Classification

Exp:

Email : Spam / Not spam?

Online Transaction : Fraudulent (Yes/No)?

Tumor : Malignant / Benign?

binary classification

2 classes ⇒ $y \in \{0, 1\}$

0: "Negative Class" (e.g., benign tumor)

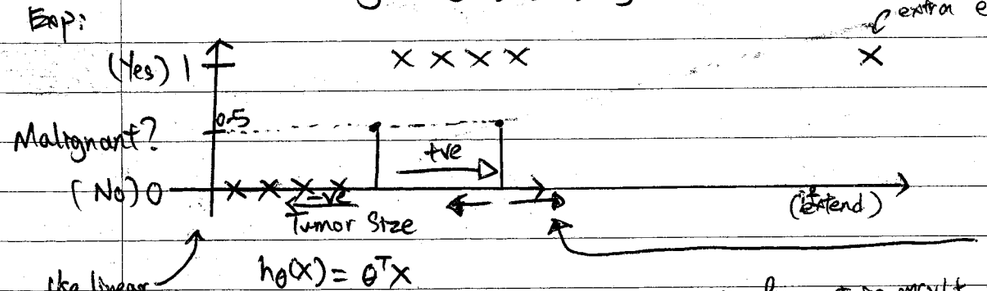
1: "Positive Class" (e.g., ~~malignant~~ malignant tumor)

absent of something

present of something

Multiclasses ⇒ $y \in \{0, 1, 2, 3\}$

Exp:



Use linear regression to draw out graph a "line of best fit", and plot threshold point to find out x.

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

▷ magnitude that must be exceeded for certain result

[Threshold] classifier output $h_\theta(x)$ at 0.5:

if $h_\theta(x) \geq 0.5$, predict "y=1"

if $h_\theta(x) < 0.5$, predict "y=0"

But when an extra example come out far away from training set, linear regression will come out with ~~a~~ different point and different threshold.

conclusion:

So apply linear regression on classification problem often not a good idea.

∴ Don't use linear regression on classification problem

Classification : $y = 0$ or 1

Linear regression $h_\theta(x)$ can be > 1 or < 0

Logistic Regression : $0 \leq h_\theta(x) \leq 1$

↳ classification algorithm that we apply to settings where label y is discrete value (1/0)

Logistic Regression

Hypothesis Representation

No.

Date

~~Things~~ Function we going to use to represent our hypothesis in classification problem.

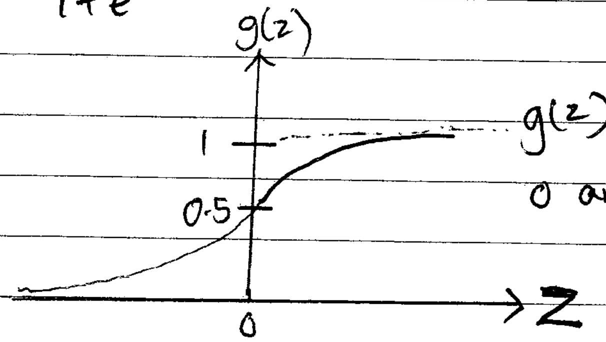
Logistic Regression Model

Want $0 \le h_\theta(x) \le 1$

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

$g(z) = \frac{1}{1+e^{-z}}$ z sigmoid function / logistic function

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



Interpretation of Hypothesis Output

$h_\theta(x)$ = estimated probability that $y=1$ on input x (new input)

tumor classification example:

If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\theta(x) = 0.7$ (patient with features x, probability $y=1$ is 0.7).

Tell patient that 70% chance of tumor being malignant (has features x)

2) $h_\theta(x) = p(y=1 | x;\theta)$ "probability that $y=1$, given x, parameterized by θ"

$y = 0$ or 1

$\Rightarrow P(y=0|x;\theta) + P(y=1|x;\theta) = 1$

$P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$

70% chance of tumor being malignant → 30% being benign

Logistic Regression

Decision Boundary

No.

Date

To understand when it makes predictions $y=1$ and when it makes prediction $y=0$

$z=0,\ e^0=1,\ g(z)=\frac{1}{2}$
$z\to\infty,\ e^{-\infty}\to 0,\ g(z)=1$
$z\to-\infty,\ e^{\infty}\to\infty,\ g(z)=0$

Logistic regression

$h_\theta(x) = g(\theta^T x) = P(y=1|x;\theta)$

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

Suppose predict "$y=1$" if $h_\theta(x) \ge 0.5$ (estimate probability of $y=1 \ge 0.5$)

$\theta^T x \ge 0$

predict "$y=0$" if $h_\theta(x) < 0.5$ (estimate probability of $y=1 < 0.5$)

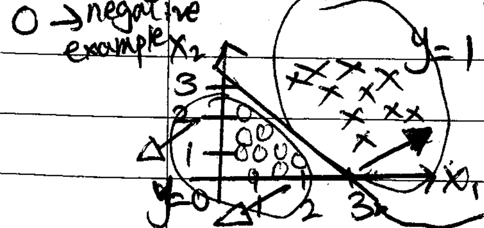
$\theta^T x < 0$

⇐ In the sigmoid function, $g(z) \ge 0.5$ when $z \ge 0$

$h_\theta(x) = g(\theta^T x) \ge 0.5$ whenever $\theta^T x \ge 0$ (z)

x → positive example

o → negative example



this line is called decision boundary

Let's say 2 examples, $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

use this try, y怎样最后 $=1 / =0$.

$h_\theta(x)=0.5$ — a straight line — $x_1+x_2=3$

Predict "$y=1$" if $(-3+x_1+x_2) \ge 0$ ($\theta^T x$) (用这个找出几时 $y=1$ 后, 再用来找 $y=0$)

$x_1+x_2 \ge 3$

[$y=1$] $x_1+x_2>3$ is upper side of straight line.

[$y=0$] $x_1+x_2<3$ is down side of s-line.

~~Des~~

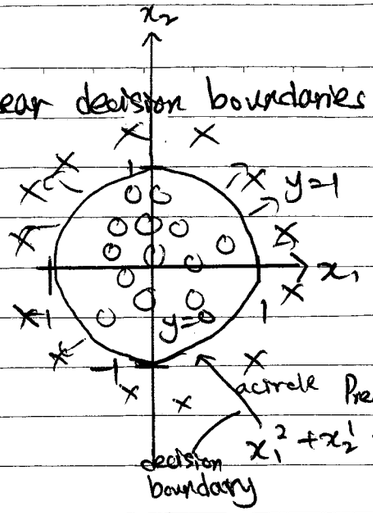
Decision boundary line = line that separate $y=1$ and $y=0$ region

We no need plot training set to plot decision boundary

No.

Date

Non-linear decision boundaries



x → positive example
O → negative example

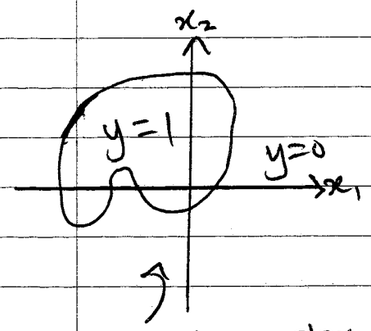
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$ $\theta = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

Predict "y=1" if $-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 \geq 1$

Decision boundary is a property, not of the training set but of the hypotheses under parameters.

(Training set) → not to define decision boundary; may be used to fit (parameter theta) → define decision boundary



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$

Could complex until like this, region in line is y=1 and outside is y=0.

Example: $\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$

$y = 1$ if $5 + (-1)x_1 + 0x_2 \geq 0$

$5 - x_1 \geq 0$ $-x_1 \geq -5$

$x_1 \leq 5$

DB is a vertical line on graph where $x_1 = 5$, everything to left is $y=1$ and to right denotes $y=0$.

→ (Higher polynomials) more Complex decision boundaries

(Higher order polynomial features)

Logistic Regression

Cost function

No.

Date

To fit parameters of theta for logistic regression.

Example of Supervised learning problem of fitting logistic regression model

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots (x^{(m)}, y^{(m)})\}$

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

$\mathbb{R}^{n+1}$ (n+1 dimensional)

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

How to choose parameters θ?

Cost Function

Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\rightarrow \text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

this works well in linear regression,

this learning algorithm have to pay if it outputs that value, if prediction is $h_\theta(x)$ and actual label is y.

What predicted

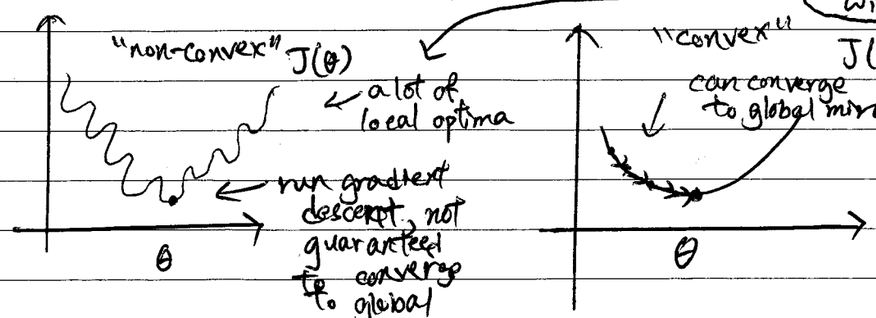
actual value we have, 0 for y.

For logistic regression $\frac{1}{1 + e^{-\theta^T x}}$ (complicated non-linear function)

But in logistic regression

It would be a non-convex function of parameters due to

if this plot plug in $J(\theta)$ and plot $J(\theta)$ will like this



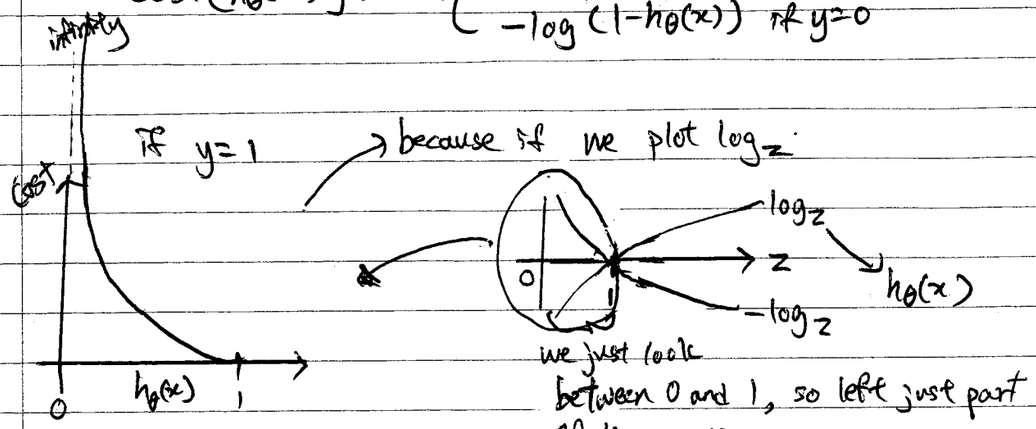
No.

Date

Logistic regression cost function

predict value ↓ , actual cost label ↓

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



if $y=1$ → because if we plot $\log_z$

we just look between 0 and 1, so left just part of the curve

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

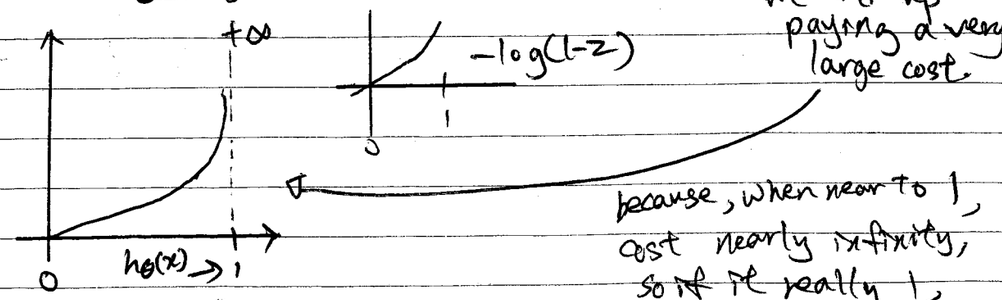
But as $h_\theta(x) \to 0$

$cost \to \infty$

chance of $y=1$ is 0. → captures intuition that if $h_\theta(x)=0$, (predict $P(y=1|x;\theta)=0$), but $y=1$, we'll penalize learning algorithm by a very large cost

[if $h_\theta(x)=y$, then $cost(h_\theta(x), y)=0$ (for $y=0$ and $y=1$)] ↓

if $y=0$, but we predict $y=1$ with almost certainty, probably 1, we end up paying a very large cost.



because, when near to 1, cost nearly infinity, so if it really 1, cost will be very large.

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

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

$Cost(h_\theta(x), y) \to \infty$ if $y=1$ and $h_\theta(x) \to 0$

- If correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, cost function will approach infinity.
- If correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function also outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.
- Writing cost function this way guarantees $J(\theta)$ is convex for logistic regression.

Logistic regression — Simplified cost function and gradient descent

cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

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

[Classification problem] Note: $y=0$ or 1 always

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

y: 0 → eqn = 0 ; $1-y$: 1−1=0 → whole eqn = 0 ; 1−0=1 → same

Only 2 possible cases, $y=1$ or $y=0$

If $y=1$, $cost(h_\theta(x), y) = -\log(h_\theta(x))$

If $y=0$, $cost(h_\theta(x), y) = \log(1-h_\theta(x))$

No.

Date

Logistic regression cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$

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

vectorized implementation is

$h = g(X\theta)$

$$J(\theta) = \frac{1}{m}\cdot\left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

To fit parameters θ:

$\min_\theta J(\theta)$ → Get θ → θ that fit our training set

To make a prediction given new x:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ ← and output our prediction as this

↳ output of hypothesis

interpret as $P(y=1 \mid x;\theta)$ estimating the probability that $y=1$

Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat { $\theta_j := \theta_j - \alpha\left[\frac{\partial}{\partial \theta_j} J(\theta)\right]$ }

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

so is like this

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

Repeat {

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

(simultaneously update all θ_j) ← simultaneously update all of your values of θ.

}

Algorithm looks identical to linear regression!

Different is that definition for this hypothesis has changed

Linear: $h_\theta(x) = \theta^T x$

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

→ So not the same thing as gradient descent for linear regression.

[Same method as I monitor linear regression to make sure it is converging, same with logistic regression.]

No.

Date

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 learning rate α is set properly and that gradient descent is running correctly?

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

Apply feature scaling can also make gradient descent run faster for logistic regression.

Advanced Optimization — algorithm concept → run faster than g.descent

↳ very large training set no problem

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$) ← to compute

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Use libraries instead of write out these sophisticated algorithm unless you are expert in numerical computing

Use a for loop to update, ideally, we use a ~~vector wise~~ implementation

vectorized implementation:

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

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

have a clever inner loop called line search algorithm that automatically tries out diff. value for α and pick good α, can also pick a diff. α for every iteration.

Advantages

- No need manually pick α
- Often faster than g.descent

Disadvantages:

- more complex

How to use algorithms in the previous page — as a function

Example:

$\theta = \begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}$ To min $J(\theta)$: $\theta_1 = 5, \theta_2 = 5$

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

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

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

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);
```

To apply one of advanced optimization algorithm to minimize cost function J. Use something like g descent but more advanced. Can implement octave function

(means you going to provide gradient to algorithm) — set gradient objective parameter to on — then called the advanced optimization function — maximum of 100 iteration

an options as a data structure stores options you want

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

(should be int, not string)

Give initial guess for θ

verify whether this algorithm has converged or not

fminunc: stand for function minimization unconstrained in Octave

@: symbol represent a pointer to the cost function that we defined up there and use it

$\theta \in \mathbb{R}^d$ $d \geq 2$ or fminunc can't work. (use "help" to check details)

```
theta = [theta(1); theta(2); ...; theta(n+1)];   % Octave indexes start from index of 1.
function [jVal, gradient] = costFunction(theta)   % function that return cost function and return the gradient
jVal = [code to compute J(theta)];
gradient(1) = [code to compute (d/d theta_0) J(theta)];
gradient(2) = [code to compute (d/d theta_1) J(theta)];
...
gradient(n+1) = [code to compute (d/d theta_n) J(theta)];
```

Compute θ_0 → Compute θ_1

if want to use this optimization to linear regression, just must put the appropriate code to compute in these.

Logistic Regression

Multi-class classification: One-vs-all / One-vs-rest (algorithm name)

When only 2 categories of data, then $y = \{0, 1\}$, now more than 2, then expand $y = \{0, 1 \ldots n\}$

Example:

Email folder/tagging: Work, Friends, Family, Hobby — $y=1$, $y=2$, $y=3$, $y=4$

Medical diagrams: Not ill, Cold, Flu — $y=1$, $y=2$, $y=3$

Weather: Sunny, Cloudy, Rain, Snow

$y \in \{0, 1 \ldots n\}$

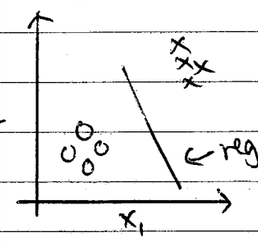
$h_\theta^{(0)}(x) = P(y=0 \mid x;\theta)$

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

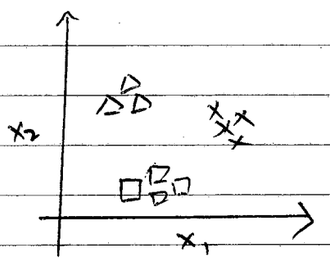
$h_\theta^{(n)}(x) = P(y=n \mid x;\theta)$

prediction $= \max(h_\theta^{(i)}(x))$

Binary classification:

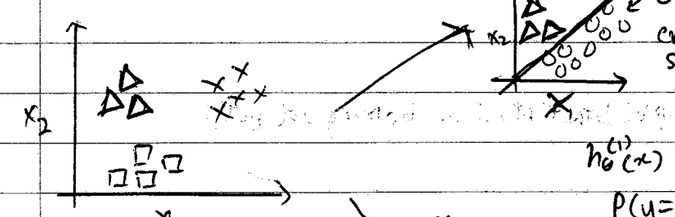


Multi-class classification:



Example:

One-vs-all (one-vs-rest):



One-vs-all classification to work on multi-class classification.

create a new sort of fake training set where class 2 and 3 assigned to -ve class and class 1 to +ve class.

do same thing to 2 and 3

$h_\theta^{(i)}(x) = P(y=i \mid x;\theta)$ $(i = 1, 2, 3)$

estimate probability that y = class i given x and parameterized by θ.

One-vs-all ⇒ Train logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict probability that $y = i$.

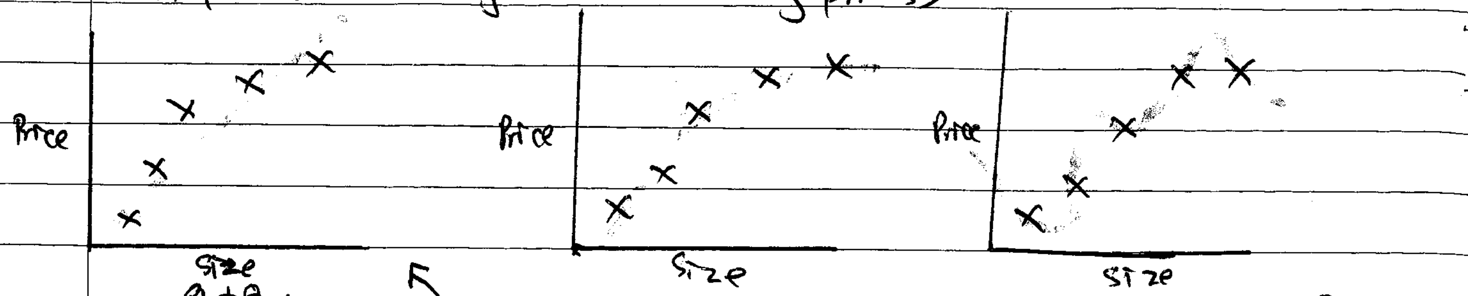
On a new input x, to make prediction, pick class i that maximizes $\max h_\theta^{(i)}(x)$ (value of i that gives us the highest probability)

↳ run 3 classifiers and pick class i that maximizes the three

Regularization

The problem of overfitting. → cause learning algorithm to perform ~~very~~ poorly

Example: Linear regression (housing prices)



$\theta_0+\theta_1 x$

If use linear regression, the housing price should move to right instead of keep up, so this algo, not very well suit training set.

Problem: "Underfitting", "High bias"

⇒ poor fit to data

$\theta_0+\theta_1 x+\theta_2 x^2$

⇒ Works pretty well

(Just right for fitting this data)

$\theta_0+\theta_1 x+\theta_2 x^2+\theta_3 x^3+\theta_4 x^4$

⇒ a curve that passes through all 5 examples, but very wiggly curve, not really a good model for predicting housing prices.

Problem: "Overfitting", "High variance"

if we fit such high order polynomial, hypothesis can fit almost any function, face of possible hypothesis too large, too variable and we don't have enough data to constrain it to get good hypothesis. and so called "overfitting"

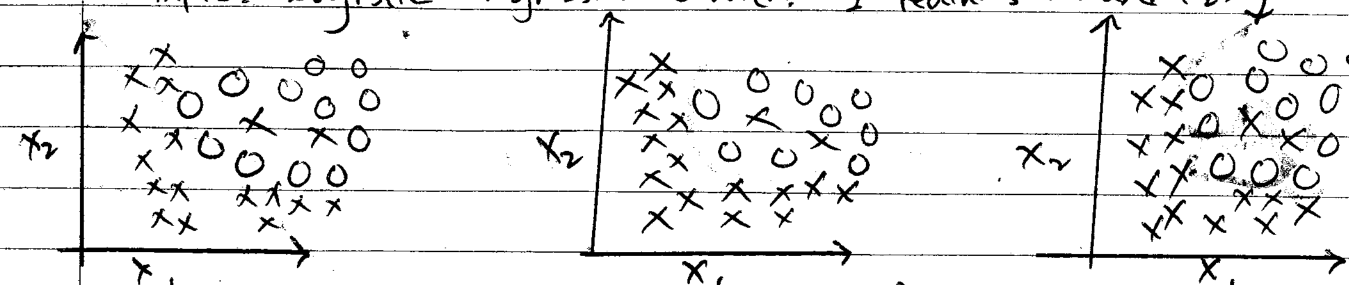
Overfitting: Problem of overfitting comes when if we have too many features, learned hypothesis will fit the training set very well

$(J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2\approx 0)$, but end up curve like this tries too hard to fit training set, in the end fail to generalize to new examples (predict prices on new examples).

generalize: how well a hypothesis applies even to new example (data that not in training set yet)

Example: Logistic regression (with 2 features x1 and x2)

And if to predict true or -ve, this boundary is not good in predicting



$h_\theta(x)=g(\theta_0+\theta_1 x_1+\theta_2 x_2)$

(g = sigmoid function)

"Underfitting"

$g(\theta_0+\theta_1 x_1+\theta_2 x_2+\theta_3 x_1^2+\theta_4 x_2^2+\theta_5 x_1 x_2)$

⇒ will get a boundary like this.

$g(\theta_0+\theta_1 x_1+\theta_2 x_1^2+\theta_3 x_1^2 x_2+\theta_4 x_1^2 x_2^2+\theta_5 x_1^2 x_2^3+\theta_6 x_1^3 x_2+\dots)$

Logistic regression may contort itself, try hard to find a d-boundary that fit every example well.

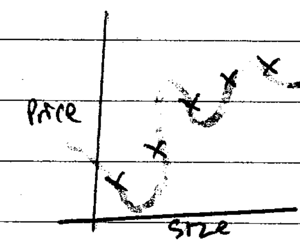
"overfitting"

(只对旧data有用，新的data来就不会很准)

If we think overfitting is occurring, do what to address it.

Addressing overfitting:

x_1 = size of house
x_2 = no. of bedrooms
x_3 = no. of floors
x_4 = age of house
x_5 = average income in neighborhood
x_6 = kitchen size
⋮
x_{100}



The way Use figure to select an appropriate degree polynomial to fit training set is not really work with many features, because it become very hard to plot data much harder to visualize it and decide what features to keep or not.

→ Options:

1. Reduce number of features.
   - Manually select which features to keep.
   - Model selection algorithm (later in course) → auto decide which features to keep/throw

   (But when throwing away features just like throw away information about ~~the problem~~)

2. Regularization:
   - Keep all the features, but reduce magnitude/values of parameters θ_j.
   - Works well when we have a lot of features, each of which contributes a bit to predicting y.

(每个都有用，所以不想throw)

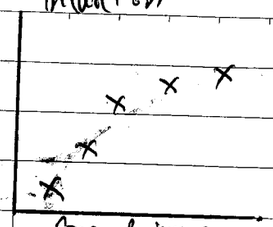
Regularization

Cost function

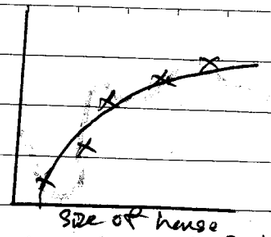
No. _____

Date _____

Intuition



$\theta_0 + \theta_1 x + \theta_2 x^2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Add extra regularization term at the end.

Suppose we penalize and make θ_3, θ_4 really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$$

Example:
Give huge num to θ_3 and θ_4

⇒ Now the only way to make new cost function small is if θ_3 and θ_4 are small.

End up $\theta_3 \approx 0$ $\theta_4 \approx 0$

Will make the graph back to (Good) Quadratic function, + tiny contribution from small term, θ_3 and θ_4

Regularization

Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$ ← small value of parameter make "simple" hypothesis

- "Simpler" hypothesis
- Less Prone to overfitting

Housing:

- Features: $x_1, x_2, \dots, x_{100}$ ← (x_1: size, x_2: num bedroom...) If we have like 100 features, it's hard to prek which are the ones that less likely to be relevant.
- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$ (too many, don't know which one to prek, to shrink it.)

No. _____

Date _____

Example

Now, take linear regression cost function

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

← modify this and shrink all parameter because don't know prick which to shrink

it starts from 1, so not going to penalize θ_0 being large.

Regularization.

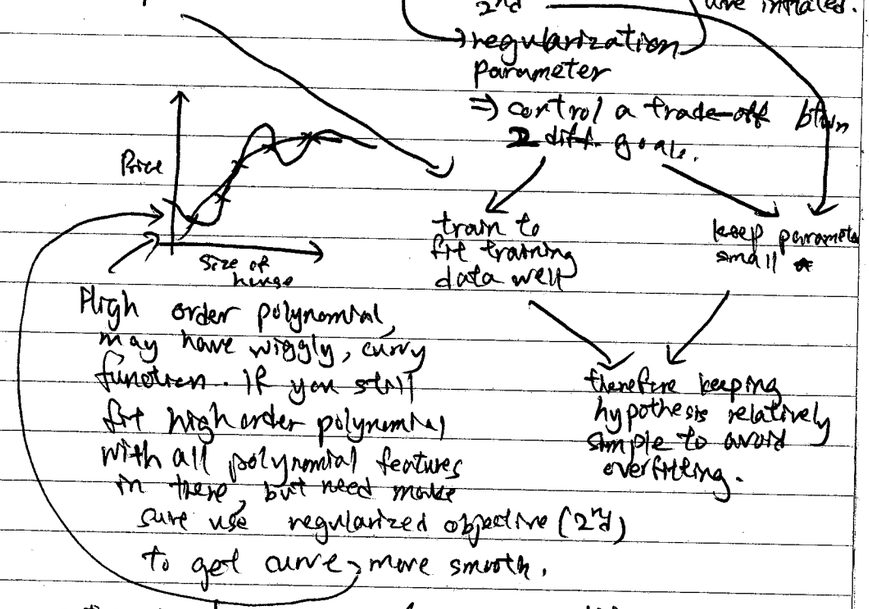
$$J(\theta) = \frac{1}{2m}\left[\underbrace{\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2}_{1^{st}} + \underbrace{\lambda \sum_{j=1}^{n} \theta_j^2}_{2^{nd}}\right]$$

regularization term

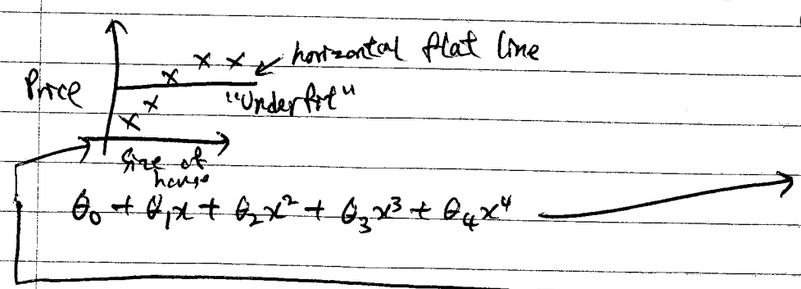
$\min_\theta J(\theta)$

λ: regularization parameter
⇒ control a trade-off btwn 2 diff. goals.

determine how much the costs of our theta parameters are inflated.



If λ set to an extremely large value (exp: $\lambda = 10^{10}$),
⇒ Algorithm results in underfitting (fails to fit even the training set)
↳ end up penalizing $\theta_1, \theta_2, \theta_3, \theta_4 \dots$ very highly ⇒ which mean end up $\theta_1 \approx 0$, $\theta_2 \approx 0$, $\theta_3 \approx 0$, $\theta_4 \approx 0$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

⇓

$h_\theta(x) = \theta_0$

⇓

make only for horizontal straight line to data.

For regularization to work well, some care should be taken to choose the regularization parameter, λ.

Regularization

Regularized linear regression

No. _____

Date _____

- Generalize 2 algorithms (g-descent, normal eqn) to the case of regularized linear regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$\min_\theta J(\theta)$

Find parameter theta that minimizes this cost function

Separate G-descent algorithm into 2 → $\frac{\partial}{\partial\theta_0}J(\theta)$

Repeat {

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

$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$

$(j = \cancel{0},1,2,3,\ldots n)$

}

Because for our regularized linear regression, we penalize $\theta_1, \theta_2 \cdots \theta_n$ but no θ_0

this square bracket inside is just $\frac{\partial}{\partial\theta_j}J(\theta)$ ↳ regularized

$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

perform update

$1 - \alpha\frac{\lambda}{m}$: <1 but +ve so to shrink θ_j

Normal equation

$$X = \begin{bmatrix}(x^{(1)})^T\\ \vdots \\ (x^{(m)})^T\end{bmatrix}$$

$m \times (n+1)$

$$y = \begin{bmatrix}y^{(1)}\\ \vdots \\ y^{(m)}\end{bmatrix} \in \mathbb{R}^m$$

$\frac{\partial}{\partial\theta_j}J(\theta) \overset{\text{set}}{=} 0$ ..... doing bunch of math, and get this eqn that minimize cost function.

$\min_\theta J(\theta)$

$$\theta = (X^TX + \lambda\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1\end{bmatrix})^{-1}X^Ty$$

Example: $n=2$, (3×3) matrix as it is $(n+1)\times(n+1)$ matrix. diagonal is one "0" and other all "1" and part apart from diagonal is all "0"

Non-invertibility (optional / advanced)

Suppose $m \le n$ (#examples, #features) ⇒ X is non-invertible if $m<n$, may be non-invertible if $m=n$

MATLAB → Octave

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

↳ will be non-invertible / singular / matrix will be degenerate

pinv, will not give a good hypothesis inv, will not work

if $\lambda > 0$,

$$\theta = \left(X^TX + \lambda\begin{bmatrix}0 & & \\ & 1 & \\ & & \ddots \\ & & & 1\end{bmatrix}\right)^{-1}X^Ty$$

because of this >0

will not be singular and will be invertible

∴ regularization take care of any non-invertibility issues of X^TX matrix as well.

Regularization

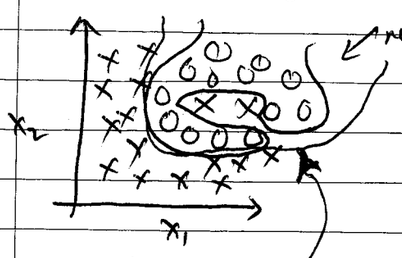
Logistic Regression

No. _____

Date _____

Adapt gradient descent and advanced optimization to work for regularized logistic regression.

Regularized logistic regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 \cdots)$$

when penalizing and parameter ~~become smaller~~, graph become like this

Cost function:

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Penalize $\theta_1, \theta_2, \cdots \theta_n$ from being too large

Gradient descent

Repeat {

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

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$$

$(j = 1, 2, 3, \cdots, n)$

}

$\frac{\partial}{\partial \theta_j} J(\theta)$ ↳ cost function used regularization

When using regularised logistic regression, Plot

$-\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$

as a ~~n~~ function of the number of iterations and make sure it's decreasing to monitor whether g-descent is working correctly or not.

θ_0 in Octave is theta(1)

θ_1 in Octave is theta(2)

θ_n in Octave is theta(n+1)

Advanced optimization

$\theta = \begin{bmatrix}\theta_0 \\ \theta_1 \\ \vdots \\ \theta_n\end{bmatrix}$ ← theta(1), theta(2)

```
function [jVal, gradient] = costFunction(theta)
  jVal = [code to compute J(theta)];
    J(θ) = [-(1/m)Σ_{i=1}^{m} y^(i) log(h_θ(x^(i))) + (1-y^(i)) log(1-h_θ(x^(i)))] + (λ/2m)Σ_{j=1}^{n} θ_j^2
                                                                  include in regularization
  gradient(1) = [code to compute ∂/∂θ_0 J(θ)];
                (1/m)Σ_{i=1}^{m} (h_θ(x^(i)) - y^(i)) x_0^(i)        ← don't change
  gradient(2) = [code to compute ∂/∂θ_1 J(θ)];
                (1/m)Σ_{i=1}^{m} (h_θ(x^(i)) - y^(i)) x_1^(i) + (λ/m)θ_1
  gradient(3) = [code to compute ∂/∂θ_2 J(θ)];
                (1/m)Σ_{i=1}^{m} (h_θ(x^(i)) - y^(i)) x_2^(i) + (λ/m)θ_2
  ⋮
  gradient(n+1) = [code to compute ∂/∂θ_n J(θ)];
```

So if implement this course function and pass into fminunc or other advanced optimization technique, will minimize new regularized cost function J of theta (J(θ))