

# CS789: Machine Learning and Neural Network

## Logistic Regression

Jakramate Bootkrajang

Department of Computer Science  
Chiang Mai University

- We have seen how we can construct a generative classifier.
- The generative classifier puts some assumption on the data (in our case Gaussian assumption)
- This lecture we will learn the counterpart of generative classifier called *discriminative* classifier
- Discriminative classifier aims to classify data directly without modelling data distribution.

# Logistic Regression (1/3)

- Recall from previous lecture: the point where decision changes from class 1 to class 0 is

$$p(y = 1|\mathbf{x}) = p(y = 0|\mathbf{x})$$

- Dividing both side by  $p(y = 0|\mathbf{x})$  and taking log, we get

$$\log \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} = 0$$

- Our decision function is then

$$f(\mathbf{x}) = \log \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})}$$

- We want to impose linear decision boundary on  $f(\mathbf{x})$  so we model it with a linear function

$$f(\mathbf{x}) = \log \frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} = \mathbf{w}^T \mathbf{x}$$

# Logistic Regression (3/3)

- From the above definition, it is then possible to find the probability supporting the prediction.
- This is done by inverting  $\log \frac{p(y=1|\mathbf{x})}{p(y=0|\mathbf{x})} = \mathbf{w}^T \mathbf{x}$  to get  $p(y = 1|\mathbf{x})$
- Which turns out to be
  - ▶  $p(y = 1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x})}$ .
  - ▶  $p(y = 0|\mathbf{x}) = 1 - p(y = 1|\mathbf{x}) = 1 - \frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x})}$
- The function  $\frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x})}$  is called the logistic function/sigmoid function.

# Comparison between NDA and LR

## NDA

- Generative
- Models the occurrence of  $x$  by some probability distribution (Gaussian is the mostly used)
- Posterior is obtained through Bayes theorem.

$$p(y = 1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x})}$$

## Logistic Regression

- Discriminative
- Does **not** try to model the occurrence of  $\mathbf{x}$ .
- However, it jumps to modelling the posterior probability directly.

$$p(y = 1|\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x})}$$

# Parameter estimation

Assuming the training data  $S$  is i.i.d (independently and identically drawn), the likelihood function of LR is given by:

$$\begin{aligned}\mathcal{L}(\theta|S) &= \mathcal{L}(\theta|(X, Y)) \\ &= p((X, Y)|\theta) \\ &= p(Y|X, \theta)p(X|\theta) \\ &= \prod_{i=1}^N p(y_i = 1|\mathbf{x}_i, \mathbf{w})^{y_i} (1 - p(y_i = 1|\mathbf{x}_i, \mathbf{w}))^{1-y_i}\end{aligned}$$

It's easier to work with a log-likelihood

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^N y_i \log p(y_i = 1|\mathbf{x}_i, \mathbf{w}) + (1 - y_i) \log p(y_i = 0|\mathbf{x}_i, \mathbf{w})$$

# Parameter estimation:

- Using the definition of  $p(y = 1|\mathbf{x})$  and  $p(y = 0|\mathbf{x})$  we can simplify things a bit.

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^M y_i \mathbf{w}^T \mathbf{x}_i - \log(1 + e^{\mathbf{w}^T \mathbf{x}_i}) \quad (1)$$

- We would like to maximise the likelihood Eq.(1).



# First order partial derivatives

- The point at which Eq.(1) is maximum is a saddle point (i.e., first derivative is zero)
- We find the first order (partial) derivatives of Eq.(1) w.r.t  $w_j$ .

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_j} = \sum_{i=1}^N y_i x_{ij} - \sum_{i=1}^N \frac{x_{ij} e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}} \quad (2)$$

- Setting Eq.(2) to zero, we find that there is no closed-form solution (we cannot isolate  $w$ )

$$\sum_{i=1}^N y_i x_{ij} - \sum_{i=1}^N \frac{x_{ij} e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}} = 0$$

# First approach to optimisation: Newton's method

- Also known as Newton-Raphson method.
- The method for finding successive better approximation to the root of a real-valued function,  $x : f(x) = 0$ .
  - ▶ The update routine is given by  $x^{new} = x^{old} - \frac{f(x)}{f'(x)}$
- Back to our problem we want to find  $\mathcal{L}'(\mathbf{w}) = 0$ , the Newton's method for our purpose is then

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \eta \frac{\mathcal{L}'(\mathbf{w})}{\mathcal{L}''(\mathbf{w})}$$

- But we need to find the second order partial derivative  $\mathcal{L}''(\mathbf{w})$

# A side note on calculus

- A **partial derivative** of differentiable function  $f(x_1, x_2, \dots, x_n)$  of several variables is its derivative w.r.t one of those variable with the others held constant.
- A **gradient** of the function  $f(x_1, x_2, \dots, x_n)$  is a vector of partial derivatives
  - ▶  $\nabla f(x_1, x_2, \dots, x_n) = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$
- A **Hessian matrix** is a square matrix of second-order partial derivative

$$\text{▶ } H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- So we massage the first partial derivative a bit

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_j} &= \sum_{i=1}^N y_i x_{ij} - \sum_{i=1}^N \frac{x_{ij} e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}} \\ &= \sum_{i=1}^N y_i x_{ij} - \sum_{i=1}^N p(y = 1 | \mathbf{x}_i) x_{ij} \\ &= \sum_{i=1}^N \left[ y_i - p(y = 1 | \mathbf{x}_i) \right] x_{ij}\end{aligned}$$

# Newton's method: The Hessian

$$\begin{aligned}\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_j \partial w_k} &= \frac{\partial \sum_{i=1}^N y_i x_{ij} - \sum_{i=1}^N \frac{x_{ij} e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}}}{\partial w_k} \\ &= - \sum_{i=1}^N \frac{(1 + e^{\mathbf{w}^T \mathbf{x}_i}) e^{\mathbf{w}^T \mathbf{x}_i} x_{ij} x_{ik} - (e^{\mathbf{w}^T \mathbf{x}_i})^2 x_{ij} x_{ik}}{(1 + e^{\mathbf{w}^T \mathbf{x}_i})^2} \\ &= - \sum_{i=1}^N x_{ij} x_{ik} p(y = 1 | \mathbf{x}_i) - x_{ij} x_{ik} p(y = 1 | \mathbf{x}_i)^2 \\ &= - \sum_{i=1}^N x_{ij} x_{ik} p(y = 1 | \mathbf{x}_i) (1 - p(y = 1 | \mathbf{x}_i))\end{aligned}$$

# Newton's method: In matrix form

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}^T (\mathbf{y} - \mathbf{p}_1)$$
$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = -\mathbf{X}^T \mathbf{Q} \mathbf{X}$$

- $\mathbf{X}$  is an  $N \times (m + 1)$  (1's augmented) input matrix
- $\mathbf{y}$  is a vector of labels
- $\mathbf{p}_1$  is a vector of  $p(y = 1 | \mathbf{x}_i, \mathbf{w}^{old})$
- $\mathbf{Q}$  is an  $N \times N$  diagonal matrix with  $\mathbf{Q}[i, i]$  being  $p(y = 1 | \mathbf{x}_i, \mathbf{w}^{old})(1 - p(y = 1 | \mathbf{x}_i, \mathbf{w}^{old}))$

# Newton's method for optimising LR: Summary

## Pseudo Code

- 1  $\mathbf{w} \leftarrow \mathbf{0}$
- 2 Make sure class label vector  $\mathbf{y}$  is in  $\{0, 1\}$  format
- 3 Compute  $\mathbf{p}_1$  by setting its elements to

$$p(y = 1|\mathbf{x}_i; \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{x}_i}}{1 + e^{\mathbf{w}^T \mathbf{x}_i}}$$

- 4 Compute diagonal matrix  $\mathbf{Q}$  by setting its diagonal elements to  $p(y = 1|\mathbf{x}_i; \mathbf{w})(1 - p(y = 1|\mathbf{x}_i; \mathbf{w}))$
- 5  $\mathbf{w}^{new} = \mathbf{w}^{old} + \eta(\mathbf{X}^T \mathbf{Q} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p}_1)$
- 6 Until stopping criteria is met (usually  $|\mathbf{w}^{new} - \mathbf{w}^{old}| < \epsilon$ )

# Problem with Newton's method

- Finding the Hessian is a tedious work.
- Further, finding the inverse of the Hessian is usually time consuming.
- Some modifications exist, e.g., [Quasi-Newton](#), for speeding up the calculation of the inverse by some approximation technique.
- Some methods even require only the first derivative, e.g, conjugate gradient method. Cool!



# Multiclass logistic regression

- Support multiclass classification
- Also known as **Multinomial logistic regression**
- The posterior probability is modelled by the softmax function

$$p(y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}_j^T \mathbf{x})}$$

- Here,  $\mathbf{w}_k$  is the weight vector corresponding to class  $k$ .

# Multiclass logistic regression

- The maximum likelihood estimate of  $\mathbf{w}_k$  is obtained by maximising the data log-likelihood.

$$\mathcal{L}(\mathbf{w}_1, \dots, \mathbf{w}_k) = \sum_{i=1}^N \sum_{k=1}^K \delta(y_i = k) \log \frac{\exp(\mathbf{w}_k^T \mathbf{x}_i)}{\sum_{j=1}^K \exp(\mathbf{w}_j^T \mathbf{x}_i)}$$

# Summary

- We learn another way to construct a classifier.
- The classifier is called *discriminative* classifier.
- Since it focuses on separating the data not modelling data generation.
- One widely used classifier of this type is the Logistic Regression.
- Optimising the parameter of the logistic regression can be done using numerical method, such as the Newton's method.