CS229 Machine Learning Note 2

Classification and logistic regression

Terminologies

- **Classification Problems**: Predict y in small num of discrete values
- **Binary classification**: y takes 0, **negative class**, and 1, **positive class**, $y^{(i)}$ is also called **label** of the sample.

2.1 Logistic Regression

New Hypotheses
Since $y \in \{0, 1\}$, change form of hypotheses $h_\theta(x)$:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

Still $\theta^T x = \theta_0 + \sum \theta_j x_j$

Logistic function or **Sigmoid function**:

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

Note:

- $z \to \infty, g(z) \to 1$, $z \to -\infty, g(z) \to 0$ (This alternates discrete (0,1) into a continuous function on (0,1))
- Not hard to show:

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

Estimate θ using MLE

Assume:

$$P(y|x;\theta) = h_\theta(x)^y(1-h_\theta(x))^{1-y} \sim \mathcal{B}er(h_\theta(x))$$

$$\begin{aligned}
\ell(\theta) &= p(\vec{y}|X;\theta) \\
&= \prod_{i=1}^{n} p(y^{(i)}|x^{(i)};\theta) \\
&= \prod_{i=1}^{n} (h_\theta(x^{(i)}))^{y^{(i)}}(1-h_\theta(x^{(i)}))^{1-y^{(i)}} \\
\ell(\theta) &= \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))
\end{aligned}$$

Optimize by $\theta := \theta + \alpha\nabla_\theta l(\theta)$: (Here is maximizing likelihood thus plus)

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

Therefore, **Stochastic Gradient Ascent Rule**:

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

Note that

- it takes **Identical Form** to **LMS**.
- different from linear regression, LR has no normal function thus can only be optimized by gradient descent.

2.2 Digression: Perceptron(感知器) Learning Algorithm

Threshold function: modify Logistic Regression to discrete case.

$$g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \qquad h_\theta(x) = g(\theta^T x)$$

Keep update rule:

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

- Perceptron regression hardly derives meaningful probabilistic interpretation
- So does deriving this by MLE

2.3 Multi-class classification

Multinomial distribution:

- $p(y|x;\theta)$ is a distribution over k possible discrete outcomes with $p = \phi_1, \dots, \phi_k$
- Introduce k groups of params $\theta_1, \dots, \theta_k$, then use $\theta_1^T x, \dots, \theta_k^T x$ to represent $\phi_1, \dots, \phi_k$
- Two problems
    - $\theta_j^T x$ is not necessarily 1.
    - Sum of $\theta_j^T x$ may not 1.

Softmax function: Addressing above, ensure output as probability vector.

$$\text{softmax}(t_1, t_2, \dots, t_k) = \begin{bmatrix} \frac{\exp(t_1)}{\sum_{j=1}^k \exp t_j} \\ \vdots \\ \frac{\exp(t_k)}{\sum_{j=1}^k \exp t_j} \end{bmatrix}$$

Logits: Inputs to the softmax function (vec t)

$$\Longrightarrow \begin{bmatrix} P(y=1|x;\theta) \\ \vdots \\ P(y=k|x;\theta) \end{bmatrix} = \text{softmax}(\theta_1^T x, \dots, \theta_k^T x) = \begin{bmatrix} \frac{\exp(\theta_1^T x)}{\sum_{j=1}^k \exp \theta_j^T x} \\ \vdots \\ \frac{\exp(\theta_k^T x)}{\sum_{j=1}^k \exp \theta_j^T x} \end{bmatrix}$$

Written as

$$P(y=i|x;\theta)=\phi_i=\frac{\exp(t_i)}{\sum_{j=1}^k \exp(t_j)}=\frac{\exp(\theta_i^T x)}{\sum_{j=1}^k \exp \theta_j^T x}$$
$$\implies \ell(\theta)=\sum_{i=1}^n -\log\left(\frac{\exp(\theta_{y^{(i)}}^T x^{(i)})}{\sum_{j=1}^k \exp(\theta_j^T x^{(i)})}\right)$$

Cross-entropy Loss: $\ell_{ce}:\mathbb{R}^k\times\{1,\dots,k\}\to\mathbb{R}_{\geq 0}$ modularize in the complex equation above:

$$\ell_{ce}((t_1,\dots,t_k),y)=-\log\left(\frac{\exp(t_y)}{\sum_{j=1}^k \exp(t_j)}\right)$$
$$\ell(\theta)=\sum_{i=1}^n \ell_{ce}((\theta_1^T x^{(i)},\dots,\theta_k^T x^{(i)}),y^{(i)})$$

By basic calculus, let $t=(t_1,\dots,t_k)$

$$\frac{\partial \ell_{ce}(t,y)}{\partial t_i}=\frac{\partial}{\partial t_i}(-t_y+\log(\sum_{j=1}^k \exp(t_j)))$$
$$=\phi_i-1\{y=i\} \text{ or in vectorized notations } \phi-e_y$$

By chain rule:

$$\frac{\partial}{\partial \theta_i}\ell_{ce}((\theta_1^T x,\dots,\theta_k^T x),y)=(\phi_i-1\{y=i\})x$$

Thus, **gradient of loss**:

$$\frac{\partial \ell(\theta)}{\partial \theta_i}=\sum_{j=1}^n(\phi_i^{(j)}-1\{y^{(j)}=i\})x^{(j)},\quad \phi_i^{(j)}=\frac{\exp(\theta_i^T x^{(j)})}{\sum_{s=1}^k \exp(\theta_s^T x^{(j)})}$$

2.4 Newton Method for Maximizing

Find $f(\theta)=0$ by updating rule:

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}$$

$$\implies \theta := \theta - \frac{\ell'(\theta)}{\ell''(\theta)}$$

Newton-Raphson method Generalize to multidimensional setting:

$$\theta := \theta - H^{-1} \nabla_\theta \ell(\theta)$$

where $\nabla_\theta \ell(\theta)$ us vector of partial derivative of $\ell(\theta)$ to θ_i's.
Hessian: d-by-d matrix, entry: $H_{ij} = \frac{\partial^2 \ell(\theta)}{\partial \theta_i \partial \theta_j}$(黑塞矩阵)

Advantages

- Faster convergence (than gradient)
- Though more expensive per iteration. (When d is small, overall faster)
- **But perform worse at high dim problems** (as deriving inverse of H is required)

Applied to maximize logistic regression, also called **Fisher scoring**