10-701

**Machine Learning**Regression

Where we are

sDensity t

Prob

**√**

u

p

n

I

Estimator

ability

s t

u

p

n

I

s t

u

p

n

I

Classifier Predict category

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

Reviews (out of 5 stars)

$ Distance Cuisine (out of

10)

| 30 | 21 | 7 |
| --- | --- | --- |
| 15 | 12 | 8 |
| 27 | 53 | 9 |

score

of these factors (how important are they with respect to the others).

• Assume we would like to build a recommender system for ***ranking*** potential restaurants based on an individuals’ preferences

• If we have many observations we may be able to recover the weights



4 8.5 2 7.8 5 6.7 3 20 5 6 5.4

**?**

Linear regression

• Given an input x we would like

to compute an output y

• For example:

Y

- Predict height from age

- Predict Google’s price from

Yahoo’s price

- Predict distance from wall

using sensor readings

X

Note that now Y can be **continuous**

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

Y

following equation:

What we are

trying to predict

y = wx+ε

Observed values

X

where w is a parameter and ε

represents measurement or

other noise

Linear regression

Y*y* = *wx* + ε

• Our goal is to estimate *w* from a training data

of <xi,yi> pairs

• One way to find such relationship is to

minimize the a least squares error:

∑ −

*y*2

argmin ( )X

*w iwxi*

*i*

• Several other approaches can be used as well

• So why least squares?

- minimizes squared distance between measurements and predicted line

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

- has a nice probabilistic interpretation - easy to compute

Solving linear regression using least squares minimization

• You should be familiar with this by now …

• We just take the derivative w.r.t. to w and set to 0:

∂

∑ ∑

2

( ) 2 ( )*y wx x y wx* − = − − ⇒

∂

*i i i*

*w*

*i*

*i i*

*i*

∑

2 ( ) 0 *x y wx*

− = ⇒

*i i i*

*i*

∑ ∑

2

*x y wx*

= ⇒

*i i i*

*i i*

*w*

=

∑ *i*

*x y i i*

∑

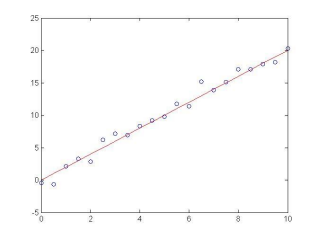
*x*

2 *i*

*i*

Regression example

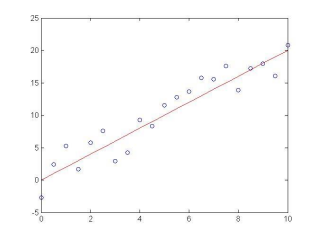
• Generated: w=2

• Recovered: w=2.03 

• Noise: std=1

Regression example

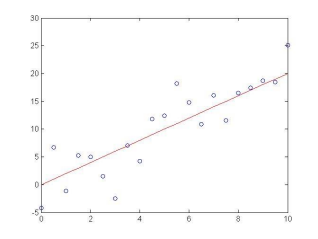
• Generated: w=2

• 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

Y

• What if the line does not?

• No problem, simply change the

model to

y = w0 + w1x+ε

• Can use least squares to determine w0, w1

∑ *i*− *i*

*y w x*

w0

∑ −

*x y w*

( )

X

*i i*

0

1

*w* 2

*wi*

=

*i*

0

=

*n*

1

∑ *i*

*x*

*i*

Bias term

• So far we assumed that the

line passes through the origin

Y

• What if the line does not?

• No problem, simply change the

model to

y = w0 + w1x+ε

Just a second, we will soon

give a simpler solution

• Can use least squares to

determine w0, w1

∑ *i*− *i*

*y w x*

w0

∑ −

*x y w*

( )

X

*i i*

0

1

*w* 2

*wi*

=

*i*

0

=

*n*

1

∑ *i*

*x*

*i*

Multivariate regression

• What if we have several inputs?

- Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task

• This becomes a multivariate linear regression problem

• Again, its easy to model:

y = w0 + w1x1+ … + wkxk + ε

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 Not all functions can be

the Google prediction task

approximated using the input

• This becomes a multivariate regression problem

values directly

• Again, its easy to model:

y = w0 + w1x1+ … + wkxk + ε

y=10+3x12-2x22+ε

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

2 2

*y w w x*  *w x*

= + + + +ε

0 1 1 *k k*

Non-Linear basis function

• What type of functions can we use?

• A few common examples:

- Polynomial: φj(x) = xjfor j=0 … n

- Gaussian: - Sigmoid: €

€

φ *j*(*x*) =(*x* − μ *j*)

2σ *j*2

φ *j*(*x*) =1

1+ exp(−*sjx*)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

*n*

∑

*y* = *wj*φ *j*(*x*) *j*= 0

• Where φ*j(x)* can be either *xj*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

*n*

∑

Our goal is to minimize the following loss function:

∑2

∑ )

*J*(w) = (*yi*− *wj*φ *j*(*xi*)

*y* = *wj*φ *j*(*x*)

*j*= 0

*w* – vector of dimension k+1

*i*

φ*(xi)* – vector of dimension k+1

*j*

Moving to vector notations we get: ∑

*J*(w) = (*yi*− wTφ(*xi*))2

*i*

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

*yi* – a scaler€

∂

∑ = 2 (*yi*− wTφ(*xi*))

∂*w*(*yi*− wTφ(*xi*))2 *i*

∑ φ(*xi*)T *i*

∑ φ(*xi*)T= 0 ⇒

Equating to 0 we get2 (*yi*− wTφ(*xi*))

*i*

⎡

⎣ ⎢ ⎤ ⎦ ⎥

∑ φ(*xi*)T= wTφ(*xi*)

*yi*

*i*

∑ φ(*xi*)T *i*

LMS for general linear regression problem ∑

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

*J*(w) = (*yi*− wTφ(*xi*))2 *i*

∂

∑ = 2 (*yi*− wTφ(*xi*))

∂*w*(*yi*− wTφ(*xi*))2

∑ φ(*xi*)T

*i*

*i*

€

Equating to 0 we get

∑ φ(*xi*)T= 0 ⇒ 2 (*yi*− wTφ(*xi*))

*i*

⎣ ⎢ ⎤ ⎦ ⎥

⎡

∑ φ(*xi*)T= wTφ(*xi*)

*yi*

*i*

⎛

∑ φ(*xi*)T *i*

Define:

Φ =

⎜ ⎜ ⎜ ⎜ ⎞ ⎠ ⎟ ⎟ ⎟ ⎟

φ0(*x*1) φ1(*x*1) φ *m*(*x*1) φ0(*x*2) φ1(*x*2) φ *m*(*x*2)

€

φ0(*xn*) φ1(*xn*) φ *m*(*xn*)

⎝

Then deriving w

we get:w = (Φ*T*Φ)−1Φ*T*y€

LMS for general linear regression problem ∑

*J*(w) = (*yi*− wTφ(*xi*))2

Deriving w we get:

*i*

w = (Φ*T*Φ)−1Φ*T*y

€

n entries vector k+1 entries vector

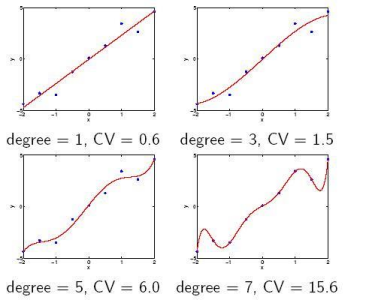
€

n by k+1 matrix

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* = wTφ(*x*) + ε

€

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

w = (Φ*T*Φ)−1Φ*T*y

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*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

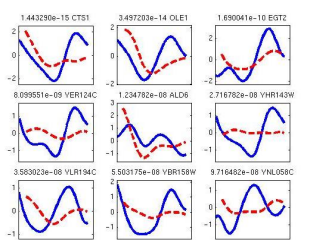
~~€~~

€

*y* = *a*3*x*3+ *b*3*x*2+ *c*3*x* + *d*3

€

Splines

• 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

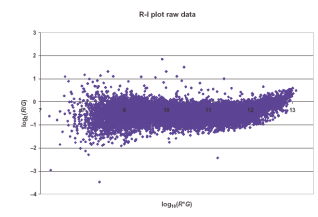
regions for which they have been defined

• 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 Ωx centered at x to assign weight to points in x’s vicinity

• Next we solve the following *weighted* regression problem ∑*i*− wTφ(*xi*))2

min wΩ*x*(*xi*)(*y*

*i*

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

€

Ωx(x1)=0.3

Ωx(x2)=0.7

Ωx(x)=0.9

**x1 x2 x**

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 Ω*x*(*xi*) =12πσ−(*x*−*xi*)2

*e*

2σ2

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