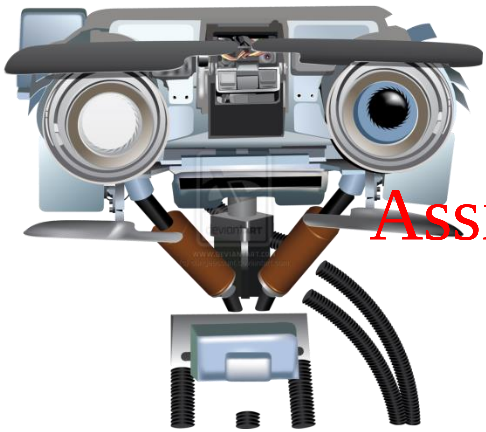
Today

- Maximum Likelihood (cont’d)
- Classification

Reminder: ps1 due at midnight



Maximum Likelihood for Linear Regression

Maximum likelihood way of estimating model parameters θ

In general, assume data is generated by some distribution

$$U \sim p(U|\theta)$$

Observations (i.i.d.)

$$\mathrm{D} = \{u^{(1)}, u^{(2)}, \dots, u^{(m)}\}$$

Maximum likelihood estimate

$$\mathcal{L}(\mathrm{D}) = \boxed{\prod_{i=1}^{m} p(u^{(i)}|\theta)}$$

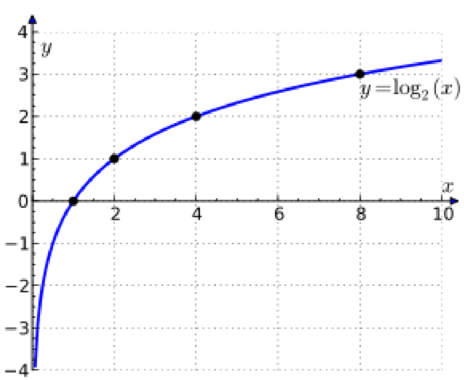
Likelihood

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}(\mathrm{D})$$

Log likelihood

$$= \underset{\theta}{\operatorname{argmax}} \boxed{\sum_{i=1}^{m} \log p(u^{(i)}|\theta)}$$

Note: *p* replaces *h*!



log(f(x)) is monotonic/increasing, same argmax as *f(x)*

i.i.d. observations

- independently identically distributed random variables

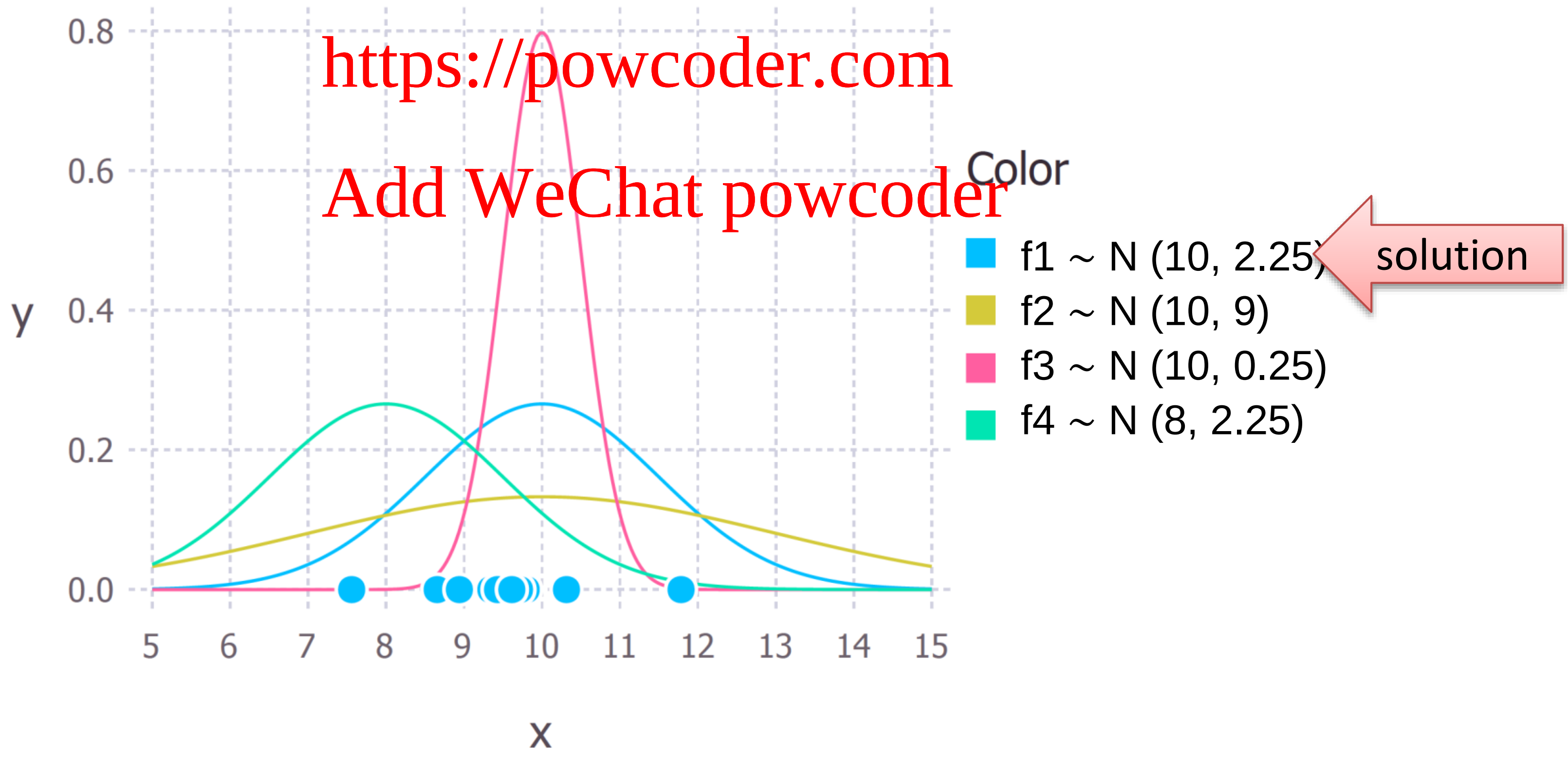
- If u^i are i.i.d. r.v.s, then

$$p(u^1, u^2, \ldots, u^m) = p(u^1)p(u^2)\ldots p(u^m)$$

- A reasonable assumption about many datasets, but not always

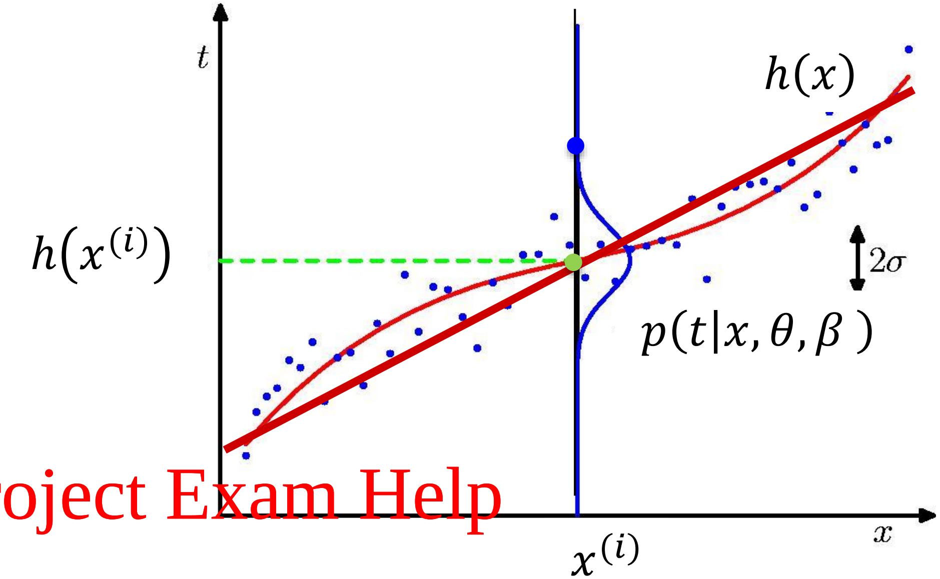
ML: Another example

- Observe a dataset of points $D = \{x^i\}_{i=1:10}$
- Assume x is generated by Normal distribution, $x \sim N(x|\mu, \sigma)$
- Find parameters $\theta_{ML} = [\mu, \sigma]$ that maximize $\prod_{i=1}^{10} N(x^i|\mu, \sigma)$



ML for Linear Regression

Assume:
$t = y + \epsilon = h(x) + \epsilon$
Noise $\epsilon \sim N(\epsilon|0, \beta^{-1})$,
where $\beta = \frac{1}{\sigma^2}$



we don't get to see y, only t

$t_i \quad h(x^{(i)})$

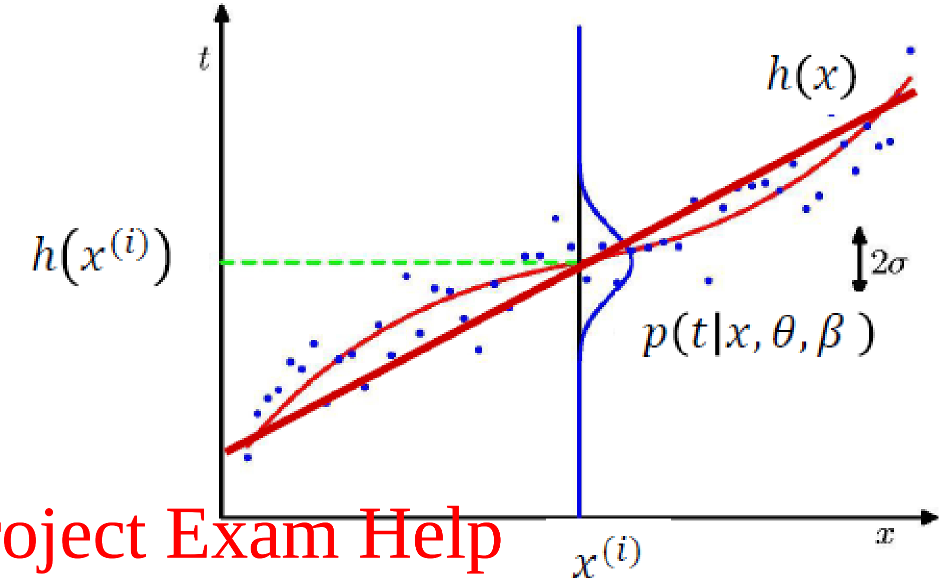
ML for Linear Regression

Assume:

$t = y + \epsilon = h(x) + \epsilon$

Noise $\epsilon \sim N(\epsilon|0, \beta^{-1})$,

where $\beta = \frac{1}{\sigma^2}$



$$p(t|x, \theta, \beta) = N(t|h(x), \beta^{-1})$$

Probability of one data point

$$p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta) = \prod_{i=1}^{m} N(t^{(i)}|h(x^{(i)}), \beta^{-1})$$

Likelihood function

Max. likelihood solution

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta) \qquad \beta_{ML} = \underset{\beta}{\operatorname{argmax}}\, p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta)$$

Want to maximize

$$p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta) = \prod_{i=1}^{m} N(t^{(i)}|h(x^{(i)}),\ \beta^{-1})$$

Easier to maximize log()

$$\ln p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta) =$$

$$-\frac{\beta}{2}\sum_{i=1}^{m}\left(h(x^{(i)}) - t^{(i)}\right)^2 + \frac{m}{2}\ln\beta - \frac{m}{2}\ln(2\pi)$$

Want to maximize w.r.t. θ

$$\ln p(\boldsymbol{t}|\boldsymbol{x},\theta,\beta) = -\frac{\beta}{2}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - t^{(i)}\right)^2 + \frac{m}{2}\ln\beta - \frac{m}{2}\ln(2\pi)$$

... but this is same as minimizing sum-of-squares cost[1]

$$\frac{1}{2m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - t^{(i)}\right)^2$$

... which is the same as our SSE cost from before!!

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

[1]multiply by $-\frac{1}{m\beta}$, changing max to min, omit last two terms (don't depend on θ)

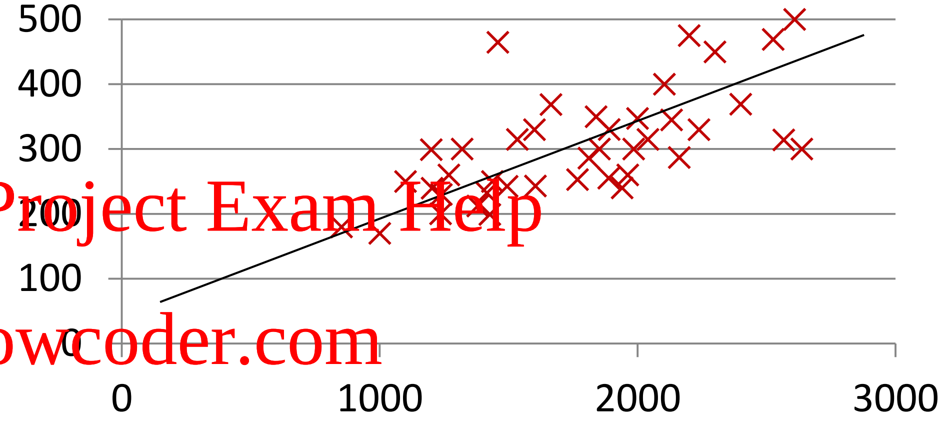
Summary: Maximum Likelihood Solution for Linear Regression

Hypothesis:

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

θ: parameters

$D = \left(x^{(i)}, t^{(i)}\right)$: data



Likelihood:

$$p(\boldsymbol{t}|\boldsymbol{x}, \theta, \beta) = \prod_{i=1}^{m} N(t^{(i)}|h_\theta\left(x^{(i)}\right),\ \beta^{-1})$$

Goal: maximize likelihood, equivalent to

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - t^{(i)}\right)^2$$

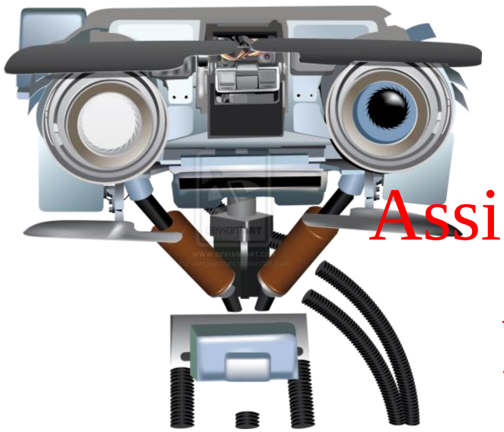
(same as minimizing SSE)

Probabilistic Motivation for SSE

- Under the Gaussian noise assumption, maximizing the probability of the data points is the same as minimizing a sum-of-squares cost function

- Also known as least squares method

- ML can be used for other hypotheses!
  – But linear regression has closed-form solution



Supervised Learning: Classification

Classification

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

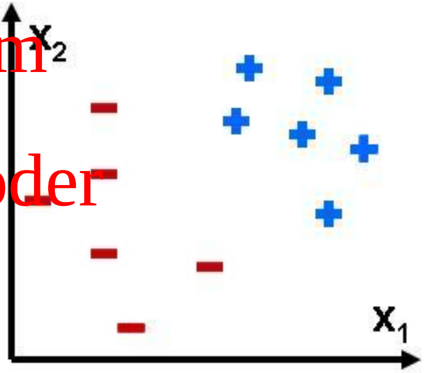
0: “Negative Class” (e.g., benign tumor)

1: “Positive Class” (e.g., malignant tumor)

Tumor: Malignant / Benign?

Email: Spam / Not Spam?

Video: Viral / Not Viral?



Classification

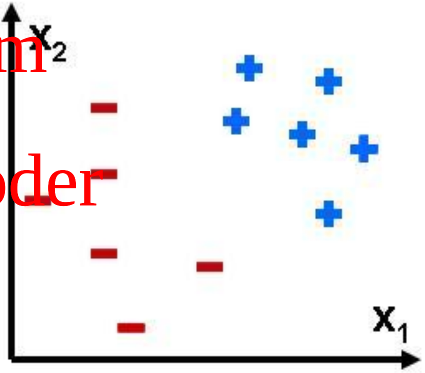
$y \in \{0,1\}$

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

1: “Positive Class” (e.g., malignant tumor)

Why not use least squares regression?

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

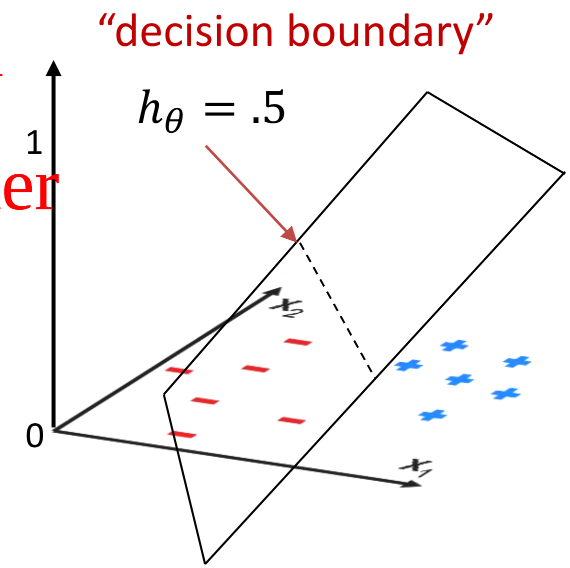


Classification

$y \in \{0,1\}$ 0: "Negative Class" (e.g., benign tumor)
1: "Positive Class" (e.g., malignant tumor)

Why not use least squares regression?

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$



- Indeed, this is possible!
  - Predict 1 if $h_\theta(x) > .5$, 0 otherwise
- However, outliers lead to problems...
- Instead, use logistic regression

Least Squares vs. Logistic Regression for Classification

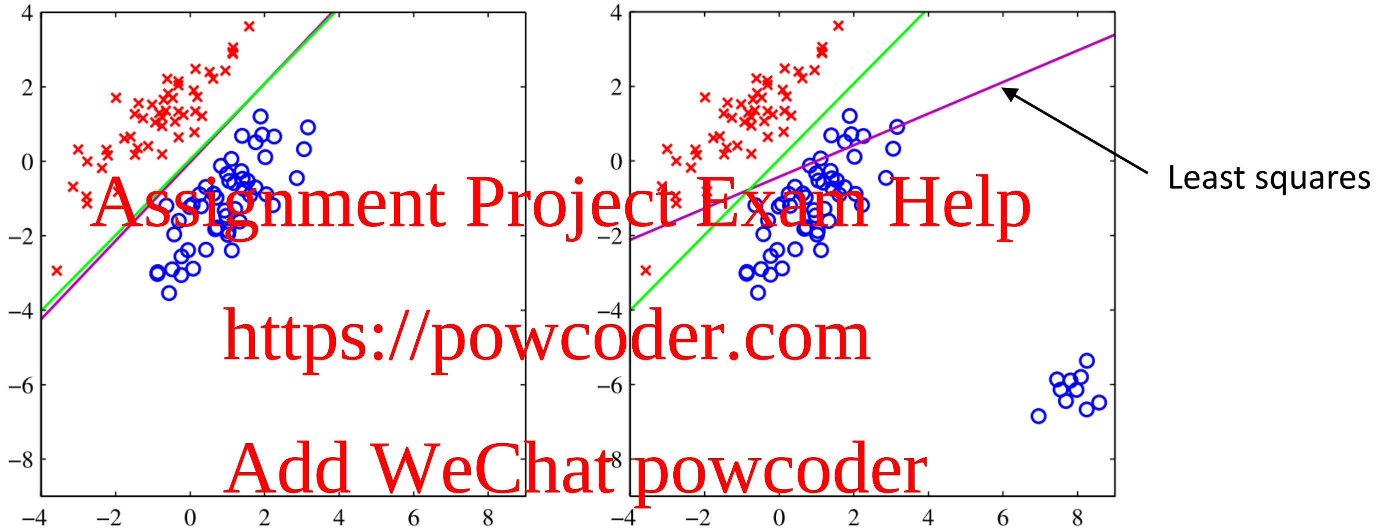


Figure 4.4 from Bishop. The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve). The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

(see Bishop 4.1.3 for more details)

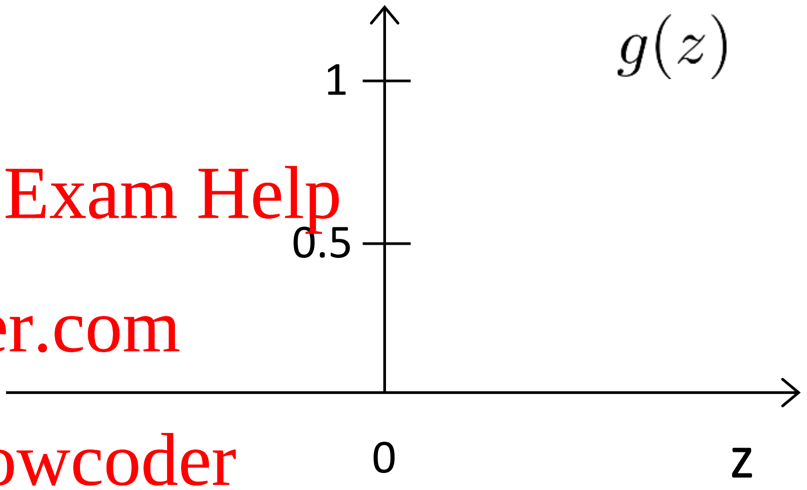
Logistic Regression

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

map to (0, 1) with “sigmoid” function

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

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

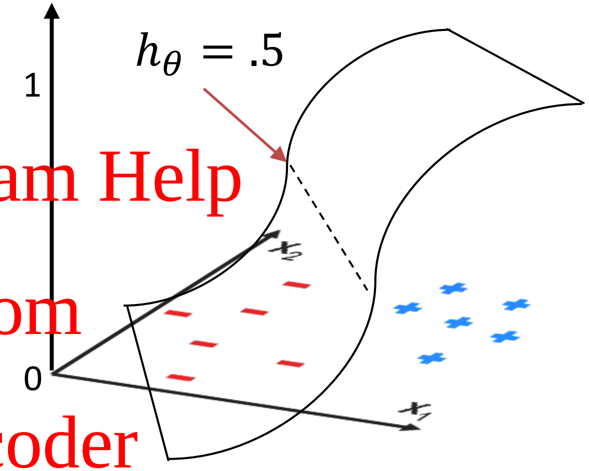


$$h_\theta(x) = p(y=1|x)$$ “probability of class 1 given input”

Logistic Regression

Hypothesis:

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



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

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

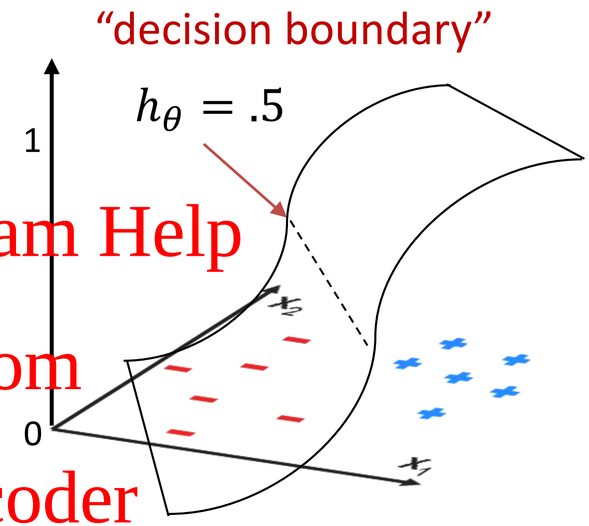
Logistic Regression Cost

Hypothesis:

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

θ: parameters

$D = \left(x^{(i)}, y^{(i)}\right)$: data



Cost Function: cross entropy

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

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

Goal: minimize cost $\quad \min_\theta J(\theta)$

Cross Entropy Cost

- Cross entropy compares distribution *q* to reference *p*

$$H(p,q) = -\sum_x p(x)\, \log q(x).$$

- Here *q* is predicted probability of $y=1$ given *x*, reference distribution is $p=y^{(i)}$, which is either *1 or 0*

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

Gradient of Cross Entropy Cost

- Cross entropy cost

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

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

- its gradient w.r.t θ is:

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

(left as exercise)

- No direct closed-form solution

Gradient descent
for Logistic Regression

Cost

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

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

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

Gradient descent
for Logistic Regression

Cost

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

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

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

Maximum Likelihood Derivation of Logistic Regression Cost

We can derive the Logistic Regression cost

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

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

using Maximum Likelihood, by writing down the likelihood function as

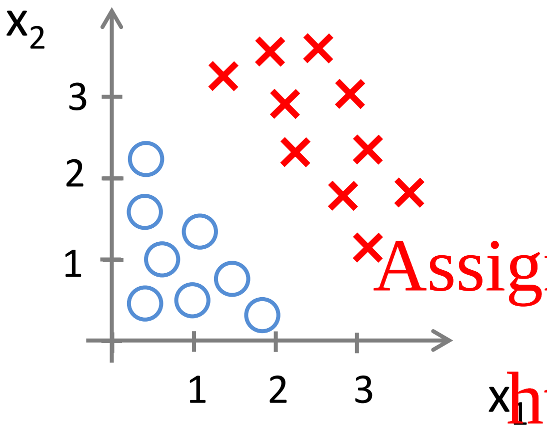
$$p(D|\theta) = \prod_{i=1}^{m} p(y=1|x^{(i)},\theta)^{y^{(i)}}\left(1-p(y=1|x^{(i)},\theta)\right)^{(1-y^{(i)})}$$

where

$$p(y=1|x^{(i)},\theta) = h_\theta(x^{(i)})$$

then taking the log.

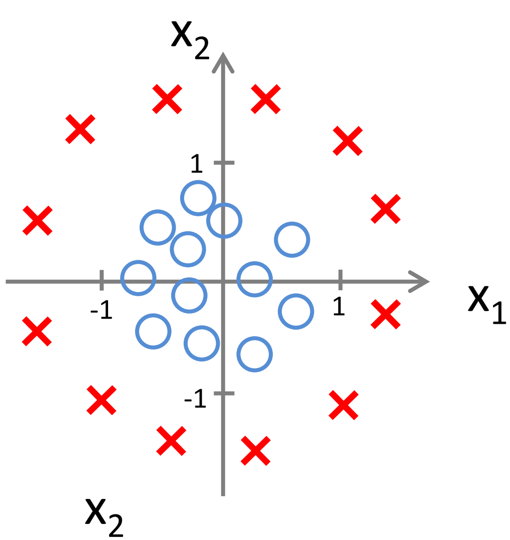
Decision boundary



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

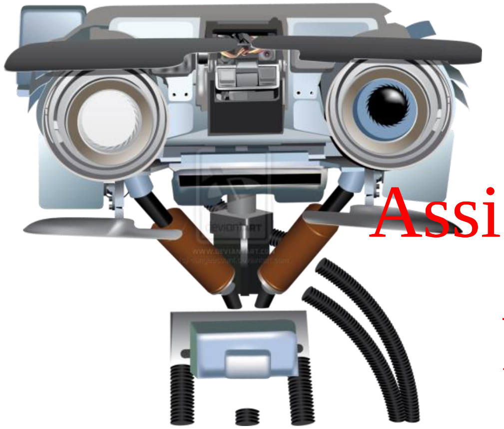
Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Non-linear decision boundaries



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

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

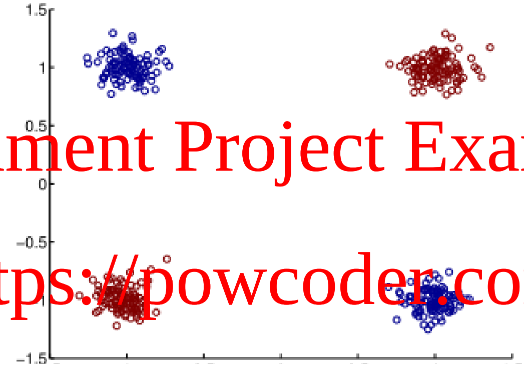


Supervised Learning II

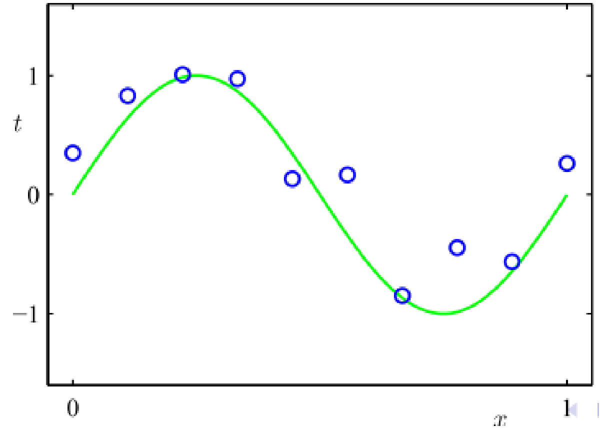
Non-linear features

What to do if data is nonlinear?

Example of nonlinear classification



Example of nonlinear regression

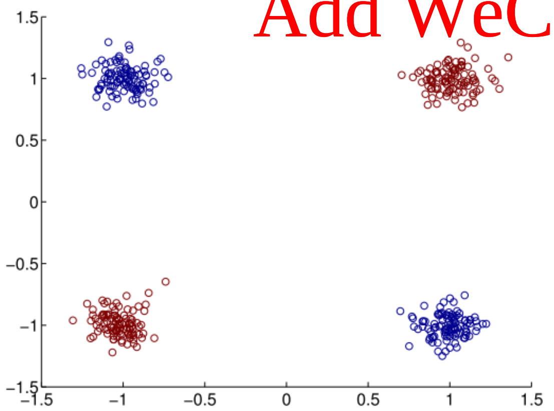


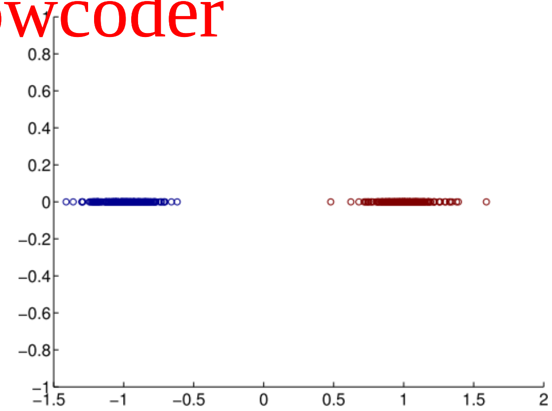
Nonlinear basis functions

Transform the input/feature

$$\phi(x) : x \in R^2 \to z = x_1 \cdot x_2$$

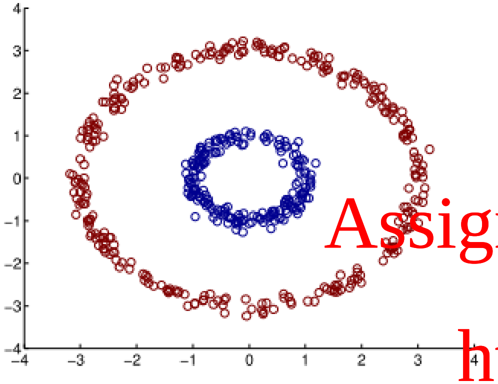
Transformed training data: linearly separable!



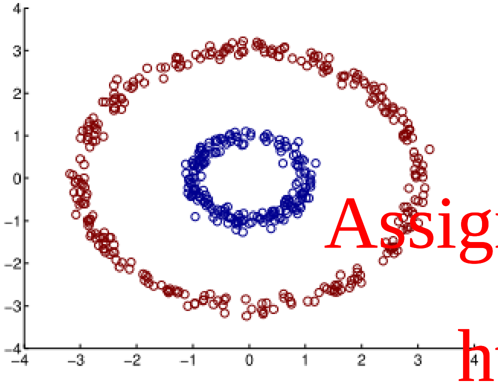


Another example

How to transform the input/feature?



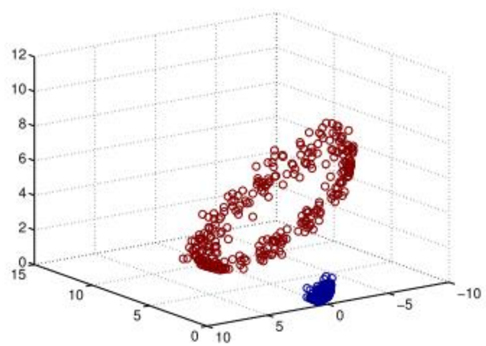
Another example



How to transform the input/feature?

$$\phi(x): x \in R^2 \rightarrow z = \begin{bmatrix} {x_1}^2 \\ x_1 \cdot x_2 \\ {x_2}^2 \end{bmatrix}$$

Transformed training data: linearly separable



Intuition: suppose $\theta = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

Then $\theta^T z = {x_1}^2 + {x_2}^2$

i.e., the sq. distance to the origin!

Non-linear basis functions

- We can use a nonlinear mapping, or basis function

$$\phi(x) : x \in R^N \rightarrow z \in R^M$$

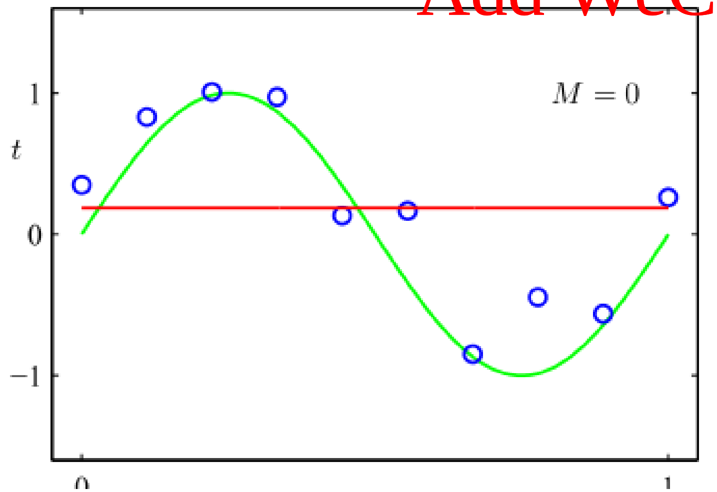
- where M is the dimensionality of the new feature/input z (or $\phi(x)$)
- Note that M could be either greater than D or less than, or the same

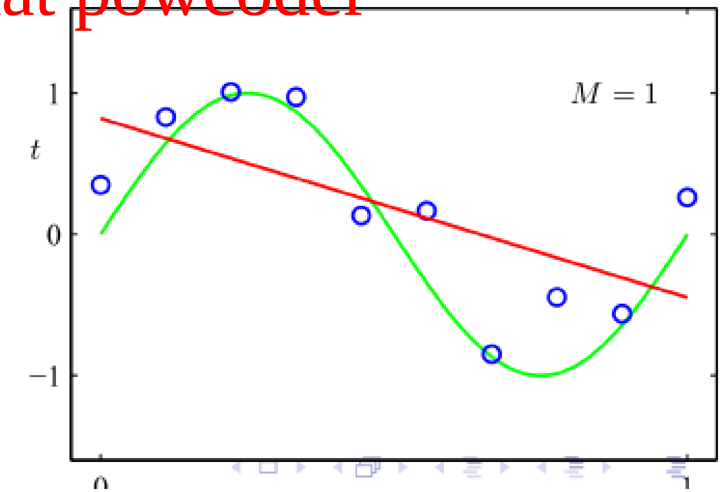
Example with regression

Polynomial basis functions

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

Fitting samples from a sine function: *underrfitting* as $f(x)$ is too simple





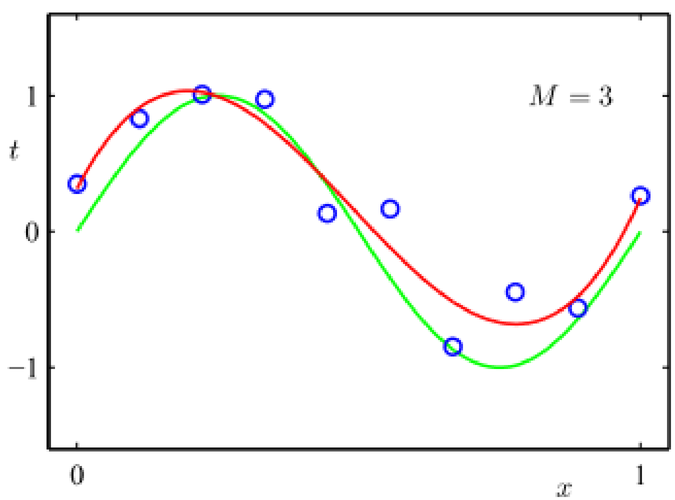
Add more polynomial basis functions

Polynomial basis functions

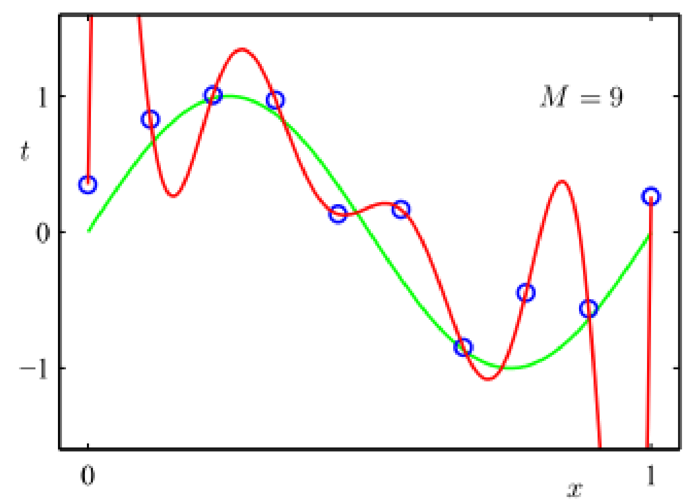
$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

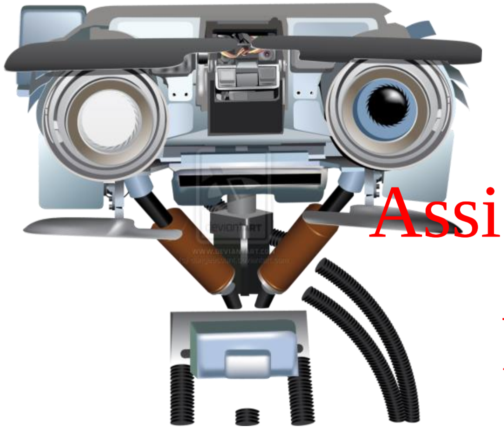
Being too adaptive leads to better results on the training data, but not so great on data that has not been seen!

M=3 *good fit*



M=9: *overfitting*





Supervised Learning II

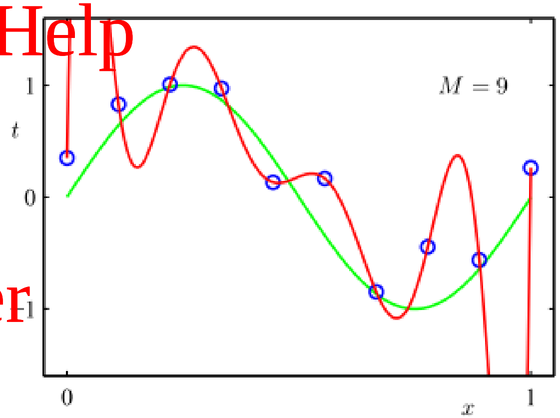
Overfitting

Overfitting

Parameters for higher-order polynomials are very large

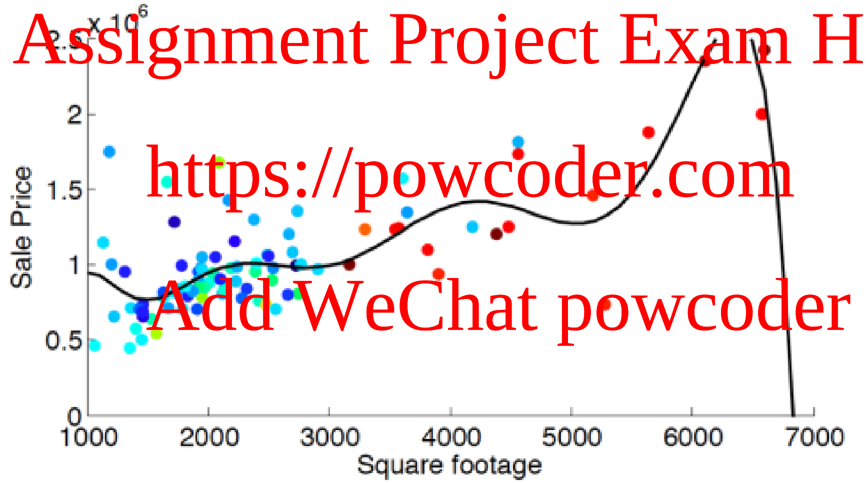
	M =0	M =1	M =3	M =9
θ_0	0.19	0.82	0.31	0.35
θ_1		-1.27	7.99	232.37
θ_2			-25.43	-5321.83
θ_3			17.37	48568.31
θ_4				-231639.30
θ_5				640042.26
θ_6				-1061800.52
θ_7				1042400.18
θ_8				-557682.99
θ_9				125201.43

M=9: *overfitting*



Overfitting disaster

Fitting the housing price data with M = 3

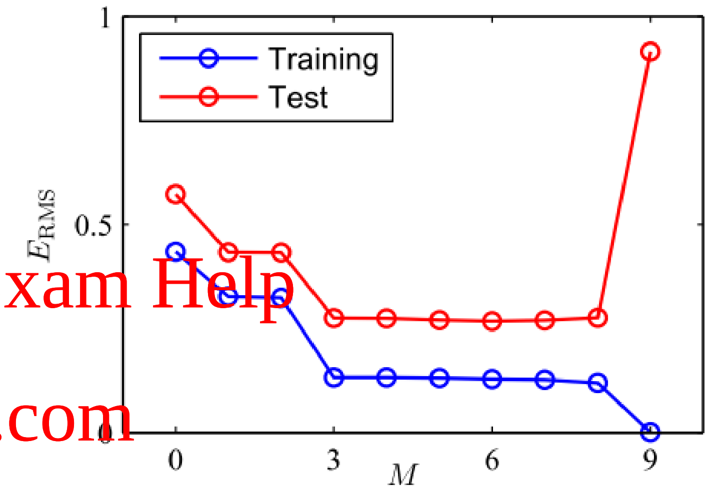


Note that the price would goes to zero (or negative) if you buy bigger houses!
This is called poor generalization/overfitting.

Detecting overfitting

Plot model complexity versus objective function on test/train data

As model becomes more complex, performance on training keeps improving while on test data it increases



Horizontal axis: measure of model complexity
In this example, we use the maximum order of the polynomial basis functions.

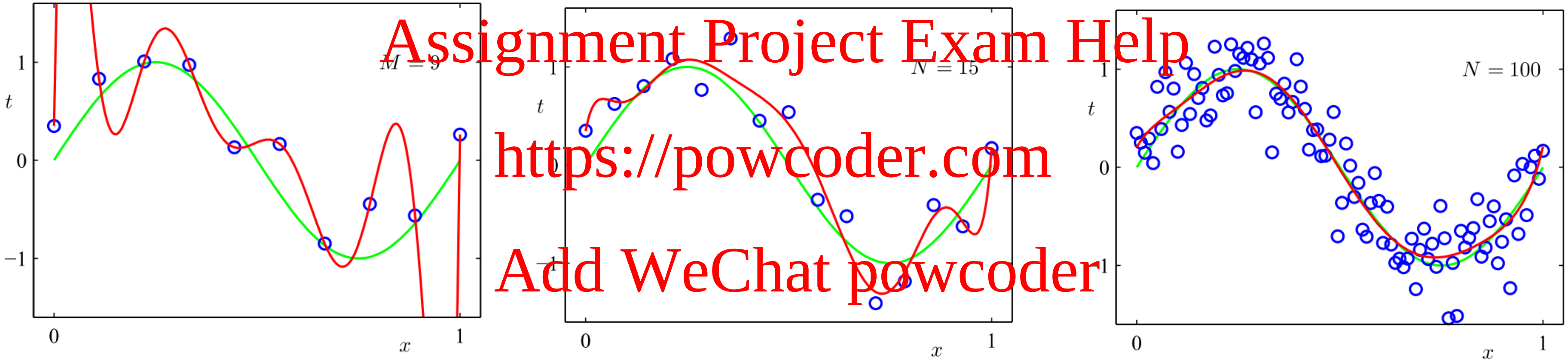
Vertical axis: For regression, it would be SSE or mean SE (MSE)
For classification, the vertical axis would be classification error rate or cross-entropy error function

Overcoming overfitting

- Basic ideas
  - Use more training data
  - Regularization methods
  - Cross-validation

Solution: use more data

M=9, increase N



What if we do not have a lot of data?

Overcoming overfitting

- Basic ideas
  - Use more training data
  - Regularization methods
  - Cross-validation

Next Class

Supervised Learning 3: Regularization

more logistic regression, regularization; bias-variance

Reading: Bishop 3.1, 3.2

Discussion/Lab this week: Intro to Numpy

PSet 2 out on Thursday