Introduction to Machine Learning

Lecture 12:

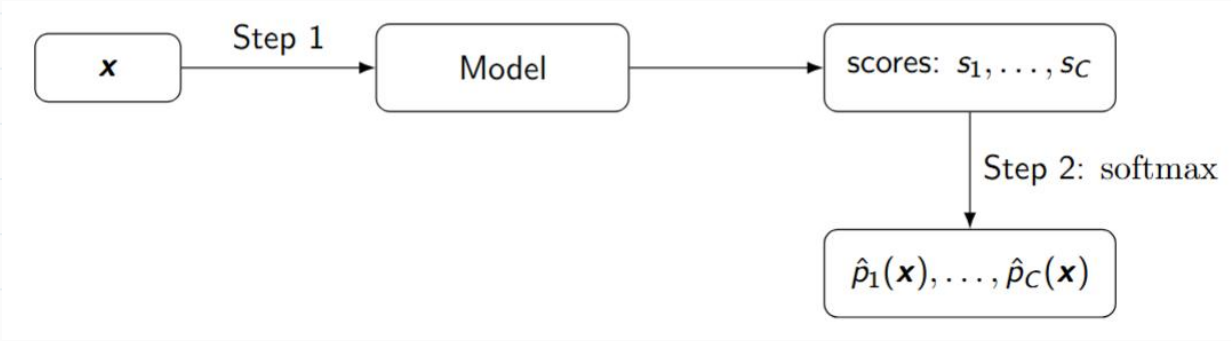
Softmax Regression, Regularization using Prior & Support Vector Machine (SVM)

Prof. Subrahmanya Swamy Peruru
Department of Electrical Engineering
IIT Kanpur

Multiclass Logistic Regression (Softmax Regression)

- We studied logistic regression in the context of **binary classification**.
- We can generalize logistic regression to **multi-class classification** using **softmax** function

Key idea: predict a score for **each class**, and convert these scores into **probabilities**.



Note: Softmax is a generalization of the sigmoid function to multi-class.

Multi-class Logistic Regression model

Step 1: Compute a real-valued score for each class (Linear)

$$s_i = \mathbf{w}_i^\top \mathbf{x} + b_i$$

Step 2: Convert scores into probabilities (Softmax):

$$\text{softmax}_i(s_1, \ldots, s_C) = \frac{\exp(s_i)}{\exp(s_1) + \exp(s_2) + \cdots + \exp(s_C)}, \qquad \text{for } i \in \{1, 2, \ldots, C\}$$

Model: For $i \in \{1, \ldots, C\}$:

$$\hat{p}_i(\mathbf{x}) = \mathbb{P}(Y = i \mid X = \mathbf{x}) = \frac{e^{\mathbf{w}_i^\top \mathbf{x} + b_i}}{\sum_{j=1}^{C} e^{\mathbf{w}_j^\top \mathbf{x} + b_j}}.$$

- **Parameters to learn:** one weight vector and bias **for each class** $\{\mathbf{w}_i, b_i\}_{i=1}^{C}$
- Learned by minimizing the negative log-likelihood (NLL).

Training and Prediction

Binary classification (recall):

$$\mathrm{NLL}(\mathbf{w}, b) = -\sum_{i=1}^{n} [y_i \log \hat{p}_1(\mathbf{x}_i) + (1 - y_i) \log(1 - \hat{p}_1(\mathbf{x}_i))] .$$

Multi-class classification:

$$\mathrm{NLL}\left(\{\mathbf{w}_i, b_i\}_{i=1}^{C}\right) = -\sum_{i=1}^{n} \sum_{j=1}^{C} \mathbb{I}\{y_i = j\} \log \hat{p}_j(\mathbf{x}_i)$$

Prediction:

$$\hat{y} = \arg\max_j \hat{p}_j(\mathbf{x})$$

Since softmax is monotonic w.r.t. the scores:

$$\hat{y} = \arg\max_j (\mathbf{w}_j^{\top} \mathbf{x} + b_j).$$

Maximum A Posteriori (MAP) Estimation

A Regularization Technique

MAP (Maximum A Posteriori) Estimation

MLE: Uses only data

MAP: In addition to data, uses prior information / belief.

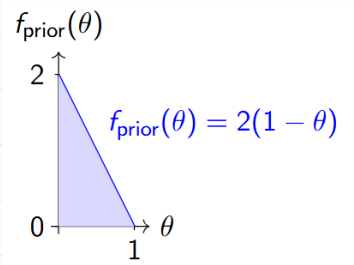
Coin Toss

$\mathrm{H} = 1, \mathrm{T} = 0$

Given:

- **Data:** 0, 1, 1, 0
- **Prior:** $f_{\text{prior}}(\theta)$ - Represents our belief (distribution) on coin bias θ **before seeing the data**

Prior Example:
coin is biased towards tail (gamblers design coin that profits them).



Task: Estimate θ

Compute the posterior distribution and find the MAP estimate

Posterior: $f_{\theta \mid \text{data}}(\theta)$ - Represents our belief about the parameter after observing data

$$\theta_{\text{MAP}} = \arg\max_{\theta} f_{\theta \mid \text{data}}(\theta)$$

MAP Estimation

Bayes Rule: For events A and B,

$$\boxed{P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)}}$$

Posterior distribution:

$$f_{\theta \mid \text{data}}(\theta) = \frac{P(\text{data} \mid \theta)\, f_{\text{prior}}(\theta)}{P(\text{data})}.$$

While maximizing over θ, we can ignore the denominator since it does not depend on θ

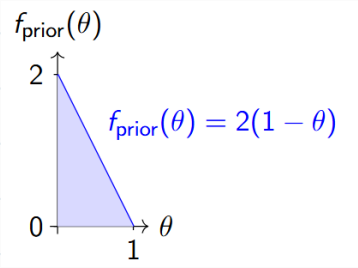
$$\begin{aligned}\theta_{\text{MAP}} &= \arg\max_\theta \; f_{\theta \mid \text{data}}(\theta) \\ &= \arg\max_\theta \; P(\text{data} \mid \theta)\, f_{\text{prior}}(\theta)\end{aligned}$$

Instead of maximizing, **take negative of log** and **minimize**:

$$\theta_{\text{MAP}} = \arg\min_\theta \left[-\log P(\,\text{data} \mid \theta\,) - \log f_{\text{prior}}(\theta) \right].$$

Loss: $\underbrace{\text{NLL}(\theta)}_{\text{data term}} + \underbrace{\left(-\log f_{\text{prior}}(\theta)\right)}_{\text{prior}}$

MAP Example



Observed data: $0, 1, 1, 0$

Likelihood: $\theta^2(1-\theta)^2$

Negative Log-likelihood $\text{NLL}(\theta) = -\log L(\theta) = -2\log\theta - 2\log(1-\theta)$

Negative Log-Prior: $\text{NLP}(\theta) = -\log f_{\text{prior}}(\theta) = -\log(2(1-\theta))$

$$\text{Loss:} \quad \underbrace{\text{NLL}(\theta)}_{\text{data term}} + \underbrace{\left(-\log f_{\text{prior}}(\theta)\right)}_{\text{prior}}$$

MAP Loss: $\text{NLL}(\theta) + \text{NLP}(\theta) = -2\log\theta - 2\log(1-\theta) - \log(2(1-\theta))$

Setting the gradient to zero gives:

$$-\frac{2}{\theta} + \frac{2}{1-\theta} + \frac{1}{1-\theta} = 0 \Rightarrow \theta_{\text{MAP}} = 0.4$$

Recall the MLE estimate: $\theta_{\text{MLE}} = 0.5$

The prior slightly pushes the MLE estimate toward zero, since it favors smaller values of θ.

Logistic Regression: Gaussian Prior on Weights

We can use the concept of **prior to push weights towards smaller values**.

Assume **i.i.d. Gaussian prior on weights**:

$$w_i \sim \mathcal{N}(0,1/\lambda), \quad \text{i.i.d.}$$

(**mean 0**, **variance $1/\lambda$**)

Large λ ⇒ variance is small ⇒ weights should be close to 0
Small λ ⇒ variance is small ⇒ large weights allowed

For a single component: $f_{\text{prior}}(w_i) \propto \exp\left(-\frac{\lambda}{2} w_i^2\right)$.

For the entire weight vector (independent components):

$$f_{\text{prior}}(\mathbf{w}) = \prod_{i=1}^{d} f_{\text{prior}}(w_i) \propto \prod_{i=1}^{d} \exp\left(-\frac{\lambda}{2} w_i^2\right)$$

$$= \exp\left(-\frac{\lambda}{2} \sum_{i=1}^{d} w_i^2\right) = \exp\left(-\frac{\lambda}{2} \| \mathbf{w} \|_2^2\right)$$

Gaussian Prior ⇒ L2 regularization

Gaussian prior: $f_{\text{prior}}(\mathbf{w}) = \exp\left(-\frac{\lambda}{2} \| \mathbf{w} \|_2^2\right)$.

MAP objective:

$$\text{NLL}(\theta) - \log f_{\text{prior}}(\mathbf{w})$$

$$= \text{NLL}(\theta) - \log\left[\exp\left(-\frac{\lambda}{2} \| \mathbf{w} \|_2^2\right)\right]$$

$$= \text{NLL}(\theta) + \frac{\lambda}{2} \| \mathbf{w} \|_2^2.$$

$$\theta_{\text{MAP}} = \arg\min_\theta \text{NLL}(\theta) + \frac{\lambda}{2} \| \mathbf{w} \|_2^2.$$

Similar as L2 regularization (ridge regression) seen in least squares.

Take away:

- L2 regularization ≡ zero-mean Gaussian prior on weights.
- λ controls how strongly small weights are encouraged.

Instead of Gaussian prior, if we assume Laplacian distribution as the prior for weights, we end up with L1 regularization (like LASSO).

Support Vector Machines (SVM)

Linear Classification via a Separating Hyperplane

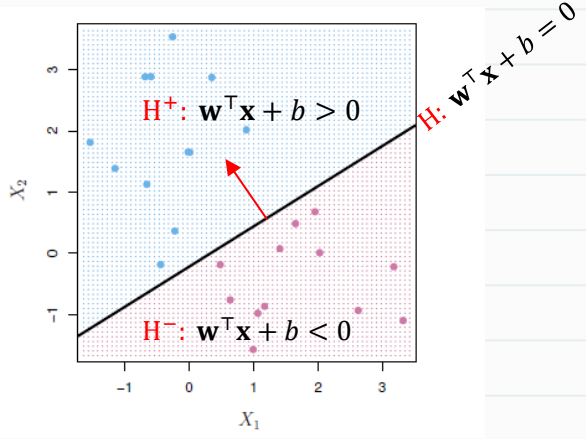
Linear Separation via Hyperplane

A hyperplane

$$\mathrm{H}: \mathbf{w}^\top \mathbf{x} + b = 0$$

splits $\mathbb{R}^d$ into two half-spaces (positive and negative):

$$\mathrm{H}^+: \mathbf{w}^\top \mathbf{x} + b > 0$$
$$\mathrm{H}^-: \mathbf{w}^\top \mathbf{x} + b < 0$$



Given data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $y_i \in \{+1, -1\}$, we need a linear classifier that places

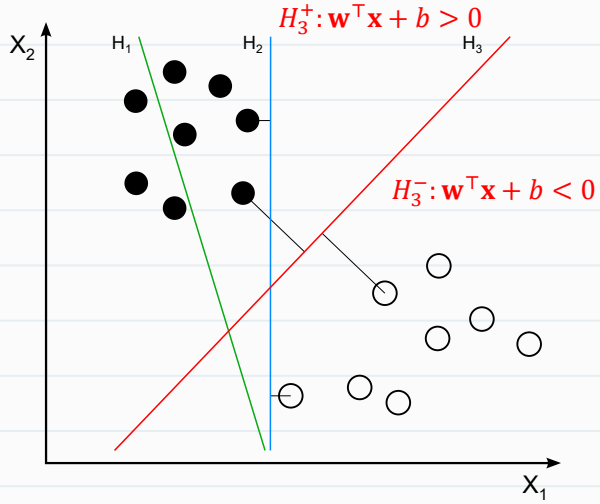
positive data points in positive half-space
negative data points in the negative half-space.

$$\mathbf{w}^\top \mathbf{x}_i + b > 0 \quad \text{if} \quad y_i = +1$$
$$\mathbf{w}^\top \mathbf{x}_i + b < 0 \quad \text{if} \quad y_i = -1$$

Equivalent **compact form**: $\boxed{y_i\,(\mathbf{w}^\top \mathbf{x}_i + b) > 0}$

Fig Source: ISLP Book

There are many separating hyperplanes: Which one to choose?

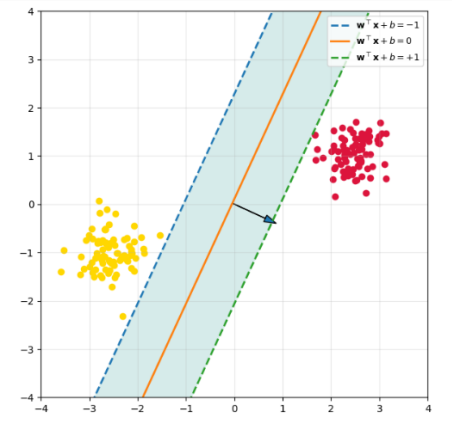


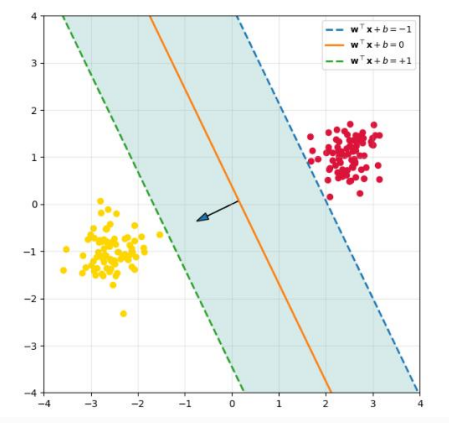
H_1 does not separate
H_2 separates
H_3 separates
Which is better among $\mathbf{H_2}$ and $\mathbf{H_3}$?

When multiple separating hyperplanes exist,
we need a principled way to choose the best one.

Fig Source: https://en.wikipedia.org/wiki/Support_vector_machine

Maximum-Margin Hyperplane





Margin = distance between these two parallel hyperplanes.

Parameters: $\mathbf{w}$, b **Demo**

Direction: $\frac{\mathbf{w}}{||w||}$

Magnitude: $||\mathbf{w}||$

Bias: b

In addition to the separating hyperplane $\mathbf{w}^\top\mathbf{x}+b=0$, introduce two more **parallel** hyperplanes:

$$\mathbf{w}^\top\mathbf{x}+b=+1, \qquad \mathbf{w}^\top\mathbf{x}+b=-1.$$

Enforce that positive class points lie on the positive-half of the $\mathbf{w}^\top\mathbf{x}+b=+1$:

$$(\mathbf{w}^\top\mathbf{x}_i+b)\geq 1 \quad \text{if } \mathrm{y_i}=+1$$

Enforce that negative class points lie on the negative-half of the $\mathbf{w}^\top\mathbf{x}+b=-1$:

$$(\mathbf{w}^\top\mathbf{x}_i+b)\leq -1 \quad \text{if } \mathrm{y_i}=-1$$

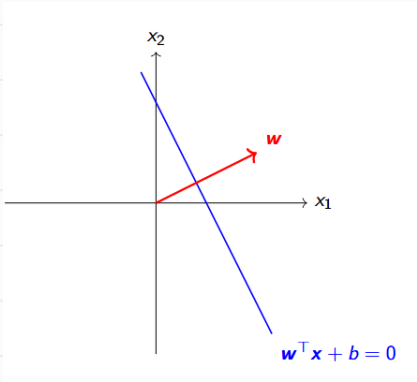
Equivalent **Compact form:** $\boxed{y_i(\mathbf{w}^\top\mathbf{x}_i+b)\geq 1}$

Normal Vector of a Hyperplane

Hyperplane:

$$\mathbf{w}^\top \mathbf{x} + b = 0$$

Normal vector: w



Proof:

Take any two points $\mathbf{x}_1, \mathbf{x}_2$ on the hyperplane:

$$\mathbf{w}^\top \mathbf{x}_1 + b = \mathbf{w}^\top \mathbf{x}_2 + b = 0$$
$$\Rightarrow \quad \mathbf{w}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0$$

Hence, every direction joining any two points $\mathbf{x}_1, \mathbf{x}_2$ is orthogonal to $\mathbf{w}$.

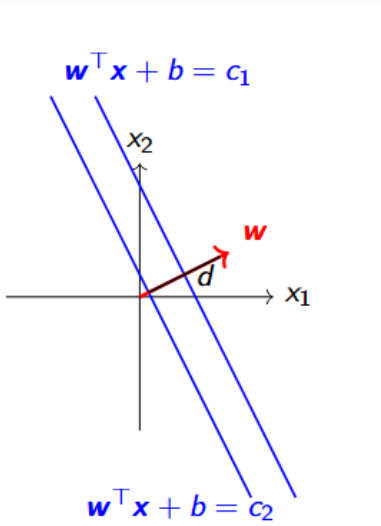
Therefore, $\mathbf{w}$ is the normal.

Distance Between Two Parallel Hyperplanes

Consider two parallel hyperplanes (same normal $\mathbf{w} \neq \mathbf{0}$):

$$H_1\colon \mathbf{w}^\top \mathbf{x} + b = c_1, \qquad H_2\colon \mathbf{w}^\top \mathbf{x} + b = c_2.$$

$$\boxed{\text{Distance}(H_1, H_2) = \frac{|c_2 - c_1|}{\| \mathbf{w} \|}}$$



Proof:

Let $\mathbf{u} = \mathbf{w}/\| \mathbf{w} \|$ be the unit normal.

Pick any $\mathbf{x}_1 \in H_1$ .Move along the normal: $\mathbf{x}_2 = \mathbf{x}_1 + \alpha \mathbf{u}$.

$$\mathbf{w}^\top \mathbf{x}_2 + b = \mathbf{w}^\top (\mathbf{x}_1 + \alpha \mathbf{u}) + b = (\mathbf{w}^\top \mathbf{x}_1 + b) + \alpha\, \mathbf{w}^\top \mathbf{u}$$
$$= c_1 + \alpha \ \| \mathbf{w} \|.$$

To land on H_2 ,set $c_1 + \alpha \| \mathbf{w} \| = c_2$,hence

$$\alpha = \frac{c_2 - c_1}{\| \mathbf{w} \|}.$$

$$\boxed{\text{Distance}(H_1, H_2) = |\alpha|}$$

Maximum-Margin Hyperplane

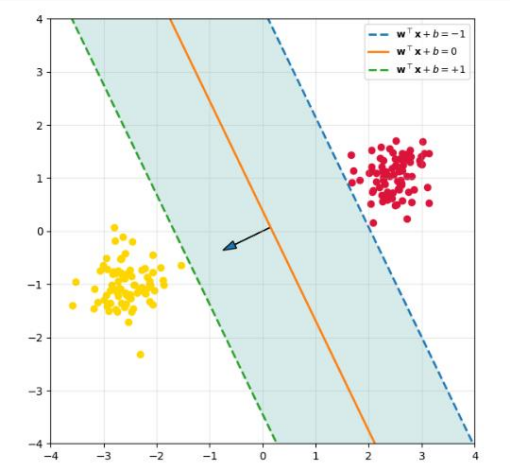
Separating hyperplane:

$$\mathbf{w}^\top \mathbf{x} + b = 0.$$

The two **parallel** hyperplanes:

$$\mathbf{w}^\top \mathbf{x} + b = +1, \qquad \mathbf{w}^\top \mathbf{x} + b = -1.$$

$$\boxed{\text{Distance}(H_1, H_2) = \frac{|c_2 - c_1|}{\| \mathbf{w} \|}}$$



Margin = distance between these two parallel hyperplanes:

$$\text{Margin} = \frac{|(+1) - (-1)|}{\| \mathbf{w} \|} = \frac{2}{\| \mathbf{w} \|}.$$

Hard-Margin SVM: Optimization Problem

Maximizing the margin

$$\max \frac{2}{\| \mathbf{w} \|} \quad \Leftrightarrow \quad \min \| \mathbf{w} \|.$$

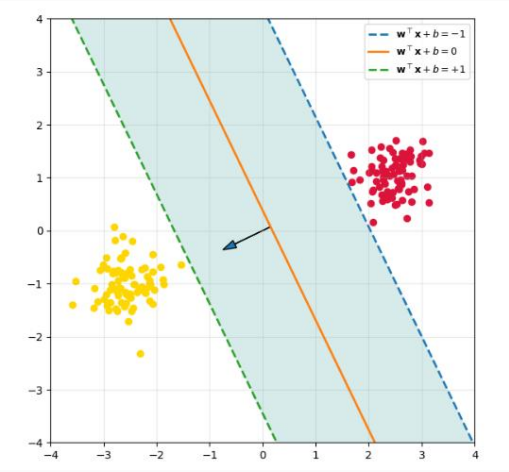
Convenient convex objective:

$$\min_{\mathbf{w},b} \frac{1}{2} \| \mathbf{w} \|^2.$$

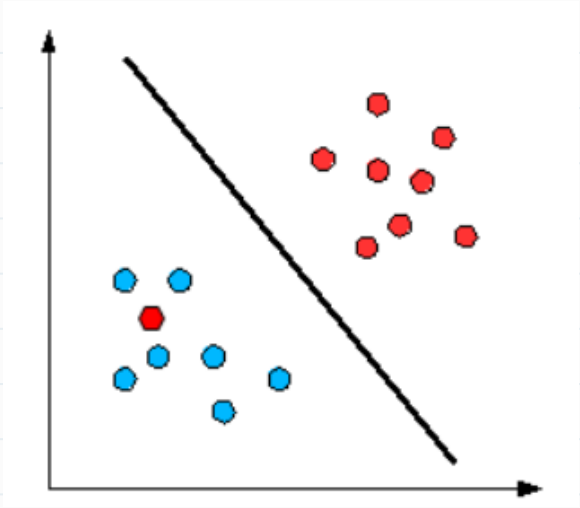
Hard-margin SVM (primal):

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2} \| \mathbf{w} \|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, n. \end{aligned}$$

Decision rule: $\hat{y}(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$.



When data is not perfectly linearly separable



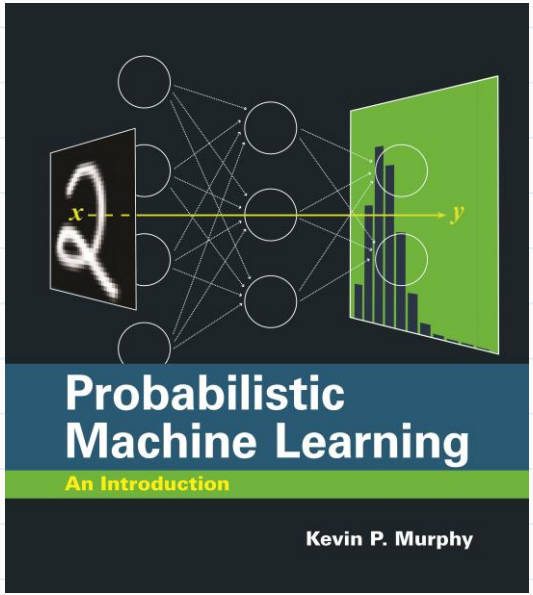
We have to use another variant called **Soft-Margin SVM**

Fig Source: https://online.stat.psu.edu/stat857/node/242/

Reference

Multi-class Logistic Regression: Chapter 10.3
SVM: Chapter 17.3

“Probabilistic Machine Learning: An Introduction” book by Kevin P. Murphy



Thank You