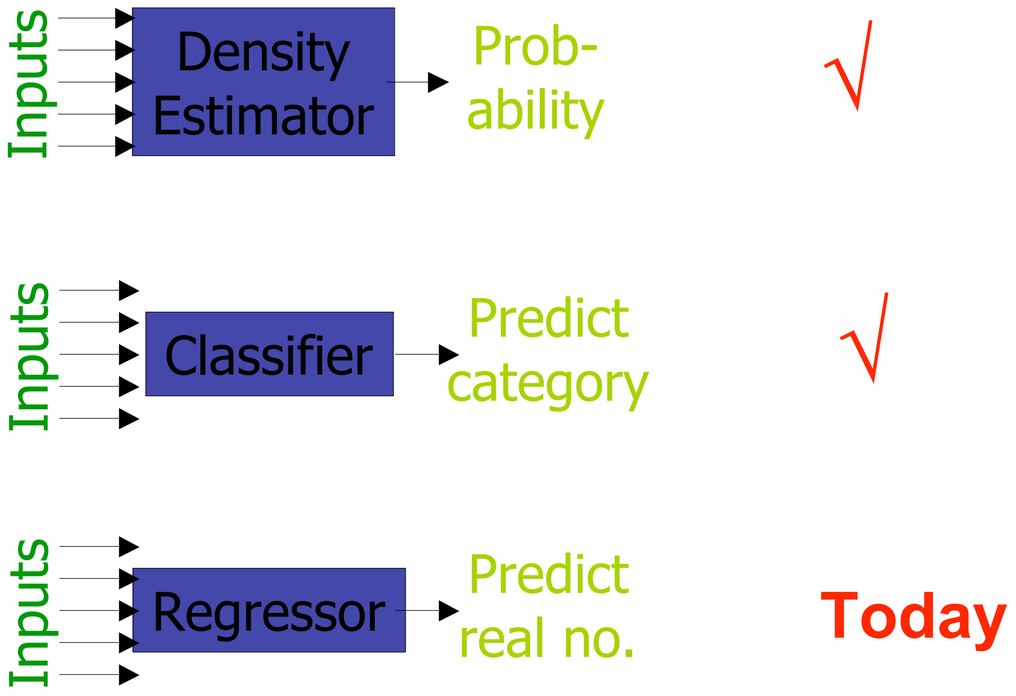
15-381: Artificial Intelligence

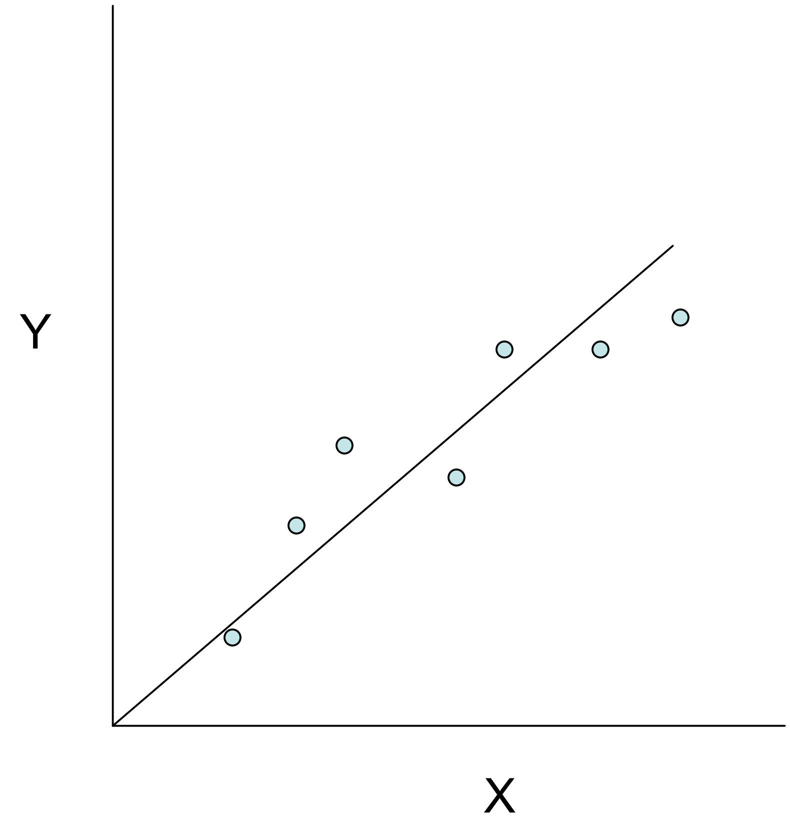
Regression and cross validation

Where we are


Inputs
Density Estimator
Prob-ability
√
Inputs
Classifier
Predict category
√
Inputs
Regressor
Predict real no.
Today

Linear regression

- Given an input x we would like to compute an output y
- For example:
  - Predict height from age
  - Predict Google’s price from Yahoo’s price
  - Predict distance from wall from sensors



Linear regression

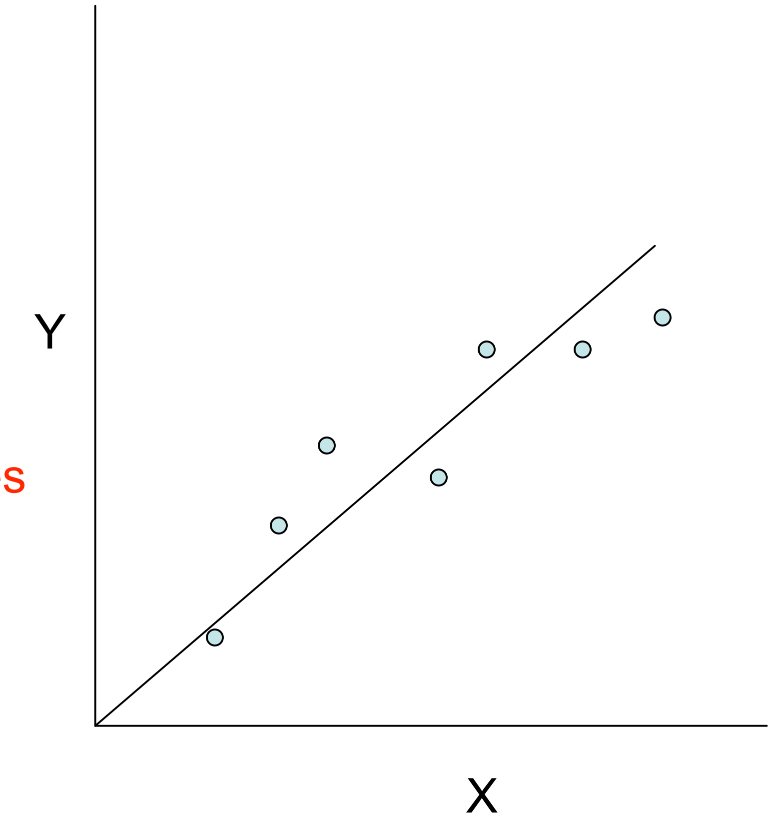
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

Observed values

What we are trying to predict

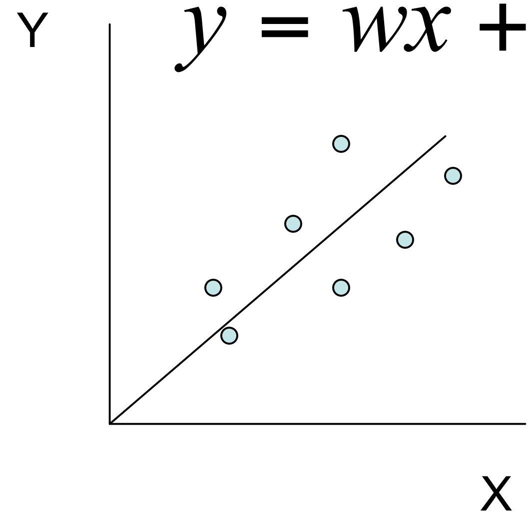
$$y = wx + \varepsilon$$

where w is a parameter and ε represents measurement or other noise



Linear regression

$$y = wx + \varepsilon$$



• Our goal is to estimate w from a training data of $<x_i,y_i>$ pairs

• This could be done using a least squares approach

$$\arg\min_w \sum_i (y_i - wx_i)^2$$

• Why least squares?

- minimizes squared distance between measurements and predicted line

- has a nice probabilistic interpretation

- easy to compute

If the noise is Gaussian with mean 0 then least squares is also the maximum likelihood estimate of w

Solving linear regression

- You should be familiar with this by now …
- We just take the derivative w.r.t. to w and set to 0:

$$\frac{\partial}{\partial w}\sum_i (y_i - wx_i)^2 = 2\sum_i -x_i(y_i - wx_i) \Rightarrow$$

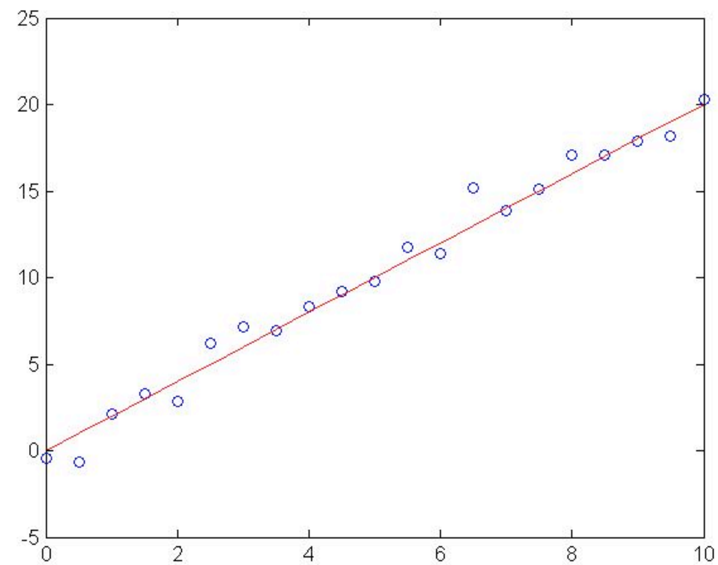
$$2\sum_i x_i(y_i - wx_i) = 0 \Rightarrow$$

$$\sum_i x_i y_i = \sum_i wx_i^2 \Rightarrow$$

$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

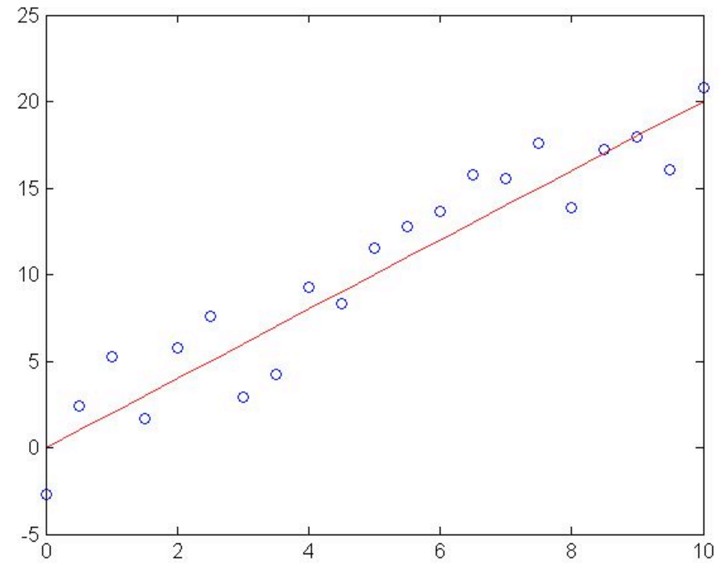
Regression example

- Generated: w=2
- Recovered: w=2.03
- Noise: std=1



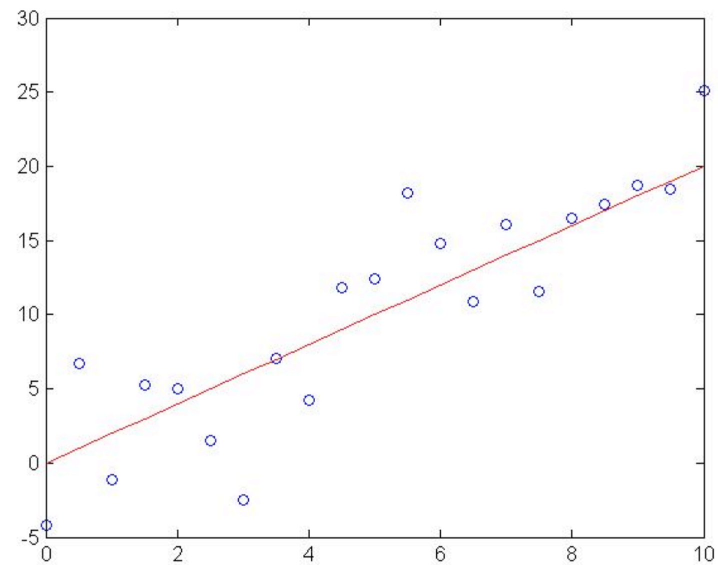
Regression example

- Generated: w=2
- Recovered: w=2.05
- Noise: std=2



Regression example

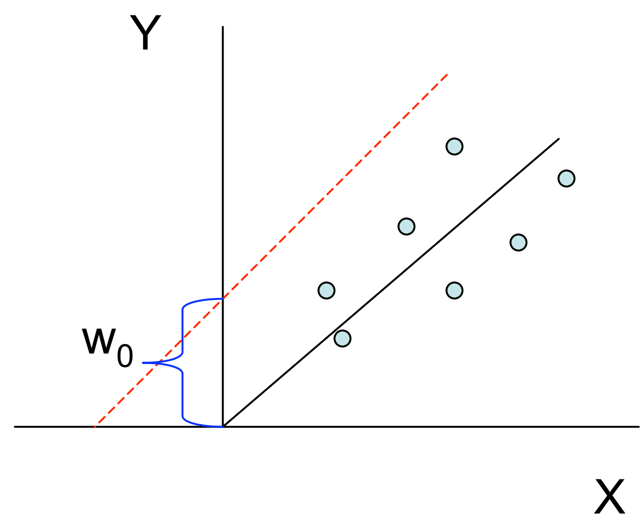
- Generated: w=2
- Recovered: w=2.08
- Noise: std=4



Affine regression

- So far we assumed that the line passes through the origin
- What if the line does not?
- No problem, simply change the model to

$$y = w_0 + w_1 x + \varepsilon$$



- Can use least squares to determine w_0 , w_1

$$w_0 = \frac{\sum_i y_i - w_1 x_i}{n} \qquad w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}$$

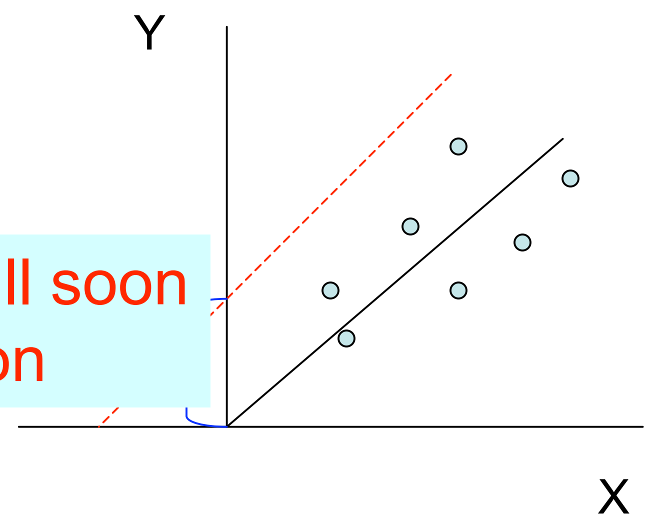
Affine regression

- So far we assumed that the line passes through the origin
- What if the line does not?
- No problem, simply change the model to

y = w

Just a second, we will soon give a simpler solution

- Can use least squares to determine w_0 , w_1



$$w_0 = \frac{\sum_i y_i - w_1 x_i}{n} \qquad w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}$$

Multivariate regression

- What if we have several inputs?

  - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task
- This becomes a multivariate regression problem
- Again, its easy to model:

$$y = w_0 + w_1x_1 + \ldots + w_kx_k + \varepsilon$$

Notations:

Lower case: variable or parameter (w_0)

Lower case bold: vector ($\mathbf{w}$)

Upper case bold: matrix ($\mathbf{X}$)

Multivariate regression: Least squares

- We are now interested in a vector $\mathbf{w}^T = [w_0, w_1, \ldots, w_k]$
- It would be useful to represent this in matrix notations:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- We can thus re-write our model as $\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}$
- The solution turns out to be: $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- This is an instance of a larger set of computational solutions which are usually referred to as ‘generalized least squares’

Multivariate regression: Least squares

- We can re-write our model as $\mathbf{y} = \mathbf{X}\mathbf{w}$
- The solution turns out to be: $\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- The is an instance of a larger set of computational solutions which are usually referred to as ‘generalized least squares’

- $\mathbf{X}^T\mathbf{X}$ is a k by k matrix
- $\mathbf{X}^T\mathbf{y}$ is a vector with k entries

Why is $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ the right solution?

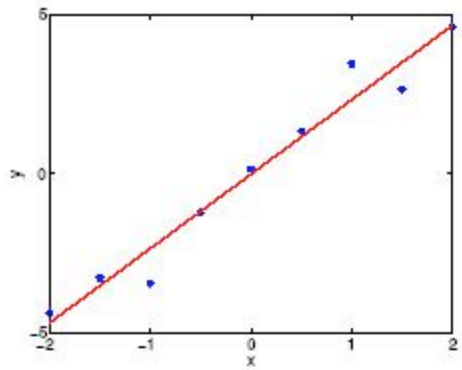
Hint: Multiply both sides of the original equation by $(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$

Beyond linear regression

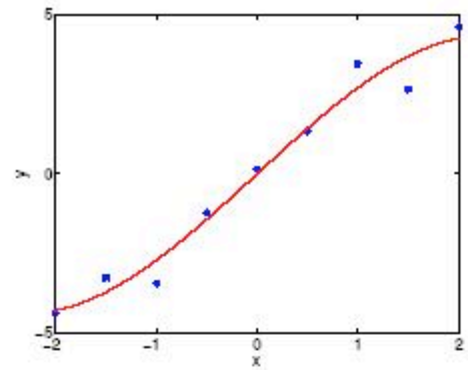
- Can also generalize these classes of functions to be non-linear functions of the inputs *x* but still linear in the parameters *w.*

$$f(x,w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m$$

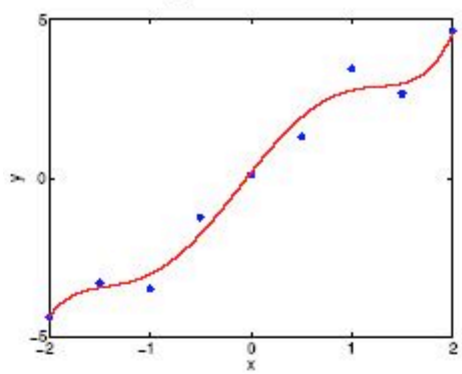
Polynomial regression examples



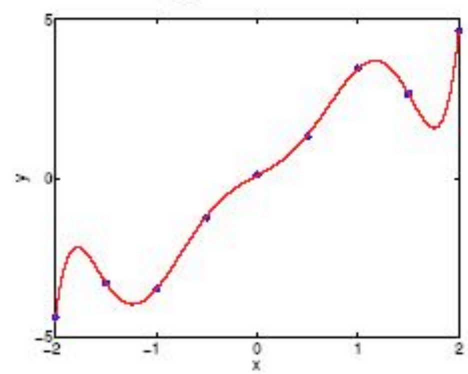
degree = 1



degree = 3



degree = 5



degree = 7

Over fitting

• With too few training examples our polynomial regression model may achieve zero training error but nevertheless has a large generalization error

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i; w_0, w_1))^2 \approx 0$$

$$E_{(x,y)\sim P}(y - f(x; w_0, w_1))^2 >> 0$$

• When the training error no longer bears any relation to the generalization error we say that the function *overfits* the (training) data

Cross validation

• Cross-validation allows us to estimate the generalization error based on training examples alone.

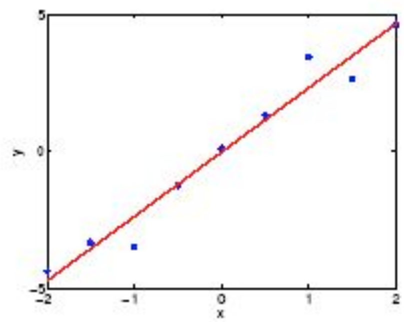
• We learn a model using a subset of the training data and estimate the generalization error using the rest of the data

• We chose the model (for example polynomial order) that minimizes the error on the *held out* data

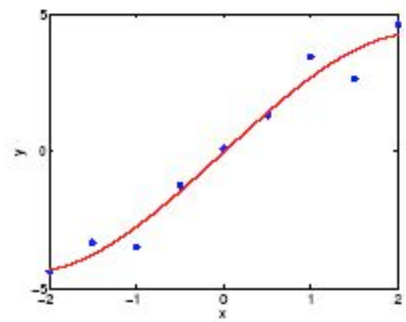
Common strategies

- Leave one out cross validation
- Leave a bigger subset
- Train and test sets

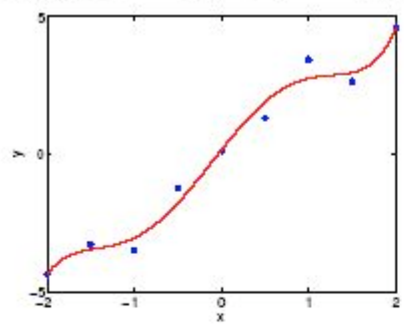
Cross validation: Example



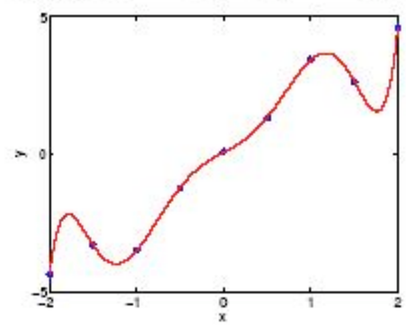
degree = 1, CV = 0.6



degree = 3, CV = 1.5



degree = 5, CV = 6.0



degree = 7, CV = 15.6