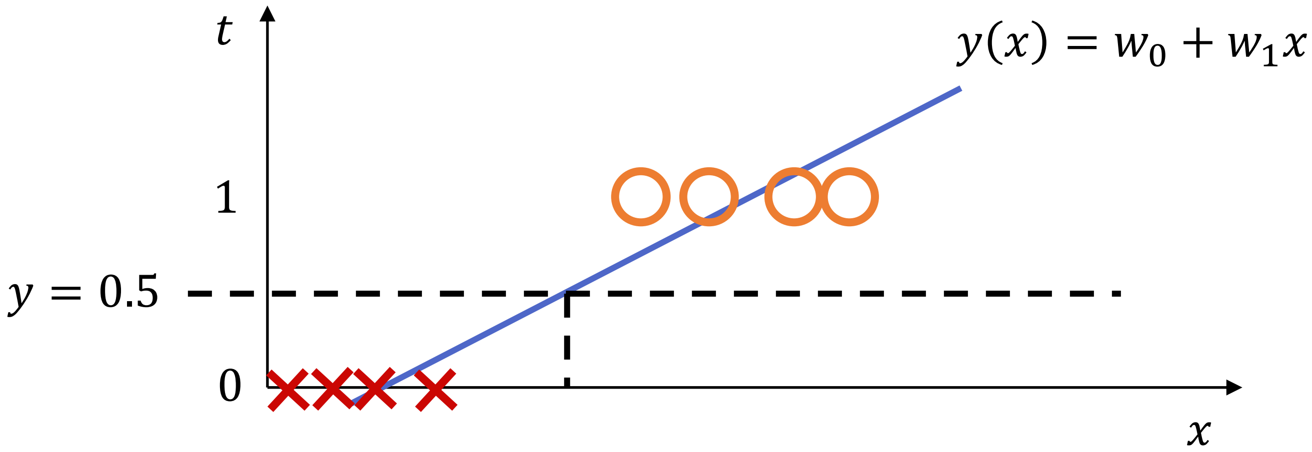
Least squares for classification

- How to learn the decision boundary $\mathbf{w}$?
- Minimize sum-of-squares error:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_{\mathrm{n}} - t_n)^2$$

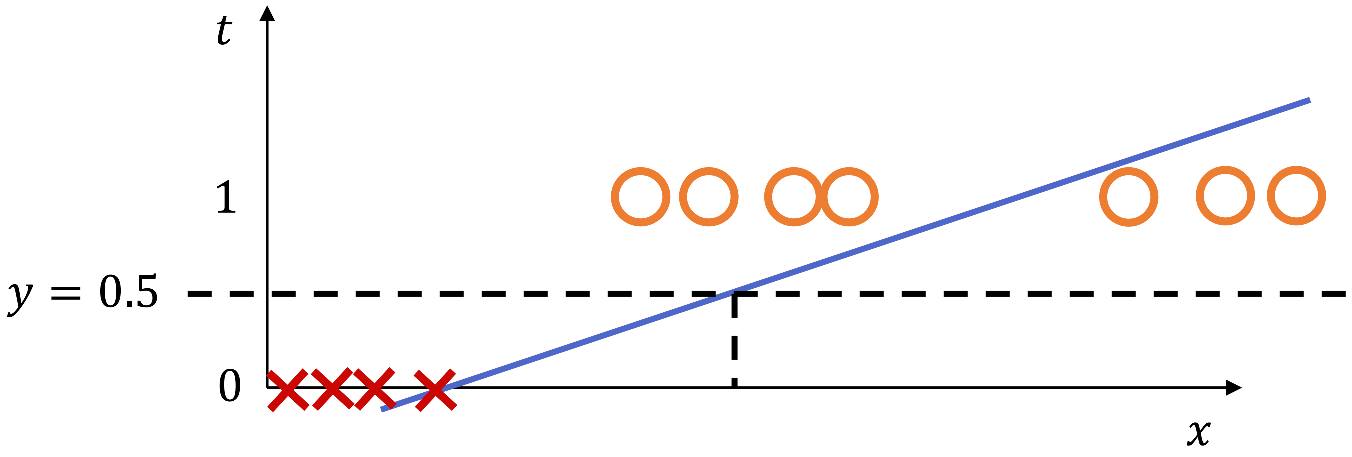
- i.e. linear regression except $t_n \in \{0,1\}$
- Threshold $\mathbf{w}^T\mathbf{x} \geq 0.5$ to get classification
- Close-form solution as in regression

Problems with least squares



Problems with least squares

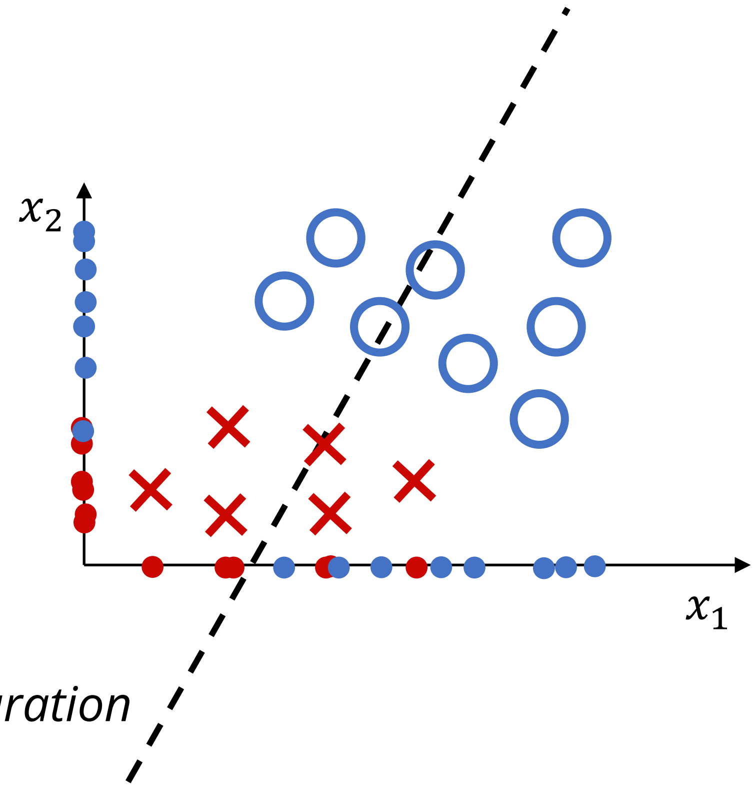
- Let's add more easy examples



- Decision boundary got worse! Incorrectly classifies previously correct examples
- Problem: function can return values outside [0,1]

Fisher's linear discriminant

- Main idea: interpret $\mathbf{w}^T\mathbf{x}$ as projection of inputs onto a line
- i.e. reduce the dimension to 1
- Threshold $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} \geq y_0$ for classification
- What is the best direction $\mathbf{w}$ for projection?



Projection for perfect separation

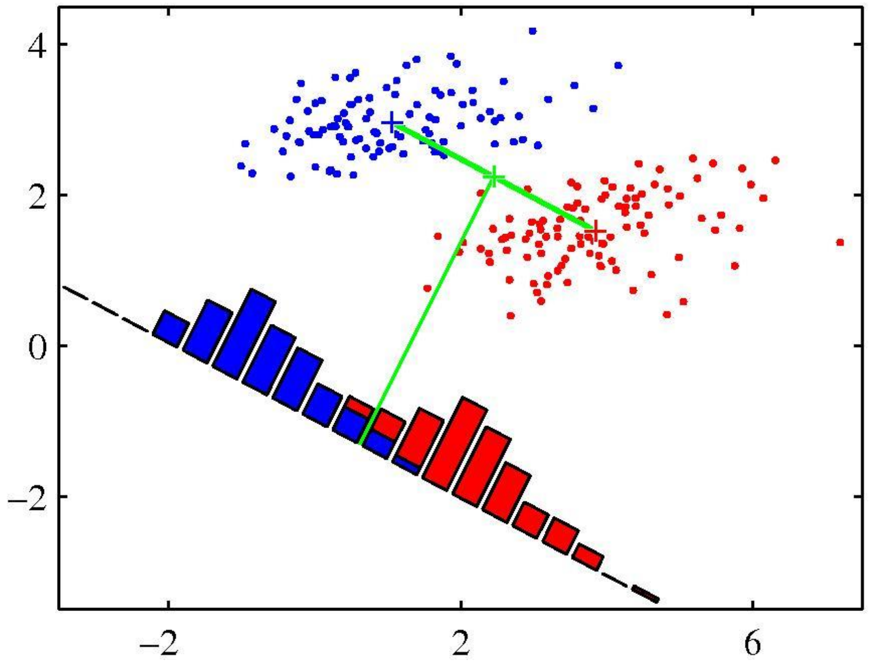
Fisher's linear discriminant

- Attempt 1: project s.t. the "average" points of each class are far apart

  i.e. project onto the line connecting the class means
- Maximize $\mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1)$

  where $\mathbf{m}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k}\mathbf{x}_n$
- $\mathbf{w}^* \propto \mathbf{m}_2 - \mathbf{m}_1$
- Problem: may overlap if there is

  within-class variance is in this direction



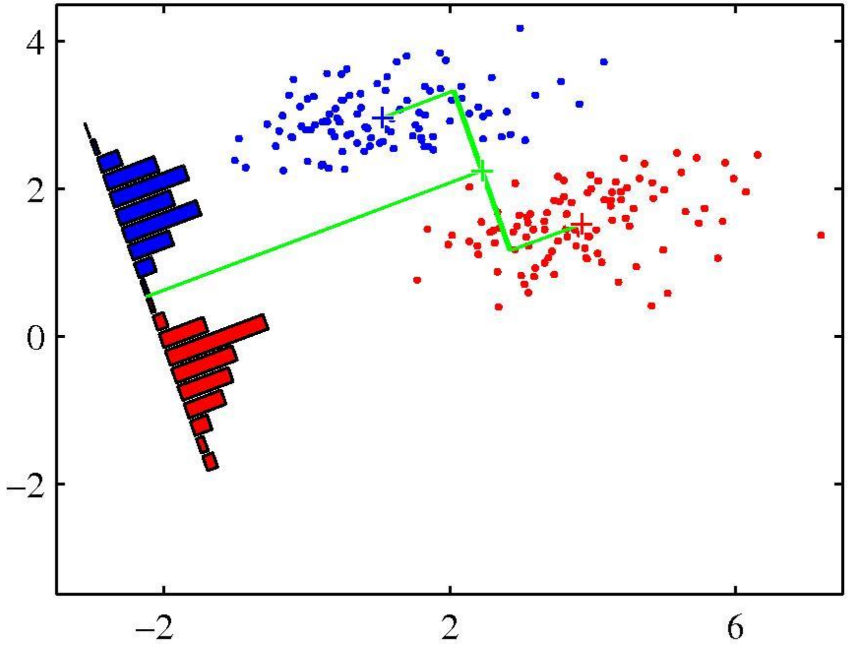
Fisher's linear discriminant

- (1) maximize the separation of means between classes, as well as

  (2) minimize the variance within each class
- Projected means: $m_k = \mathbf{w}^T \mathbf{m}_k$
- Projected variance: $s_k^2 = \sum_{n \in \mathcal{C}_k} (\mathbf{w}^T \mathbf{x}_n - m_k)^2$
- Fisher criterion:

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

  maximize w.r.t. $\mathbf{w}$



Fisher’s linear discriminant

- Fisher criterion: $J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$
- In matrix notation:

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

- Between-class covariance matrix:

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

- Total within-class covariance matrix:

$$\boldsymbol{S}_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T$$

Fisher's linear discriminant

- Fisher criterion is maximized when:
$$\mathbf{w}^T \boldsymbol{S}_B \mathbf{w}(\boldsymbol{S}_W \mathbf{w}) = \mathbf{w}^T \boldsymbol{S}_W \mathbf{w}(\boldsymbol{S}_B \mathbf{w})$$
- We only care about the direction of $\mathbf{w}$, and not its magnitude: drop the scalar factors:
$$\boldsymbol{S}_W \mathbf{w} = \boldsymbol{S}_B \mathbf{w}$$
- $\boldsymbol{S_B}\mathbf{w} = (\mathbf{m_2} - \mathbf{m_1})(\mathbf{m_2} - \mathbf{m_1})^{\mathrm{T}}\mathbf{w} = \text{scalar} \times (\mathbf{m_2} - \mathbf{m_1}) \propto (\mathbf{m_2} - \mathbf{m_1})$
- Therefore: $\mathbf{w} \propto \boldsymbol{S}_W^{-1}(\mathbf{m_2} - \mathbf{m_1})$
- If $\boldsymbol{S}_W^{-1} = \lambda I$ *(i.e. isotropic), then* $\mathbf{w} \propto (\mathbf{m_2} - \mathbf{m_1})$

    Within-class covariance is spherical

The perceptron algorithm

- Foundations for neural networks
- Limited to binary class.
- For ease of notation: $t = 1$ for class 1, $t = -1$ for class 2
- Discriminant model of the form:

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x}) \quad \text{where } f(a) = \begin{cases} +1, a \geq 0 \\ -1, a < 0 \end{cases}$$

- Recall: sum-of-squares error was problematic (penalizes predictions outside of (0,1) range even though correctly classified)
- Main idea: penalize only misclassified examples

The perceptron algorithm

$$y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x}) \quad \text{where } f(a) = \begin{cases} +1, a \geq 0 \\ -1, a < 0 \end{cases}$$

- An example $\mathbf{x}_n$ is misclassified iff
$$\mathbf{w}^T\mathbf{x}_n \cdot t_n < 0$$
- Goal: if possible, find $\mathbf{w}$ s.t. $\mathbf{w}^T\mathbf{x}_n \cdot t_n \geq 0$ for all n
- *Perceptron criterion*:
$$E_P(\mathbf{w}) = -\sum_{n\in\mathcal{M}} \mathbf{w}^T\mathbf{x}_n t_n$$
- $\mathcal{M}$ is the set of misclassified examples

The perceptron algorithm

- Perceptron rule: iterative update using a misclassified example $\mathbf{x}_\mathrm{n}$ (c.f. minimizing the perceptron criterion using stochastic gradient descent):

$$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)} - \eta \nabla E_p(\mathbf{w}) = \mathbf{w}^{(k)} + \eta \mathbf{x}_\mathrm{n} t_n$$

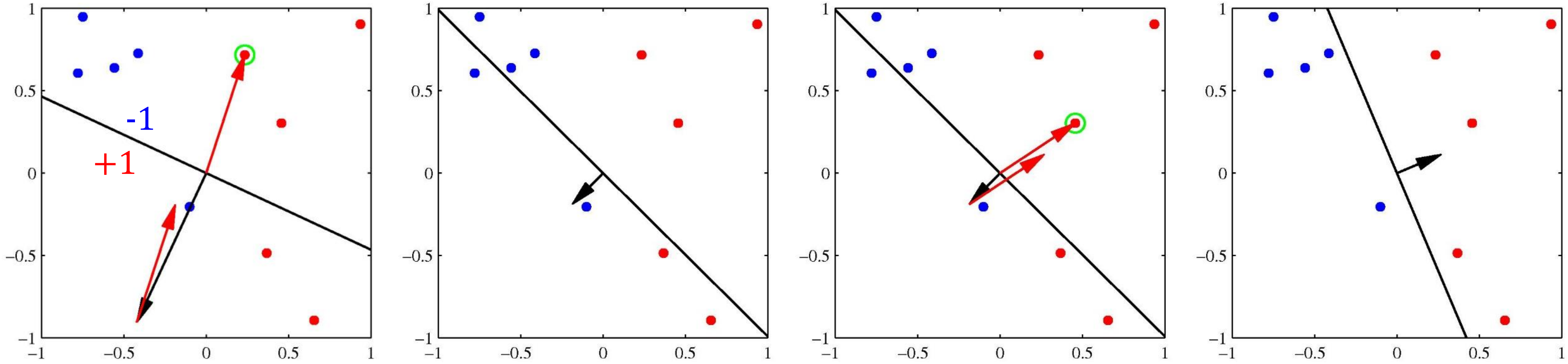
traditionally $\eta = 1$

- After a single update using $\mathbf{x}_\mathrm{n}$, its contribution to the error is reduced:

$$-\mathbf{w}^{(k+1)T}\mathbf{x}_\mathrm{n} t_n = -\mathbf{w}^{(k)T}\mathbf{x}_\mathrm{n} t_n - \eta(\mathbf{x}_\mathrm{n} t_n)^T \mathbf{x}_\mathrm{n} t_n < -\mathbf{w}^{(k)T}\mathbf{x}_\mathrm{n} t_n$$

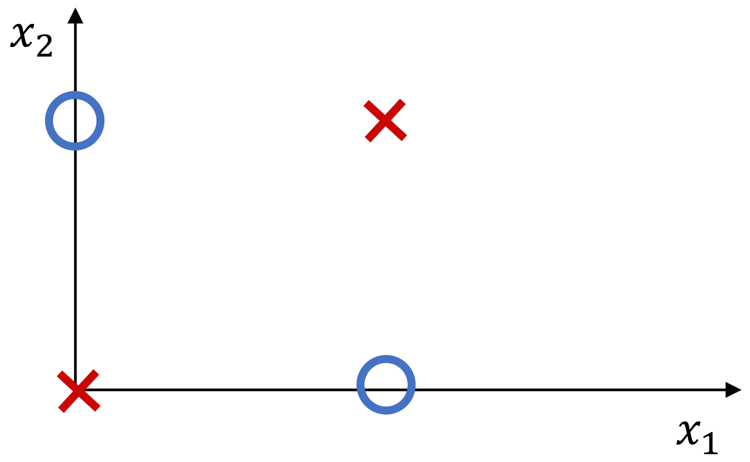
- However, the changed weight vector could cause previously correct examples to be misclassified. I.e., the total error is not guaranteed to decrease after every iteration.

The perceptron algorithm



The perceptron algorithm

- *The perceptron convergence theorem:* if the data is linearly separable, the perceptron learning algorithm finds the solution in a finite number of steps.
- But it may take very long…
- If the data is *not linearly separable*, perceptron algorithm will never converge

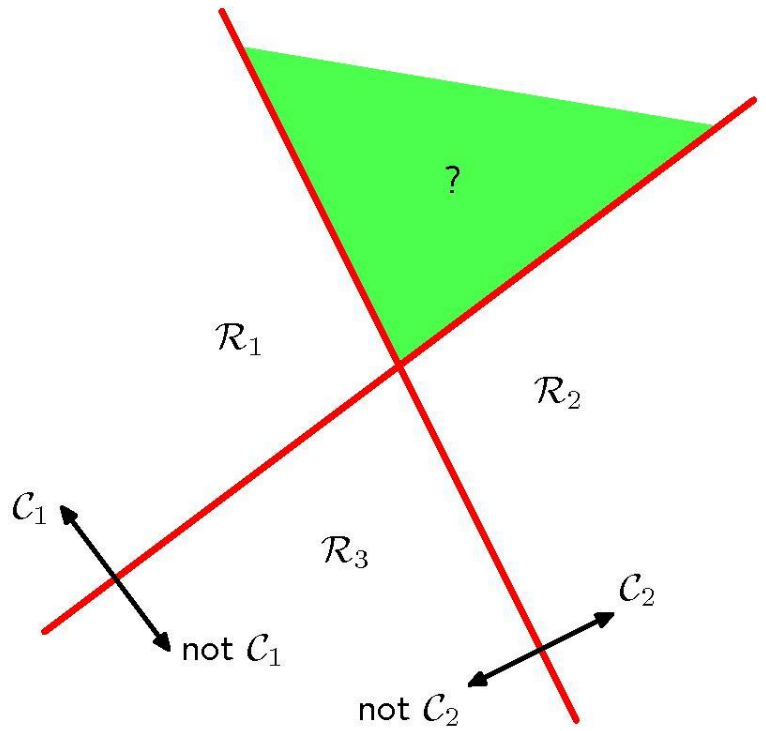


Discriminant functions: multi-class

- How do we extend linear discriminants to K > 2 classes?
- 1st attempt: use K-1 binary classifiers, each classifying $\mathcal{C}_k$ vs. not $\mathcal{C}_k$

  “one-vs-the-rest”
- Does not work!

  May lead to ambiguous regions



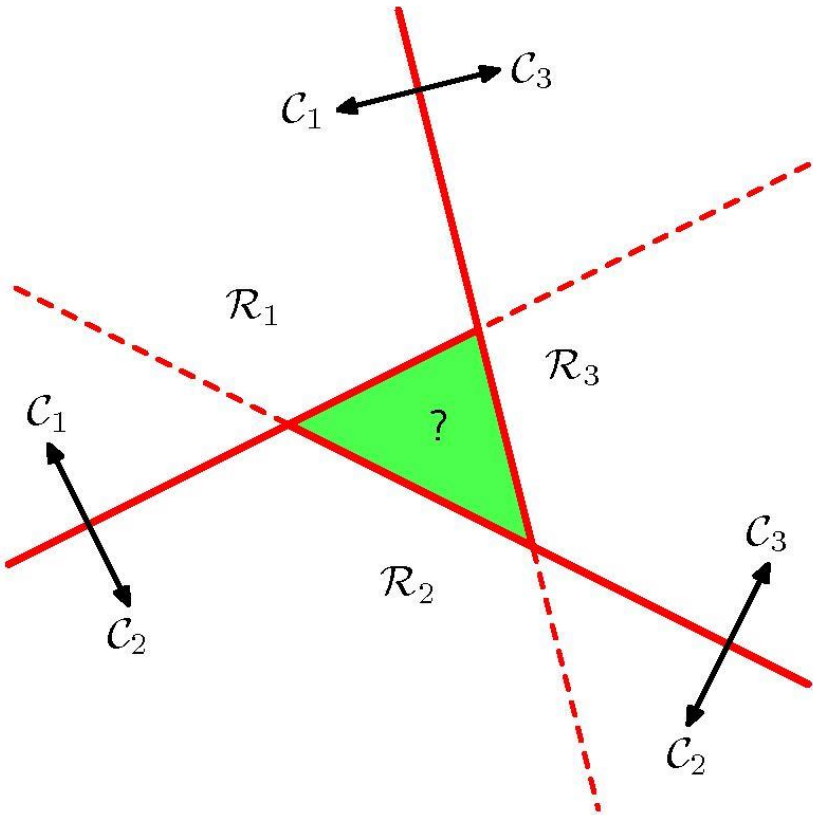
Discriminant functions: multi-class

- 2^{nd} attempt: use $K(K-1)/2$ binary classifiers, each classifying $\mathcal{C}_j$ vs. $\mathcal{C}_k$ for a pair $j \neq k$

  "one-vs-one"

- Classify according to a majority vote
- Does not work either!

  May lead to ambiguous regions



Discriminant functions: multi-class

- A single K-class discriminant comprising K linear functions:

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- Assign $\mathbf{x}$ to $\operatorname{argmax}_k y_k(\mathbf{x})$

  i.e. $\mathbf{x} \mapsto \mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$

- The decision boundary between $\mathcal{C}_j$ and $\mathcal{C}_k$ is given by a (D-1)-dimensional hyperplane $y_k(\mathbf{x}) = y_j(\mathbf{x})$
- Decision regions are convex