10-601
Machine Learning

Regression

Types of classifiers

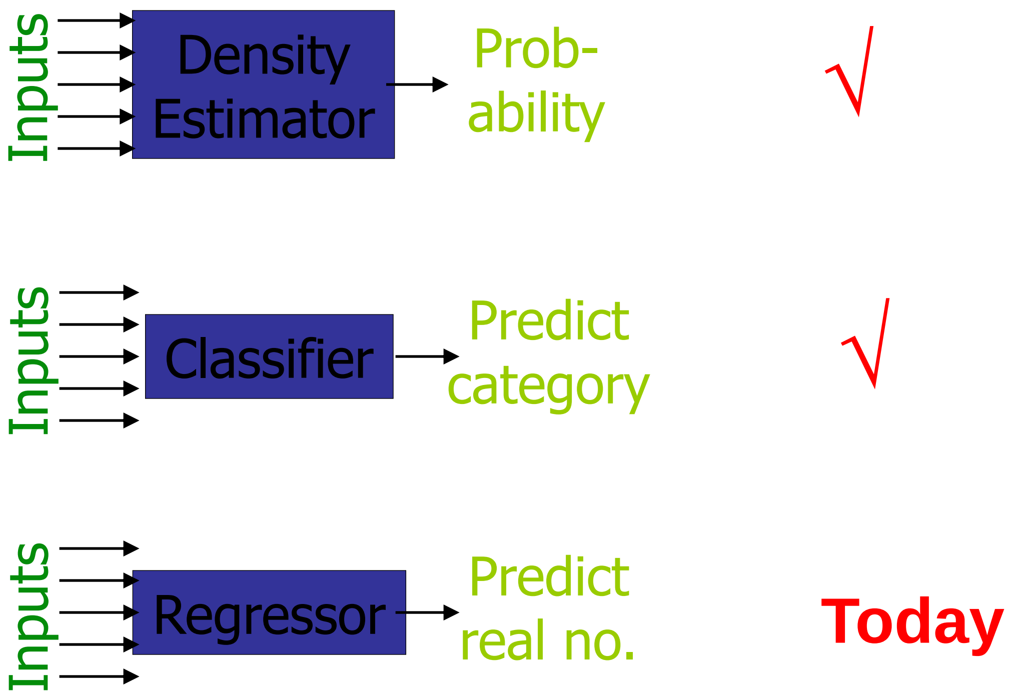
- We can divide the large variety of classification approaches into three major types

  1. Instance based classifiers
     - Use observation directly (no models)
     - e.g. K nearest neighbors

  2. Generative:
     - build a generative statistical model
     - e.g., Bayesian networks

  3. Discriminative
     - directly estimate a decision rule/boundary
     - e.g., decision tree

Where we are


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

Choosing a restaurant

- In everyday life we need to make decisions by taking into account lots of factors
- The question is what weight we put on each of these factors (how important are they with respect to the others).
- Assume we would like to build a recommender system based on an individuals' preferences
- If we have many observations we may be able to recover the weights

Reviews (out of 5 stars)	$	Distance	Cuisine (out of 10)
4	30	21	7
2	15	12	8
5	27	53	9
3	20	5	6

?

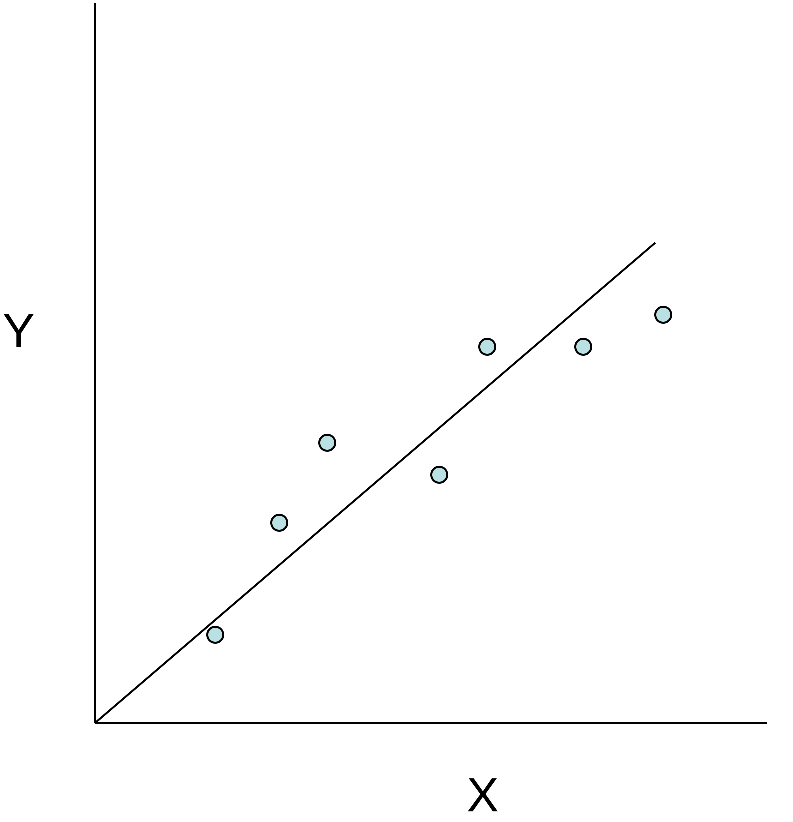
Linear regression

- 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

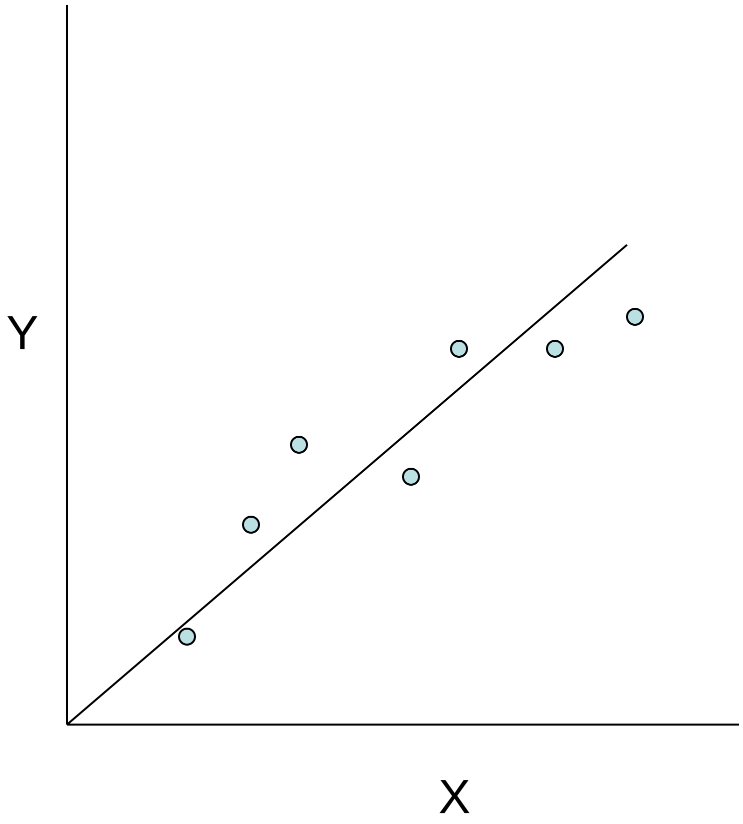
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

Observed values

What we are trying to predict

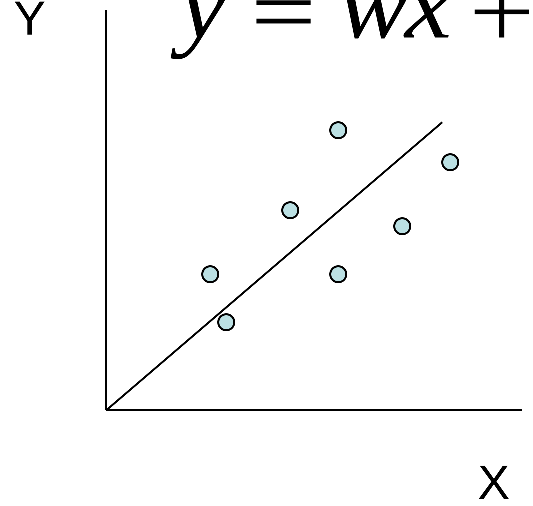
$$y = wx+\varepsilon$$

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:

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

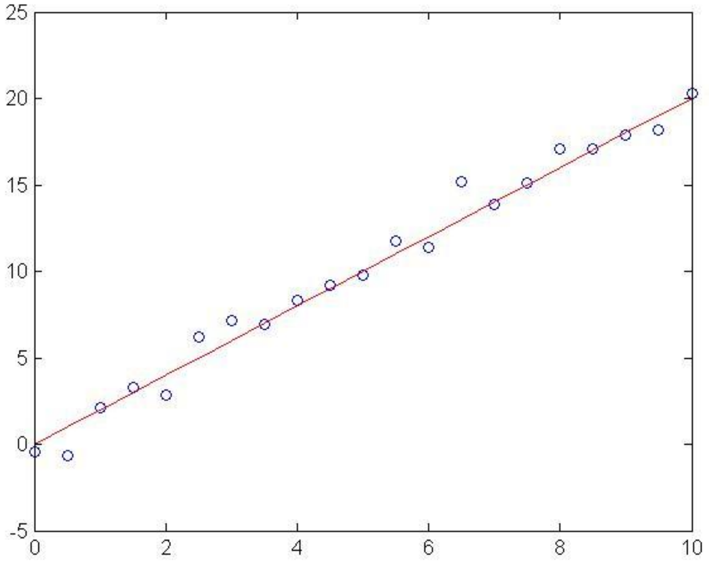
$$2\sum_i x_i(y_i - wx_i) = 0 \Rightarrow$$

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

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

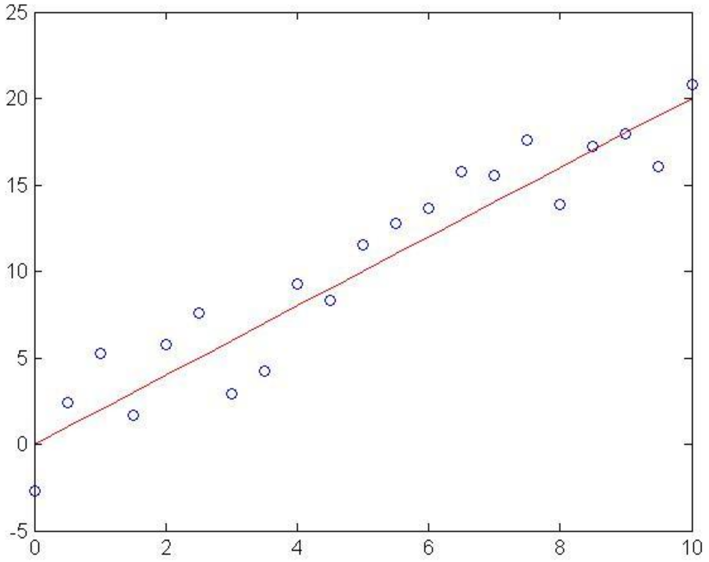
Regression example

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



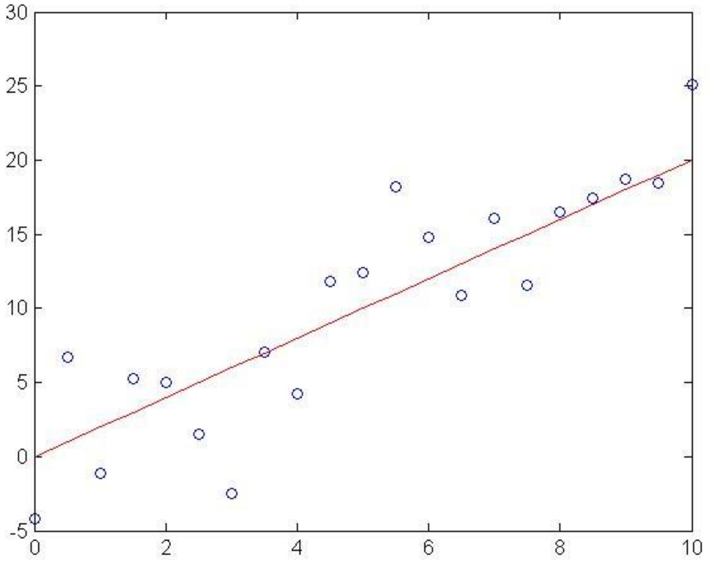
Regression example

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



Regression example

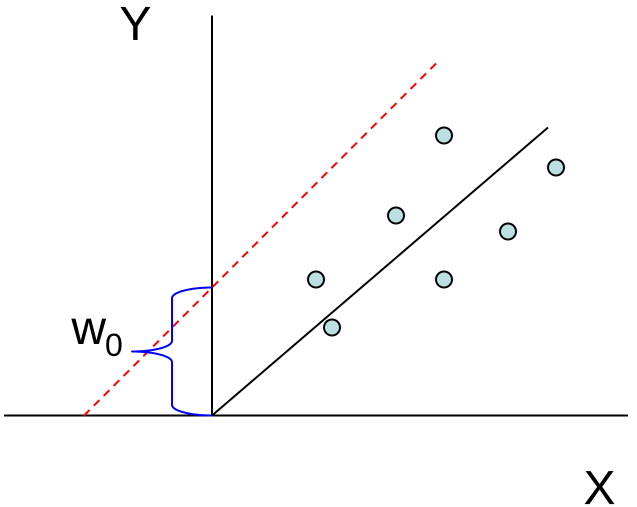
- Generated: w=2
- Recovered: w=2.08
- Noise: std=4



Bias term

- 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

$$w_0 = \frac{\sum_i y_i - w_1 x_i}{n}$$

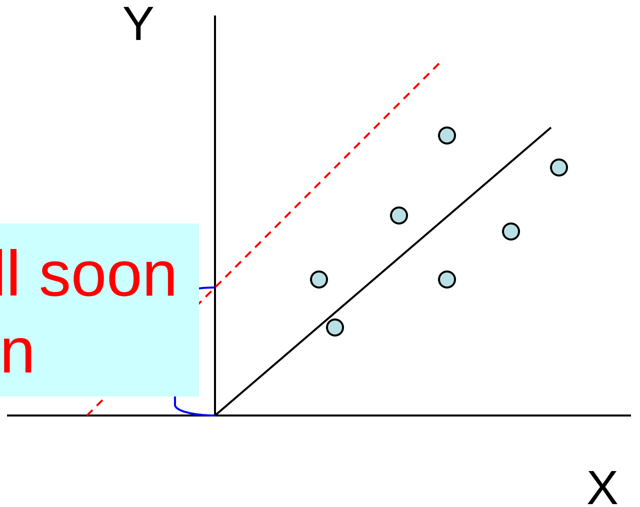
$$w_1 = \frac{\sum_i x_i (y_i - w_0)}{\sum_i x_i^2}$$

Bias term

- 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:

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

Google's stock price

Microsoft's stock price

Yahoo's stock price

Multivariate regression

- What if we have several inputs?

  - Stock prices for Yahoo, Microsoft and Ebay for the Goo

- This be n problem
- Again, its easy to model:

Not all functions can be approximated using the input values directly

$$y = w_0 + w_1 x_1 + \ldots + w_k x_k + \varepsilon$$

$$y=10+3x_1^2-2x_2^2+\varepsilon$$

In some cases we would like to use polynomial or other terms based on the input data, are these still linear regression problems?

Yes. As long as the coefficients are linear the equation is still a linear regression problem!

Non-Linear basis function

- So far we only used the observed values
- However, linear regression can be applied in the same way to functions of these values
- As long as these functions can be directly computed from the observed values the parameters are still linear in the data and the problem remains a linear regression problem

$$y = w_0 + w_1 x_1^2 + \ldots + w_k x_k^2 + \varepsilon$$

Non-Linear basis function

- What type of functions can we use?
- A few common examples:

- Polynomial: $\phi_j(x) = x^j$ for j=0 … n

- Gaussian: $$\phi_j(x) = \frac{(x-\mu_j)}{2\sigma_j^2}$$

- Sigmoid: $$\phi_j(x) = \frac{1}{1+\exp(-s_j x)}$$

Any function of the input values can be used. The solution for the parameters of the regression remains the same.

General linear regression problem

- Using our new notations for the basis function linear regression can be written as

$$y = \sum_{j=0}^{n} w_j \phi_j(x)$$

- Where $\phi_j(x)$ can be either x_j for multivariate regression or one of the non linear basis we defined
- Once again we can use 'least squares' to find the optimal solution.

LMS for the general linear regression problem

$$y = \sum_{j=0}^{n} w_j \phi_j(x)$$

Our goal is to minimize the following loss function:

$$J(\mathrm{w}) = \sum_i (y^i - \sum_j w_j \phi_j(x^i))^2$$

w – vector of dimension k+1

$\phi(x^i)$ – vector of dimension k+1

y^i – a scaler

Moving to vector notations we get:

$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2$$

We take the derivative w.r.t **w**

$$\frac{\partial}{\partial w} \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2 = 2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}}$$

Equating to 0 we get

$$2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}} = 0 \Rightarrow$$

$$\sum_i y^i \phi(x^i)^{\mathrm{T}} = \mathrm{w}^{\mathrm{T}} \left[\sum_i \phi(x^i) \phi(x^i)^{\mathrm{T}} \right]$$

LMS for general linear regression problem

$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2$$

We take the derivative w.r.t **w**

$$\frac{\partial}{\partial w} \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i))^2 = 2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}}$$

Equating to 0 we get

$$2 \sum_i (y^i - \mathrm{w}^{\mathrm{T}} \phi(x^i)) \phi(x^i)^{\mathrm{T}} = 0 \Rightarrow$$

$$\sum_i y^i \phi(x^i)^{\mathrm{T}} = \mathrm{w}^{\mathrm{T}} \left[\sum_i \phi(x^i) \phi(x^i)^{\mathrm{T}} \right]$$

Define:

$$\Phi = \begin{pmatrix} \phi_0(x^1) & \phi_1(x^1) & \cdots & \phi_m(x^1) \\ \phi_0(x^2) & \phi_1(x^2) & \cdots & \phi_m(x^2) \\ \vdots & \vdots & \cdots & \vdots \\ \phi_0(x^n) & \phi_1(x^n) & \cdots & \phi_m(x^n) \end{pmatrix}$$

Then deriving w
we get:

$$\mathrm{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{y}$$

LMS for general linear regression problem

$$J(\mathrm{w}) = \sum_i (y^i - \mathrm{w}^T \phi(x^i))^2$$

Deriving w we get: $\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$

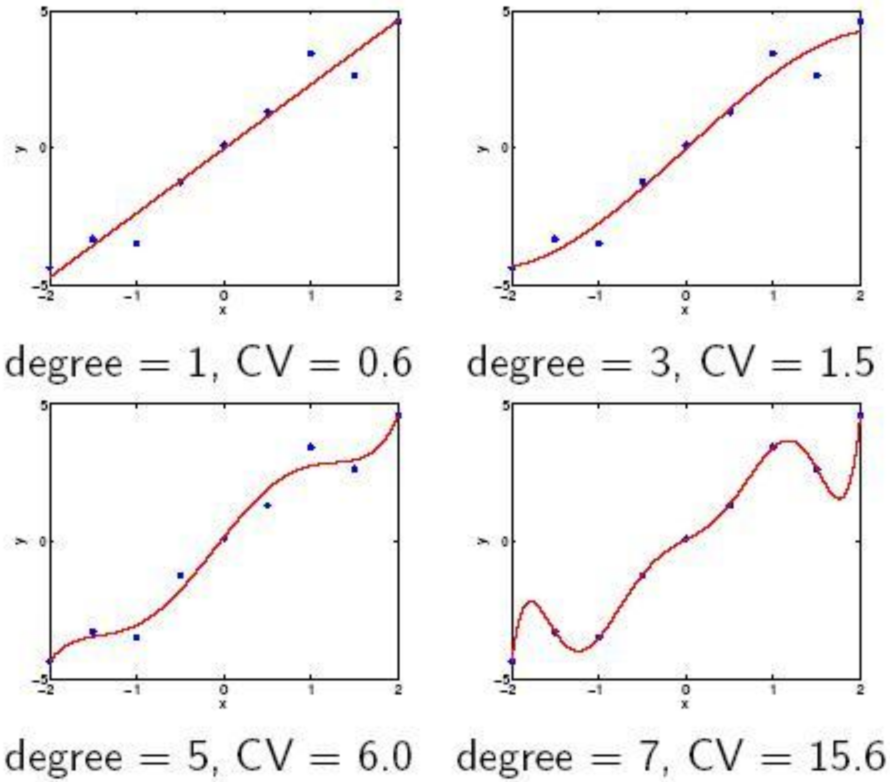
k+1 entries vector

n by k+1 matrix

n entries vector

This solution is also known as ‘psuedo inverse’

Example: Polynomial regression



A probabilistic interpretation

Our least squares minimization solution can also be motivated by a probabilistic in interpretation of the regression problem:

$$y = \mathrm{w}^{\mathrm{T}} \phi(x) + \varepsilon$$

where…

- the noise signals (ε) are independent
- the noise has a normal distribution with mean 0 and unknown variance σ^2

Then p(y|w,x) has a normal distribution with

- mean $\mathrm{w}^{\mathrm{T}}\phi(x)$
- variance σ^2

A probabilistic interpretation

Our least squares minimization solution can also be motivated by a probabilistic in interpretation of the regression problem: $y = \mathrm{w}^{\mathrm{T}} \phi(x) + \varepsilon$

Then p(*y*|*w*,*x*) has a normal distribution with

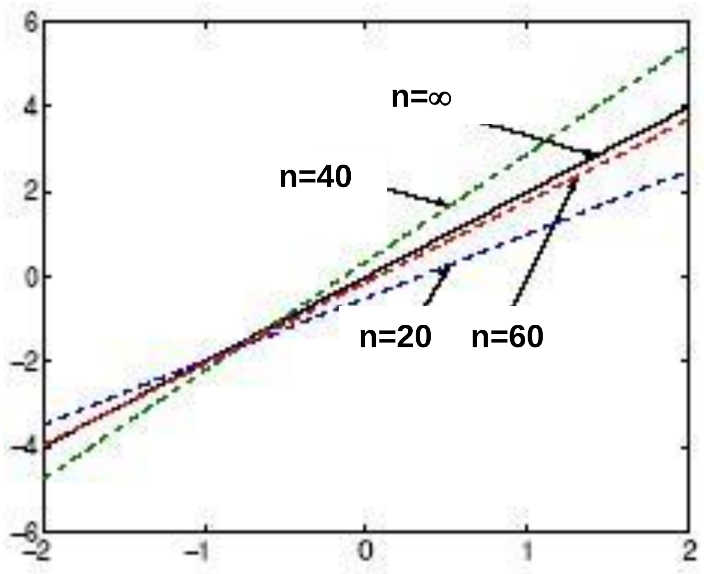
- mean $\mathrm{w}^{\mathrm{T}} \phi(x)$
- variance σ^2

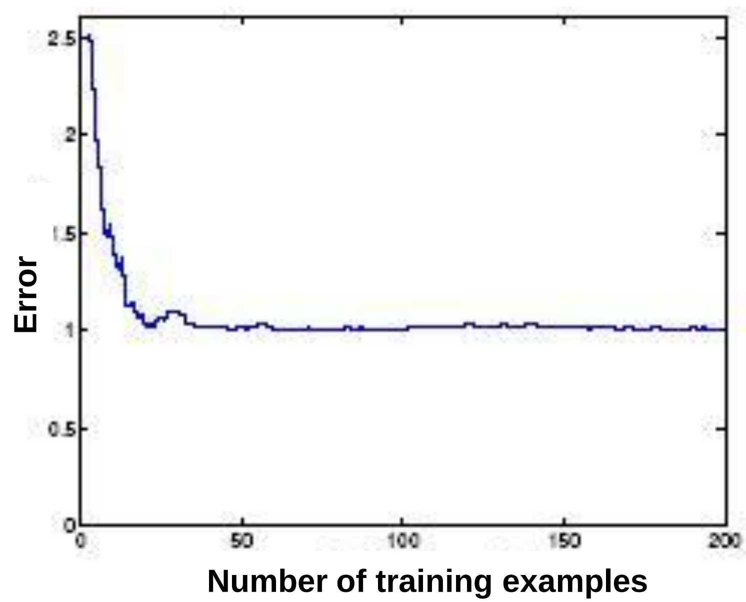
The MLE for w in this model is the same as the solution we derived for least squares

$$\mathrm{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{y}$$

Errors

As the number of training examples increases our solution gets “better”



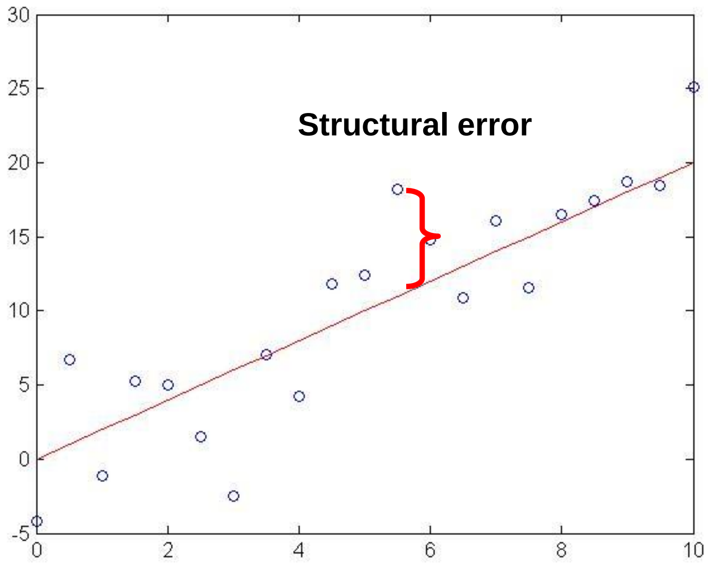


Two types of errors

- **Structural error** measures the error introduced by the limited function class (infinite training data):

$$\min_{\mathrm{w}} E_{(x,y)\sim P}(y - \mathrm{w}^{\mathrm{T}}\phi(x))^2 = E_{(x,y)\sim P}(y - \mathrm{w}^{*\mathrm{T}}\phi(x))^2$$

where ($\mathrm{w}^{*\mathrm{T}}$) are the optimal linear regression parameters.

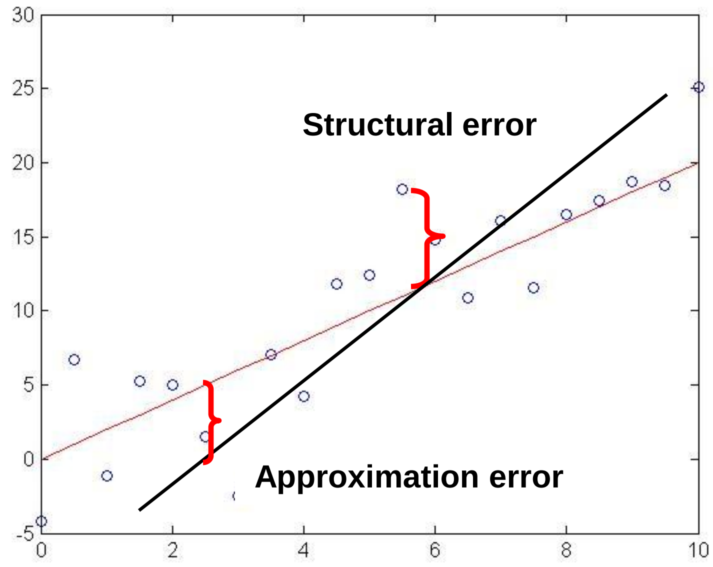


Two types of error

- **Approximation error** measures how close we can get to the optimal linear predictions with limited training data:

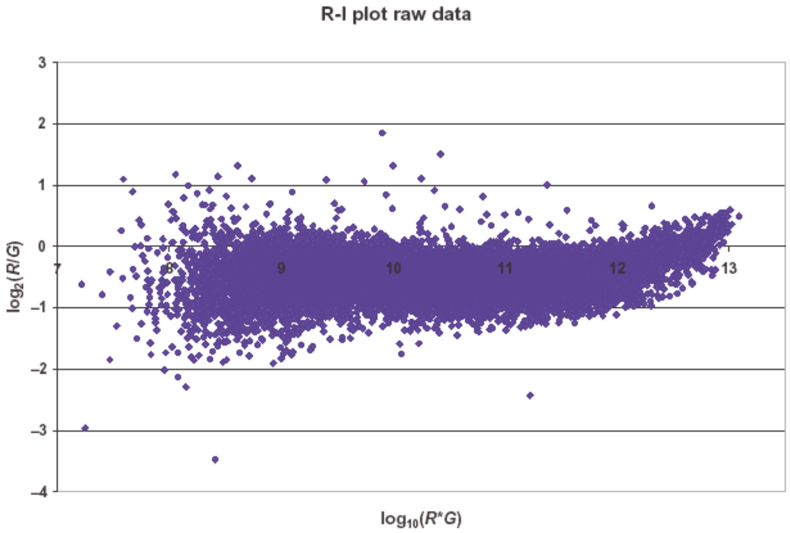
$$E_{(x,y)\sim P}(\mathrm{w}^{*\mathrm{T}}\phi(x) - \hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2$$

where $(\hat{\mathrm{w}})$ are the parameter estimates based on a small training set (therefore themselves random variables).



Other types of linear regression

- Linear regression is a useful model for many problems
- However, the parameters we learn for this model are **global**; they are the same regardless of the value of the input x
- Extension to linear regression adjust their parameters based on the region of the input we are dealing with



Splines

- Instead of fitting one function for the entire region, fit a set of piecewise (usually cubic) polynomials satisfying continuity and smoothness constraints.
- Results in smooth and flexible functions without too many parameters
- Need to define the regions in advance (usually uniform)

$$y = a_2 x^3 + b_2 x^2 + c_2 x + d_2$$

$$y = a_1 x^3 + b_1 x^2 + c_1 x + d_1$$

$$y = a_3 x^3 + b_3 x^2 + c_3 x + d_3$$

Splines

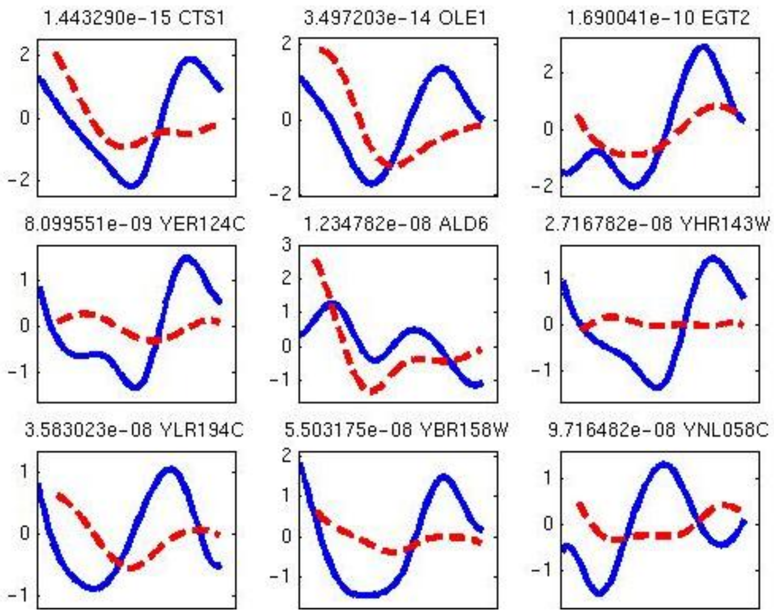
- The polynomials are not independent
- For cubic splines we require that they agree in the border point on the value, the values of the first derivative and the value of the second derivative
- How many free parameters do we actually have?

$$y = a_2x^3 + b_2x^2 + c_2x + d_2$$

$$y = a_1x^3 + b_1x^2 + c_1x + d_1 \qquad y = a_3x^3 + b_3x^2 + c_3x + d_3$$

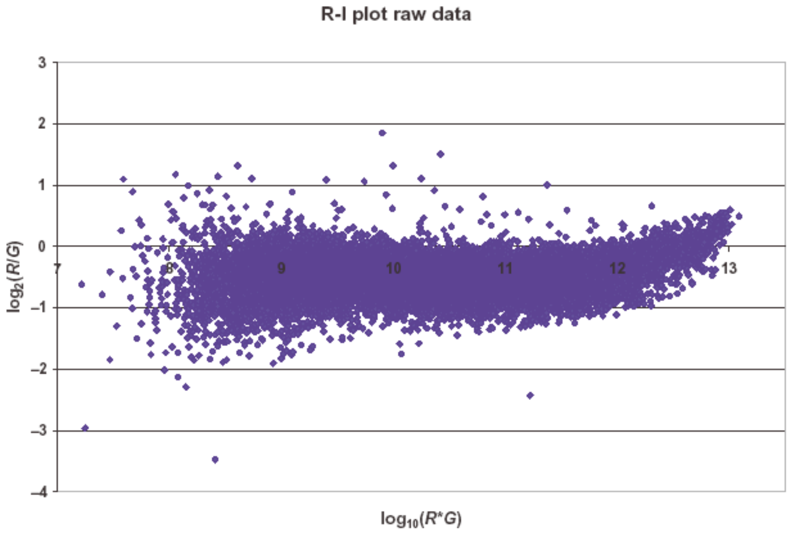
Splines

- Splines sometimes contain additional requirements for the first and last polynomial (for example, having them start at 0)
- Once Splines are fitted to the data they can be used to predict new values in the same way as regular linear regression, though they are limited to the support regions for which they have been defined
- Note the range of functions that can be displayed with relatively small number of polynomials (in the example I am using 5)



Locally weighted models

- Splines rely on a fixed region for each polynomial and the weight of all points within the region is the same.
- An alternative option is to set the region based on the density of the input data and have points closer to the point we are trying to estimate have a higher weight

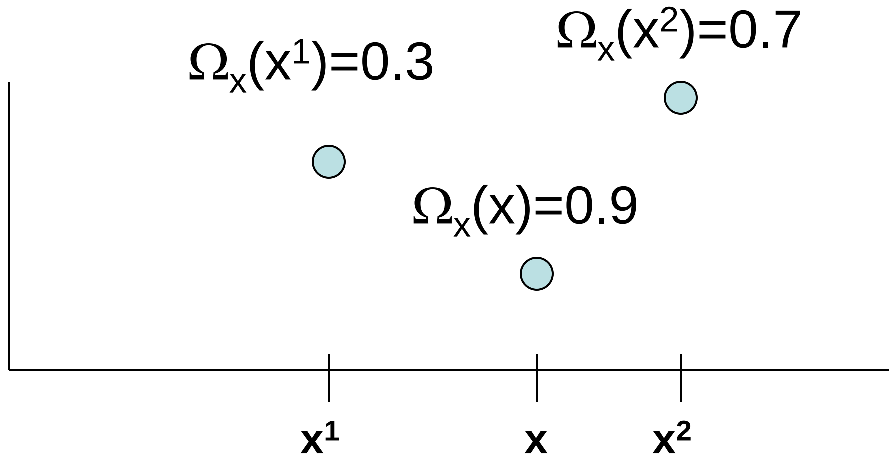


Weighted regression

- For a point x we use weight function Ω_x centered at x to assign weight to points in x's vicinity
- Next we solve the following *weighted* regression problem

$$\min_{\mathrm{w}} \sum_i \Omega_x(x^i)(y^i - \mathrm{w}^T \phi(x^i))^2$$

- The solution is the same as our general solution (the weight is given for every input)



Determining the weights

- There are a number of ways to determine the weights
- One options is to use a Gaussian centered at *x*, such that

$$\Omega_x(x^i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x^i)^2}{2\sigma^2}}$$

σ^2 is a parameter that should be selected by the user

More on these weights when we discuss kernels

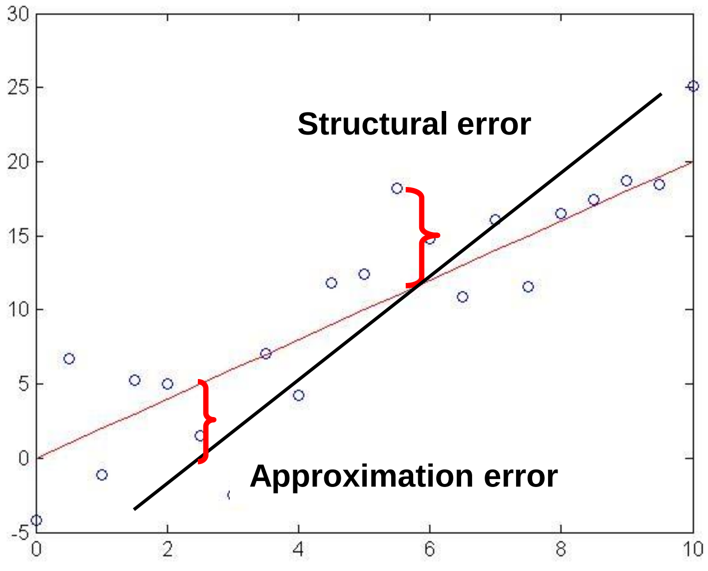
Important points

- Linear regression
  - basic model
  - as a function of the input
- Solving linear regression
- Error in linear regression
- Advanced regression models

Error decomposition

Lets write the global error in terms of the structural and estimation errors

$$E_{(x,y)\sim P}(y-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2 = E_{(x,y)\sim P}((y-\mathrm{w}^{*\mathrm{T}}\phi(x))+(\mathrm{w}^{*\mathrm{T}}\phi(x)-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2$$



Error decomposition

$$
\begin{aligned}
E_{(x,y)\sim P}(y-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2 &= E_{(x,y)\sim P}((y-\mathrm{w}^{*\mathrm{T}}\phi(x))+(\mathrm{w}^{*\mathrm{T}}\phi(x)-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2 \\
&= E_{(x,y)\sim P}(y-\mathrm{w}^{*\mathrm{T}}\phi(x))^2 \\
&\quad +2E_{(x,y)\sim P}(y-\mathrm{w}^{*\mathrm{T}}\phi(x))(\mathrm{w}^{*\mathrm{T}}\phi(x)-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x)) \\
&\quad +E_{(x,y)\sim P}(\mathrm{w}^{*\mathrm{T}}\phi(x)-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2
\end{aligned}
$$

Must be 0. The estimation error (ε) for the optimal parameters (w*) is, by definition, decorelated from any linear function of the input data.

Error is decomposable!

The expected error of our linear regression function decomposes into the sum of structural and approximation errors

$$E_{(x,y)\sim P}(y-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2 = E_{(x,y)\sim P}(y-\mathrm{w}^{*\mathrm{T}}\phi(x))^2 + E_{(x,y)\sim P}(\mathrm{w}^{*\mathrm{T}}\phi(x)-\hat{\mathrm{w}}^{\mathrm{T}}\phi(x))^2$$

Good news: Adding more data can only help in our regression problem