

10-701

Machine Learning

Regression

Where we are



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 for **ranking** potential restaurants 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)	score
4	30	21	7	8.5
2	15	12	8	7.8
5	27	53	9	6.7
3	20	5	6	5.4

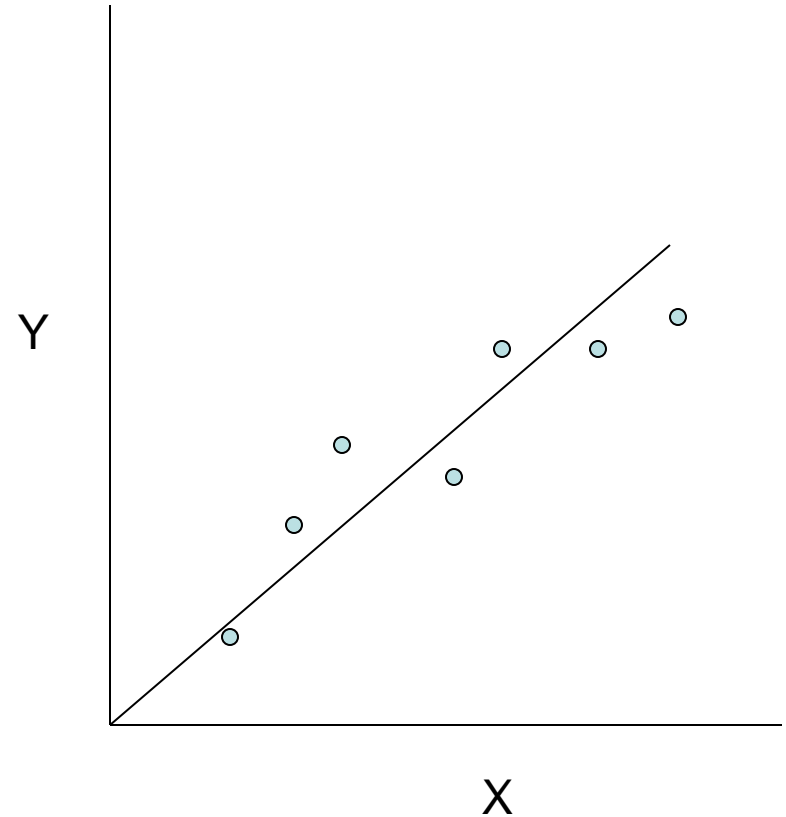


?



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 using sensor readings



Note that now Y can be
continuous

Linear regression

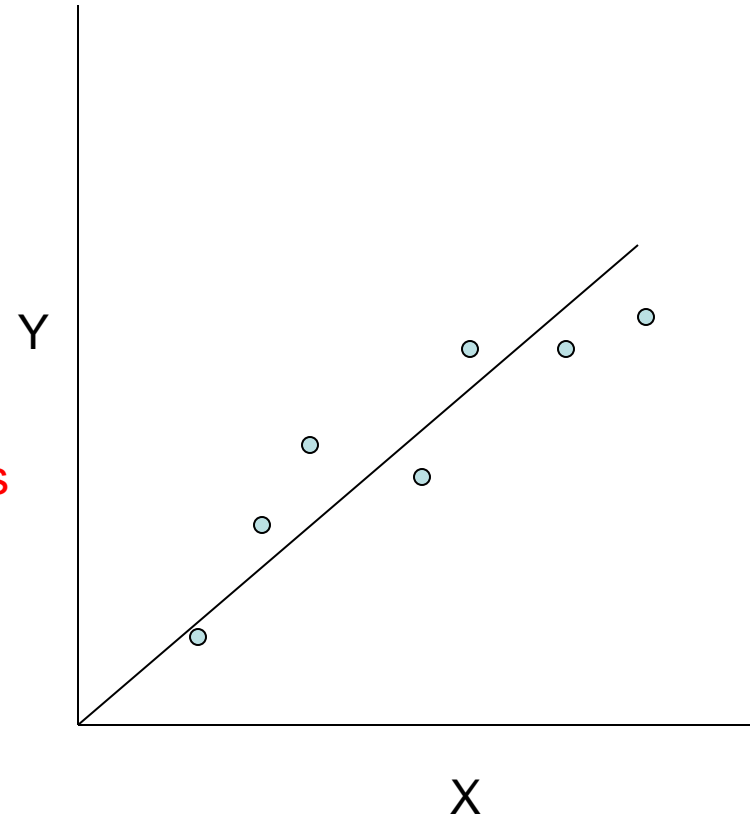
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are trying to predict \swarrow

$$y = wX + \varepsilon$$

\nearrow Observed values

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

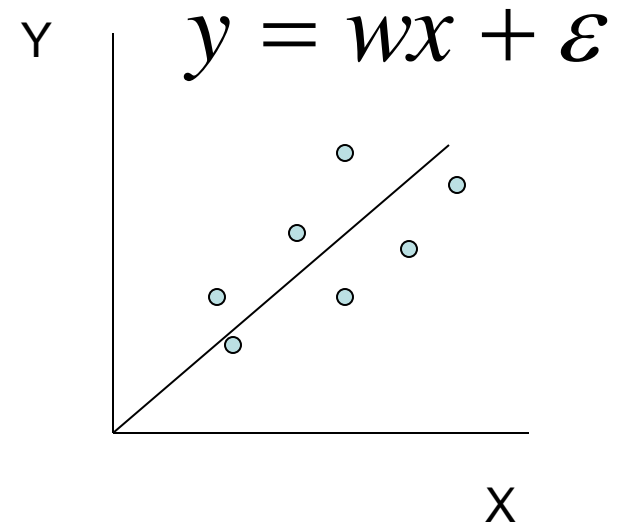


Linear regression

- Our goal is to estimate w from a training data of $\langle x_i, y_i \rangle$ pairs
- One way to find such relationship is to minimize the a least squares error:

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

- Several other approaches can be used as well
- So 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 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:

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

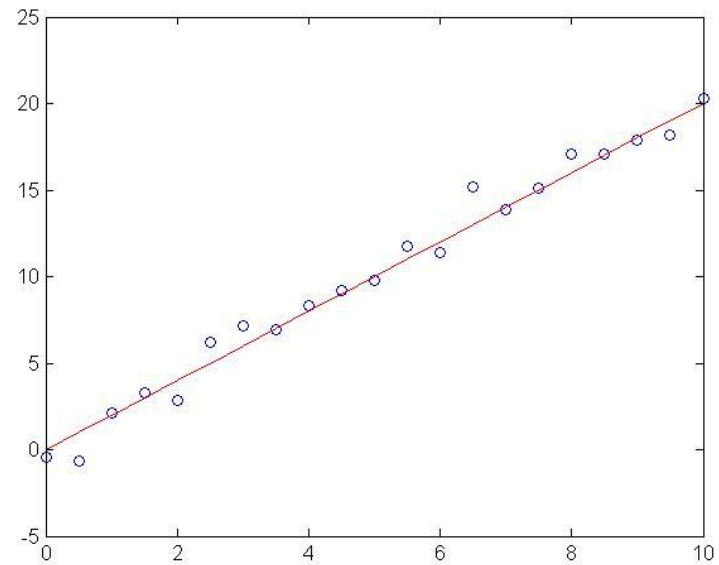
$$2 \sum_i x_i (y_i - wx_i) = 0 \Rightarrow$$

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

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

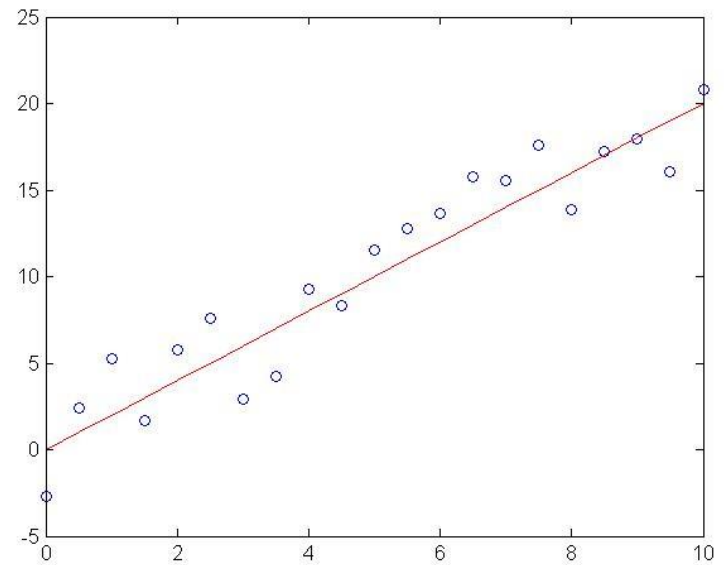
Regression example

- Generated: $w=2$
- Recovered: $w=2.03$
- Noise: $\text{std}=1$



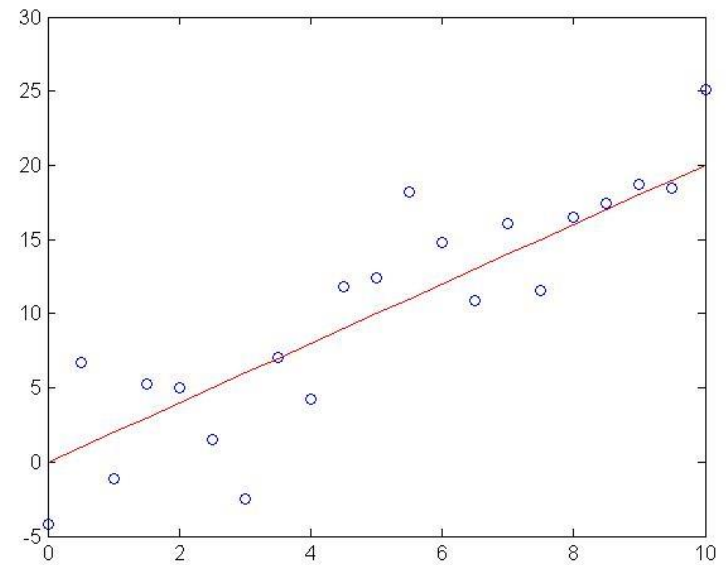
Regression example

- Generated: $w=2$
- Recovered: $w=2.05$
- Noise: $\text{std}=2$



Regression example

- Generated: $w=2$
- Recovered: $w=2.08$
- Noise: $\text{std}=4$



Bias term

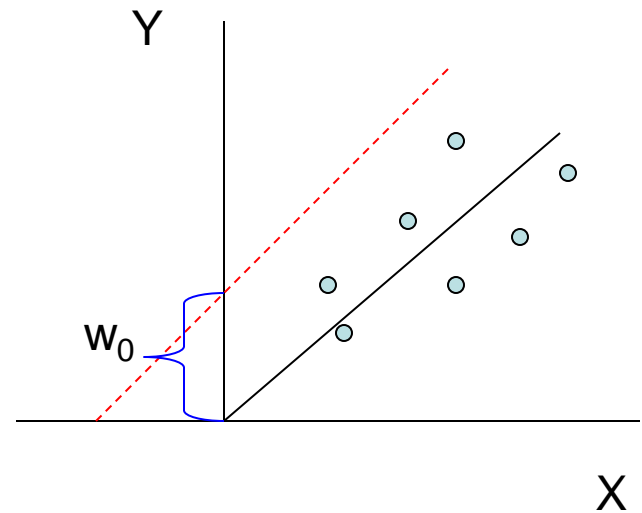
- So far we assumed that the line passes through the origin
- What if the line does not?
- No problem, simply change the model to

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

- Can use least squares to determine w_0 , w_1

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

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

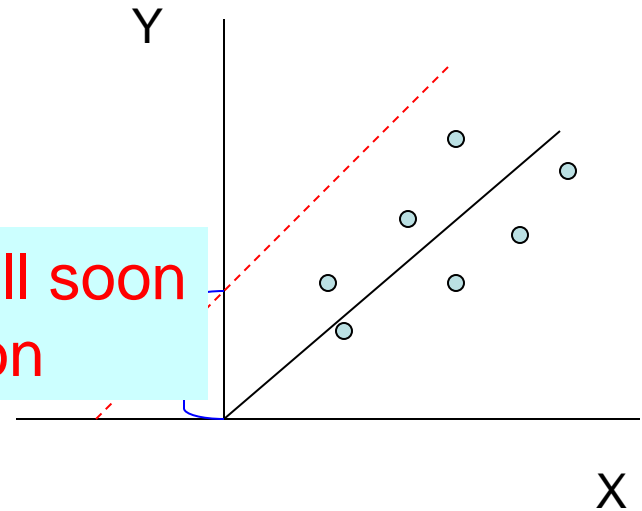


Bias term

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

$y = w_0 + w_1 x$ Just a second, we will soon give a simpler solution

- Can use least squares to determine w_0 , w_1



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

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

Multivariate regression

- What if we have several inputs?
 - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task
- This becomes a multivariate linear regression problem
- Again, its easy to model:

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

Google's stock price



Yahoo's stock price

Microsoft's stock price

Multivariate regression

- What if we have several inputs?
 - Stock prices for Yahoo, Microsoft and Ebay for the Google stock price problem
- This becomes a linear regression problem
- Again, its easy to model:

Not all functions can be approximated using the input values directly

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

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

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

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

Non-Linear basis function

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

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

- 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

Our goal is to minimize the following loss function:

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

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

\mathbf{w} – vector of dimension $k+1$
 $\phi(x^i)$ – vector of dimension $k+1$
 y^i – a scalar

Moving to vector notations we get:

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

We take the derivative w.r.t \mathbf{w}

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

Equating to 0 we get $2 \sum_i (y^i - \mathbf{w}^T \phi(x^i)) \phi(x^i)^T = 0 \Rightarrow$

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

LMS for general linear regression problem

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

We take the derivative w.r.t \mathbf{w}

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

Equating to 0 we get

$$2 \sum_i (y^i - \mathbf{w}^T \phi(x^i)) \phi(x^i)^T = 0 \Rightarrow \\ \sum_i y^i \phi(x^i)^T = \mathbf{w}^T \left[\sum_i \phi(x^i) \phi(x^i)^T \right]$$

Define:

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

Then deriving \mathbf{w}
we get:

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

LMS for general linear regression problem

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

Deriving \mathbf{w} we get:

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

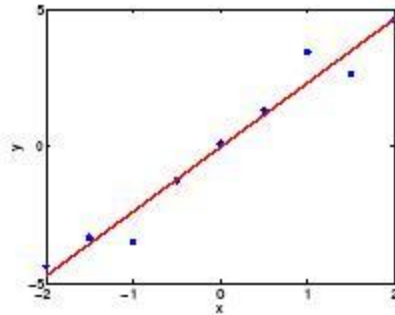
\mathbf{w} is a $k+1$ entries vector

$\Phi^T \Phi$ is an n by $k+1$ matrix

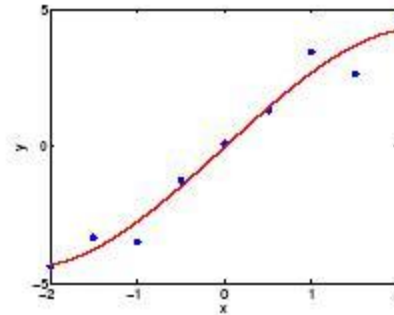
\mathbf{y} is an n entries vector

This solution is
also known as
'psuedo inverse'

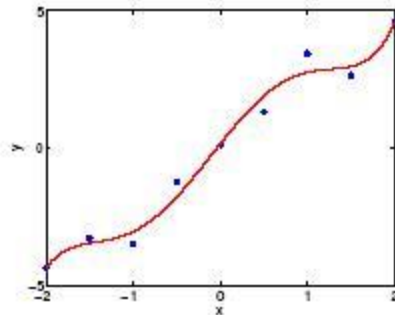
Example: Polynomial regression



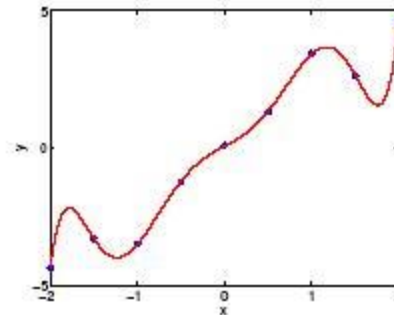
degree = 1, CV = 0.6



degree = 3, CV = 1.5



degree = 5, CV = 6.0



degree = 7, CV = 15.6

A probabilistic interpretation

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

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

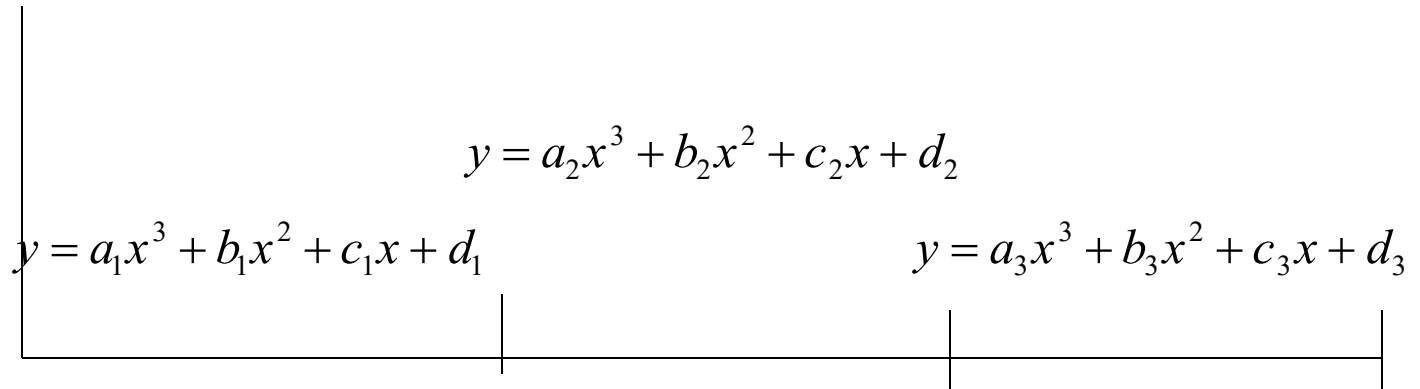
$$\mathbf{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}$$

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

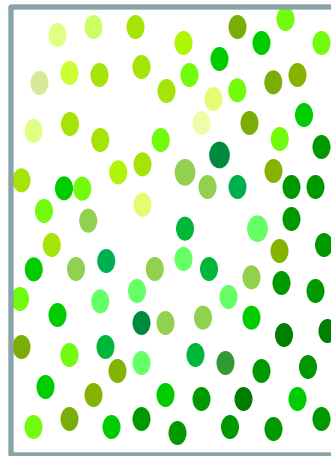
- Instead of fitting one function for the entire region, fit a set of piecewise (usually cubic) polynomials satisfying continuity and smoothness constraints.
- Results in smooth and flexible functions without too many parameters
- Need to define the regions in advance (usually uniform)



LOCAL, KERNEL REGRESSION

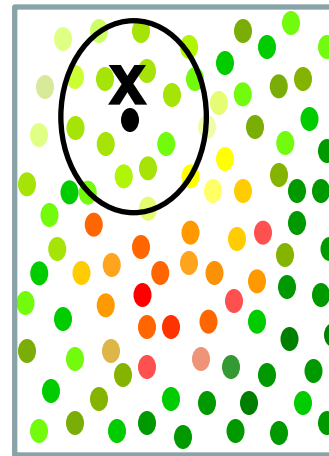
Local Kernel Regression

- What is the temperature in the room?
at location x ?



$$\hat{T} = \frac{1}{n} \sum_{i=1}^n Y_i$$

Average



$$\hat{T}(x) = \frac{\sum_{i=1}^n Y_i \mathbf{1}_{\|X_i - x\| \leq h}}{\sum_{i=1}^n \mathbf{1}_{\|X_i - x\| \leq h}}$$

“Local” Average

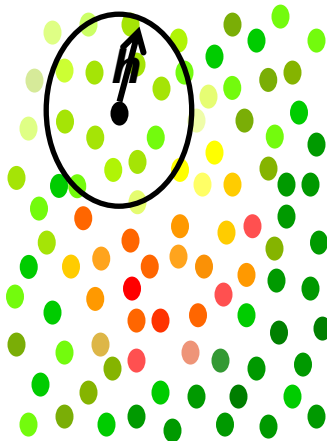
Local Average Regression

#pts in h ball around X

$$\Rightarrow \hat{f}_n(X) = \hat{\beta} = \sum_{i=1}^n w_i Y_i$$

$$= \frac{1}{n_X^h} \sum_{i=1}^n Y_i \mathbf{1}_{|X-X_i| \leq h}$$

Sum of Ys in h ball around X



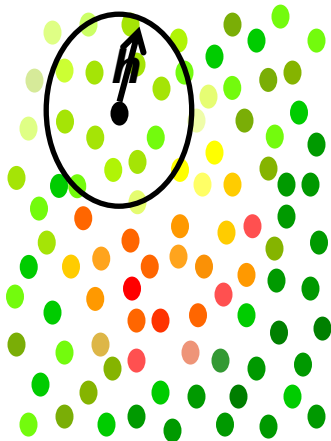
**Recall: NN classifier
with majority vote**

Here we use Average instead

Nadaraya-Watson Kernel Regression

$$\Rightarrow \hat{f}_n(X) = \hat{\beta} = \sum_{i=1}^n w_i Y_i$$
$$= \frac{1}{n_X h} \sum_{i=1}^n Y_i \mathbf{1}_{|X-X_i| \leq h}$$

Sum of Ys in h ball around X



$$w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{X-X_i}{h}\right)}$$

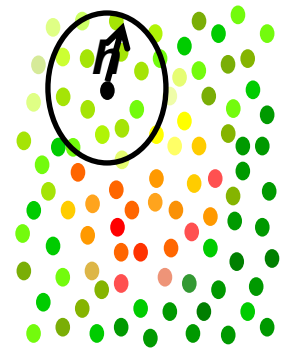
boxcar kernel :

$$K\left(\frac{X-X_i}{h}\right) = \mathbf{1}_{|X-X_i| \leq h}$$

Local Kernel Regression

- Nonparametric estimator akin to kNN
- Nadaraya-Watson Kernel Estimator

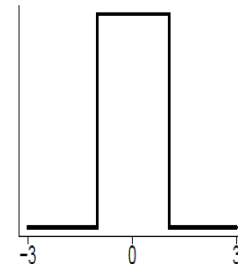
$$\hat{f}_n(X) = \sum_{i=1}^n w_i Y_i \quad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{X-X_i}{h}\right)}$$



- Weight each training point based on distance to test point
- Boxcar kernel yields local average

boxcar kernel :

$$K(x) = \frac{1}{2}I(x),$$



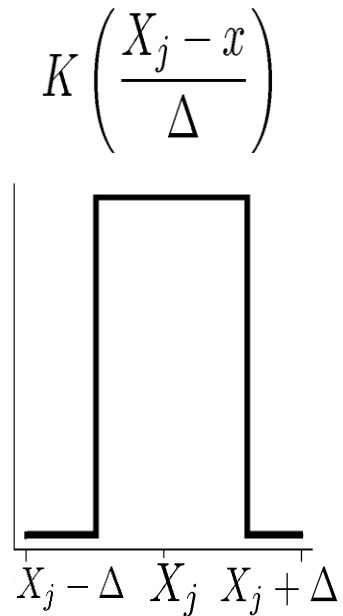
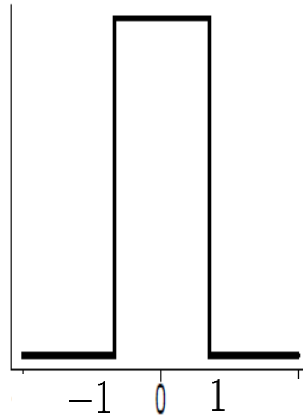
Kernels

$$K(x) \geq 0,$$

$$\int K(x)dx = 1$$

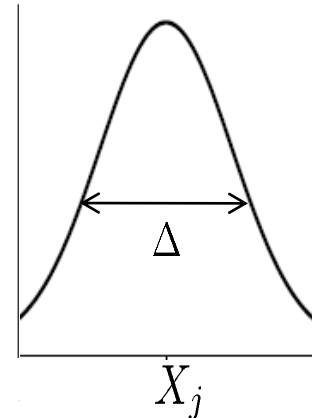
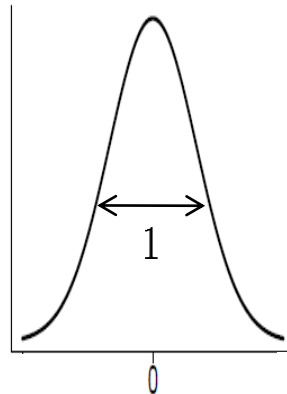
boxcar kernel :

$$K(x) = \frac{1}{2}I(x),$$



Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$$



Kernel Regression as Weighted Least Squares

$$\min_f \frac{1}{n} \sum_{i=1}^n w_i (f(X_i) - Y_i)^2$$

$$\frac{1}{n} \sum_{i=1}^n w_i = 1$$

Weighted Least Squares

Weigh each training point based on distance to test point

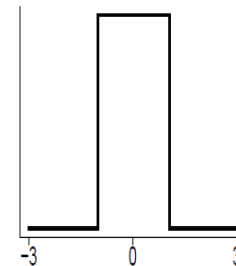
$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{X - X_i}{h}\right)}$$

K – Kernel

h – Bandwidth of kernel

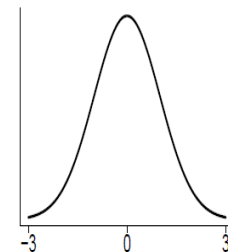
boxcar kernel :

$$K(x) = \frac{1}{2} I(x),$$



Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$



Kernel Regression as Weighted Least Squares

set $f(X_i) = \beta$ (a constant)

$$\min_{\beta} \sum_{i=1}^n w_i (\underbrace{\beta}_{\text{constant}} - Y_i)^2$$

$$w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^n K\left(\frac{X-X_i}{h}\right)}$$

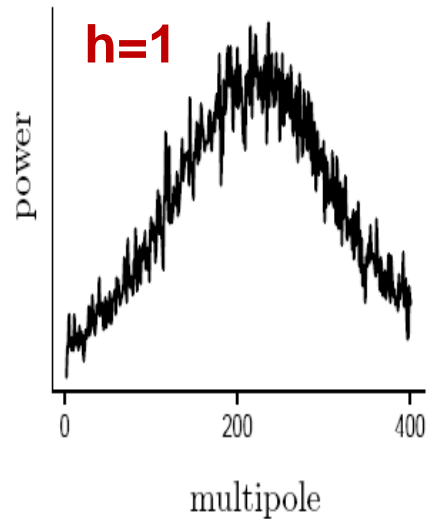
$$\frac{\partial J(\beta)}{\partial \beta} = 2 \sum_{i=1}^n w_i (\beta - Y_i) = 0$$

Notice that $\sum_{i=1}^n w_i = 1$

$$\Rightarrow \hat{f}_n(X) = \hat{\beta} = \sum_{i=1}^n w_i Y_i$$

Choice of kernel bandwidth h

Too small



$h=10$

Too small

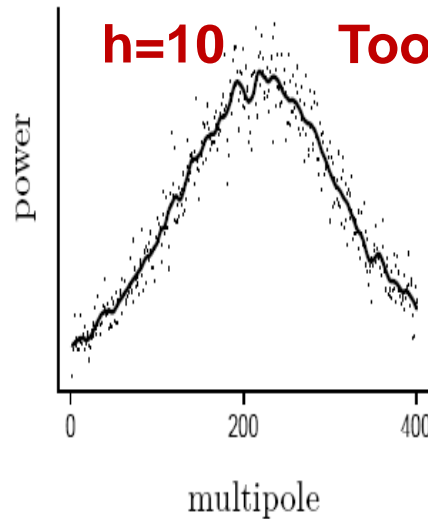
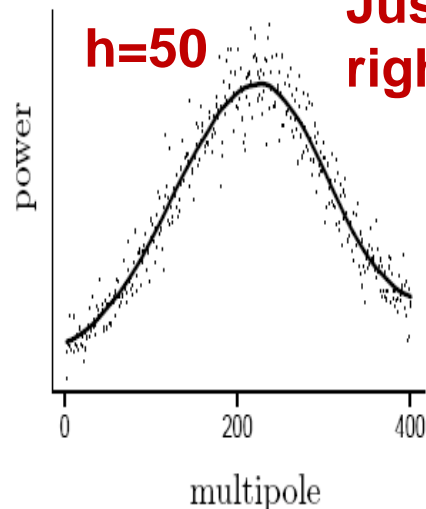


Image Source:
Larry's book – All
of Nonparametric
Statistics

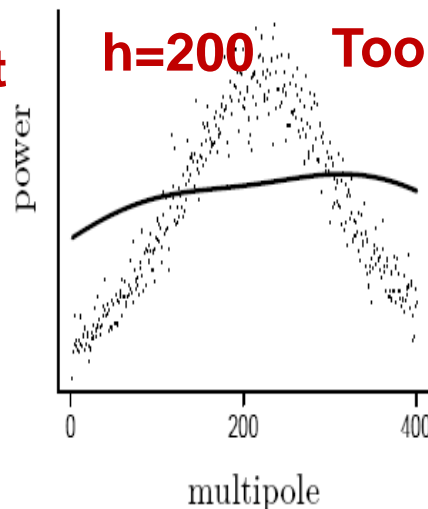
$h=50$

**Just
right**

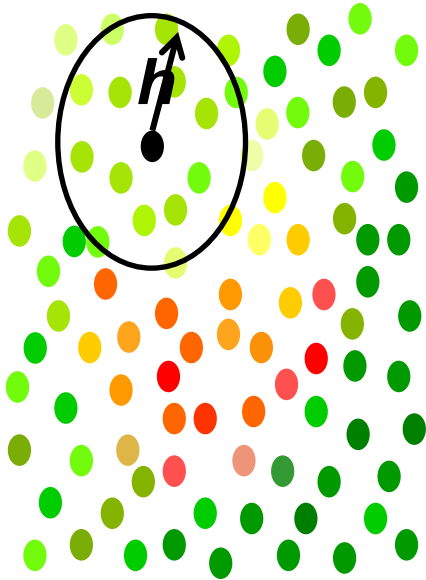


$h=200$

Too large



Choice of Bandwidth



Should depend on n , # training data
(determines variance)

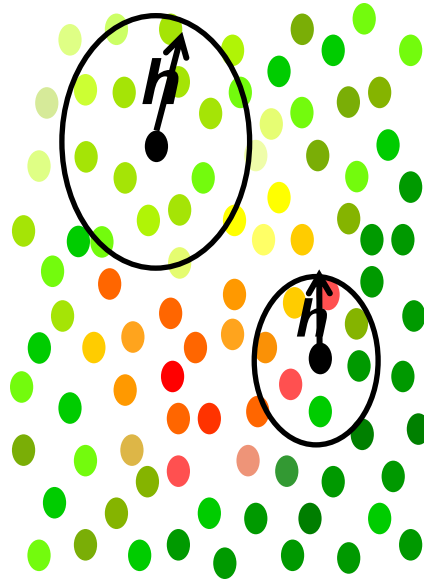
Should depend on smoothness of
function
(determines bias)

Large Bandwidth – average more data points, reduce noise (Lower variance)

Small Bandwidth – less smoothing, more accurate fit (Lower bias)

Bias – Variance tradeoff

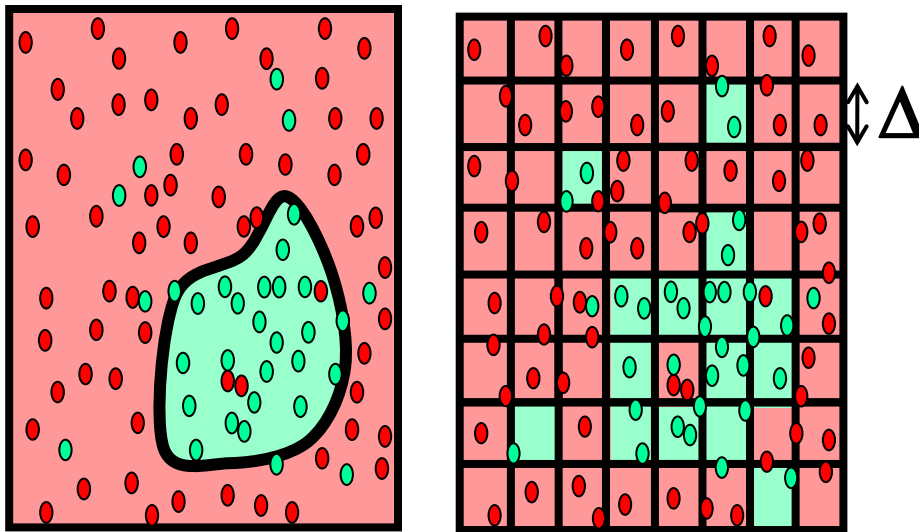
Spatially adaptive regression



If function smoothness varies spatially, we want to allow bandwidth h to depend on X

Local polynomials, splines, wavelets, regression trees ...

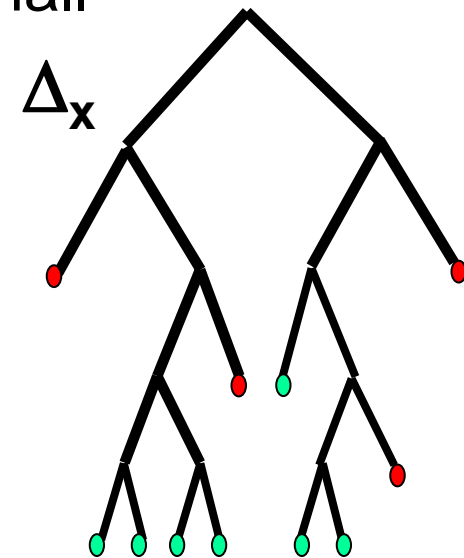
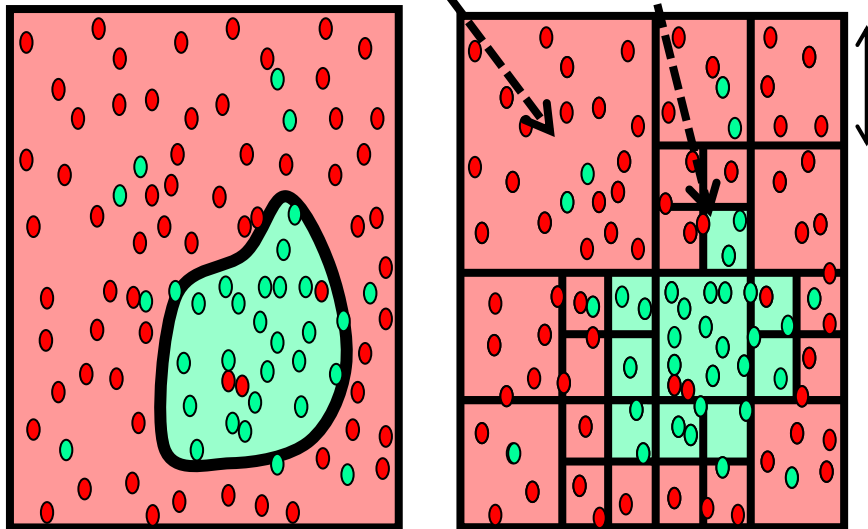
Local prediction



Histogram Classifier

Local Adaptive prediction

Let neighborhood size adapt to data –
small neighborhoods near decision
boundary (small bias), large
neighborhoods elsewhere (small
variance)



Majority vote
at each leaf

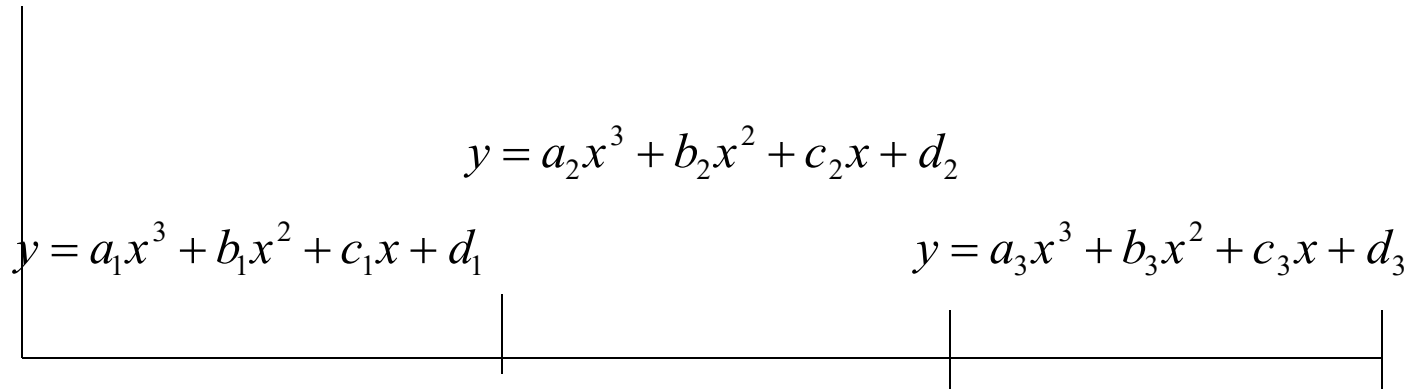
Decision Tree Classifier

Important points

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

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?



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)

