MAT8034: Machine Learning

Classification and Logistic Regression

Fang Kong

https://fangkongx.github.io/Teaching/MAT8034/Spring2025/index.html

Part of slide credit: Stanford CS229

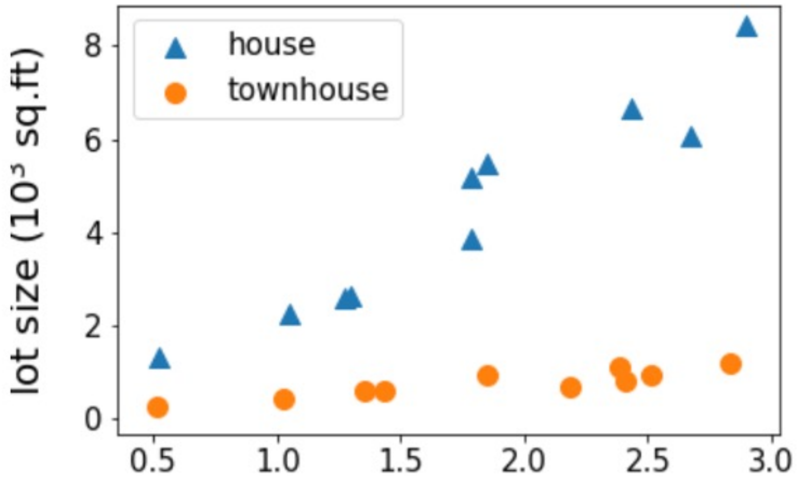
Outline

- Logistic regression
- Digression: the perceptron learning algorithm
- Newton's method
- Multi-class classification

Logistic regression

Intuition of logistic regression

- Hope to use the linear method to solve the classification problem
- Given a a training set:$\{(x^{(i)}, y^{(i)}), i = 1,2, \dots, n\}$, let $y^{(i)} \in \{0,1\}$



- Build the connection between p and $\theta^\top x = \theta_0 + \theta_1 x_1 + \theta_2 x_2$
- $p \in (0,1)$ but $\theta^\top x \in (-\infty, +\infty)$

Intuition of logistic regression

- Consider the odd: p/(1-p) $\in (0, +\infty)$
- Consider the log odd:
  - Logit(p) = log p/(1-p) $\in (-\infty, +\infty)$

- Good properties:
  - p->0, logit -> $-\infty$; p->1, logit -> $+\infty$
  - Symmetry: Logit(p)=-Logit(1-p)
  - Use linear model to approximate the logit: $\theta^\top x$= Logit(p)= log p/(1-p)
  - $p = \frac{1}{1+\exp(-\theta^\top x)} := \mathrm{sigmoid}(\theta^\top x)$

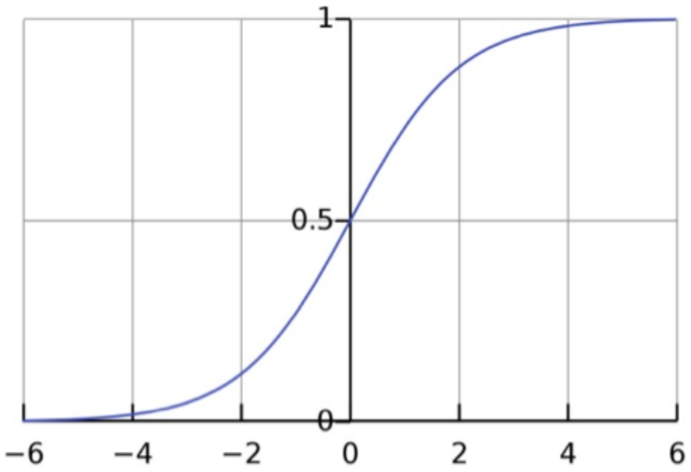
Logistic Regression

- Given a training set $\{(x^{(i)}, y^{(i)})$ for $i = 1, \dots, n\}$ let $y^{(i)} \in \{0, 1\}$. Want $h_\theta(x) \in [0, 1]$. Let's pick a smooth function:

$$h_\theta(x) = g(\theta^T x)$$

- Here, g is a link function. There are *many*... but we'll pick one!

$$g(z) = \frac{1}{1 + e^{-z}}.$$



How do we interpret $h_\theta(x)$?

$$P(y = 1 \mid x; \theta) = h_\theta(x)$$
$$P(y = 0 \mid x; \theta) = 1 - h_\theta(x)$$

Likelihood function

- Let's write the Likelihood function. Recall:

$$
\begin{aligned}
P(y = 1 \mid x; \theta) =& h_\theta(x) \\
P(y = 0 \mid x; \theta) =& 1 - h_\theta(x)
\end{aligned}
$$

- Then,

$$
\begin{aligned}
L(\theta) =& P(y \mid X; \theta) = \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}; \theta) \\
=& \prod_{i=1}^{n} h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}} \quad \text{exponents encode “if-then”}
\end{aligned}
$$

- Taking logs to compute the log likelihood $\ell(\theta)$ we have:

$$
\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))
$$

Gradient ascent for log likelihood

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} \ell(\theta) &= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y \frac{1}{g(\theta^T x)} - (1-y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} \theta^T x \\
&= \left(y(1 - g(\theta^T x)) - (1-y) g(\theta^T x) \right) x_j \\
&= (y - h_\theta(x))\, x_j
\end{aligned}
$$

$$
\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}
$$

Another view: logistic loss

- In linear regression
  - The loss function is $J(h_\theta(x^{(i)}), y) = \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
- For the classification
  - Define the loss function

$$\ell_{\text{logistic}}(t, y) \triangleq y \log(1 + \exp(-t)) + (1 - y) \log(1 + \exp(t))\,. \qquad (2.3)$$

  - When $y = 1$, minimizing the loss gets $t \rightarrow +\infty, p \rightarrow 1$
  - When $y = 0$, minimizing the loss gets $t \rightarrow -\infty, p \rightarrow 0$

Another view: logistic loss

- For the classification
  - Define the loss function

$$\ell_{\text{logistic}}(t, y) \triangleq y \log(1 + \exp(-t)) + (1 - y) \log(1 + \exp(t)) \,. \qquad (2.3)$$

  - The relationship between the loss and log likelihood $-\ell(\theta) = \ell_{\text{logistic}}(\theta^\top x, y)$

$$\frac{\partial \ell_{\text{logistic}}(t, y)}{\partial t} = y \frac{-\exp(-t)}{1 + \exp(-t)} + (1 - y) \frac{1}{1 + \exp(-t)} \qquad (2.5)$$

$$= 1/(1 + \exp(-t)) - y. \qquad (2.6)$$

Then, using the chain rule, we have that

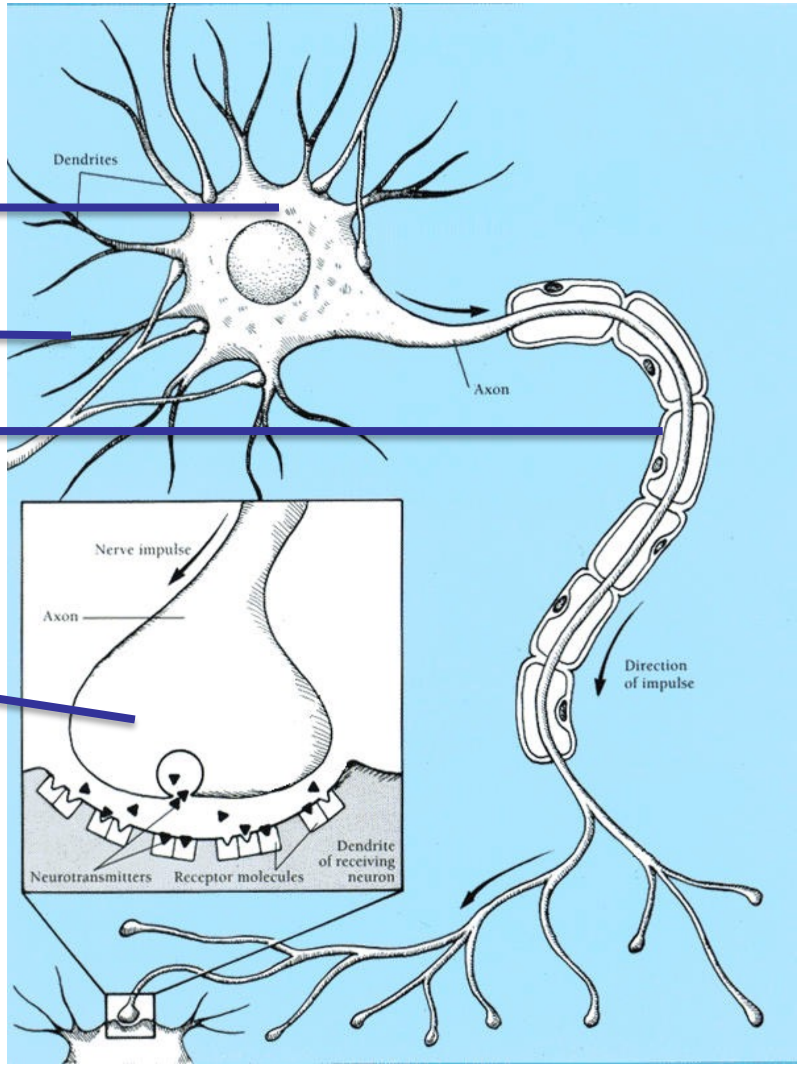
$$\frac{\partial}{\partial \theta_j} \ell(\theta) = -\frac{\partial \ell_{\text{logistic}}(t, y)}{\partial t} \cdot \frac{\partial t}{\partial \theta_j} \qquad (2.7)$$

$$= (y - 1/(1 + \exp(-t))) \cdot x_j = (y - h_\theta(x)) x_j \,, \qquad (2.8)$$

Connection with the perceptron

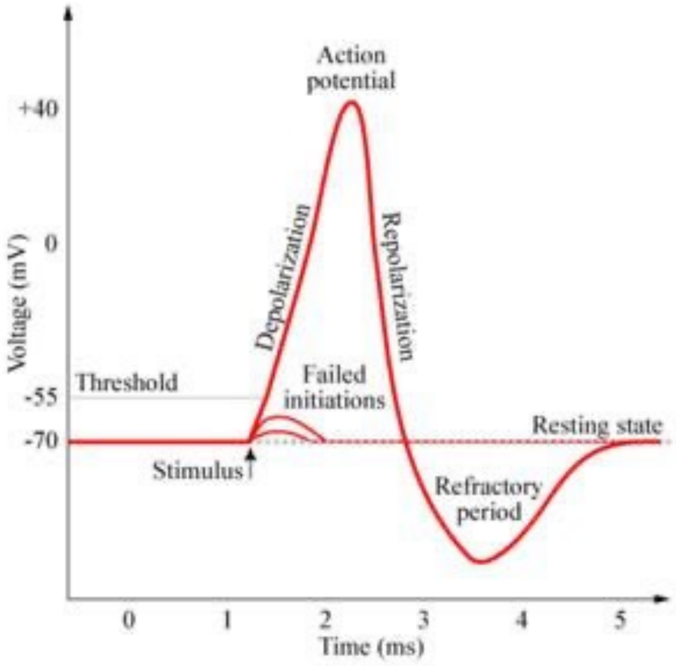
Biological neuron structure

- 细胞结构
  - 细胞体
  - 树突
  - 轴突
  - 突触末梢



Biological neural communication

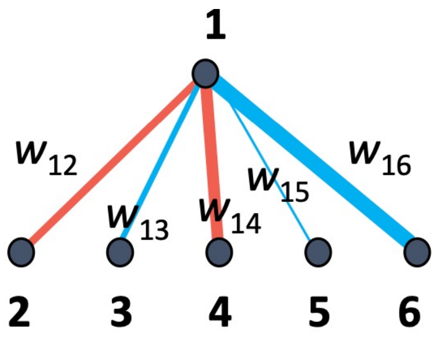
- 细胞膜间的电位表现出的电信号称为动作电位
- 电信号从细胞体中产生，沿着轴突往下传，并且导致突触末梢释放神经递质介质
- 介质通过化学扩散从突触传递到其他神经元的树突
- 神经递质可以是兴奋的或者是抑制的
- 如果从其他神经元来的神经递质是兴奋的且超过某个阈值，将会触发一个动作电位



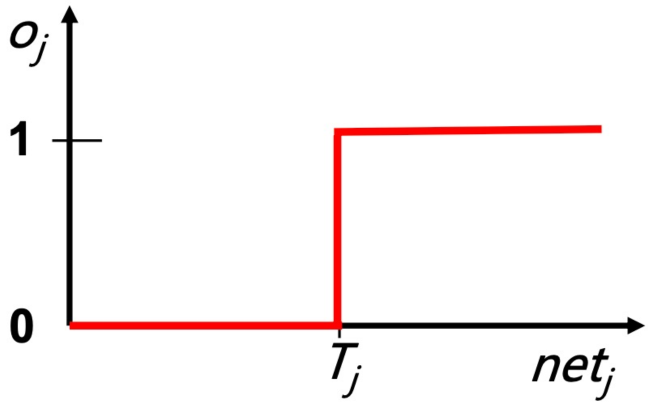
McCulloch-Pitts neuron model [1943]

- Model the network as a graph, where the units are nodes, and the synaptic connections are weighted edges from node i to node j, with the weight as $w_{j,i}$
- The input of the unit is:

$$\text{net}_j = \sum_i w_{j,i} \cdot o_i$$

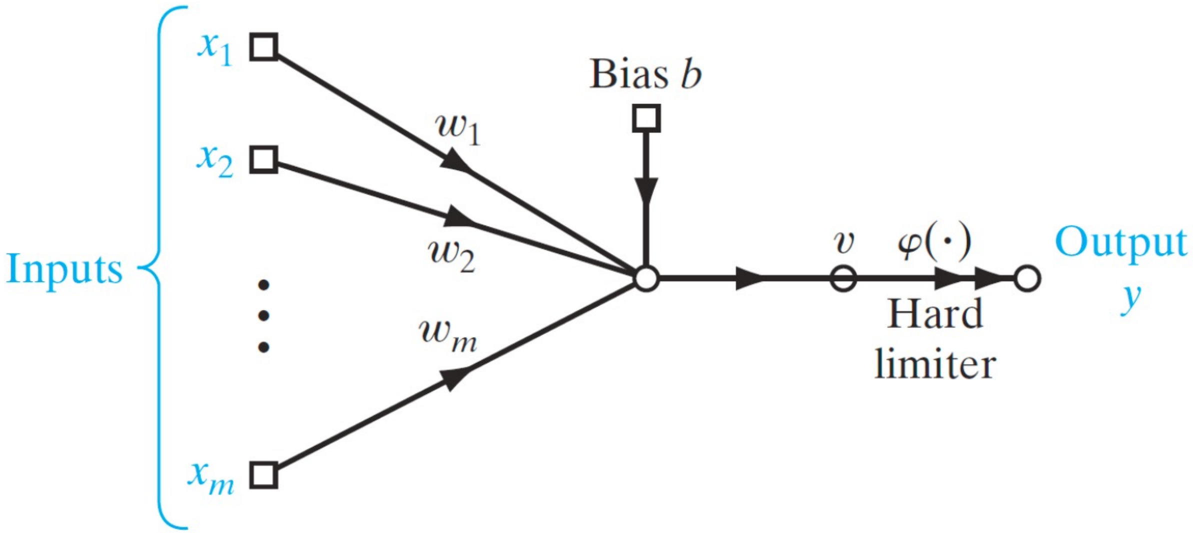


- The output of the unit is:
  - 0 if $\text{net}_j < T_j$; 1 otherwise
  - T_j is the threshold



Slide credit: Ray Mooney

Single-layer perception by Rosenblatt [1958]



- Rosenblatt [1958] 进一步提出感知机作为第一个在“老师”指导下进行学习的模型（即监督学习）
- 专注在如何找到合适的用于二分类任务的权重w_m
  - $y = 1$：类别1
  - $y = -1$：类别2

预测

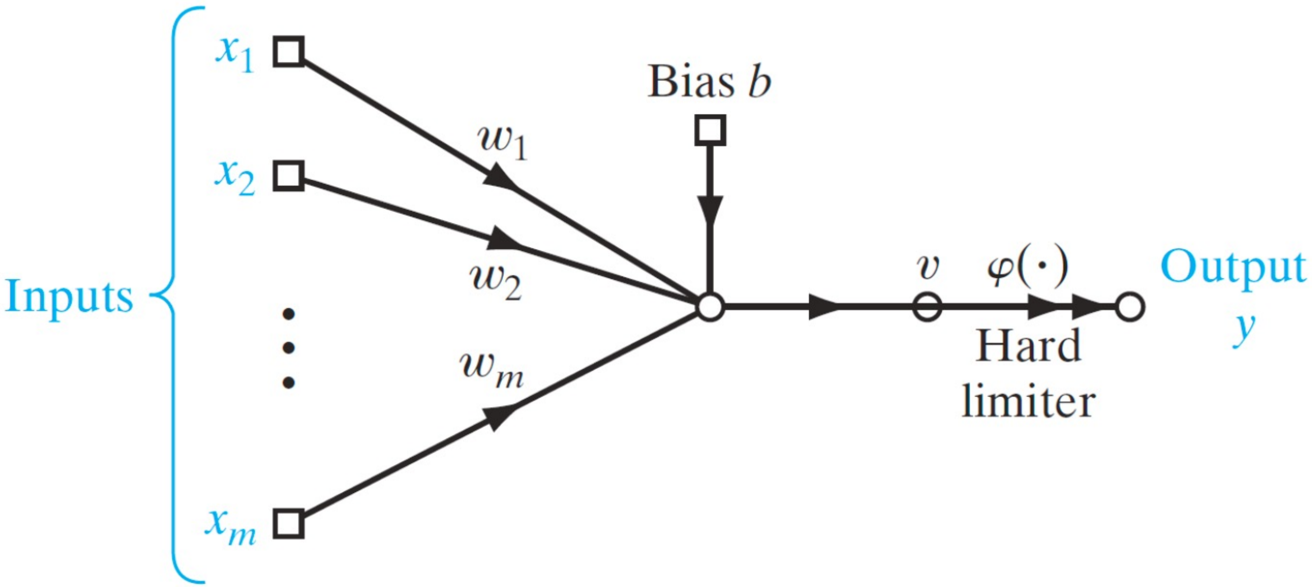
$$\hat{y} = \varphi(\sum_{i=1}^{m} w_i x_i + b)$$

激活函数

$$\varphi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Slide credit: Weinan Zhang

Training perception



预测

$$\hat{y} = \varphi(\sum_{i=1}^{m} w_i x_i + b)$$

激活函数

$$\varphi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- 训练

$$w_i = w_i + \eta(y - \hat{y})x_i$$
$$b = b + \eta(y - \hat{y})$$

- 下列规则等价：
  - 如果输出正确，则不进行操作
  - 如果输出高了，降低正输入的权重
  - 如果输出低了，增加正输入的权重

Slide credit: Weinan Zhang

Newton’s method

Another algorithm to maximize $\ell(\theta)$

Newton's method: formuation

- Returning to logistic regression with g(z) being the sigmoid function
- A different algorithm for maximizing the log likelihood $\ell(\theta)$

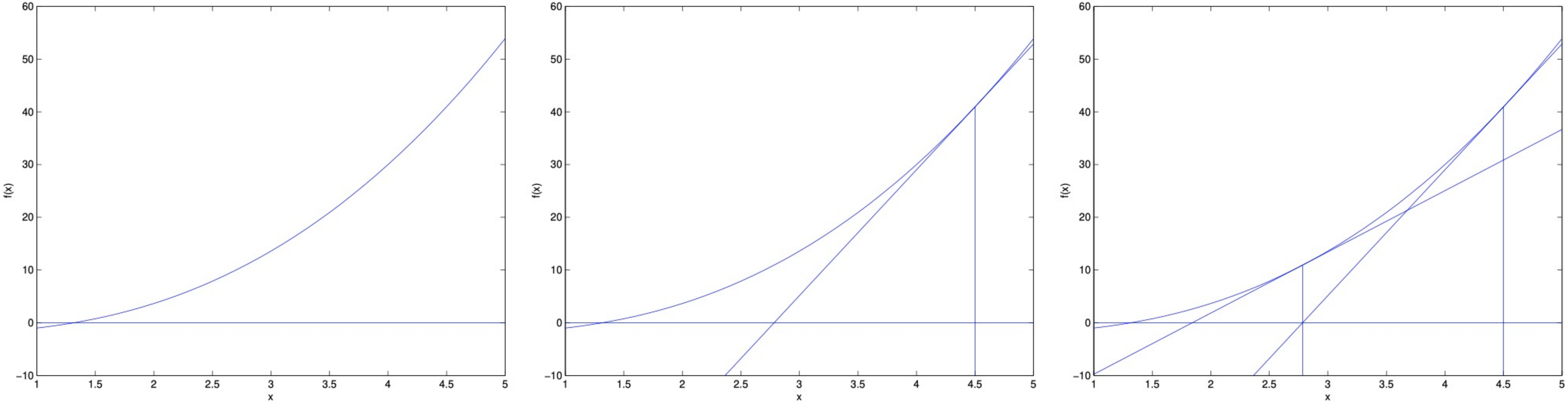
- To maximize $\ell(\theta)$, hope to find θ such that $\nabla\ell(\theta) = 0$

- New formulation

$$\text{Given } f : \mathbb{R}^d \to \mathbb{R} \text{ find } \theta \text{ s.t. } f(\theta) = 0.$$

Newton's method

Given $f : \mathbb{R}^d \to \mathbb{R}$ find θ s.t. $f(\theta) = 0$



Newton's method

- Suppose $\theta_n - \theta_{n+1} = \Delta$
- $\frac{f(\theta_n)-0}{\Delta} = f'(\theta_n)$
- $\theta_n - \theta_{n+1} = \Delta = \frac{f(\theta_n)}{f'(\theta_n)}$
- So the update rule in 1d $\quad \theta := \theta - \frac{f(\theta)}{f'(\theta)}$
- To maximizing the log likelihood? $\quad \theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}$

Generalization to the multidimensional setting

- For the likelihood, i.e., $f(\theta) = \nabla_\theta \ell(\theta)$ we need to generalize to a vector-valued function which has:

$$\theta^{(t+1)} = \theta^{(t)} - \left(H(\theta^{(t)})\right)^{-1} \nabla_\theta \ell(\theta^{(t)}).$$

in which $H_{i,j}(\theta) = \frac{\partial}{\partial\theta_i \partial\theta_j} \ell(\theta)$.

Properties of Newton’s method

- Convergence rate?
  - Use the Hessian information to determine step size, more adaptive
  - May converge very fast
- Computational cost?
  - Computing Hessian requires $O(d^2)$

Multi-class classification

Problem formulation

- Suppose we want to choose among k discrete values, e.g., $\{\text{'Cat'}, \text{'Dog'}, \text{'Car'}, \text{'Bus'}\}$ so $k = 4$.

- We encode with **one-hot** vectors i.e. $y \in \{0,1\}^k$ and $\sum_{j=1}^{k} y_j = 1$.

$$\underset{\text{'Cat'}}{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}} \quad \underset{\text{'Dog'}}{\begin{pmatrix}0\\1\\0\\0\end{pmatrix}} \quad \underset{\text{'Car'}}{\begin{pmatrix}0\\0\\1\\0\end{pmatrix}} \quad \underset{\text{'Bus'}}{\begin{pmatrix}0\\0\\0\\1\end{pmatrix}}$$

- In this case, $p(y|x;\theta)$ is a distribution over k discrete outcomes

Objective

- Introduce $\theta_1^\top x$, $\theta_2^\top x$, …, $\theta_k^\top x$ to represent the corresponding probabilities
- Hope:
  - Each probability $\in [0,1]$
  - The sum over all probabilities is 1

Softmax function

- Define the softmax function softmax : $\mathbb{R}^k \to \mathbb{R}^k$ as

$$\operatorname{softmax}(t_1, \ldots, t_k) = \begin{bmatrix} \frac{\exp(t_1)}{\sum_{j=1}^k \exp(t_j)} \\ \vdots \\ \frac{\exp(t_k)}{\sum_{j=1}^k \exp(t_j)} \end{bmatrix}. \tag{2.9}$$

- Let $(t_1, \ldots, t_k) = (\theta_1^\top x, \cdots, \theta_k^\top x)$

$$\begin{bmatrix} P(y = 1 \mid x; \theta) \\ \vdots \\ P(y = k \mid x; \theta) \end{bmatrix} = \operatorname{softmax}(t_1, \cdots, t_k) = \begin{bmatrix} \frac{\exp(\theta_1^\top x)}{\sum_{j=1}^k \exp(\theta_j^\top x)} \\ \vdots \\ \frac{\exp(\theta_k^\top x)}{\sum_{j=1}^k \exp(\theta_j^\top x)} \end{bmatrix}$$

Quiz

- Does k = 2 case agree with logistic regression?

$$P(y = j|x;\theta) = \frac{e^{\theta_j^T x}}{e^{\theta_1^T x} + e^{\theta_2^T x}}$$

How to optimize?

- Compute the negative log likelihood function

$$-\log p(y \mid x, \theta) = -\log\left(\frac{\exp(t_y)}{\sum_{j=1}^{k}\exp(t_j)}\right) = -\log\left(\frac{\exp(\theta_y^\top x)}{\sum_{j=1}^{k}\exp(\theta_j^\top x)}\right)$$

- Define the cross-entropy loss function

$$\ell_{\mathrm{ce}}((t_1, \ldots, t_k), y) = -\log\left(\frac{\exp(t_y)}{\sum_{j=1}^{k}\exp(t_j)}\right)$$

- Over n training examples?

$$\ell(\theta) = \sum_{i=1}^{n} \ell_{\mathrm{ce}}((\theta_1^\top x^{(i)}, \ldots, \theta_k^\top x^{(i)}), y^{(i)})$$

Gradient descent to minimize the loss

$$\frac{\partial \ell_{\mathrm{ce}}(t,y)}{\partial t_i} = \phi_i - 1\{y=i\}\,, \tag{2.16}$$

where $1\{\cdot\}$ is the indicator function, that is, $1\{y=i\}=1$ if $y=i$, and $1\{y=i\}=0$ if $y \neq i$. Alternatively, in vectorized notations, we have the following form which will be useful for Chapter 7:

$$\frac{\partial \ell_{\mathrm{ce}}(t,y)}{\partial t} = \phi - e_y\,, \tag{2.17}$$

where $e_s \in \mathbb{R}^k$ is the s-th natural basis vector (where the s-th entry is 1 and all other entries are zeros.) Using Chain rule, we have that

$$\frac{\partial \ell_{\mathrm{ce}}((\theta_1^\top x, \dots, \theta_k^\top x), y)}{\partial \theta_i} = \frac{\partial \ell(t,y)}{\partial t_i} \cdot \frac{\partial t_i}{\partial \theta_i} = (\phi_i - 1\{y=i\}) \cdot x\,. \tag{2.18}$$

Therefore, the gradient of the loss with respect to the part of parameter θ_i is

$$\frac{\partial \ell(\theta)}{\partial \theta_i} = \sum_{j=1}^{n} (\phi_i^{(j)} - 1\{y^{(j)}=i\}) \cdot x^{(j)}\,, \tag{2.19}$$

Summary

- Two-class classfication
  - Logistic regression
    - Intuition, optimization
  - Digression: the perceptron learning algorithm
  - Newton’s method
    - Use second-order information
- Multi-class classification
  - Softmax function