PERCEPTRON AND SUPPORT VECTOR MACHINE

DATA/MSML 603: Principles of Machine Learning

Perceptron

- Interested in finding a hyperplane that will separate the samples from two different labels

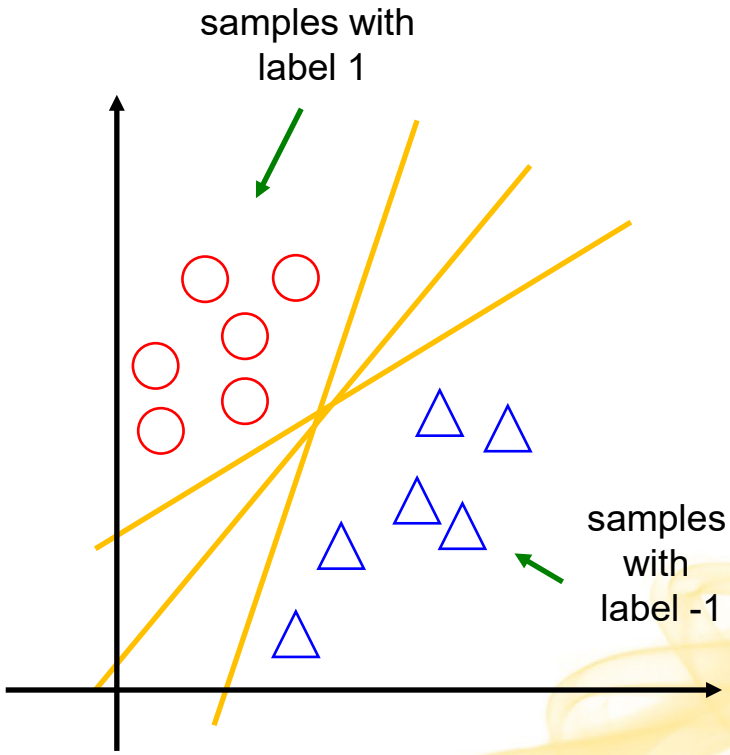
$$\begin{cases} \mathbf{w}^T\mathbf{x}^k + b > 0 \text{ if } y^k = +1 \\ \mathbf{w}^T\mathbf{x}^k + b < 0 \text{ if } y^k = -1 \end{cases} \Leftrightarrow -y^k(\mathbf{w}^T\mathbf{x}^k + b) < 0 \text{ for all } k = 1, \ldots, n$$

- Perceptron function:

$$\sum_{k=1}^{n} \max\left(0, -y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)$$

- Optimization problem:

$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \max\left(0, -y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)$$



separable case

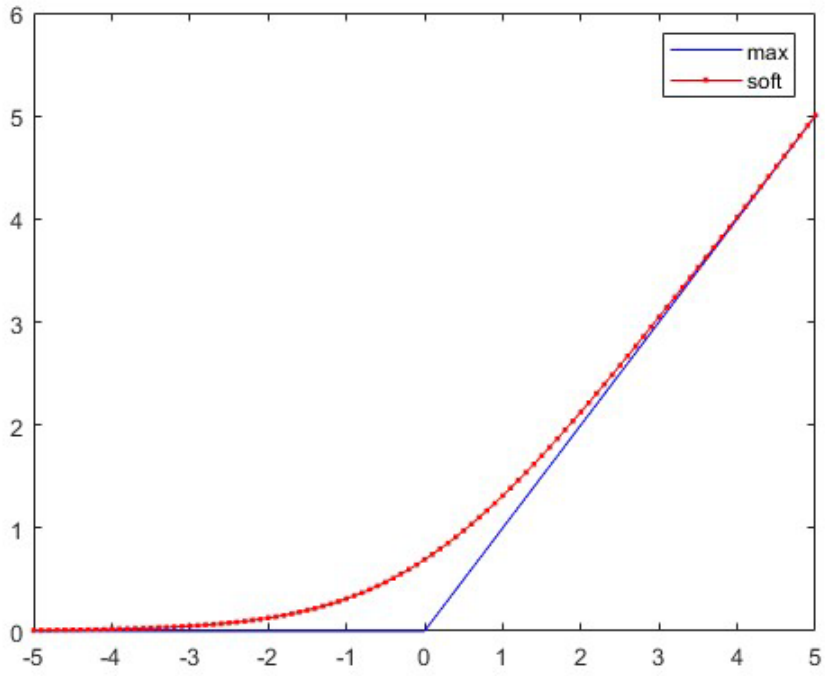
Perceptron

- Technical issues
  - Trivial and undesirable solution: $\mathbf{w} = \mathbf{0},\ b = 0$ (yay~ ??) 
  - While the perceptron function is continuous, it is not differentiable everywhere
    - For numerical solvers, having a differentiable objective function is beneficial (we will talk about this in the next lecture)
- Question: How can we fix these problems?
- Answer: Replace non-differentiable “max” function with smooth “softmax” function

Perceptron

- Softmax function:

$$\text{soft}(s_1, s_2) = \log(e^{s_1} + e^{s_2})$$



- After replacing max with softmax

$$\sum_{k=1}^{n} \max\left(0, -y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)$$

$$\sum_{k=1}^{n} \text{soft}\left(0, -y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)$$

$$= \sum_{k=1}^{n} \log\left(1 + \exp\left(-y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)\right)$$

Softmax cost function

- Does not have the trivial solution of zero vector

Perceptron

- Softmax minimization problem

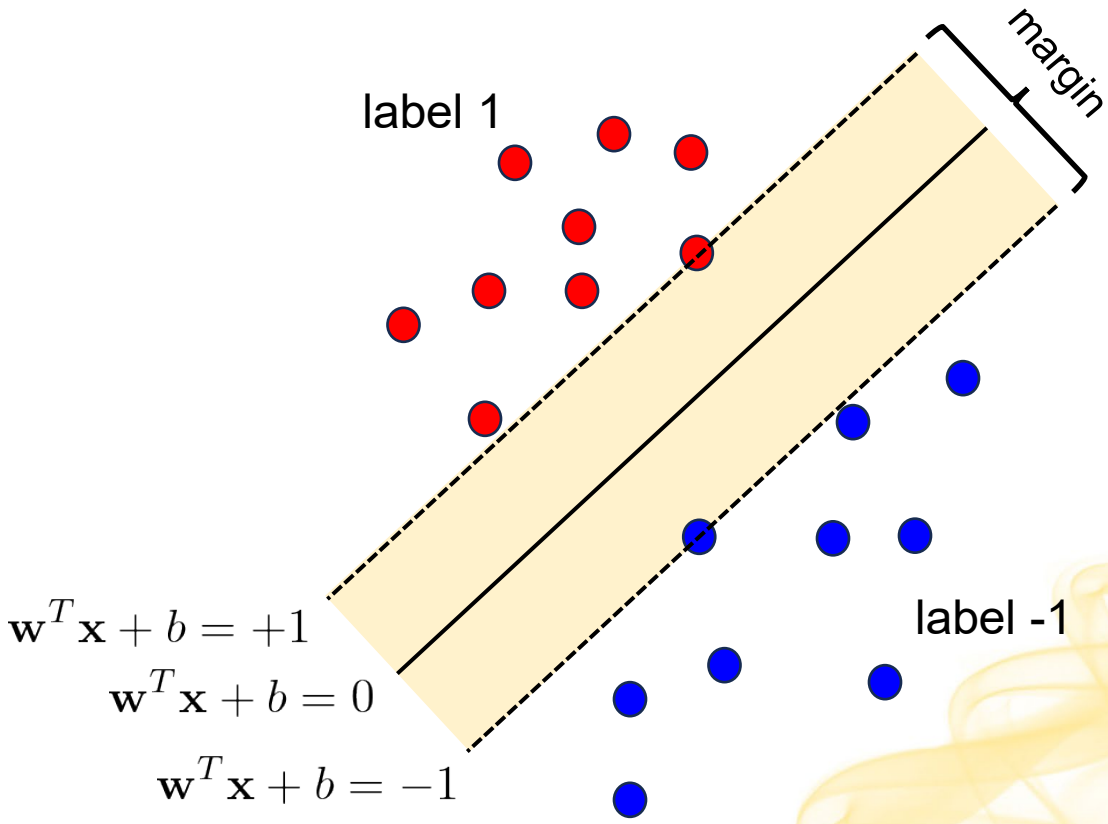
$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \log\left(1+\exp\left(-y^k(\mathbf{w}^T\mathbf{x}^k+b)\right)\right) \tag{6-1}$$

  - Differentiable and can be solved using gradient-based methods

Margin Perceptron

- Want to find a hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ that satisfies the following:
  - All samples $\mathbf{x}$ from label $y = +1$ satisfy
    $$\mathbf{w}^T\mathbf{x} + b \geq +1$$
  - All samples $\mathbf{x}$ from label $y = -1$ satisfy
    $$\mathbf{w}^T\mathbf{x} + b \leq -1$$
  - For each label, at least one sample satisfies equality
- Combining them ⇨ $y(\mathbf{w}^T\mathbf{x} + b) \geq 1$ or equivalently

$$\max\left(0, 1 - y(\mathbf{w}^T\mathbf{x} + b)\right) = 0$$



Margin Perceptron

- Margin perceptron cost function and optimization

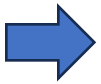
$$\sum_{k=1}^{n} \max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) \quad \Rightarrow \quad \boxed{\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)} \qquad (6\text{-}2)$$

  - Does not have a trivial solution
  - Value is equal to 0 if linearly separable
  - Still a valid cost function even when the data is not linearly separable
    - Value is strictly positive

Margin Perceptron

- Still has the same issue of not being differentiable everywhere
  - Can be handled using softmax approximation
    - Similar to the softmax approximation of the original perceptron with comparable performance
- Squared margin perceptron

$$\sum_{k=1}^{n} \left(\max\left(0, 1 - y^k (\mathbf{w}^T \mathbf{x}^k + b)\right) \right)^2$$



$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \left(\max\left(0, 1 - y^k (\mathbf{w}^T \mathbf{x}^k + b)\right) \right)^2$$

  - Solutions coincide when linearly separable

Slightly Different View of Perceptron

- Consider a step function for binary classification with labels +1 and -1

$$\text{sign}(\mathbf{w}^T\mathbf{x}+b) = \begin{cases} +1 & \text{if } \mathbf{w}^T\mathbf{x}+b > 0 \\ -1 & \text{if } \mathbf{w}^T\mathbf{x}+b < 0 \end{cases}$$

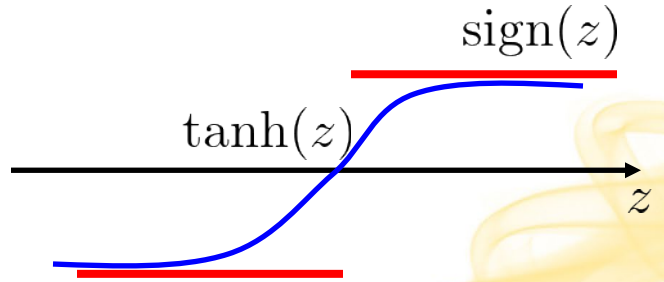
  - Constraints can be written as $\text{sign}\big(y_k(\mathbf{w}^T\mathbf{x}_k+b)\big) = 1$

- Oftentimes we want to have a smoother step function: define

$$\sigma(\mathbf{w}^T\mathbf{x}+b) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}} \in [0,1]$$

  - We can approximate the step function using

$$\tanh(\mathbf{w}^T\mathbf{x}+b) = 2\sigma(\mathbf{w}^T\mathbf{x}+b) - 1$$



Slightly Different View of Perceptron

- Rewriting the conditions $\text{sign}\left(y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) = 1$ using the approximation to the step function

$$\tanh\left(y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) \approx 1$$

or, equivalently,

$$\frac{1}{1 + e^{-y^k(\mathbf{w}^T\mathbf{x}^k + b)}} \approx 1 \quad \Leftrightarrow \quad \log\left(1 + e^{-y^k(\mathbf{w}^T\mathbf{x}^k + b)}\right) \approx 0$$

$$\Rightarrow \quad \text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \log\left(1 + \exp\left(-y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)\right)$$

logistic regression

(6-3)

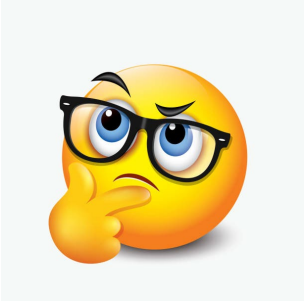
Original perceptron problem with softmax function !!!

Support Vector Machine

- Suppose that we want to find a hyperplane $(\mathbf{w}, b)$ with the largest margin subject to the conditions for margin perceptron

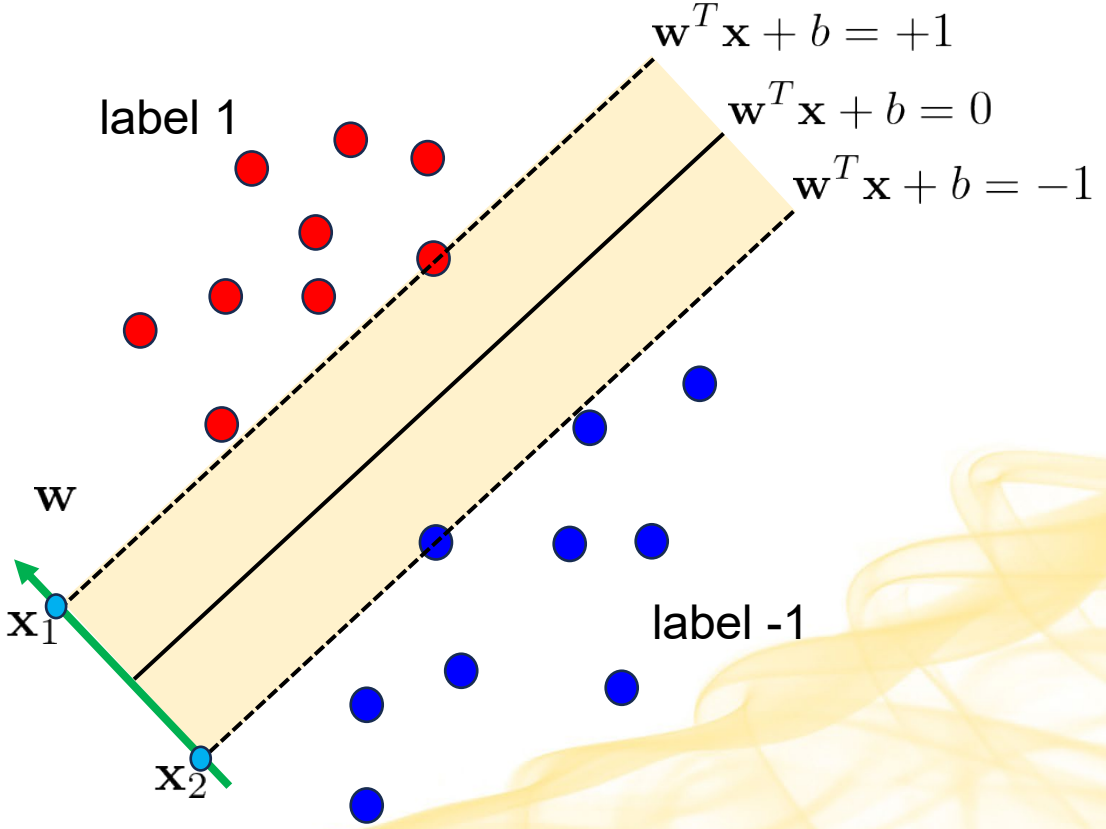
$$y^k(\mathbf{w}^T\mathbf{x}^k + b) \geq 1 \quad \text{for all } k = 1, \ldots, n \quad \Leftrightarrow \quad \max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) = 0 \quad \text{for all } k = 1, \ldots, n$$

- Question: How do we formulate the problem of finding the hyperplane with the largest margin?
  - We need a way to tell the computer how to measure the “margin” and maximize it



Support Vector Machine

- Consider a fixed hyperplane $(\mathbf{w}, b)$ found using margin perceptron
- Choose two points, one on each boundary, such that $\mathbf{x}_1 - \mathbf{x}_2$ is parallel to $\mathbf{w}$
- Distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ equal to the margin, i.e., the distance between the two hyperplanes



Support Vector Machine

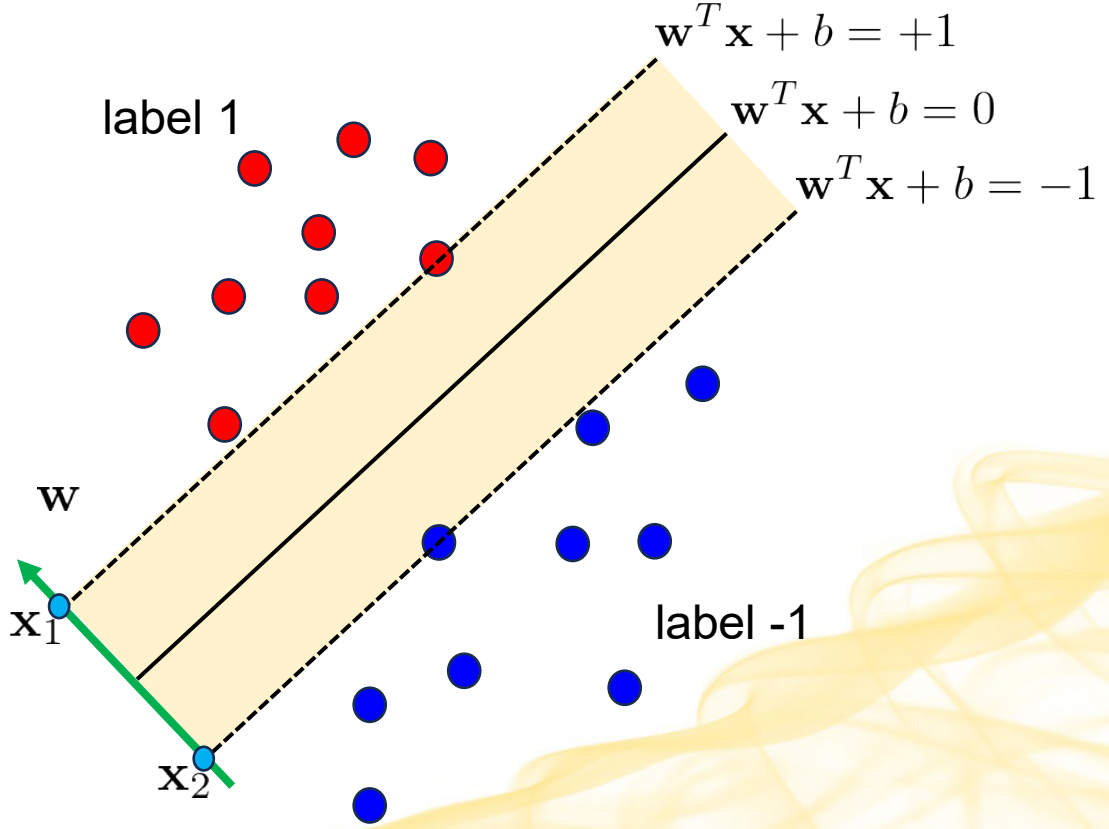
- First, note

$$\mathbf{x}_1^T\mathbf{w} + b - (\mathbf{x}_2^T\mathbf{w} + b) = (\mathbf{x}_1 - \mathbf{x}_2)^T\mathbf{w} = 2$$

- Because $\mathbf{x}_1 - \mathbf{x}_2$ and $\mathbf{w}$ are parallel, the angle between them is zero

$$(\mathbf{x}_1 - \mathbf{x}_2)^T\mathbf{w} = \langle \mathbf{x}_1 - \mathbf{x}_2, \mathbf{w} \rangle$$
$$= \|\mathbf{x}_1 - \mathbf{x}_2\| \cdot \|\mathbf{w}\| = 2$$

$$\text{margin} = \|\mathbf{x}_1 - \mathbf{x}_2\| = \frac{2}{\|\mathbf{w}\|}$$



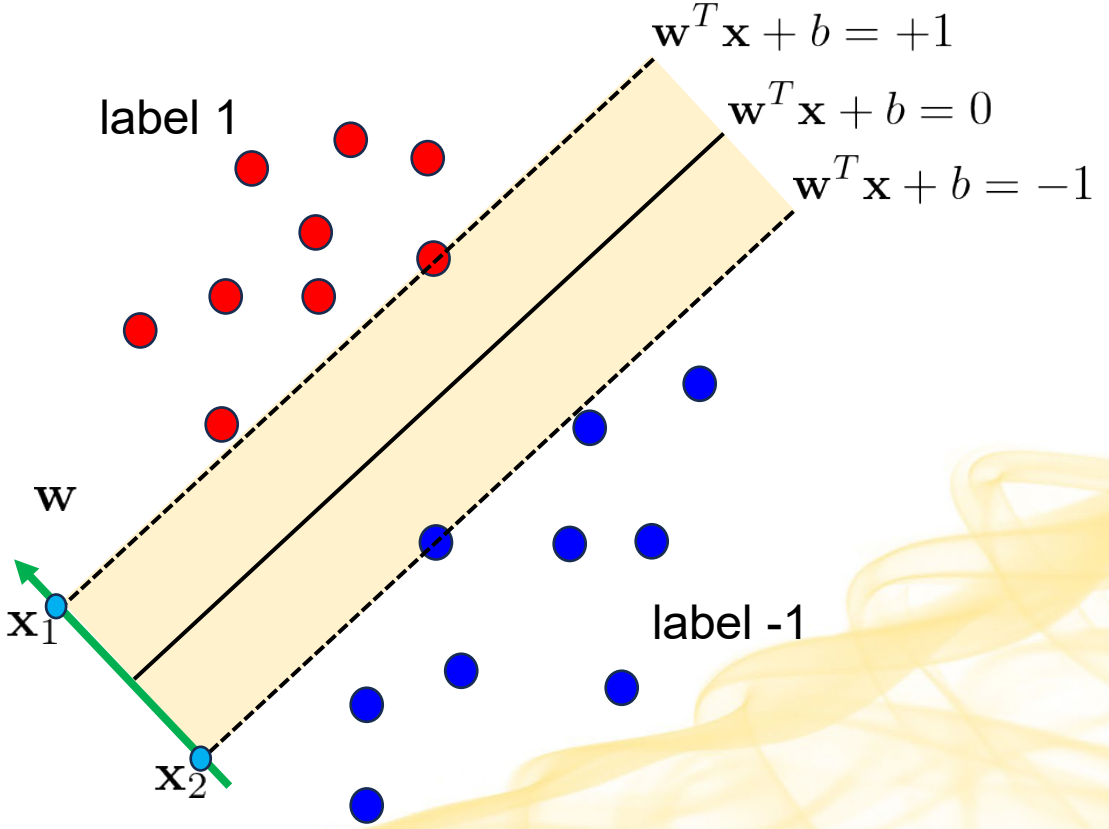
Support Vector Machine

- In order to maximize the margin, we should minimize $\|\mathbf{w}\|$

1. Hard-margin support vector for linearly separable case

$$\begin{aligned} &\text{minimize}_{\mathbf{w},b} && \|\mathbf{w}\|_2^2 \\ &\text{subject to} && \max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) = 0, \\ & && k = 1, \ldots, n \end{aligned}$$

   - Samples that lie on the boundaries or are closest to the hyperplane $\{\mathbf{x} \mid \mathbf{w}^T\mathbf{x} + b = 0\}$ are called "support vectors"



Support Vector Machine

2. Soft-margin support vector machine
   - Issue with hard-margin SVM : In practice, we do not know in advance whether or not the data is linearly separable
   - When the data is not linearly separable, there is no hyperplane that satisfies the constraints for all samples

$$\max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) = 0, \;\; k = 1, \ldots, n$$

   - We need to “relax” the constraints and allow some constraints to be violated
     - Instead, we penalize the constraints that are violated by our solution and incorporate the penalty into the cost function to be minimized

Support Vector Machine

2. Soft-margin support vector machine

$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) + \lambda\|\mathbf{w}\|_2^2 \qquad (6\text{-}4)$$

where $\lambda \geq 0$

- The smaller the weight λ is, the more emphasis we put on satisfying the constraints
- Soft-margin SVM cost function has an extra term $\lambda\|\mathbf{w}\|_2^2$, which is an ℓ_2 regularizer

---

Margin perceptron:

$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \max\left(0, 1 - y^k(\mathbf{w}^T\mathbf{x}^k + b)\right) \qquad (6\text{-}2)$$

Support Vector Machine

2. Differentiable version of soft-margin support vector machine

$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \log\left(1 + e^{-y^k(\mathbf{w}^T\mathbf{x}^k + b)}\right) + \lambda\|\mathbf{w}\|_2^2 \qquad (6\text{-}5)$$

where $\lambda \geq 0$

- Called “log-loss” support vector machine
- ℓ_2-regularized form of logistic regression

---

logistic regression:

$$\text{minimize}_{\mathbf{w},b} \sum_{k=1}^{n} \log\left(1 + \exp\left(-y^k(\mathbf{w}^T\mathbf{x}^k + b)\right)\right) \qquad (6\text{-}3)$$

Multiclass Classification: *c* Classes

- Two types of algorithms
    1. One-versus-all (OvA) multiclass classification
    2. Multiclass softmax classification

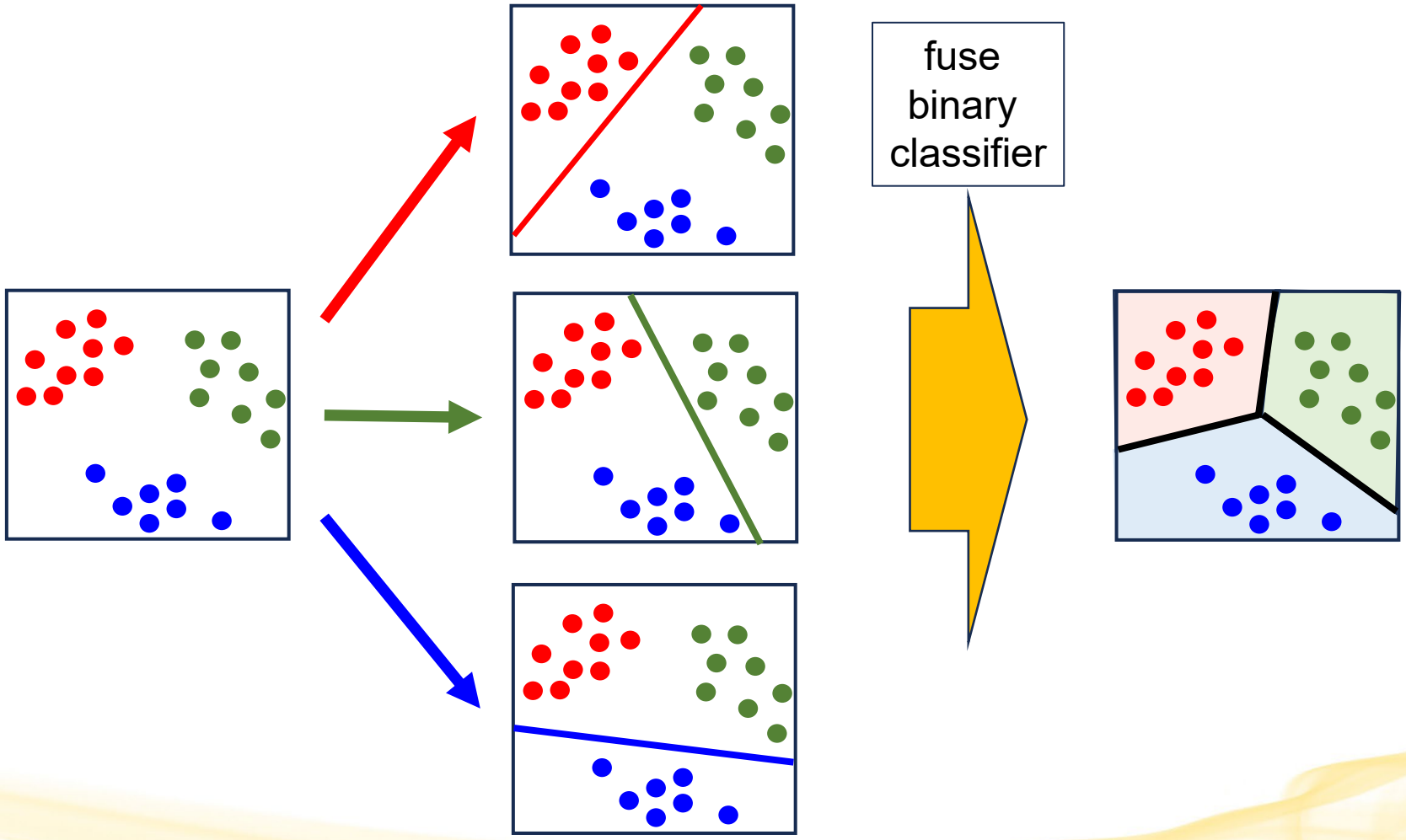
Multiclass Classification: *c* Classes

- One-versus-all (OvA) multiclass classification
  - For each class, we design/learn a binary (linear) classifier that distinguishes the class from the rest of the classes: for each label $j = 1, 2, \ldots, c$

$$\begin{cases} \mathbf{w}_j^T \mathbf{x} + b_j > 0 & \text{for samples from label } j \\ \mathbf{w}_j^T \mathbf{x} + b_j < 0 & \text{for samples from all other labels} \end{cases}$$

  - Fuse the binary classifiers to construct a multiclass classifier

Multiclass Classification: *c* Classes

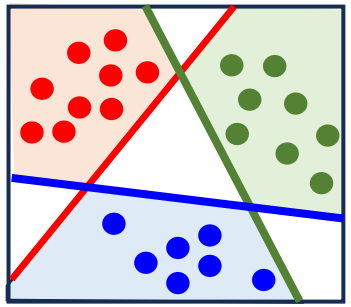


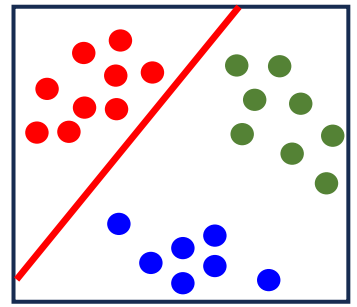
Multiclass Classification: *c* Classes

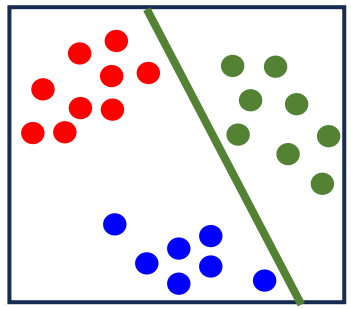
- Question: How do we fuse the binary classifiers?
- Naïve approach: Choose label j if

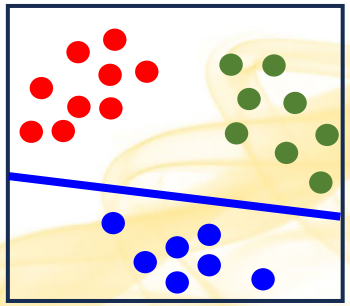
$$\begin{cases} \mathbf{w}_j^T \mathbf{x} + b_j > 0 \\ \mathbf{w}_i^T \mathbf{x} + b_i < 0 \quad \text{if } i \neq j \end{cases}$$

  - Works poorly in practice
    - Some regions not assigned to any label
    - Some regions assigned to more than one label







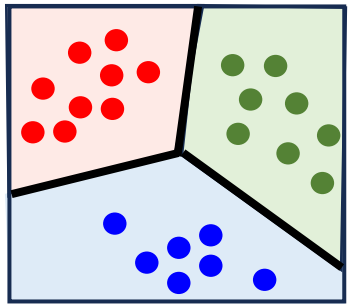


Multiclass Classification: *c* Classes

- Fusion rule: Assign $\mathbf{x}$ to label given by

$$y \in \text{argmax}_{j=1,\ldots,c} \ \mathbf{w}_j^T \mathbf{x} + b_j$$

  - Generalization of the naïve approach
  - Works well in many cases
  - Boundaries can be obtained from a set of linear equations



$$\mathbf{w}_j^T \mathbf{x} + b_j = \mathbf{w}_i^T \mathbf{x} + b_i, \ i \neq j \iff (\mathbf{w}_j - \mathbf{w}_i)^T \mathbf{x} + (b_j - b_i) = 0$$

Multiclass Classification: *c* Classes

- One-versus-all classification first learns *c* linear classifiers separately and fuse them afterwards to construct the final multiclass classifier
- Multiclass softmax classification determines the *c* classifiers jointly by learning all of their parameters simultaneously based on the fusion rule
- Recall: fusion rule $\alpha : \mathbb{R}^d \to \{1, 2, \ldots, c\} =: \mathcal{C}$ chooses the label according to

$$\alpha(\mathbf{x}) \in \operatorname{argmax}_{j=1,\ldots,c} \mathbf{w}_j^T \mathbf{x} + b_j$$

$$\mathbf{w}_{\alpha(\mathbf{x}^k)}^T \mathbf{x}^k + b_{\alpha(\mathbf{x}^k)} = \max_{j=1,\ldots,c} \mathbf{w}_j^T \mathbf{x}^k + b_j \iff \max_{j=1,\ldots,c} \mathbf{w}_j^T \mathbf{x}^k + b_j - \left(\mathbf{w}_{\alpha(\mathbf{x}^k)}^T \mathbf{x}^k + b_{\alpha(\mathbf{x}^k)}\right) = 0$$

$$\text{for all } k = 1, \ldots, n \qquad \text{for all } k = 1, \ldots, n$$

Multiclass Classification: *c* Classes

- Tune the parameters so that following condition holds for all samples

$$\max_{j=1,\ldots,c} \mathbf{w}_j^T \mathbf{x}^k + b_j - \left(\mathbf{w}_{y^k}^T \mathbf{x}^k + b_{y^k}\right) = 0, \;\; k = 1, 2, \ldots, n$$

- Summing over all samples

$$\text{minimize}_{(\mathbf{W},\mathbf{b})} \sum_{i=1}^{c} \sum_{k \in \mathcal{D}_i} \max_{j=1,\ldots,c} \mathbf{w}_j^T \mathbf{x}^k + b_j - \left(\mathbf{w}_{y^k}^T \mathbf{x}^k + b_{y^k}\right) \tag{6-6}$$

where $\mathbf{W} = (\mathbf{w}_i : i = 1, \ldots, c), \;\; \mathbf{b} = (b_i : i = 1, \ldots, c)$

Multiclass Classification: *c* Classes

- Same old issues with

$$\text{minimize}_{(\mathbf{W},\mathbf{b})} \sum_{i=1}^{c} \sum_{k \in \mathcal{D}_i} \left[\max_{j=1,\ldots,c} \mathbf{w}_j^T \mathbf{x}^k + b_j - \left(\mathbf{w}_{y^k}^T \mathbf{x}^k + b_{y^k} \right) \right]$$

  - Trivial solution
  - Nondifferentiable

- Softmax function : $\text{soft}(s_1, \ldots, s_c) = \log \left(\sum_{i=1}^{c} e^{s_i} \right)$

Multiclass Classification: *c* Classes

- Softmax function :

$$\text{minimize}_{(\mathbf{W},\mathbf{b})} \sum_{i=1}^{c} \sum_{k \in \mathcal{D}_i} \left[\log\left(\sum_{j=1}^{c} \exp\left(\mathbf{w}_j^T \mathbf{x}^k + b_j \right) \right) - \left(\mathbf{w}_{y^k}^T \mathbf{x}^k + b_{y^k} \right) \right]$$

$$\mathbf{w}_{y^k}^T \mathbf{x}^k + b_{y^k} = \log\left(\exp(\mathbf{w}_{y^k}^T \mathbf{x}^k + b_{y^k}) \right)$$

$$\text{minimize}_{(\mathbf{W},\mathbf{b})} \sum_{i=1}^{c} \sum_{k \in \mathcal{D}_i} \left[\log\left(1 + \sum_{j \neq y^k} \exp\left((\mathbf{w}_j - \mathbf{w}_{y^k})^T \mathbf{x}^k + (b_j - b_{y^k}) \right) \right) \right]$$

multiclass softmax cost function or softmax regression