logistic regression model

cost function

given a training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

with m examples denoted as $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$; where $x_0 = 1, y \in \{0,1\}$

and a hypothesis: $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

the parameters θ can be determined as follows:

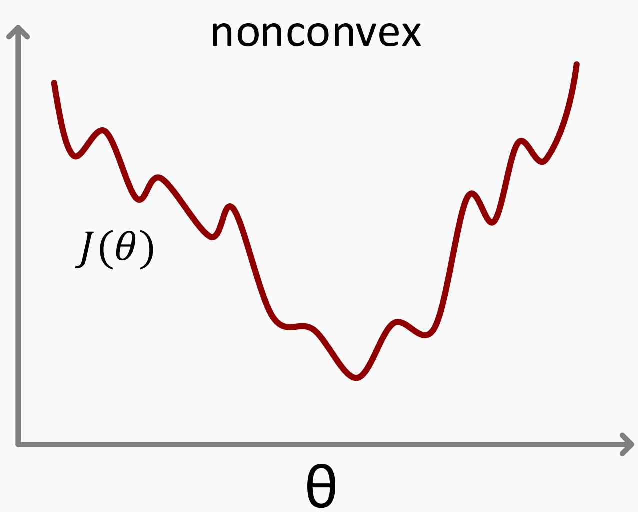
linear regression:

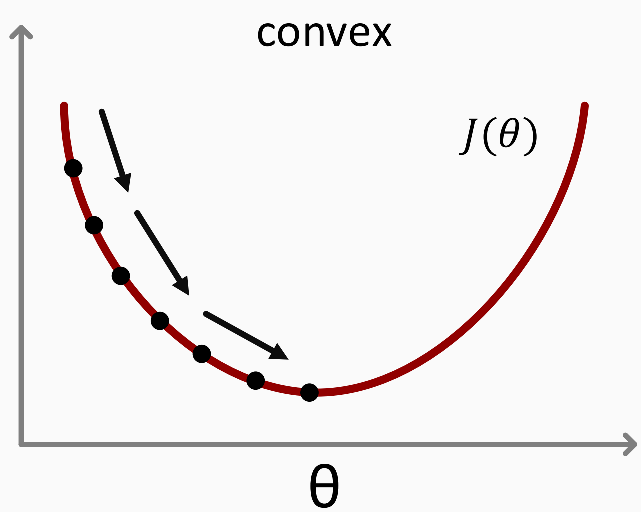
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2} \rightarrow \mathrm{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$\mathrm{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

the simplified cost functions serves purpose as to measure the cost the training set will have to pay when $h_\theta(x^{(i)})$ is predicted in terms of actual $y^{(i)}$: being $\frac{1}{2}$ the squared error

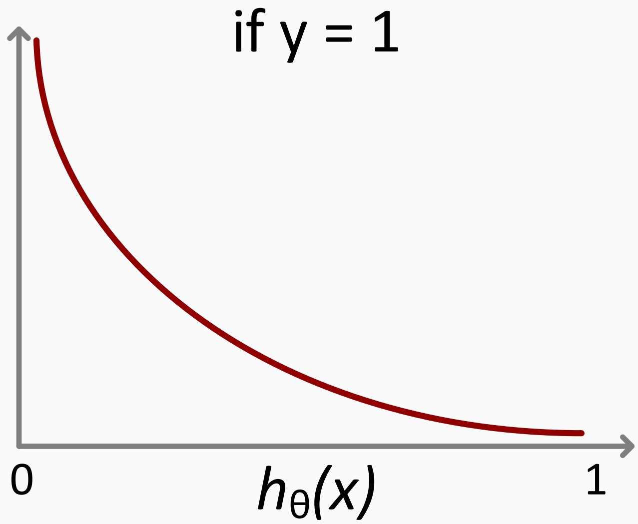
the Linear Regression Cost Function cannot be applied to logistic regression as it would not create a Convex Function; Gradient Descent would fail to find the global minimum:





instead, logistic regression: $\mathrm{cost}\left(h_\theta(x^{(i)}), y^{(i)}\right) = \begin{cases} -\log\left(h_\theta(x^{(i)})\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x^{(i)})\right) & \text{if } y = 0 \end{cases}$

the cost function is illustrated in each class as follows:

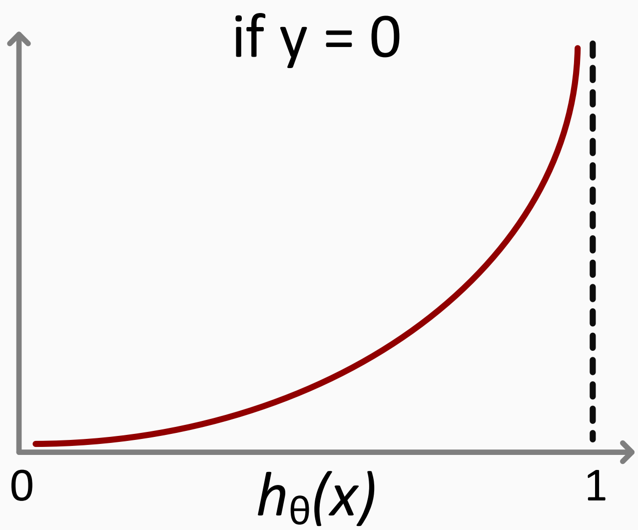


cost $= 0$ if $y = 1, h_\theta(x) = 1$

but as $h_\theta(x) \to 0$

cost $\to \infty$

the illustration (left) captures the intuition if $h_\theta(x^{(i)}) = 0$, (predict $P(y = 1|x:\theta) = 0$), but $y = 1$, the learning algorithm will be heavily penalized with a significant measured cost



cost $= 0$ if $y = 0, h_\theta(x) = 0$

but as $h_\theta(x) \to 1$

cost $\to \infty$

the illustration (left) captures the intuition if $h_\theta(x^{(i)}) = 1$, (predict $P(y = 0|x:\theta) = 0$), but $y = 0$, the learning algorithm will be heavily penalized with a significant measured cost

In logistic regression, the cost function for our hypothesis outputting (predicting) $h_\theta(x)$ on a training example that has label $y \in \{0, 1\}$ is:

$$\text{cost}(h_\theta(x), y) = \begin{cases} -\log h_\theta(x) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Which of the following are true? Check all that apply.

- [x] If $h_\theta(x) = y$, then $\text{cost}(h_\theta(x), y) = 0$ (for $y = 0$ and $y = 1$).

Correct Response

- [x] If $y = 0$, then $\text{cost}(h_\theta(x), y) \to \infty$ as $h_\theta(x) \to 1$.

Correct Response

- [ ] If $y = 0$, then $\text{cost}(h_\theta(x), y) \to \infty$ as $h_\theta(x) \to 0$.

Correct Response

- [x] Regardless of whether $y = 0$ or $y = 1$, if $h_\theta(x) = 0.5$, then $\text{cost}(h_\theta(x), y) > 0$.

Correct Response

simplified cost function and gradient descent

logistic regression:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$\mathrm{cost}(h_\theta(x), y) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x)\right) & \text{if } y = 0 \end{cases}$$

note: $y = 0$ or $y = 1$ always

explanation of simplifying the cost function $\mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$ above:

$$\mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = -y\log\left(h_\theta(x)\right) - (1-y)\log\left(1 - h_\theta(x)\right)$$

if **y=1**: $\mathrm{cost}(h_\theta(x), y) = -\log\left(h_\theta(x)\right)$

if **y=0**: $\mathrm{cost}(h_\theta(x), y) = -\log\left(1 - h_\theta(x)\right)$

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{cost}\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

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

principle of likelihood maximization derived function (statistical rational behind cost function used)

to fit parameters θ:

the process of minimizing $J(\theta)$; $\min_\theta J(\theta)$ will provide the parameters θ

to make a prediction given a new x training example:

output $h_\theta(x) = \frac{1}{1+e^{-\theta^T X}}$; $P(y = 1|x\!:\theta)$

in order to minimize $J(\theta)$; $\min_\theta J(\theta)$ (gradient descent)

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

updating all values of parameter θ is best performed as a vectorized

repeat until convergence {

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

$\rightarrow$ (simultaneously update all θ_j)}

note the identical cosmetic similarity between the logistic and linear regression update rule is attributed to the altered definition of $h_\theta(x)$ from $h_\theta(x) = \theta^T X$ to $h_\theta(x) = \frac{1}{1+e^{-\theta^T X}}$

Suppose you are running gradient descent to fit a logistic regression model with parameter $\theta \in \mathbb{R}^{n+1}$. Which of the following is a reasonable way to make sure the learning rate α is set properly and that gradient descent is running correctly?

- Plot $J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$ as a function of the number of iterations (i.e. the horizontal axis is the iteration number) and make sure $J(\theta)$ is decreasing on every iteration.
- Plot $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$ as a function of the number of iterations and make sure $J(\theta)$ is decreasing on every iteration.

  Correct Response
- Plot $J(\theta)$ as a function of θ and make sure it is decreasing on every iteration.
- Plot $J(\theta)$ as a function of θ and make sure it is convex.

One iteration of gradient descent simultaneously performs these updates:

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

$$\vdots$$

$$\theta_n := \theta_n - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_n^{(i)}$$

We would like a vectorized implementation of the form $\theta := \theta - \alpha\delta$(for some vector $\delta \in \mathbb{R}^{n+1}$).

What should the vectorized implementation be?

- $\theta := \theta - \alpha\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}]$

  Correct Response
- $\theta := \theta - \alpha\frac{1}{m}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})] \cdot x^{(i)}$
- $\theta := \theta - \alpha\frac{1}{m}x^{(i)}[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})]$

advanced optimization

optimization algorithm

cost function $J(\theta)$ wanting to $\min_\theta J(\theta)$

given θ, code is written to compute

$$J(\theta) \text{ and } a\frac{\partial}{\partial\theta_j}J(\theta)$$

(for $j = 0, 1, \ldots, n$)

gradient descent:

repeat {

$$\theta_j := \theta_j - a\frac{\partial}{\partial\theta_j}J(\theta)$$

}

optimization algorithms options:

- gradient descent
- conjugate gradient
- bfgs
- l-bfgs

advantages:

- unnecessary to select α manually
- often faster than gradient descent

disadvantages:

- generally more complex

optimization through example:

goal to $\min_{\theta} J(\theta)$ assuming $\theta_1 = 5,\ \theta_2 = 5$:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = 2(\theta_2 - 5)$$

optimization in octave:

```
function [jVal, gradient]
          = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
        (theta(2)-5)^2
  gradient = zeros(2,1)
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

after implementing the cost function, call the advanced optimization function in octave:

```
options = optimset('GrabObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag]...
      = fminunc(@costFunction, initialTheta, options);
```

`'GrabObj'` in reference to gradient objective set to `'on'` in reference to the fact that a gradient is going to be provided to the algorithm. `initialTheta` initializes the parameters $\boldsymbol{\theta}$ are at $\mathbf{0}$. the advanced function `fminunc` is called to compute `optTheta`, the learning rate $\boldsymbol{\alpha}$ autonomously:

```
octave-3.2.4.exe:1> PS1('>> ')
>> cd 'C:\Users\ang\Desktop'
>>
>> options = optimset('GradObj','on', 'MaxIter', '100');
>> initialTheta = zeros(2,1)
initialTheta =

   0
   0

<ag] = fminunc(@costFunction, initialTheta, options)
```

```
<ag] = fminunc(@costFunction, initialTheta, options)
optTheta =

   5.0000
   5.0000

functionVal = 1.5777e-030
exitFlag =  1
>>
```

application of algorithm optimization to logistic regression

$$\text{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \leftarrow \text{theta(1)} \\ \leftarrow \text{theta(2)} \\ \vdots \\ \leftarrow \text{theta(n)} \end{matrix}$$

note the definition of theta being indexed at 1 in octave as opposed to 0 in the

```
function [jVal, gradient] = costFunction(theta)
    jVal =            [code to compute J(θ)];
    gradient(1) =     [code to compute ∂/∂θ_0 J(θ)];
    gradient(2) =     [code to compute ∂/∂θ_1 J(θ)];
    ⋮
    gradient(n+1) =   [code to compute ∂/∂θ_n J(θ)];
```

the above code requires a user to input code to compute both a cost function and relative gradients

Suppose you want to use an advanced optimization algorithm to minimize the cost function for logistic regression with parameters θ_0 and θ_1. You write the following code:

```
function [jVal, gradient] = costFunction(theta)
  jVal = % code to compute J(theta)
  gradient(1) = CODE#1 % derivative for theta_0
  gradient(2) = CODE#2 % derivative for theta_1
```

What should CODE#1 and CODE#2 above compute?

- CODE#1 and CODE#2 should compute $J(\theta)$.
- CODE#1 should be theta(1) and CODE#2 should be theta(2).
- CODE#1 should compute $\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}] (= \frac{\partial}{\partial\theta_0} J(\theta))$ and

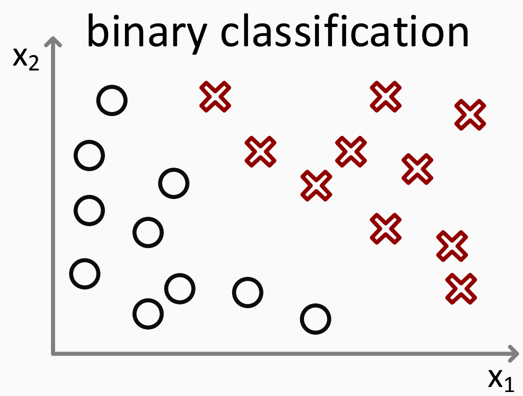
  CODE#2 should compute $\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}] (= \frac{\partial}{\partial\theta_1} J(\theta))$

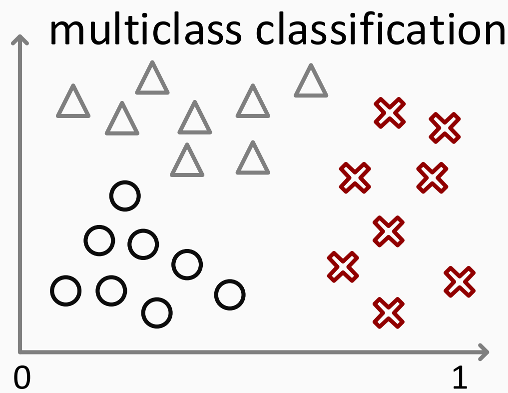
Correct Response

multiclass classification

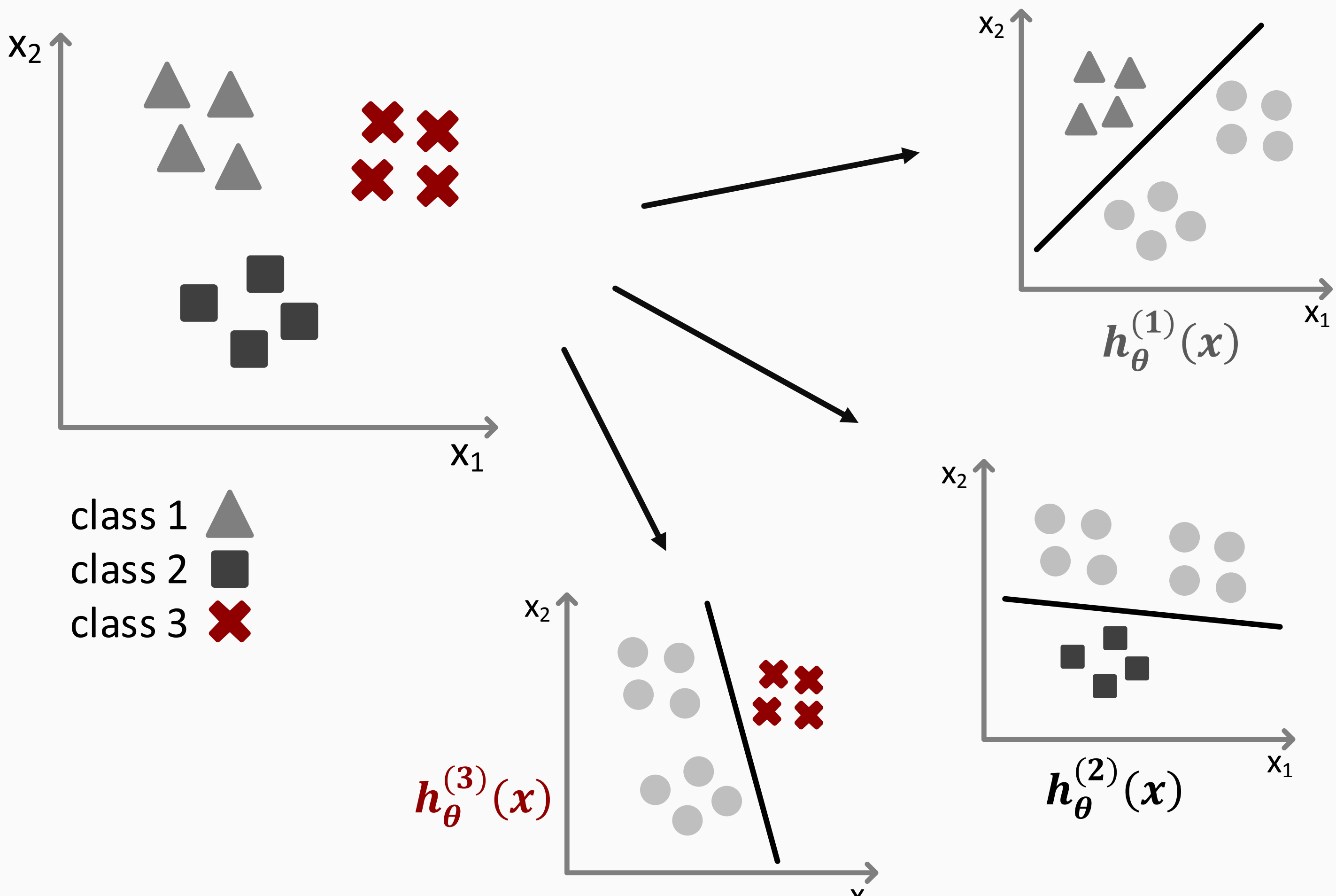
multiclass classification: one-vs-all

when classification problems have more than a binary classification of 0 or 1





this is possible by separating examples into individual binary classification problems



formally, multiclass classification trains a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i that maximizes $\max_i h_\theta^{(i)}(x)$

$$h_\theta^{(i)}(x) = P(y = i|x;\theta)$$

where $(i = 1,2,3)$