CS229: Machine Learning

Junchuan Zhao

May 18, 2022

Abstract

This document includes my notes of CS229: Machine Learning.

1 Lecture 2

1.1 Linear Regression

Hypothesis: $h(x) = \sum_{j=0}^{n} \theta_j x_j$ (n: number of features)
Linear Regression: the hypothesis is the linear combination of the training dataset.
Loss function (goal): $\min_{\theta} \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$
Parameters: θ, m, n, x, y

1.2 Gradient Descent

1.2.1 Batch Gradient Descent

Basic Ideas: start with some θ, keep changing θ to reduce $J(\theta)$, repeat until convergent
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ (α: learning rate)
$\frac{\partial}{\partial \theta_j} J(\theta) = (h_\theta(x) - y) \cdot x_j$
$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x)^i - y^i) \cdot x_j^i$ (α: learning rate)
Batch Gradient Descent: all the training data as a batch when optimizing the loss function. (-): not good for large scale dataset, very expensive.

1.2.2 Stochastic Gradient Descent

Algorithm 1 Stochastic Gradient Descent

Require: the parameter of j^{th} feature
Ensure: the updated parameter of j^{th} feature

```
for i = 1 to m do
    θ_j := θ_j − α · (h_θ(x)^i − y^i) · x_j^i;
end for
```

stochastic gradient descent heads over to the global optimum.

1.2.3 Total Equations

A Total Equation for Stochastic Gradient Descent: $\nabla_\theta J(\theta) = \mathbf{0}$
If A is square ($A \in \mathbb{R}_{n\times n}$)
 Trace of A: $tr(A) = \sum_i A_{ii}$
Features of Trace:

- $tr(A) = tr(A^T)$
- $f(A) = tr(AB)$, $\nabla_A f(A) = B^T$
- $tr(AB) = tr(BA)$
- $tr(ABC) = tr(CAB)$
- $\nabla_A tr(AA^T C) = CA + C^T A$

Loss Function: $\nabla_\theta J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$
$= \frac{1}{2}(\theta^T X^T - y^T)(X\theta - y)$
$= \frac{1}{2}(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta)$
$= X^T X\theta - X^T y$
Loss Function: $X^T X\theta - X^T y = \mathbf{0} \iff X^T X\theta = X^T y$
$\theta = (X^T X)^{-1} X^T y$

2 Lecture 3

2.1 Locally Weighted Regression

"Parametric" learning algorithm: fit fixed set of parameters (θ_i) to data.
"Non-parametric learning algorithm": amount of data/parameters you need to keep grows (linearly) with the size of data.

Linear Regression: to evaluate h at certain x
Fit θ to minimize
 $\frac{1}{2}\sum_i (y^i - \theta^T x^i)^2$,
return $\theta^T x$

Linear Regression: to evaluate h at local region of x
Fit θ to minimize
 $\sum_i w^i(y^i - \theta^T x^i)^2$, where w^i is a "weight function".
common choice for w^i is $w^i = e^{(-\frac{(x^i - x)^2}{2\tau^2})}$
If $|x^i - x|$ is small, then $w^i \approx 1$.
If $|x^i - x|$ is large, then $w^i \approx 0$.
τ: bandwidth, control a larger or narrower window.

2.2 Why Square Error?

Assume $y^i = \theta^T x^i + \epsilon^i$, where $\epsilon^i \sim N(0, \sigma^2)$ models effects of random noise
$p(y^i|x^i;\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y^i - \theta^T x^i)^2}{2\sigma^2}}$
$\iff y^i|x^i;\theta \sim N(\theta^T x^i, \sigma^2)$

Difference between Likelihood (L) and Probability (P): L variess parameters, P varies datapoints. Assume the datapoints are IID,

The "likelihood" of θ: $L(\theta) = p(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{i=1}^{m} p(y^i|x^i;\theta)$

$l(\theta) = \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{(\dots)} = \sum_{i=1}^{m} [\log \frac{1}{\sqrt{2\pi}\sigma} + \log e^{(\dots)}]$

$= m \log \frac{1}{\sqrt{2\pi}\sigma} - \sum_{i=1}^{m} \frac{(y^i - \theta^T x^i)^2}{2\sigma^2}$

MLE: maximum likelihood estimation. Choose θ to maximize $L(\theta)$

i.e. choose θ to minimize $\frac{1}{2} \sum_{i=1}^{m} (y^i - \theta^T x^i)^2$, which is actually $J(\theta)$

2.3 Classification

Binary Classification: $y \in \{0, 1\}$

2.3.1 Logistic Regression

We want $h_\theta(x) \in [0, 1]$, so we define

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

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

An important thing is that: the partial derivative of the sigmoid function is a **concave** function, which means it has the global maximum.

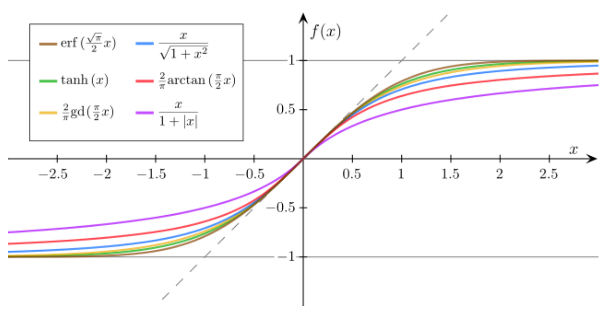

Figure 1: The graph of sigmoid function.

Different from Linear Regression $h_\theta(x) = \theta^T x$, Logistic Regression is $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$.

$p(y = 1|x;\theta) = h_\theta(x)$, $p(y = 0|x;\theta) = 1 - h_\theta(x)$

In sum, $p(y|x;\theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$

$L(\theta) = p(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{i=1}^{m} h_\theta(x^i)^{y^i} (1 - h_\theta(x^i))^{1-y^i}$

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

Choose θ to maximize $l(\theta)$, we use Batch gradient ascent.

Batch Gradient Ascent:

$\theta_j := \theta_j + \alpha \frac{\partial}{\partial \theta_j} l(\theta)$

$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i$, same as the one in linear regression. (A larger category called generalized linear model (GLM))

Note: no **normal equation** for logistic regression solution.

2.3.2 Newton's Method

The basic idea: have some function f, we want to fit θ, **s.t.** $f(\theta) = 0 \iff$ want maximizing $l(\theta)$ $\iff$ want $l'(\theta) = 0$

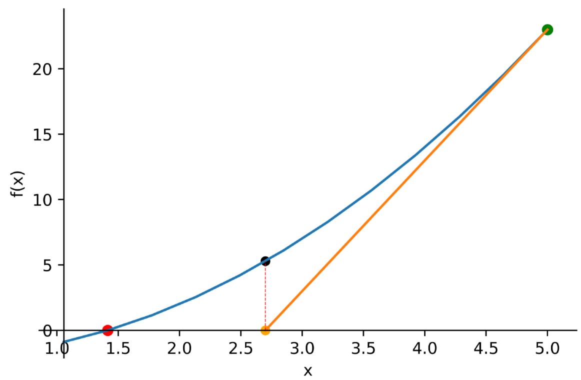

Figure 2: Newton's Method.

$\theta^{t+1} := \theta^t - \frac{f(\theta^t)}{f'(\theta^t)}$, let $f(\theta) = l'(\theta)$
$\theta^{t+1} := \theta^t - \frac{l'(\theta^t)}{l''(\theta^t)}$

"**Quadratic convergent**": 0.01 error $\longrightarrow$ 0.0001 error $\longrightarrow$ 0.00000001 error.

When θ is a vector: $(\theta \in \mathbb{R}^{n+1})$
$\theta^{t+1} := \theta^t + \alpha H^{-1} \nabla_\theta l(\theta)$
where

$$\nabla_\theta l(\theta) = \begin{bmatrix} \frac{\partial l}{\partial \theta_{00}} & \frac{\partial l}{\partial \theta_{01}} & \cdots & \frac{\partial l}{\partial \theta_{0n}} \\ \frac{\partial l}{\partial \theta_{10}} & \frac{\partial l}{\partial \theta_{11}} & \cdots & \frac{\partial l}{\partial \theta_{1n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial l}{\partial \theta_{n0}} & \frac{\partial l}{\partial \theta_{n1}} & \cdots & \frac{\partial l}{\partial \theta_{nn}} \end{bmatrix}$$

$H \in \mathbb{R}^{n+1 \times n+1}$ is the Hessian Matrix, $H_{ij} = \frac{\partial^2 l}{\partial \theta_i \partial \theta_j}$
Note: If the dataset is large, the size of H will be too large to compute.

3 Lecture 4

3.1 Perceptron

Binary step function:

$$g(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases} \tag{1}$$

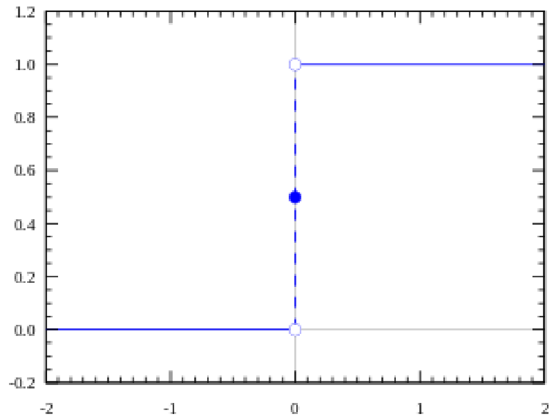

Figure 3: The graph of binary step.

Batch Gradient Ascent: $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$

3.2 Exponential Family

3.2.1 Defination and Examples of Exponential Family

Probability Density Function (PDF): $p(y;\eta) = b(y)\text{exp}[\eta^T T(y) - a(\eta)]$
y: data
η: natural parameter
$T(y)$: sufficient statistic
$b(y)$: base measure
$a(\eta)$: log partition

Example 1: Bernoulli (Binary Data): $p(y;\phi) = \phi^y(1-\phi)^{(1-y)}$
$\phi =$ probability of the event
$p(y;\phi) = \exp[y\log(\frac{\phi}{1-\phi}) + \log(1-\phi)]$
$b(y) = 1$
$T(y) = y$
$\eta = \log\frac{\phi}{1-\phi} \Rightarrow \phi = \frac{1}{1+e^{-\eta}}$
$a(\eta) = -\log(1-\phi) = -\log(1-\frac{1}{1+e^{-\eta}}) = \log(1+e^\eta)$

Example 2: Gaussian (assume variance = 1): $p(y;\mu) = \frac{1}{\sqrt{2\pi}}e^{(-\frac{(y-\mu)^2}{2})} = \frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}\exp(\mu y - \frac{\mu^2}{2})$
$b(y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}}$
$T(y) = y$
$\eta = \mu$
$a(\eta) = \frac{\mu^2}{2} = \frac{\eta^2}{2}$

3.2.2 Properties of Exponential Family

- MLE w.r.t η is concave, NLL is convex
- $E[y;\eta] = \frac{\partial}{\partial\eta}a(\eta)$
- $Var[y;\eta] = \frac{\partial^2}{\partial\eta^2}a(\eta)$

Real - Gaussian; Binary - Bernoulli; Count - Poisson; $\mathbb{R}^+$ - Gamma, Exponential; Distribution - Beta, Dirichlet (Bayesian).

3.3 Generalized Linear Model (GLM)

3.3.1 GLM Assumptions

Assumptions/Design Choices
(1) $y|x;\theta \sim$ Exponential Family (η)
(2) $\eta = \theta^T x$, $\theta \in \mathbb{R}^n$ and $x \in \mathbb{R}^n$
(3) Test Time: output $h_\theta(x) = E[y|x;\theta]$

3.3.2 GLM Pipeline

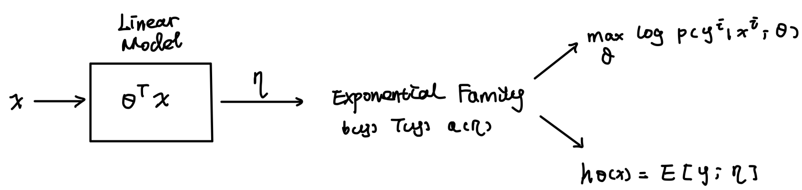

Figure 4: The pipeline for GLM.

3.3.3 GLM Training

Learning Update Rule: $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$

Terminology:
η - natural parameter
$g(\eta) = \mu = E(y;\eta)$ - canonical response function
$\eta = g^{-1}(\mu)$ - canonical link function

Three parameterizations:
Model Param: θ (**Learn Parameters**)
Natural Param: η (Design Choice: $\theta^T x$)
Canonical Param: ϕ - Bernoulli, μ σ - Gaussian, λ - Poisson (Canonical Response: $g(\cdot)$)

3.3.4 Review Logistic Regression

$h_\theta(x) = E[y|x;\theta] = \phi$ (the mean of the Bernoulli distribution)
$h_\theta(x) = \phi = \frac{1}{1+e^{-\eta}} = \frac{1}{1+e^{-\theta^T x}}$

3.4 Softmax Regression

Task: multi-class Classification
k - number of classes, $x^i \in \mathbb{R}^n$, Label $y^i \in \{0,1\}^k$ (one-hot vector)
Each class has its own parameters: $\theta_{class} \in \mathbb{R}^n$

$$p(y=i|x;\theta)=\frac{e^{\theta_i^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}$$

$$l(\theta)=\sum_{i=1}^{m} log \prod_{l=1}^{k} (\frac{e^{\theta_l^T x^i}}{\sum_{j=1}^{k} e^{\theta_j^T x^i}})^{\{y^i=l\}}$$