Introduction to Machine Learning

Lecture 11:

Logistic Regression (Classification Algorithm)

Prof. Subrahmanya Swamy Peruru
Department of Electrical Engineering
IIT Kanpur

Probabilistic View of Classification

Assume there is an underlying data distribution $(X, Y) \sim \mathcal{D}$.

Bayes Optimal Classifier (0–1 Loss):

$$f^* = \operatorname{argmin}_f \mathbb{E}_{(X,Y)\sim\mathcal{D}}[\mathbf{1}\{Y \neq f(X)\}]$$

Upon solving, the optimal classifier can be obtained as:

$$f^\star(\mathbf{x}) = \arg\max_{c\in\{1,\ldots,C\}} \mathbb{P}(Y = c \mid X = \mathbf{x}).$$

Example:
Suppose we have 3 classes and **if the data distribution is given to us:**

For a given input $\mathbf{x}$, we compare:
$P(Y = 1 \mid X = \mathbf{x}) = 0.7, \quad P(Y = 2 \mid X = \mathbf{x}) = 0.2, \quad P(Y = 3 \mid X = \mathbf{x}) = 0.1$

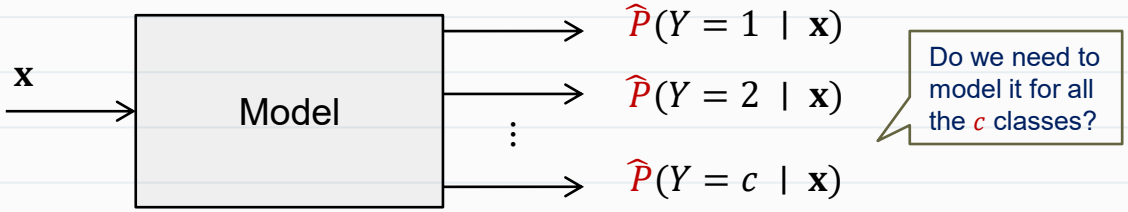
Since $P(Y = 1 \mid X = \mathbf{x}) = 0.7$, is the largest, the Bayes optimal prediction is

$$f^*(\mathbf{x}) = 1$$

To find $f^\star$, we need to know the true data distribution $\mathcal{D}$, which is not known to us.

Learning $P(Y \mid X)$ from Data

We learn an approximation model $\widehat{P}(Y \mid X = \mathbf{x})$ from data



Now consider binary classification: $Y \in \{0,1\}$

It is sufficient to model $\widehat{P}(1 \mid \mathbf{x})$.
Since $\widehat{P}(0 \mid \mathbf{x})$ can be automatically obtained using $\widehat{P}(0 \mid \mathbf{x}) = 1 - \widehat{P}(1 \mid \mathbf{x})$

Key question:

- Can we use a linear model for $\widehat{P}(1 \mid \mathbf{x})$?
- Can $\mathbf{w}^T\mathbf{x} + b$ represent a probability?

Why Linear Output is Not Enough

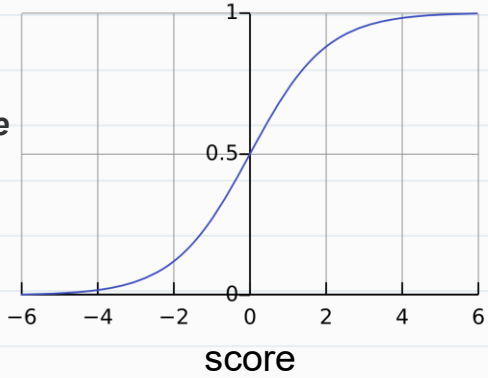
The linear score $\mathbf{w}^T\mathbf{x} + b$ cannot directly represent probabilities:

- Can be negative
- Can exceed 1

Idea: Instead of interpreting $\mathbf{w}^T\mathbf{x} + b$ as a probability, treat it as a ***confidence score***

- Large positive score ⇒ class 1 likely
- Large negative score ⇒ class 0 likely
- Zero score ⇒ not sure of the class

Sigmoid function



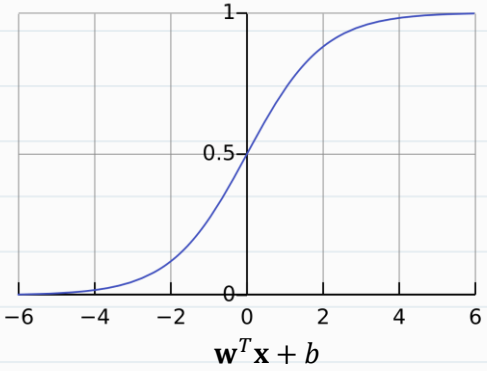
We need a function (**sigmoid**) that **converts scores into probabilities**.

$$\sigma(z) = \frac{e^z}{1 + e^z}$$

- Maps $\mathbb{R} \to (0,1)$
- $\sigma(0) = \frac{1}{2}$
- $z \to +\infty \Rightarrow \sigma(z) \to 1$
- $z \to -\infty \Rightarrow \sigma(z) \to 0$

Logistic Regression Model

$$P(Y = 1 \mid \mathbf{x}; \mathbf{w}, \mathrm{b}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$



Score:

$$\text{score} = \mathbf{w}^T\mathbf{x} + b$$

Probability model:

$$P(Y = 1 \mid \mathbf{x}; \mathbf{w}, \mathrm{b}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

Standard Prediction rule:

$$\hat{y} = \begin{cases} 1 & \text{if } P(Y = 1 \mid \mathbf{x}, \mathbf{w}, \mathrm{b}) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

Equivalent form: Predict as class 1 if

$$\mathbf{w}^T\mathbf{x} + b > 0$$

Model structure fixed and Decision rule defined.

Remaining task: Estimate $(\mathbf{w}, \mathrm{b})$ from data

Maximum Likelihood: Warm-up

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

Bernoulli random variable:

$$P(Y = 1) = \theta, \quad P(Y = 0) = 1 - \theta$$

Compact form:

$$P(Y = y) = \theta^{y}(1 - \theta)^{1-y}$$

We have data: The outcome of n independent coin tosses

$$\{y_1, y_2, \dots, y_n\}$$

Goal is to estimate the parameter: Bias of the coin θ

Likelihood function of θ: If θ is the parameter of the coin, how likely is it to observe that data?

$$L(\theta) = \prod_{i=1}^{n} P\,(Y = y_i \mid \theta)$$

MLE estimate:

$$\theta_{\mathrm{ML}} = \operatorname*{argmax}_{\theta} L(\theta)$$

Negative Log-Likelihood

Likelihood

$$L(\theta) = \prod_{i=1}^{n} P\left(y_i \mid \theta\right)$$

Log-Likelihood

- Products are hard to optimize; sums are easier
- Log converts products into sums $\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log P(y_i \mid \theta)$
- Logarithm is a strictly increasing (monotonic) function $\Rightarrow \underset{\theta}{\operatorname{argmax}}\, L(\theta) = \underset{\theta}{\operatorname{argmax}} \log L(\theta)$

From maximization to minimization

Optimization algorithms are usually written as minimization problems, so we define:

$$\mathrm{NLL}(\theta) = -\log L(\theta)$$

Minimizing Negative log-likelihood

$$\underset{\theta}{\operatorname{argmin}}\, \mathrm{NLL}(\theta)$$

Example

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

Likelihood:

$$L(\theta) = \prod_{i=1}^{n} P\,(y_i \mid \theta\,)$$
$$= P\,(0 \mid \theta\,) \times P(1 \mid \theta\,) \times P(1 \mid \theta\,) \times P(0 \mid \theta\,)$$
$$= (1-\theta) \times \theta \times\ \theta \times (1-\theta)$$
$$= \theta^2(1-\theta)^2$$

Negative Log-likelihood (Convex function):

$$\mathrm{NNL}(\theta) = -\log L(\theta)$$
$$= -2\log\theta - 2\log(1-\theta)$$

Setting the gradient to zero gives:

$$\frac{2}{\theta} - \frac{2}{1-\theta} = 0$$
$$\Rightarrow \theta_{\mathrm{ML}} = \frac{1}{2}$$

Likelihood for logistic regression

Probability modelled by sigmoid function:

$$P(\mathrm{y}=1 \mid \mathbf{x};\mathbf{w},\mathrm{b}) = \sigma(\mathbf{w}^T\mathbf{x}+b)$$
$$P(\mathrm{y}=0 \mid \mathbf{x};\mathbf{w},\mathrm{b}) = 1-\sigma(\mathbf{w}^T\mathbf{x}+b)$$

Assuming i.i.d. given data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)$

The likelihood function of the parameters unknown parameters $(\mathbf{w}, b)$:

$$L(\mathbf{w},b) = \prod_{i=1}^{n} P\left(y_i \mid \mathbf{x}_i;\mathbf{w},b\right)$$

Let $\hat{p}_i = P(\mathrm{y_i}=1 \mid \mathbf{x}_i;\mathbf{w},\mathrm{b})$ be the probability that i^{th} data point is predicted as class 1

Negative log-likelihood (NLL):

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

Minimizing the NLL loss (solved using Gradient Descent):

$$\min_{\mathbf{w},b} \mathrm{NLL}(\mathbf{w},b)$$

Log-likelihood and Cross-entropy

Minimizing Negative log-likelihood is equivalent to minimizing cross-entropy loss.

Negative log-likelihood (NLL):

$$y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i)$$

Cross Entropy:

$$\boxed{H(p, q) = -\sum_x p\,(x) \log q(x)}$$

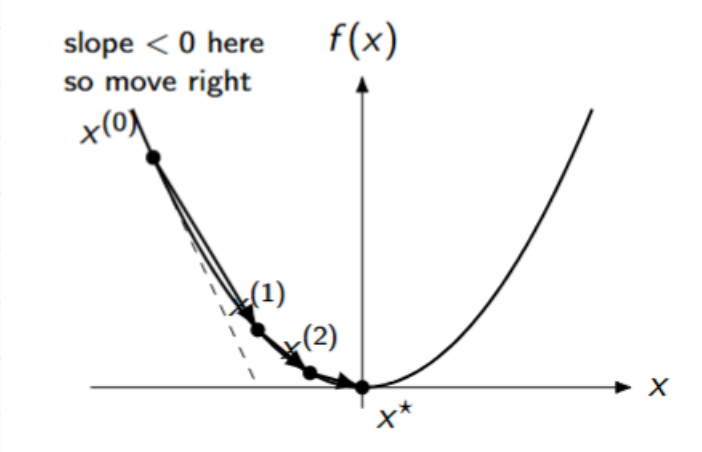
Gradient Descent

Goal: minimize a differentiable function

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}).$$

Key intuition (1-D): $f(x): \mathbb{R} \to \mathbb{R}$

- The derivative $f'(x)$ is the *slope* of f at x.
- If $f'(x) < 0$ (downward slope), moving *right* decreases f.
- If $f'(x) > 0$ (upward slope), moving *left* decreases f.



Example: $f(x) = x^2 \Rightarrow \quad f'(x) = 2x$

- If $x < 0$, then $f'(x) = 2x < 0 \Rightarrow$ step to the right.
- If $x > 0$, then $f'(x) > 0 \Rightarrow$ step to the left.
- Minimum at $x^\star = 0$.

Update rule (1-D): $x^{(t+1)} = x^{(t)} - \eta\, f'(x^{(t)}), \qquad \eta_t > 0$ (step size / learning rate).

Gradient descent (vector form): $f(\mathbf{x}): \mathbb{R}^d \to \mathbb{R}$

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \nabla f\left(\mathbf{x}^{(t)}\right).$$

Gradient Descent (Step size and Initialization)

Two knobs you choose:

- **Initial point** $\mathbf{x}^{(0)}$ (where you start) influences convergence point (multiple local minima in non-convex functions)
- **Step size** η_t:
  - Too large ⇒ may overshoot / diverge / oscillate.
  - Too small ⇒ very slow progress.
  - Often: small constant η, or decreasing η_t (e.g., $\eta_t \propto 1/t$)

Descent idea (local): for small enough η_t,

$$f\left(\mathbf{x}^{(t+1)}\right) \leq f\left(\mathbf{x}^{(t)}\right)$$ (typically holds for smooth functions with a suitable step size).

Stochastic Gradient Descent (SGD) (more details in future lectures):

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \eta_t \widehat{\nabla f}\left(\mathbf{x}^{(t)}\right),$$

where $\widehat{\nabla f}$ is a noisy/mini-batch estimate of the true gradient (cheaper per iteration; adds jitter that can help exploration).

Thank You