Learning From Data
Lecture 2: Linear Regression & Logistic Regression

Yang Li yangli@sz.tsinghua.edu.cn

September 24, 2021

Today's Lecture

Supervised Learning (Part I)

- Linear Regression
- Binary Classification
- Multi-Class Classification

Review: Supervised Learning

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

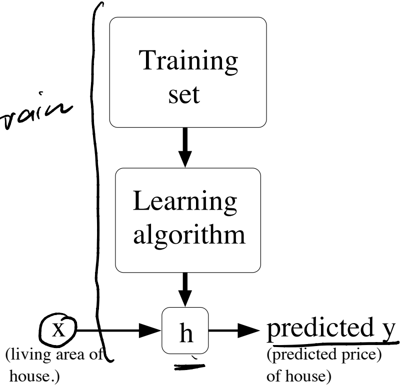
$\mathbb{R}^d$

Review: Supervised Learning

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

train

test



Review: Supervised Learning

$\mathcal{Y}$ – target space

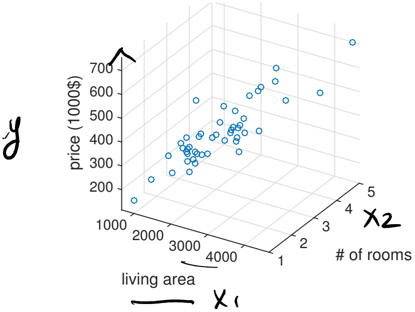
- y is discrete (categorical): **classification problem**
- y is continuous (real value): **regression problem**

y is continuous

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

input space   output space

$\mathcal{X} = \mathbb{R}^2 \quad \mathcal{Y} = \mathbb{R} \qquad h: \mathcal{X} \to \mathcal{Y}$

$y \in \mathcal{Y}$

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

$$h(x) = \theta_0^{\cdot\,1} + \theta_1 x_1 + \theta_2 x_2$$

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

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

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

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

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])
```

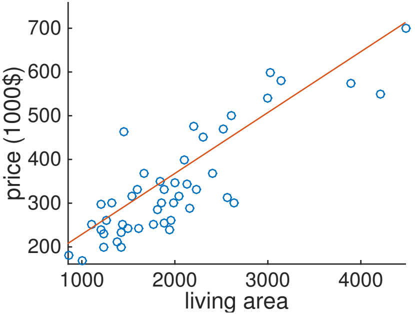
θ_0 θ_1 θ_2

make a prediction of new feature x:

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

evaluating the model

→ prediction



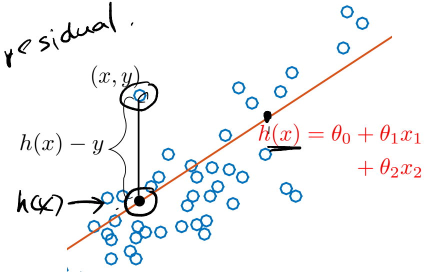
Model Estimation

How to estimate model parameters θ (or $\underset{\frown}{w}$ and $\underset{\sim}{b}$) from data?

Model Estimation

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

Least Square Estimation

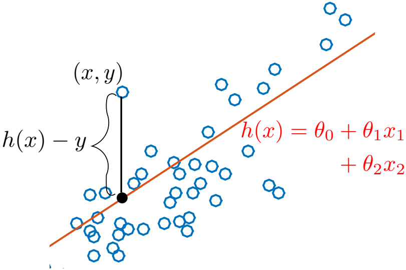


geometric approach

Model Estimation

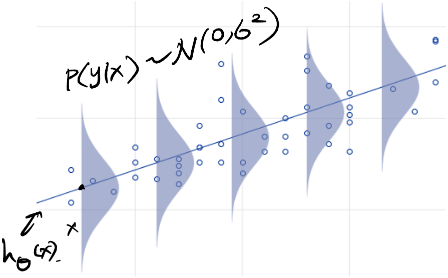
How to estimate model parameters θ (or w and b) from data?

Least Square Estimation



geometric approach

Maximum Likelihood Estimation



Probabilistic approach

Ordinary Least Square

of training samples

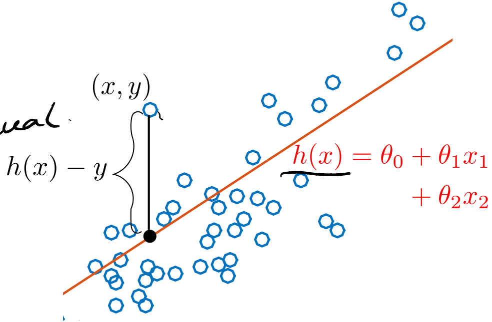
true label

Cost function:

prediction

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

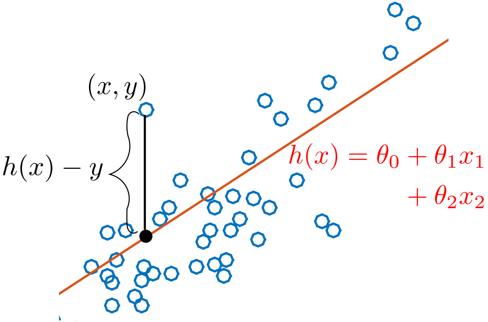
residual



Ordinary Least Square

Cost function:

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



Ordinary Least Square

Cost function:

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

Ordinary Least Square

$\theta^T x^{(i)}$

Cost function:

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

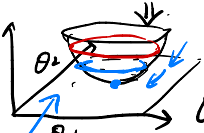
The **ordinary Least square problem** is:

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

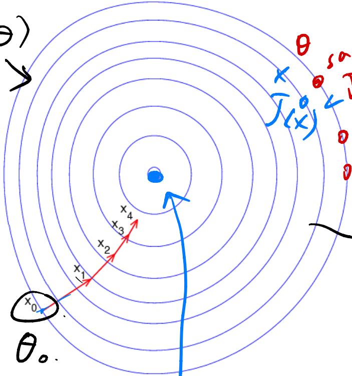
How to minimize $J(\theta)$?

- 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 $J(\theta)$.



$J(\theta)$

θ_0

level set plot

$\min_{\theta} J(\theta)$

θ same $J(\theta) = J(\theta')$

θ'

$J(x)$

Key idea

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

gradient

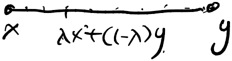
$$\theta := \theta - \alpha \nabla J(\theta)$$

α is the **learning rate**

0.1

Review: Convex function

Vector space S, a subset $C \subseteq S$ is convex if for any $x, y \in C$, its $\overline{xy} \subseteq C$. $\Rightarrow$ for any $0 \le \lambda \le 1$, affine combination $\lambda x + (1-\lambda) y \in C$.



Definition

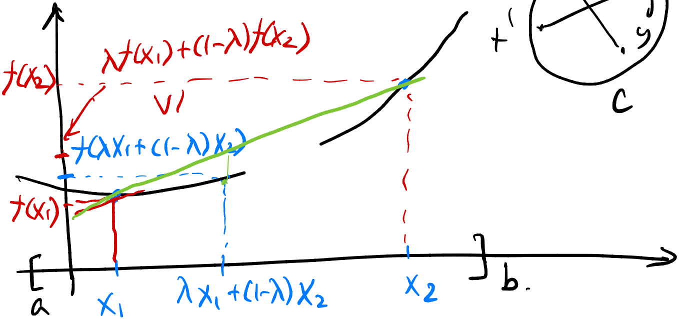
A function $f(x)$ is **convex** on a convex set C if for any $x_1, x_2 \in C$ and $0 \le \lambda \le 1$,

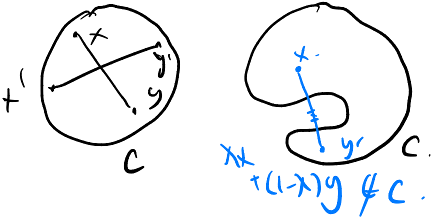
$f: C \rightarrow \mathbb{R}$

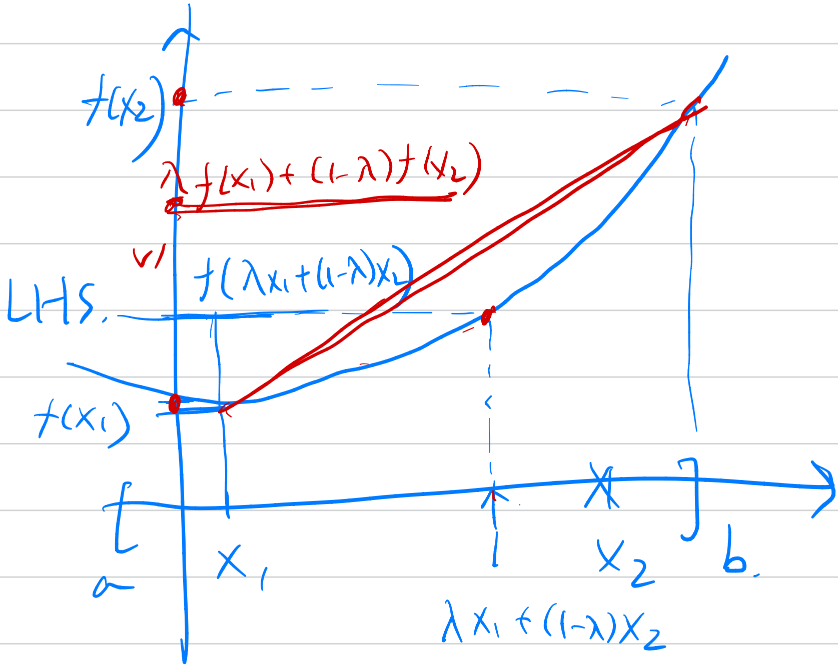
$$f(\lambda x_1 + (1-\lambda)x_2) \le \lambda f(x_1) + (1-\lambda) f(x_2)$$

$\lambda = 0.6$

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







0.6

Review: Convex function

not defined →

$f(x): C \to \mathbb{R}$

$x \in C$

Definition

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

$\in C$

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

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,

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

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

$$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \leftarrow \text{\# features.}$$

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

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

$$\frac{\partial}{\partial \theta_j}\sum_{k=1}^{n}\theta_k x_k^{(i)} = \sum_{k=1}^{n}\frac{\partial}{\partial \theta_j}\theta_k x_k^{(i)}$$

$$= \frac{\partial}{\partial \theta_j}\theta_j x_j^{(i)}$$

$$= x_j^{(i)}$$

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

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

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

Gradient descent for ordinary least square

$\theta \leftarrow \theta - \alpha \nabla J(\theta)$

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

Gradient descent update: $\theta := \theta - \alpha \underbrace{\nabla J(\theta)}$ $= \begin{bmatrix} \nabla J(\theta)_1 \\ \nabla J(\theta)_2 \\ \vdots \end{bmatrix}$

$\begin{bmatrix} \theta_1 \\ \vdots \\ \theta_j \end{bmatrix}$

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
}
```

while not converged :

for j in range(n) :

$\theta_j = \ldots\ldots$

Gradient descent for ordinary least square

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

Batch Gradient Descent

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

$O(m)$

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

Batch gradient descent

for every j

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

for every $i=1,\dots m$

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

θ_j.

Stochastic gradient descent

$\theta = [\theta_1, \cdots, \theta_n]$.

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

mini-batch

m

Stochastic gradient descent

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

θ is updated each time a training example is read

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

Minimize $J(\theta)$ Analytically

The matrix notation

$$\begin{bmatrix} x^{(i)} \end{bmatrix} \qquad\qquad (m \times 1).$$

$$X = \begin{array}{c} \\ m \end{array} \left\{ \begin{bmatrix} \overbrace{\text{—}\ \left(x^{(1)}\right)^T\ \text{—}}^{n} \\ \text{—}\ \left(x^{(2)}\right)^T\ \text{—} \\ \vdots \\ \text{—}\ \left(x^{(m)}\right)^T\ \text{—} \end{bmatrix} \right., \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

X is called the **design matrix**.

Minimize $J(\theta)$ Analytically

$$z = X\theta - y = \begin{bmatrix} \theta^T x^{(1)} \\ \theta^T x^{(2)} \\ \theta^T x^{(3)} \\ \vdots \\ \theta^T x^{(m)} \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} = \begin{bmatrix} \theta^T x^{(1)} - y^{(1)} \\ \vdots \\ \theta^T x^{(i)} - y^{(i)} \\ \vdots \\ \theta^T x^{(m)} - y^{(m)} \end{bmatrix}$$

The matrix notation

$$z^T z = \sum_{i=1}^{m} z_i^2 \qquad z \in \mathbb{R}^m$$

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

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

$$\boxed{\frac{1}{2}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)^2}$$

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

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

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

$f(x) = \frac{1}{2}(Qx - y)^T(Qx - y)^T$

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

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

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

Hint: (let $x \leftarrow \theta$, $Q \leftarrow X$)

$X \in \mathbb{R}^{m \times n}$

$x, y \in \mathbb{R}^n$, $\quad Q \in \mathbb{R}^{n \times n}$. $\quad \ell : \mathbb{R}^n \to \mathbb{R}$.

$$\ell(x) = \frac{1}{2}(Qx - y)^T(Qx - y)$$

$$= \frac{1}{2}(x^T Q^T - y^T)(Qx - y)$$

$$= \frac{1}{2}(x^T Q^T Q x - y^T Q x - x^T Q^T y + y^T y)$$

$$= \frac{1}{2} x^T Q^T Q x - y^T Q \cdot x + \frac{1}{2} y^T y$$

$$\nabla_x \ell(x) = \frac{1}{2} \cdot \frac{\partial (x^T Q^T Q x)}{\partial x} - \frac{\partial (y^T Q \cdot x)}{\partial x} + 0.$$

$$= \frac{1}{2} \cdot \left(Q^T Q + (Q^T Q)^T\right) \cdot x - Q^T y$$

$$= Q^T Q \cdot x - Q^T y$$

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

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

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

quadratic . form

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

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

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

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

$$\nabla_\theta J(\theta) = \nabla_\theta \left[\frac{1}{2}(X\theta - y)^T(X\theta - y)\right]$$
$$= X^T X\theta - X^T y = 0.$$

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

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

The Normal equation

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

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 $\nabla J(\theta) = 0$.

$X \in \mathbb{R}^{m\times m} \rightarrow \theta = X^{-1}y$

The Normal equation

$X \in \mathbb{R}^{m\times n} \rightarrow \underline{(X^TX)^{-1}X^T}\, y$

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

$(X^TX)^{-1}X^T$ 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 steepest GD.	

Which method to use?

X: $(m \times n)$ $n \times n$.

$(X^TX)^{-1}$

Avoid numerical issue: add small purterbation: $(X^TX + \lambda I)^{-1}$ $\begin{bmatrix} \lambda & & \\ & \lambda & \\ & & \lambda \end{bmatrix}$

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

$f(\theta)$

Numerically solve for θ in $\boxed{\nabla_\theta J(\theta) = 0}$

Newton's method

Solves real functions $f(x) = 0$ by iterative approximation:

- Start an initial guess x
- Update x until convergence

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

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

$x_n \leftarrow x$ at nth iteration

Geometric intuition of Newton's method

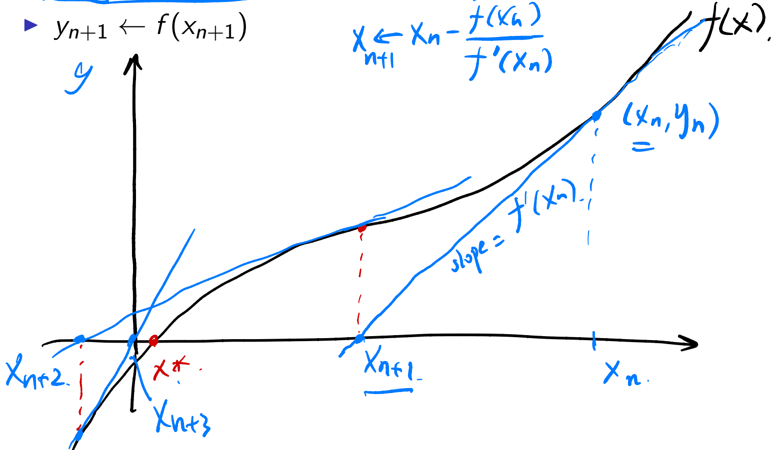
At step $n+1$:

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

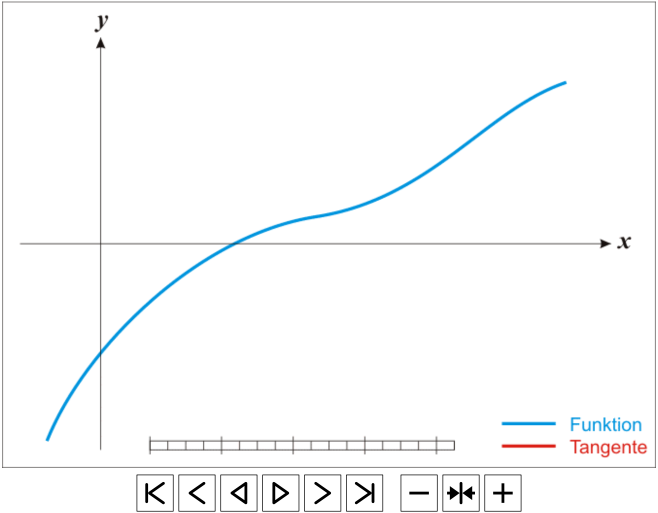
$$y = f'(x)(x - x_n) + f(x_n)$$

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

$$x_{n+1} \leftarrow x_n - \frac{f(x_n)}{f'(x_n)}$$



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 $\nabla_\theta J(\theta) = 0$:

- θ is one-dimensional:

$\frac{\partial}{\partial \theta} J(\theta)$

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

$\frac{\partial^2}{\partial \theta^2} J(\theta)$

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

Hessian matrix.

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

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

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

- θ is one-dimensional:

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

- $\cancel{x}$ θ n- is multidimensional:

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

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)
}
```

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

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:

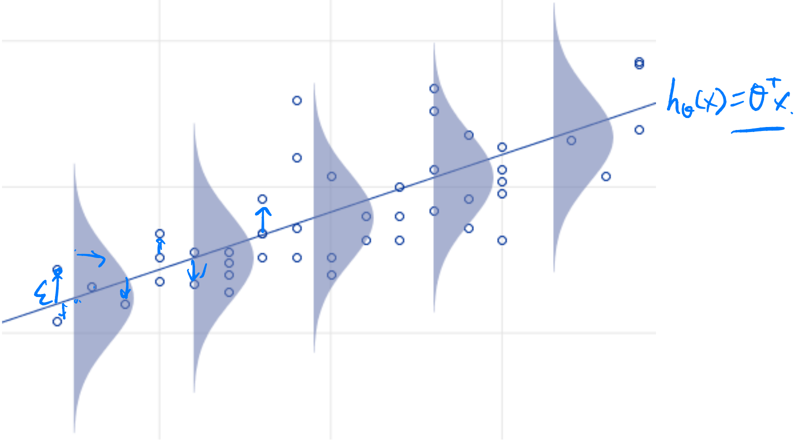
- Needs fewer interations than batch gradient descent
- Computing H^{-1} is time consuming
- Faster in practice when n is small

Maximum Likelihood Estimation

Consider target y is modeled as

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

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



Maximum Likelihood Estimation

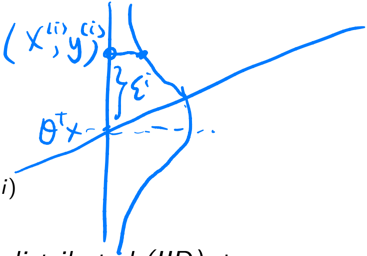
Consider target y is modeled as

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

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

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

$e^{\left(-\frac{\varepsilon^{(i)^2}}{2\sigma^2}\right)}$

Maximum Likelihood Estimation

Consider target y is modeled as

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

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

$\varepsilon^{(i)} = y^{(i)} - \theta^T x^{(i)}$

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

$x^{(i)}, y^{(i)}$

$\|$

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

Maximum Likelihood Estimation

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

$$L(\theta) = \underbrace{p(\overset{\text{label}}{\vec{y}}|\overset{\text{input.}}{X};\theta)} = \prod_{i=1}^{m} \underbrace{p(y^{(i)}|x^{(i)};\theta)}$$ due to. i.i.d.

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:

$$\theta_{MLE} = \operatorname*{argmax}_{\theta} L(\theta)$$

Maximum Likelihood Estimation

We compute log likelihood,

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

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

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

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

$$\min_{\theta} J(\theta)$$

Maximum Likelihood Estimation

We compute log likelihood,

$$\begin{aligned}\log L(\theta) &= \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) \\ &= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y^{(i)} - \theta^T x)^2}{2\sigma^2}\right)\end{aligned}$$

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

log-likelihood

i.i.d., $\varepsilon^i \sim N(0, \sigma^2)$

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

gradient ascent $\theta = \theta + \alpha \nabla \ell(\theta)$

Linear Regression Summary

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

- Least square regression (geometry approach)
- Maximum likelihood estimation (probabilistic modeling approach)

Linear Regression Summary

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

- Least square regression (geometry approach)
- Maximum likelihood estimation (probabilistic modeling approach)

Other estimation methods exist, e.g. Bayesian estimation

Linear Regression Summary

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

- Least square regression (geometry approach)
- Maximum likelihood estimation (probabilistic modeling approach)

Other estimation methods exist, e.g. Bayesian estimation

How to solve for solutions ?

- normal equation (close-form solution)
- gradient descent
- newton's method

iterative

Logistic Regression

A binary classification problem

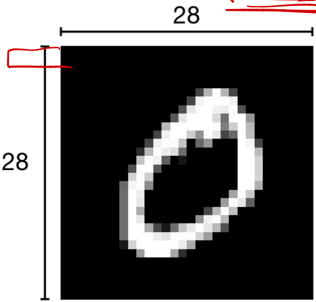
Classify binary digits

- 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]

(255, 255, 255) →

(0-255)



28×28

784

row 1

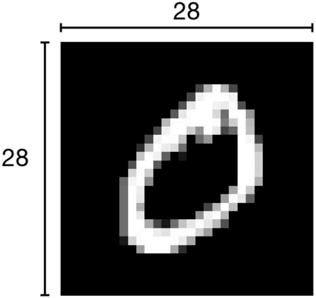
row 2

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

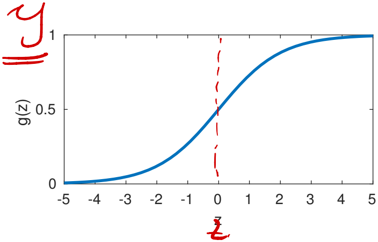
- negative class: $y^{(i)} = 0$
- positive class: $y^{(i)} = 1$

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))

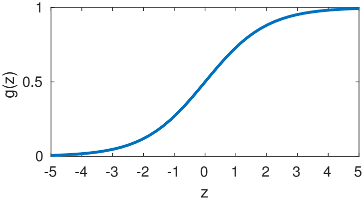


Logistic Regression Hypothesis Function

Sigmoid function

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

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

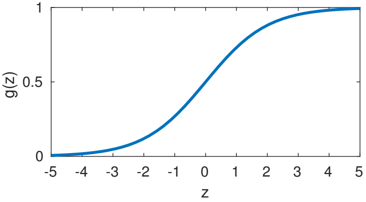


Logistic Regression Hypothesis Function

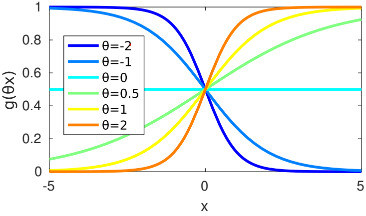
Sigmoid function

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

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



10, θ



Hypothesis function for logistic regression:

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

Review: Bernoulli Distribution

A discrete probability distribution of a binary random variable $x \in \{0, 1\}$:

$$p(x) = \begin{cases} \lambda & \text{if } x = 1 \\ 1 - \lambda & \text{if } x = 0 \end{cases}$$

$$= \cancel{p^x(1-p)^{1-x}} = \lambda^x(1-\lambda)^{1-x}$$

biased.

p(head) = λ

p(tail) = $1-\lambda$

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**. $\lambda = h_\theta(x)$.

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

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**.

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

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

Maximum likelihood estimation for logistic regression

Logistic regression assumes $y|x$ is **Bernoulli distributed**.

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

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

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

log

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

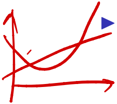
$= h_\theta(x)^y (1 - h_\theta(x))^{1-y}$.

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

Maximum likelihood estimation for logistic regression

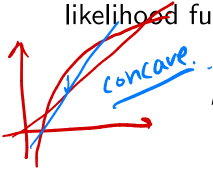
Logistic regression assumes $y|x$ is **Bernoulli distributed**.

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



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

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

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

$\frac{\partial}{\partial \theta_j}(1 - h_\theta(x^{(i)}))$

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

$$\nabla_\theta l(\theta) = \begin{bmatrix} \vdots \\ \frac{\partial}{\partial \theta_j} l(\theta) \\ \vdots \end{bmatrix}$$

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

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

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

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

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

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

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

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

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

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:

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

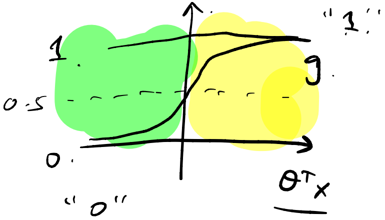
Stocastic Gradient Ascent

```
Repeat until convergence{
  for i = 1...m {
    θj = θj + α(y^(i) − hθ(x^(i)))xj^(i)   for every j=1,..,n.
  }
}
```

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

- 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

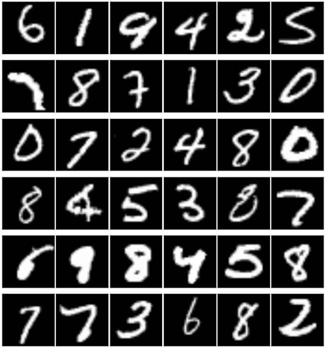
- Multiple Binary Classifiers
- Softmax Regression

Multi-class classification

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

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

MNIST Samples



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

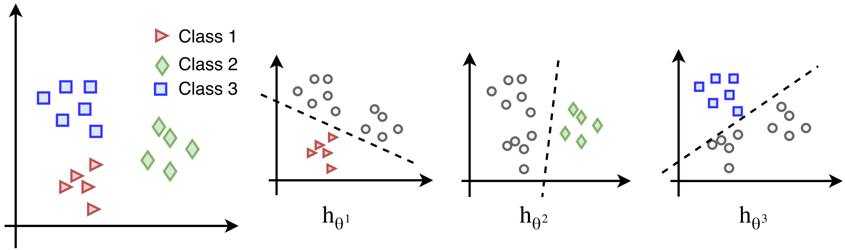
Naive Approach: Convert to binary classification

One-Vs-Rest

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

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

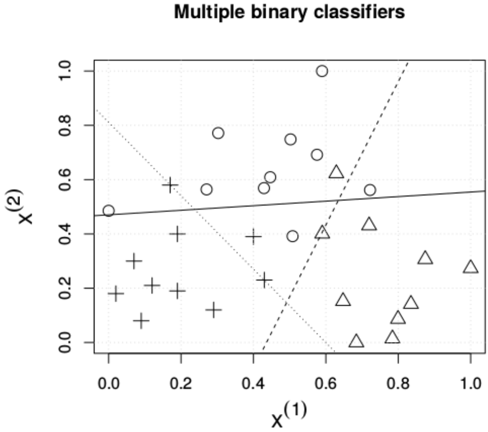
$$\hat{y} = \underset{i}{\operatorname{argmax}}\, h_i(x)$$



Multiple binary classifiers

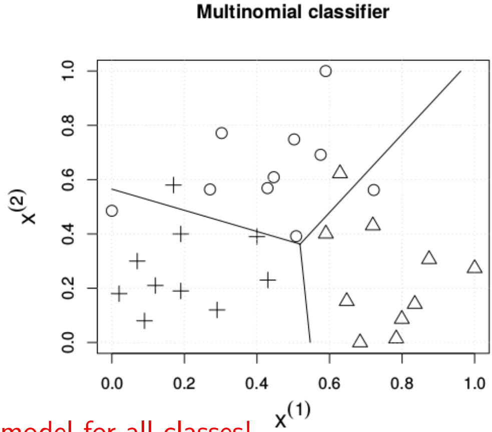
Drawbacks of One-Vs-Rest:

- 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



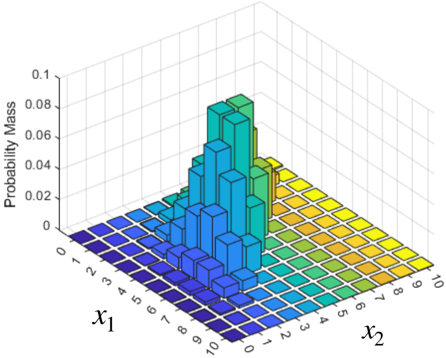
Learn one model for all classes!

Review: Multinomial Distribution

Models the probability of counts for each side of a k-sided die rolled m times, each side with independent probability ϕ_i

$$\phi_1 + \cdots + \phi_k = 1$$

$$k = 3,\ n = 10 \qquad \phi = \left[\frac{1}{2}, \frac{1}{3}, \frac{1}{6}\right]$$



Extend logistic regression: Softmax Regression

Assume $p(y|x)$ is **multinomial distributed**, $k = |\mathcal{Y}|$

Extend logistic regression: Softmax Regression

Assume $p(y|x)$ is **multinomial distributed**, $k = |\mathcal{Y}|$

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**, $k = |\mathcal{Y}|$

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

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

Softmax Regression

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

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

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

- 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

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

Proof.

Relationship with Logistic Regression

When $\mathrm{K} = 2$,

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

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 $\theta * = \theta - \begin{bmatrix} \theta_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \theta_1 - \theta_2 \\ 0 \end{bmatrix}$,

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

When to use Softmax?

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