Learning From Data
Lecture 2: Linear Regression & Logistic Regression

Yang Li yangli@sz.tsinghua.edu.cn

September 25, 2020

Today's Lecture

Supervised Learning (Part I)

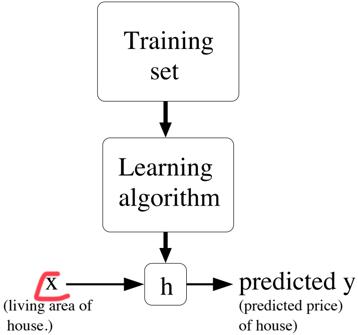
- Linear Regression
- Binary Classification
- Multi-Class Classification

Review: Supervised Learning

- Input space: $\mathcal{X}$, Target space: $\mathcal{Y}$

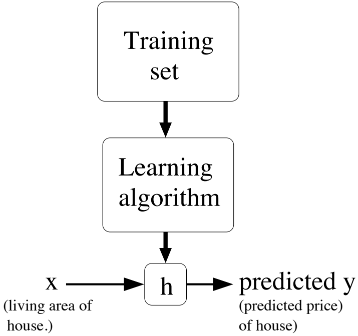
Review: Supervised Learning

- Input space: $\mathcal{X}$, Target space: $\mathcal{Y}$
- Given training examples, we want to learn a **hypothesis** function $h : \mathcal{X} \to \mathcal{Y}$ so that $h(x)$ is a "good" predictor for the corresponding y.



Review: Supervised Learning

- Input space: $\mathcal{X}$, Target space: $\mathcal{Y}$
- Given training examples, we want to learn a **hypothesis** function $h : \mathcal{X} \to \mathcal{Y}$ so that $h(x)$ is a "good" predictor for the corresponding y.



- y is discrete (categorical): **classification problem**
- y is continuous (real value): **regression problem**

Review: Inference vs Learning

Given training data of x and y,

Inference

knowing the structure of f, find good models to describe f. i.e. model the data generation process $\leftarrow$ focus of statistics

Prediction

given **future** data samples of x, predict the corresponding output data y using f. $\leftarrow$ focus of machine learning

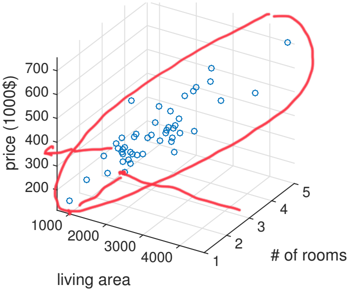
Linear Regression

Linear Regression Model
Ordinary Least Square
Maximum Likelihood Estimation

Linear Regression

Example: predict Portland housing price

Living area (ft^2)	# bedrooms	Price ($1000)
x_1	x_2	y
2104	3	400
1600	3	330
2400	3	369
⋮	⋮	⋮



Linear Approximation

A linear model

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θ_i's are called **parameters**.

Linear Approximation

A linear model

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θ_i's are called **parameters**.

Using vector notation,

$$h(x) = \theta^T x, \quad \text{where } \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

Alternative Notation

$\theta_1 \quad \theta_2 \quad \theta_0$

$$h(x) = w_1 x_1 + w_2 x_2 + b$$

w_1, w_2 are called **weights**, b is called the **bias**

$b \in \mathbb{R}$

$$h(x) = w^T x + b, \quad \text{where } w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Apply model to new data

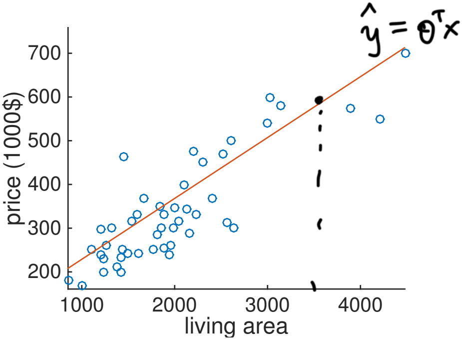
Suppose we have the optimal parameters θ , e.g.

```
> h = LinearRegression().fit(X, y)
> theta = h.coef
array([89.60, 0.1392, -8.738])
```

θ_0 θ_1 θ_2

make a prediction of new feature x:

$$\hat{y} = h_\theta(x) = \theta^T x$$



Model Estimation

How to estimate model parameters θ (or w and b) from data?

Model Estimation

How to estimate model parameters θ (or w and b) from data?

Least Square Estimation

Minimize sum of the prediction error squared (least square error) with respect to θ

Model Estimation

How to estimate model parameters θ (or w and b) from data?

Least Square Estimation

Minimize sum of the prediction error squared (least square error) with respect to θ

Maximum Likelihood Estimation

- Assume the data are generated from $h(x)$ with some noise distribution.
- Determines the parameters θ most likely to produce the observed data.

Model Estimation

How to estimate model parameters θ (or w and b) from data?

Least Square Estimation

Minimize sum of the prediction error squared (least square error) with respect to θ

Maximum Likelihood Estimation

- Assume the data are generated from $h(x)$ with some noise distribution.
- Determines the parameters θ most likely to produce the observed data.

Other estimation methods exist, e.g. Bayesian estimation

Ordinary Least Square

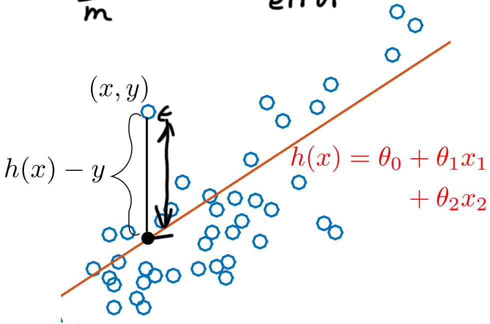
Cost function:

prediction, ground truth

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

$\frac{1}{m}$

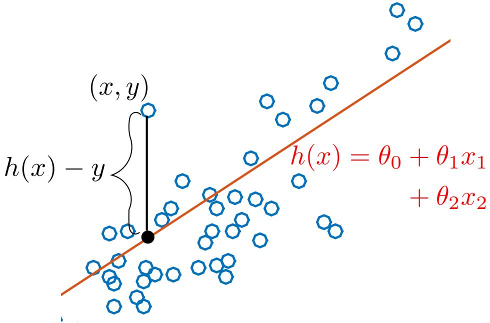
error



Ordinary Least Square

Cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$



- This model is called **ordinary least square**

Ordinary Least Square

Cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

- This model is called **ordinary least square**

Ordinary Least Square

Cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2$$

- This model is called **ordinary least square**

Ordinary Least square problem

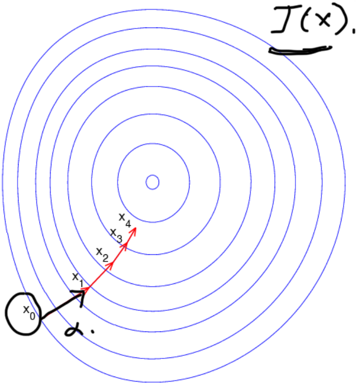
$$\begin{aligned} &\min_{\theta} J(\theta) \\ =&\min_{\theta} \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 \end{aligned}$$

How to minimize $J(\theta)$?

- Numerical solution: gradient descent, Newton's method
- Analytical solution: normal equation

Gradient descent

A first-order iterative optimization algorithm for finding the minimum of a function $J(\theta)$.



Key idea

Start at an initial guess, repeatedly change θ to decrease $J(\theta)$:

$$\theta := \theta - \alpha \nabla J(\theta)$$

α is the **learning rate**

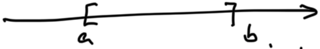
Review: Convex function

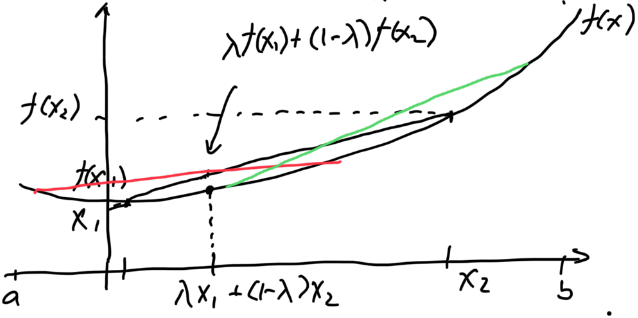
Definition

A function $f(x)$ is **convex** on a convex set C if for any $x_1, x_2 \in C$ and $0 \leq \lambda \leq 1$,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$$

e.g. C is an interval $[a, b]$





Review: Convex function

Definition

A function $f(x)$ is **convex** on a convex set C if for any $x_1, x_2 \in C$ and $0 \leq \lambda \leq 1$,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(y_2)$$

e.g. C is an interval $[a, b]$

Theorem

If $J(\theta)$ is convex, gradient descent finds the global minimum.

For the ordinary least square problem,
$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 = \frac{1}{2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$,

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix}, \text{ where } \frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j} \frac{1}{2}\sum_{i=1}^{m} (\theta^T x^i - y^i)^2$$

$$= \frac{1}{2}\sum_{i=1}^{m} 2(\theta^T x^i - y^i)\underbrace{\frac{\partial}{\partial \theta_j}(\theta^T x^i - y^i)}_{x_j^i}$$

$$= \sum_{i=1}^{m} x_j^i (\theta^T x^i - y^i)$$

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

For the ordinary least square problem,
$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h(x^{(i)}) - y^{(i)})^2 = \frac{1}{2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2$,

$$\nabla J(\theta) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{bmatrix}, \text{ where } \frac{\partial J(\theta)}{\partial \theta_j} = \frac{\partial}{\partial \theta_j}\left[\frac{1}{2}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^2\right]$$

$$= \sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right) x_j^{(i)}$$

Gradient descent for ordinary least square

Gradient of cost function: $\nabla J(\theta) = \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)}\right) x_j^{(i)}$

Gradient descent update: $\theta := \theta - \alpha \nabla J(\theta)$

$-(y^i - \theta^T x^i)$

Batch Gradient Descent

```
Repeat until convergence{
    θ_j = θ_j + α Σ_{i=1}^{m} (y^(i) − h_θ(x^(i))) x_j^(i)   for every j
}
```

$\nabla J(\theta)$

Gradient descent for ordinary least square

Gradient of cost function: $\nabla J(\theta) = \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)}\right) x_j^{(i)}$
Gradient descent update: $\theta := \theta - \alpha \nabla J(\theta)$

Batch Gradient Descent

```
Repeat until convergence{
    θ_j = θ_j + α Σ_{i=1}^{m} (y^(i) − h_θ(x^(i))) x_j^(i) for every j
}
```

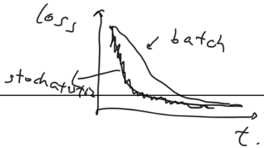
θ is only updated after we have seen all m training samples.

Batch gradient descent

Repeat until convergence{

$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$ for every j

}



Stochastic gradient descent

Repeat until convergence{

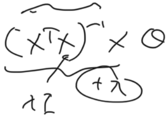
  for $i = 1 \dots m$ {

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

  }

}

θ is updated each time a training example is read



Batch gradient descent

```
Repeat until convergence{
  θ_j = θ_j + α Σ_{i=1}^{m} (y^(i) − h_θ(x^(i)))x_j^(i) for every j
}
```

$$\text{Repeat until convergence}\{\quad \theta_j = \theta_j + \alpha \sum_{i=1}^{m} (y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} \text{ for every j} \quad\}$$

Stochastic gradient descent

```
Repeat until convergence{
  for i = 1...m {
    θ_j = θ_j + α(y^(i) − h_θ(x^(i)))x_j^(i) for every j
  }
}
```

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

θ is updated each time a training example is read

- Stochastic gradient descent gets θ close to minimum much faster
- Good for regression on large data

Minimize $J(\theta)$ Analytically

The matrix notation

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

X is called the **design matrix**.

Minimize $J(\theta)$ Analytically

The matrix notation

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}, \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

X is called the **design matrix**. The least square function can be written as

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y) = \frac{1}{2}\sum_{i=1}^{m}(\theta^T x^i - y^i)^2$$

$$\underset{(m\times n)}{\begin{bmatrix} — x^1 — \\ — x^2 — \\ \vdots \\ — x^m — \end{bmatrix}} \underset{(n\times 1)}{\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}} = \underset{X\theta}{\begin{pmatrix} x^{(1)T}\theta \\ x^{(2)T}\theta \\ \vdots \\ x^{(m)T}\theta \end{pmatrix}} - \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{pmatrix} =$$

$(m\times 1)$

Compute the gradient of $J(\theta)$:

$tr(A) = tr(A^T)$

$tr(x) = x$

$tr(AB) = tr(BA)$

$tr(ABC) = tr(CAB) = tr(BCA)$

- $\nabla_A tr(AB) = B^T = \nabla_A tr(BA)$
- $\nabla_x x^T A x = (A + A^T)x.$

if $A = A^T$, $= 2Ax.$

$A = X^T X$

$$\nabla_\theta J(\theta) = \nabla_\theta \left[\frac{1}{2}(X\theta - y)^T (X\theta - y) \right]$$

$$= \frac{1}{2}\left(\nabla_\theta \left[\theta^T X^T X\theta - y^T X\theta - \theta^T X^T y + y^T y\right]\right)$$

$$= \frac{1}{2}\left(\nabla_\theta[\theta^T X^T X\theta] - \nabla_\theta[y^T X\theta - \theta^T X^T y] + \nabla_\theta[y^T y]\right)$$

$(1\times m)(m\times n)(n\times 1)$

$$= \frac{1}{2}\left(2X^T X\theta - \nabla_\theta\left(tr(y^T X\theta + \theta^T X^T y)\right)\right)$$

$tr(y^T X\theta) + tr(\theta^T X^T y)$

$$= \frac{1}{2}\left(2X^T X\theta - \nabla_\theta\, 2tr(y^T X\theta)\right)$$

$$= \frac{1}{2}\left(2X^T X\theta - 2X^T y\right)$$

$$= X^T X\theta - X^T y. \quad = 0.$$

$$\theta = (X^T X)^{-1} X^T y$$

pseudo-inverse of X.

$X\theta = Y.$

$\theta = X^{-1} Y.$

ill-condional

A^{-1}

conditional # of A

$$\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} \leftarrow \text{max singular value}$$

Compute the gradient of $J(\theta)$:

$$\nabla_\theta J(\theta) = \nabla_\theta \left[\frac{1}{2} (X\theta - y)^T (X\theta - y) \right]$$

$$=$$

Compute the gradient of $J(\theta)$:

$$\begin{aligned}\nabla_\theta J(\theta) =& \nabla_\theta \left[\frac{1}{2}(X\theta - y)^T(X\theta - y)\right] \\ =& X^T X\theta - X^T y\end{aligned}$$

Since $J(\theta)$ is **convex**, x is a global minimum of $J(\theta)$ when $\nabla J(\theta) = 0$.

Compute the gradient of $J(\theta)$:

$$\begin{aligned}\nabla_\theta J(\theta) =& \nabla_\theta \left[\frac{1}{2}(X\theta - y)^T(X\theta - y)\right] \\ =& X^TX\theta - X^Ty\end{aligned}$$

Since $J(\theta)$ is **convex**, x is a global minimum of $J(\theta)$ when $\nabla J(\theta) = 0$.

The Normal equation

$$\theta = (X^TX)^{-1}X^Ty$$

Compute the gradient of $J(\theta)$:

$$\begin{aligned}\nabla_\theta J(\theta) =& \nabla_\theta \left[\frac{1}{2}(X\theta - y)^T(X\theta - y)\right] \\ =& X^TX\theta - X^Ty\end{aligned}$$

Since $J(\theta)$ is **convex**, x is a global minimum of $J(\theta)$ when $\nabla J(\theta) = 0$.

The Normal equation

$$\theta = (X^TX)^{-1}X^Ty$$

$(X^TX)^{-1}X^T$ is called the **Moore-Penrose pseudoinverse of** X

Which method to use?

gradient descent	**normal equation**
iterative solution	exact solution

Which method to use?

gradient descent	normal equation
iterative solution	exact solution
need to choose proper learning parameter α for cost function to converge	

Which method to use?

gradient descent	**normal equation**
iterative solution	exact solution
need to choose proper learning parameter α for cost function to converge	numerically unstable when X is ill-conditioned. e.g. features are highly correlated

Which method to use?

gradient descent	**normal equation**
iterative solution	exact solution
need to choose proper learning parameter α for cost function to converge	numerically unstable when X is ill-conditioned. e.g. features are highly correlated
works well for large number of samples m	

Which method to use?

gradient descent	**normal equation**
iterative solution	exact solution
need to choose proper learning parameter α for cost function to converge	numerically unstable when X is ill-conditioned. e.g. features are highly correlated
works well for large number of samples m	solving equation is slow when m is large

Minimize $J(\theta)$ using Newton's Method

Newton's method solves real functions $f(x) = 0$ by iterative approximation

- Update rule: $x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)}$

Minimize $J(\theta)$ using Newton's Method

Newton's method solves real functions $f(x) = 0$ by iterative approximation

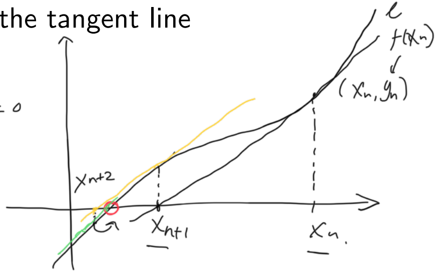
- Update rule: $x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)}$

Geometric intuition of Newton's method

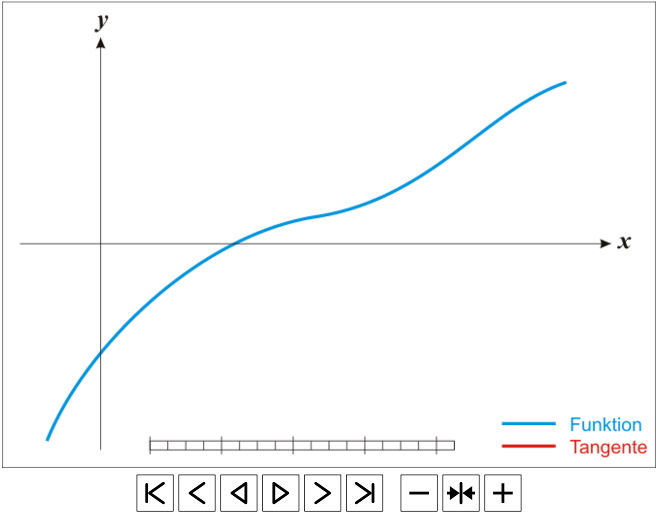
- Find tangent line of f at (x_n, y_n)
- $x_{n+1} \leftarrow$ x-intercept of the tangent line
- $y_{n+1} \leftarrow f(x_{n+1})$

$l: \; y = f'(x_n)(x - x_n) + f(x_n) = 0$

$x = -\frac{f(x_n)}{f'(x_n)} + x_n$



Newton's Method Demo



https://en.wikipedia.org/wiki/File:NewtonIteration_Ani.gif

Minimize $J(\theta)$ using Newton's Method

Newton's method for optimization $\min_\theta J(\theta)$

Use newton's method to solve $\nabla_\theta J(\theta) = 0$:

- x is one-dimensional:

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

Minimize $J(\theta)$ using Newton's Method

Newton's method for optimization $\min_\theta J(\theta)$

Use newton's method to solve $\nabla_\theta J(\theta) = 0$:

- x is one-dimensional:

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

- x is multidimensional:

$$H(\theta) = \begin{bmatrix} \frac{\partial^2 J(\theta)}{\partial \theta_1^2} & \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 J(\theta)}{\partial \theta_1 \partial \theta_n} \\ \frac{\partial^2 J(\theta)}{\partial \theta_2 \partial \theta_1} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2 J(\theta)}{\partial \theta_n \partial \theta_1} & \cdots & \cdots & \frac{\partial^2 J(\theta)}{\partial \theta_n^2} \end{bmatrix}$$

$$\theta = \theta - H^{-1}(\theta) \nabla J(\theta)$$

where H is the Hessian matrix of $J(\theta)$.

a.k.a Newton-Raphson method

Newton's Method for Optimization

```
Initialize θ
While θ has not coverged {
  θ := θ − H^{-1}(θ)∇J(θ)
}
```

$$\theta := \theta - H^{-1}(\theta)\nabla J(\theta)$$

Newton's Method for Optimization

```
Initialize θ
While θ has not coverged {
  θ := θ − H^{-1}(θ)∇J(θ)
}
```

Performance of Newton's method:

- Needs fewer interations than batch gradient descent

Newton's Method for Optimization

```
Initialize θ
While θ has not coverged {
  θ := θ − H^{-1}(θ)∇J(θ)
}
```

Performance of Newton's method:

- Needs fewer interations than batch gradient descent
- Computing H^{-1} is time consuming

Newton's Method for Optimization

Initialize θ
While θ has not coverged {
  $\theta := \theta - H^{-1}(\theta)\nabla J(\theta)$
}

Performance of Newton's method:

- Needs fewer interations than batch gradient descent
- Computing H^{-1} is time consuming
- Faster in practice when n is small

Maximum Likelihood Estimation

Consider target y is modeled as

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

and suppose $\epsilon^{(i)}$ are *independently and identically distributed (IID)* to Gaussian distribution $\mathcal{N}(0, \sigma^2)$,

IID.

Maximum Likelihood Estimation

Consider target y is modeled as

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

and suppose $\epsilon^{(i)}$ are *independently and identically distributed (IID)* to Gaussian distribution $\mathcal{N}(0, \sigma^2)$, then

$$p(\epsilon^{(i)}) =$$

Maximum Likelihood Estimation

Consider target y is modeled as

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

and suppose $\epsilon^{(i)}$ are *independently and identically distributed (IID)* to Gaussian distribution $\mathcal{N}(0, \sigma^2)$, then

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{{\epsilon^{(i)}}^2}{2\sigma^2}\right)$$

Maximum Likelihood Estimation

Consider target y is modeled as

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

and suppose $\epsilon^{(i)}$ are *independently and identically distributed (IID)* to Gaussian distribution $\mathcal{N}(0, \sigma^2)$, then

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{{\epsilon^{(i)}}^2}{2\sigma^2}\right)$$

deterministic parameter

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

Maximum Likelihood Estimation

The **likelihood** of this model with respect to θ is

$$L(\theta) = \underbrace{p(\vec{y}|X;\theta)} = \prod_{i=1}^{m} \underbrace{p(y^{(i)}|x^{(i)};\theta)}$$

Maximum Likelihood Estimation

The **likelihood** of this model with respect to θ is

$$L(\theta) = p(\vec{y}|X;\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

Maximum likelihood estimation of θ:

$$\theta_{MLE} = \underset{\theta}{\operatorname{argmax}}\, L(\theta)$$

Maximum Likelihood Estimation

We compute log likelihood,

$\ell(\theta)$

$$\log L(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta)$$

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

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

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

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

$\max_\theta \ell(\theta) \leftrightarrow \min_\theta J(\theta)$

least square

Maximum Likelihood Estimation

We compute log likelihood,

$$\log L(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta)$$

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

Maximum Likelihood Estimation

We compute log likelihood,

$$\log L(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta)$$

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

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

Maximum Likelihood Estimation

We compute log likelihood,

$$\log L(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta)$$

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

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

Then $\text{argmax}_\theta \log L(\theta) \equiv \text{argmin}_\theta \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$.

Maximum Likelihood Estimation

We compute log likelihood,

$$\log L(\theta) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta)$$

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

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

Then $\text{argmax}_\theta \log L(\theta) \equiv \text{argmin}_\theta \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$.

Under the assumptions on $\epsilon^{(i)}$, least-squares regression corresponds to the maximum likelihood estimate of θ.

Linear Regression Summary

- Least square regression
- Solving least square:
  - gradient descent
  - normal equation
  - newton's method
- Probabilistic interpretation: maximum likelihood

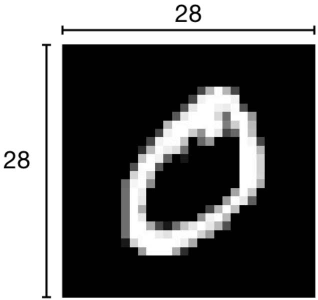
Logistic Regression

A binary classification problem

Classify binary digits

- Training data: 12600 grayscale images of handwritten digits

- Each image is represent by a vector $x^{(i)}$ of dimension $28 \times 28 = 784$
- Vectors $x^{(i)}$ are normalized to [0,1]

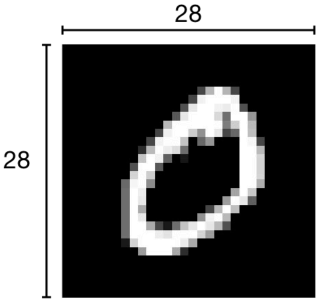


A binary classification problem

Classify binary digits

- Training data: 12600 grayscale images of handwritten digits

- Each image is represent by a vector $x^{(i)}$ of dimension $28 \times 28 = 784$
- Vectors $x^{(i)}$ are normalized to [0,1]



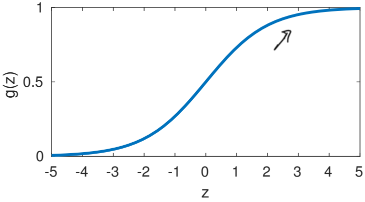
Binary classification: $\mathcal{Y} = \{0, 1\}$

- negative class: $y^{(i)} = 0$
- positive class: $y^{(i)} = 1$

Logistic Regression Hypothesis Function

Sigmoid function

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



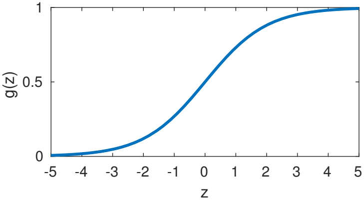
- $g : \mathbb{R} \to (0, 1)$
- $g'(z) = g(z)(1 - g(z))$

Logistic Regression Hypothesis Function

Sigmoid function

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

- $g : \mathbb{R} \rightarrow (0, 1)$
- $g'(z) = g(z)(1 - g(z))$

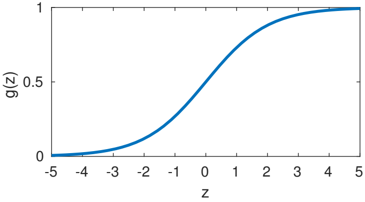


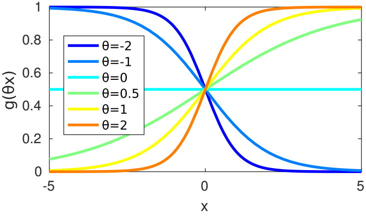
Logistic Regression Hypothesis Function

Sigmoid function

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

- $g : \mathbb{R} \to (0,1)$
- $g'(z) = g(z)(1-g(z))$





Hypothesis function for logistic regression:

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

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**.
e.g. tossing a coin with $p(head) = h_\theta(x)$

- $p(y = 1 \mid x; \theta) = h_\theta(x)$ $= \phi$ $\quad y|x \sim \text{Bernoulli}(\phi)$.
- $p(y = 0 \mid x; \theta) = 1 - h_\theta(x)$ $= 1 - \phi$.

  p(tail) ↑

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**.
e.g. tossing a coin with $p(head) = h_\theta(x)$

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

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**.
e.g. tossing a coin with $p(head) = h_\theta(x)$

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

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Given m independently generated training examples, the likelihood function is:

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

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

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**.
e.g. tossing a coin with $p(head) = h_\theta(x)$

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

$$p(y \mid x; \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}$$

Given m independently generated training examples, the likelihood function is:

$$L(\theta) = p(\vec{y}|X; \theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta)$$

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

$l(\theta)$ is concave!

Maximum likelihood estimation for logistic regression

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

Solve $\text{argmax}_\theta \, l(\theta)$ using gradient ascent:

$$g'(x) = g(x)(1 - g(x))$$

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

$$\frac{\partial}{\partial \theta_j} g(\theta^T x^i) = g(\theta^T x^i)(1 - g(\theta^T x^i)) \frac{\partial}{\partial \theta_j} \theta^T x^i$$

$$= h_\theta(x^i)(1 - h_\theta(x^i)) x_j^i$$

$$\frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} y^i \frac{1}{h_\theta(x^i)} \left(\frac{\partial}{\partial \theta_j} h_\theta(x^i)\right) + \frac{1 - y^i}{1 - h_\theta(x^i)} \frac{\partial}{\partial \theta_j} (-1) h_\theta(x^i)$$

$$= \sum_{i=1}^{m} \left[y^i \frac{1}{h_\theta(x^i)} - (1 - y^i) \frac{1}{1 - h_\theta(x^i)} \right] \frac{\partial}{\partial \theta_j} h_\theta(x^i)$$

$$= \sum_{i=1}^{m} \left(\frac{y^i h_\theta(x^i)(1 - h_\theta(x^i)) x_j^i}{h_\theta(x^i)} - \frac{(1 - y^i)}{1 - h_\theta(x)} h_\theta(x^i)(1 - h_\theta(x^i)) x_j^i \right)$$

$$= \sum_{i=1}^{m} \left(y^i (1 - h_\theta(x^i)) - (1 - y^i) h_\theta(x^i) \right) x_j^i$$

$$= \sum_{i=1}^{m} \left(y^i - h_\theta(x^i) \right) x_j^i$$

$\theta^T x$, $g(\theta^T x)$

Maximum likelihood estimation for logistic regression

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

Solve $\text{argmax}_\theta \, l(\theta)$ using gradient ascent:

$$\frac{\partial l(\theta)}{\partial \theta_j} = \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

Stocastic Gradient Ascent

```
Repeat until convergence{
  for i = 1...m {
    θ_j = θ_j + α(y^(i) − h_θ(x^(i)))x_j^(i)   for every j
  }
}
```

- Update rule has the same form as least square regression, but with different hypothesis function h_θ

Binary Digit Classification

Using the learned classifier

Given an image x, the predicted label is

$$\hat{y} = \begin{cases} 1 & g(\theta^T x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Binary digit classification results

	sample size	accuracy
Training	16200	100%
Testing	1225	100%

- Testing accuracy is 100% since this problem is relatively easy.

Multi-Class Classification

Multiple Binary Classifiers

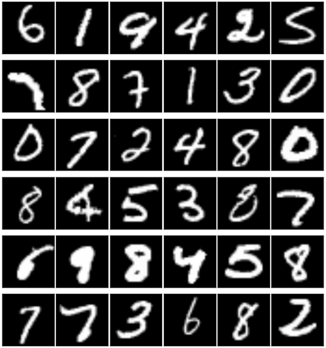
Softmax Regression

Multi-class classification

Each data sample belong to one of $k > 2$ different classes.

$$\mathcal{Y} = \{1, \ldots, k\}$$

MNIST Samples $|\mathcal{Y}| = 10.$



Given new sample $x \in \mathbb{R}^k$, predict which class it belongs.

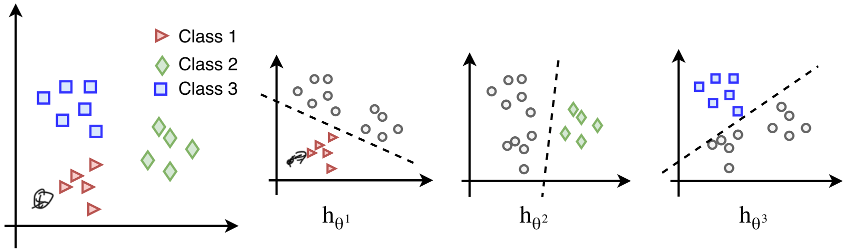
Naive Approach: Convert to binary classification

One-Vs-Rest

Learn k classifiers $h_1, \dots, h_k$. Each h_i classify one class against the rest of the classes.

Given a new data sample x, its predicted label $\hat{y}$:

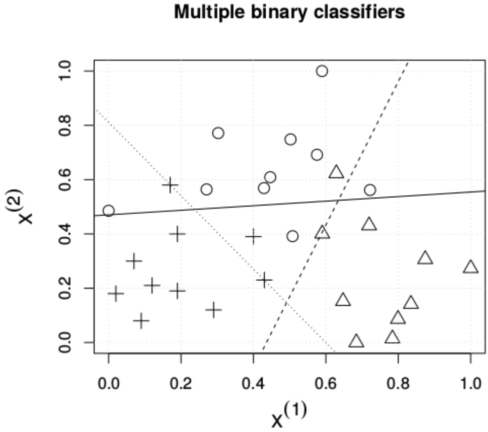
$$\hat{y} = \operatorname*{argmax}_i h_i(x)$$



Multiple binary classifiers

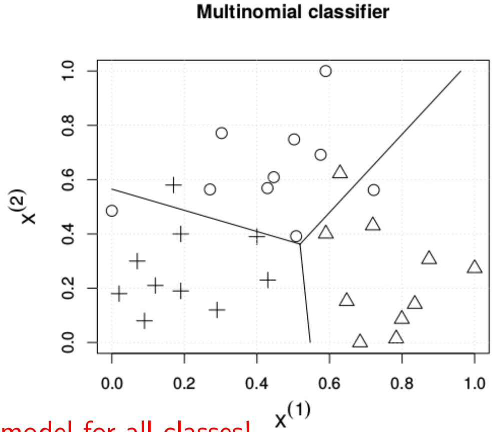
Drawbacks of One-Vs-Rest:

- Class unbalance: more negative samples than positive samples
- Different classifiers may have different confidence scales



Drawbacks of One-Vs-Rest:

- Class imbalance: more negative samples than positive samples
- Different classifiers may have different confidence scales



Learn one model for all classes!

Extend logistic regression: Softmax Regression

Assume $p(y|x)$ is **multinomial distributed**.
e.g. outcomes of rolling a k-sided die n times, each side has independent probability $\underbrace{\phi_1, \ldots \phi_k}_{h_\theta(x)}$

Extend logistic regression: Softmax Regression

Assume $p(y|x)$ is **multinomial distributed**.
e.g. outcomes of rolling a k-sided die n times, each side has independent probability $\phi_1, \ldots \phi_k$

Hypothesis function for sample x:

$$h_\theta(x) = \begin{bmatrix} p(y=1|x;\theta) \\ \vdots \\ p(y=k|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x_j}} \begin{bmatrix} e^{\theta_1^T x} \\ \vdots \\ e^{\theta_k^T x} \end{bmatrix} = \text{softmax}(\theta^T x)$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{(z_j)}}$$

Extend logistic regression: Softmax Regression

Assume $p(y|x)$ is **multinomial distributed**.
e.g. outcomes of rolling a k-sided die m times, each side has independent probability $\phi_1, \dots \phi_k$

Hypothesis function for sample x:

$$h_\theta(x) = \begin{bmatrix} p(y=1|x;\theta) \\ \vdots \\ p(y=k|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x_j}} \begin{bmatrix} e^{\theta_1^T x} \\ \vdots \\ e^{\theta_k^T x} \end{bmatrix} = \text{softmax}(\theta^T x)$$

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{(z_j)}}$$

Parameters: $\theta = \begin{bmatrix} - & \theta_1^T & - \\ & \vdots & \\ - & \theta_k^T & - \end{bmatrix}$

Softmax Regression

Given $(x^{(i)}, y^{(i)}), i = 1, \ldots, m$, the log-likelihood of the Softmax model is

$$\ell(\theta) = \sum_{i=1}^{m} \log \underbrace{p(y^{(i)}|x^{(i)};\theta)}$$

$$= \sum_{i=1}^{m} \log \underbrace{\prod_{l=1}^{k} p(y^{(i)} = l|x^{(i)})}^{\mathbf{1}\{y^{(i)}=l\}}$$

$\mathbf{1}\{y^{(i)}=l\} = \begin{cases} 1 & y^i = l. \\ 0. & y^i \neq l. \end{cases}$

Softmax Regression

Given $(x^{(i)}, y^{(i)}), i = 1, \ldots, m$, the log-likelihood of the Softmax model is

$$\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{m} \log p(y^{(i)} | x^{(i)}; \theta) \\
&= \sum_{i=1}^{m} \log \prod_{l=1}^{k} p(y^{(i)} = l | x^{(i)})^{\mathbf{1}\{y^{(i)} = l\}} \\
&= \sum_{i=1}^{m} \sum_{l=1}^{k} \mathbf{1}\{y^{(i)} = l\} \log p(y^{(i)} = l | x^{(i)})
\end{aligned}$$

Softmax Regression

Given $(x^{(i)}, y^{(i)}), i = 1, \ldots, m$, the log-likelihood of the Softmax model is

$$\begin{aligned}
\ell(\theta) &= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) \\
&= \sum_{i=1}^{m} \log \prod_{l=1}^{k} p(y^{(i)} = l|x^{(i)})^{\mathbf{1}\{y^{(i)}=l\}} \\
&= \sum_{i=1}^{m} \sum_{l=1}^{k} \mathbf{1}\{y^{(i)} = l\} \log p(y^{(i)} = l|x^{(i)}) \\
&= \sum_{i=1}^{m} \sum_{l=1}^{k} \mathbf{1}\{y^{(i)} = l\} \log \frac{e^{\theta_l^T x^{(i)}}}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}
\end{aligned}$$

Softmax Regression

Derive the stochastic gradient descent update:

- Find $\nabla_{\theta_l} \ell(\theta)$

$$\nabla_{\theta_l} \ell(\theta) = \sum_{i=1}^{m} \left[\left(\mathbf{1}\{y^{(i)} = l\} - P\left(y^{(i)} = l | x^{(i)}; \theta \right) \right) x^{(i)} \right]$$

Property of Softmax Regression

$$P(y^i = l \mid x^i) = \frac{e^{\theta_l^T x^i}}{\sum_{j=1}^{k} e^{\theta_j^T x^i}}$$

- Parameters $\theta_1, \ldots \theta_k$ are not independent: $\sum_j p(y = j|x) = \sum_j \phi_j = 1$
- Knowning $k-1$ parameters completely determines model.

Invariant to scalar addition

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_k \end{bmatrix}, \quad \psi = \begin{bmatrix} \psi \\ \vdots \\ \psi \end{bmatrix}$$

$$p(y|x;\theta) = p(y|x;\theta - \psi)$$

Proof. $$p(y=l|x;\theta-\psi) = \frac{e^{(\theta^l-\psi)x}}{\sum_{j=1}^{k} e^{(\theta_j-\psi)x}} = \frac{e^{\theta_l^T x}\cdot(e^{-\psi x})}{\sum_{j=1}^{k} e^{\theta_j^T x}\cdot(e^{-\psi x})} = \frac{e^{-\psi x}}{e^{-\psi x}} \cdot \underbrace{\frac{e^{\theta_l^T x}}{\sum_{j=1}^{k} e^{\theta_j^T x}}}_{P(y^i=l|x^i)}$$

Relationship with Logistic Regression

When K = 2,

$$h_\theta(x) = \frac{1}{e^{\theta_1^T x} + e^{\theta_2^T x}} \begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \end{bmatrix}$$

Relationship with Logistic Regression

When K = 2,

$$h_\theta(x) = \frac{1}{e^{\theta_1^T x} + e^{\theta_2^T x}} \begin{bmatrix} e^{\theta_1^T x} \\ e^{\theta_2^T x} \end{bmatrix}$$

Replace $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ with $\theta - \begin{bmatrix} \theta_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \theta_1 - \theta_2 \\ 0 \end{bmatrix}$, $= \tilde{\theta}$ since $p(y|x;\theta)$ is invariant to scalar addition,

$$h_\theta(x) = \frac{1}{e^{\theta_1^T - \theta_2^T x} + e^{0x}} \begin{bmatrix} e^{(\theta_1 - \theta_2)^T x} \\ e^{0^T x} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{e^{(\theta_1-\theta_2)^T x}}{1+e^{(\theta_1-\theta_2)^T x}} \\ \frac{1}{1+e^{(\theta_1-\theta_2)^T x}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{1+e^{-(\theta_1-\theta_2)^T x}} \\ 1 - \frac{1}{1+e^{-(\theta_1-\theta_2)^T x}} \end{bmatrix} = \begin{bmatrix} g(\tilde{\theta}^T x) \\ 1 - g(\tilde{\theta}^T x) \end{bmatrix}$$

When to use Softmax?

OH : 6:30 - 8:30

Rm 1108A.

- When classes are mutually exclusive: use Softmax
- Not mutually exclusive: multiple binary classifiers may be better

multi-label classification

↑ when labels are not mutually exclusive.