10701

Ensemble of trees: Begging and Random Forest

Bagging

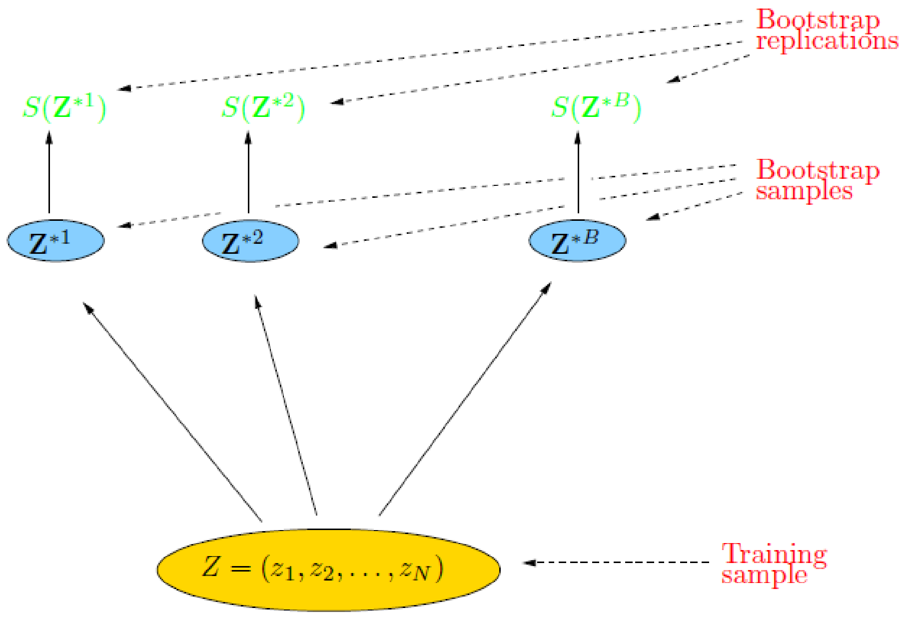
- Bagging or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

- For classification, a *committee* of trees each cast a vote for the predicted class.

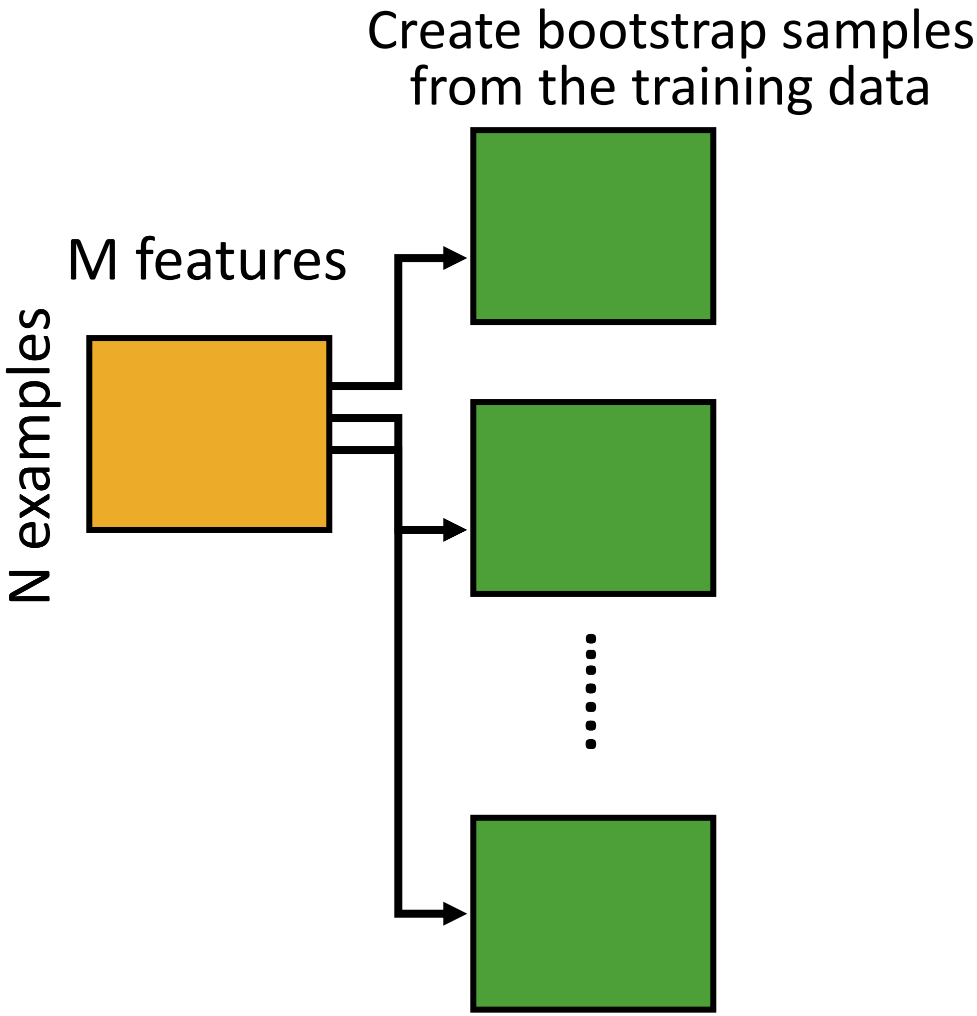
Bootstrap

The basic idea:

randomly draw datasets *with replacement* from the training data, each sample of *the same size*

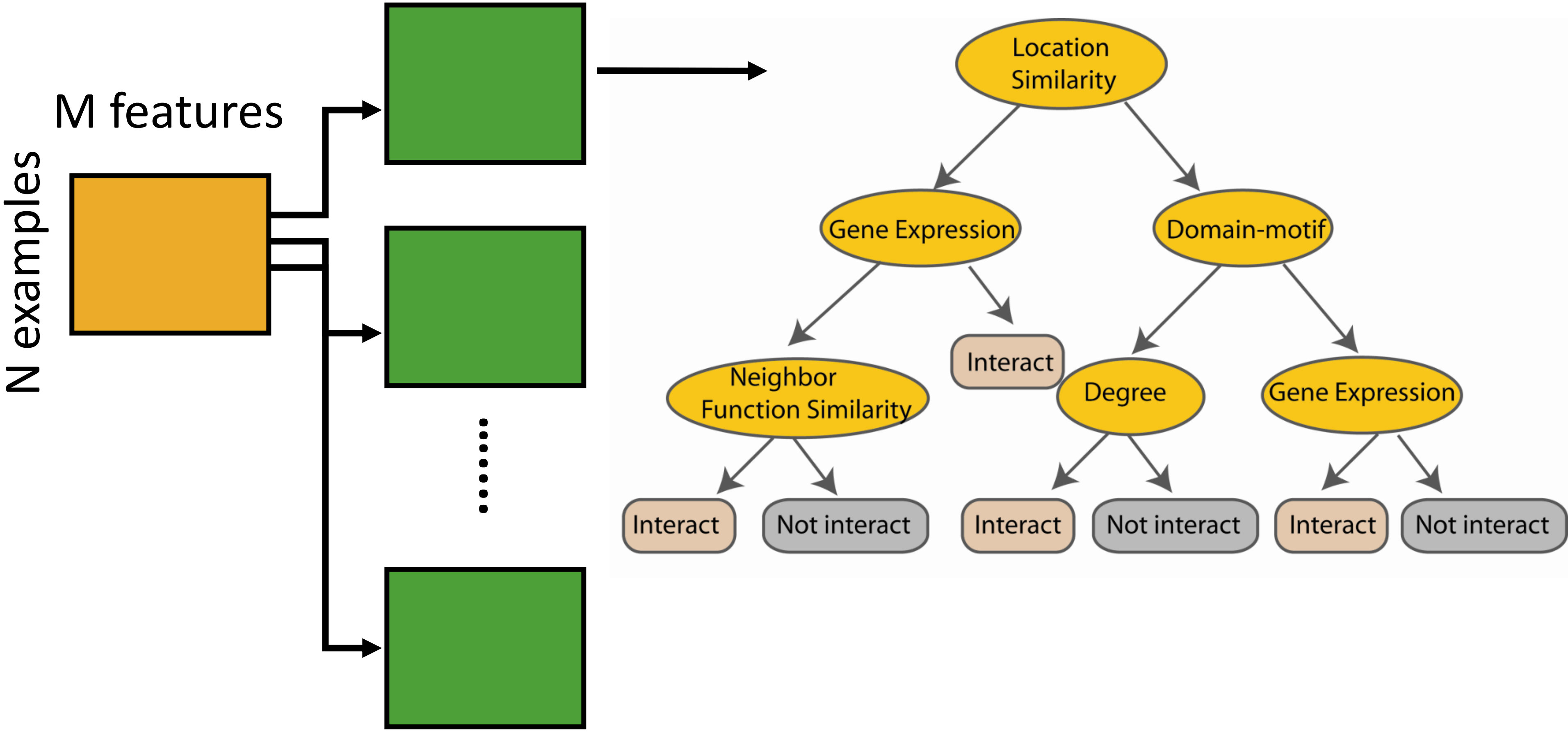


Bagging

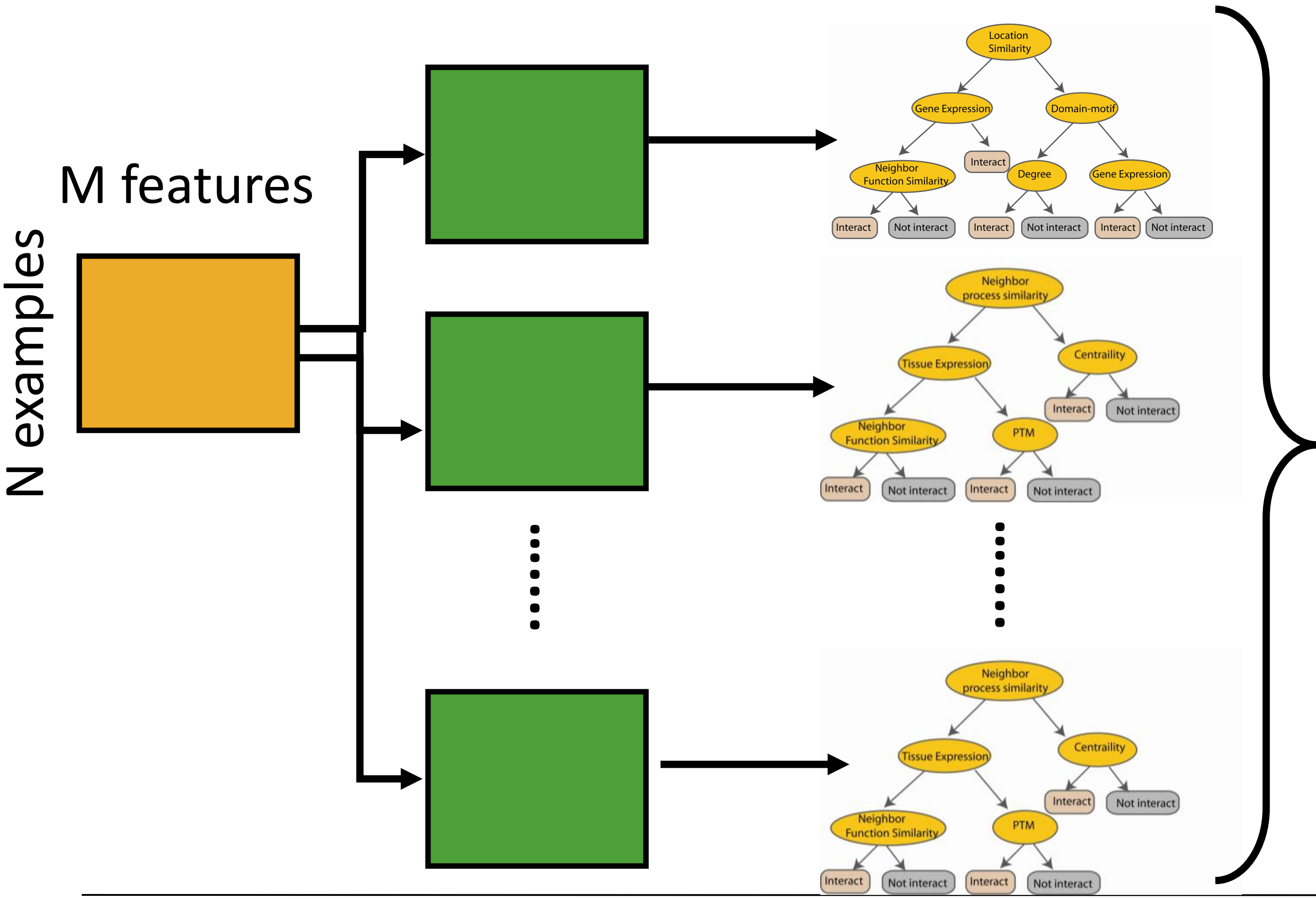

Create bootstrap samples
from the training data
M features
N examples

Random Forest Classifier

Construct a decision tree



Bagging tree classifier



Take the majority vote

Bagging

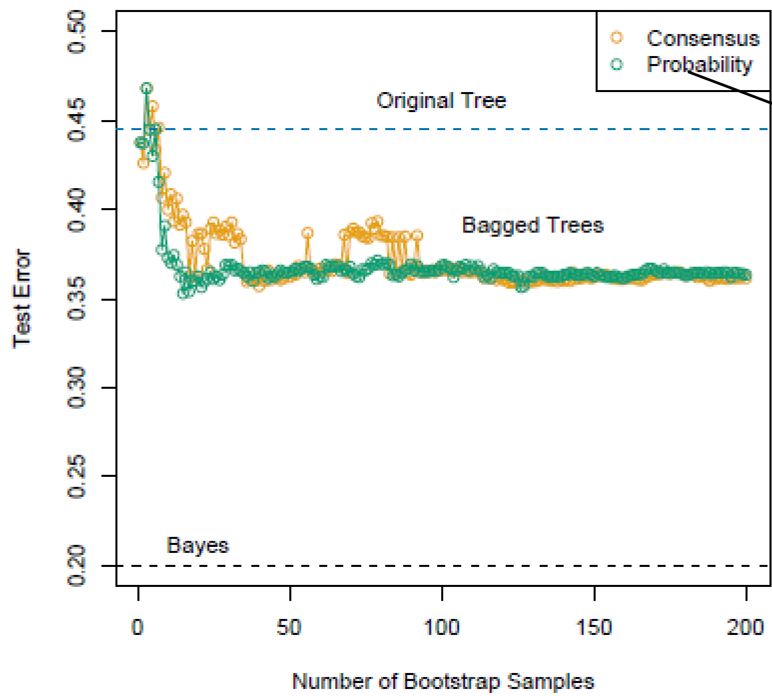
Z = $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

Z^{*b} where = 1,.., B..

The prediction at input x when bootstrap sample *b* is used for training

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

Bagging



Treat the voting Proportions as probabilities

FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

bagging helps under squared-error loss, in short because averaging reduces

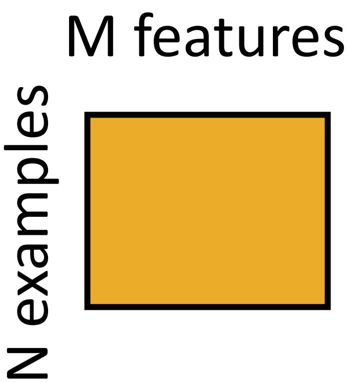
Hastie

Random forest classifier

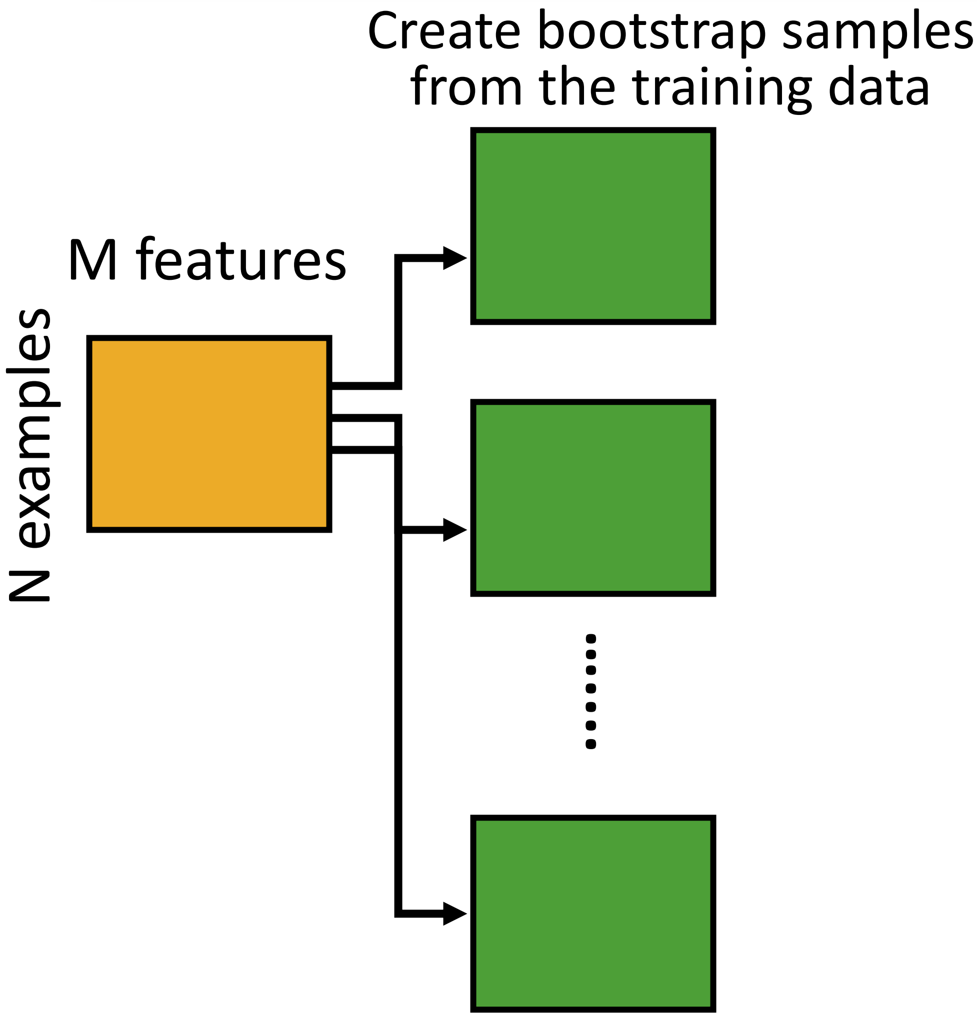
Random forest classifier, an extension to bagging which uses *a subset of the features* rather than the samples.

Random Forest Classifier

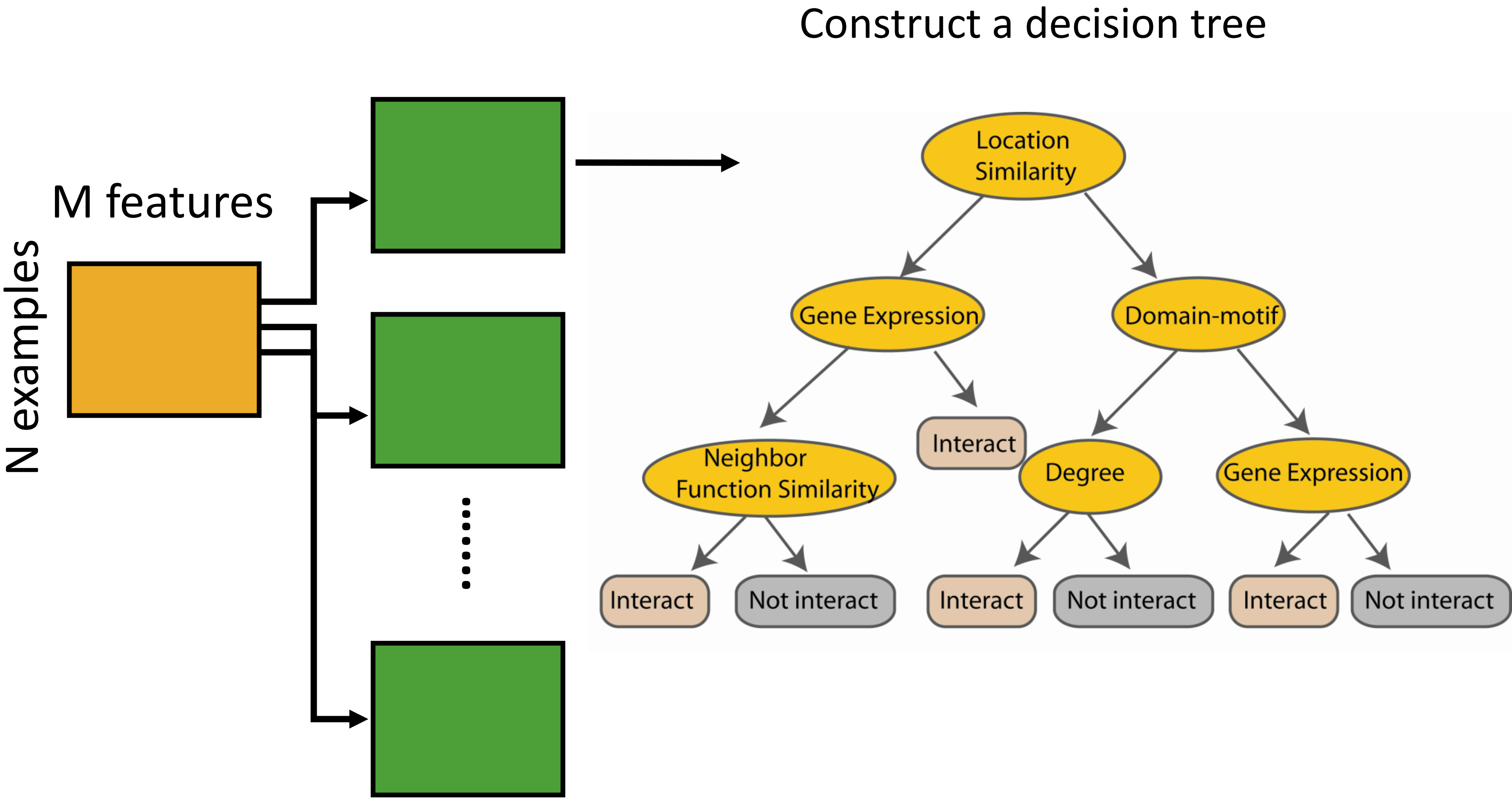
Training Data



Random Forest Classifier

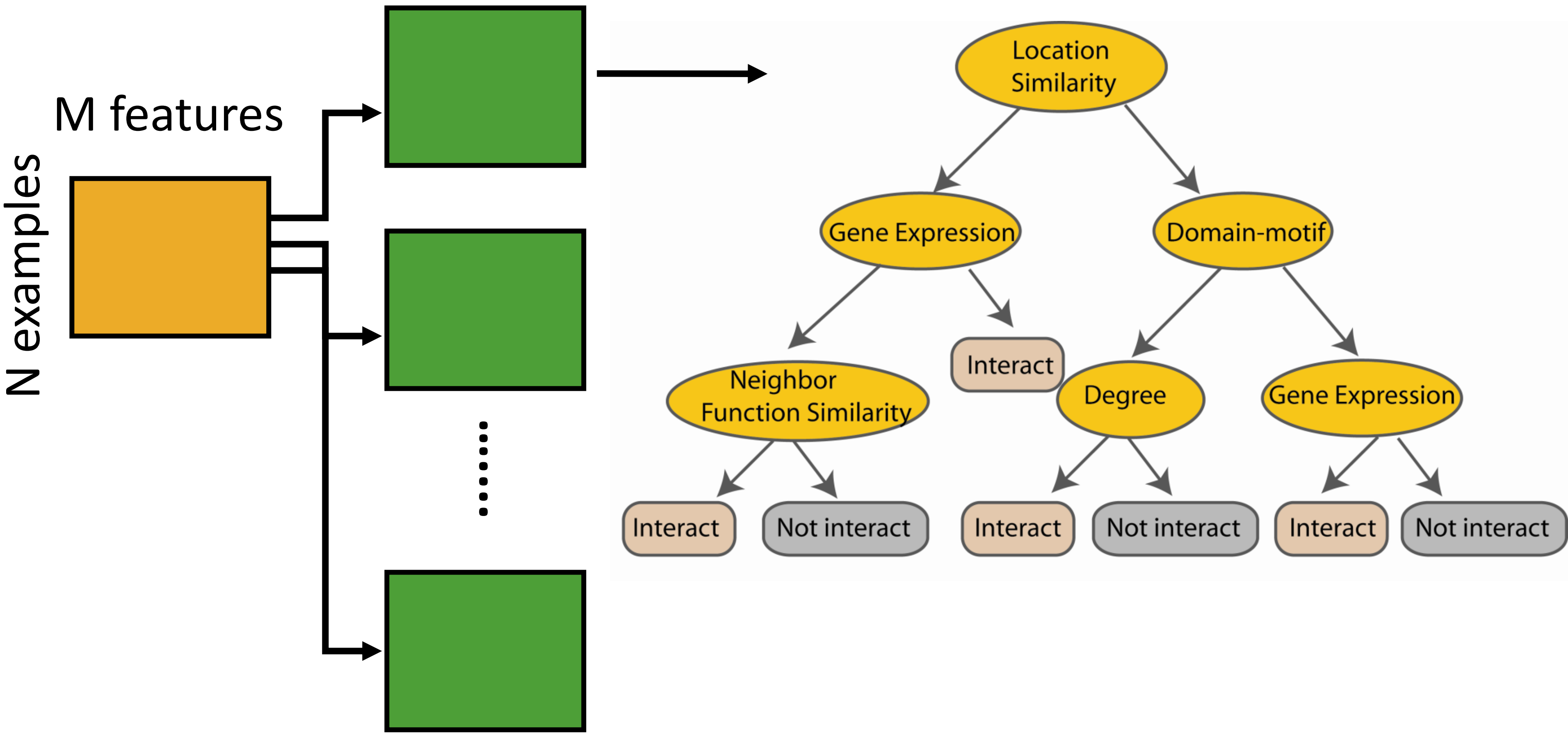


Random Forest Classifier



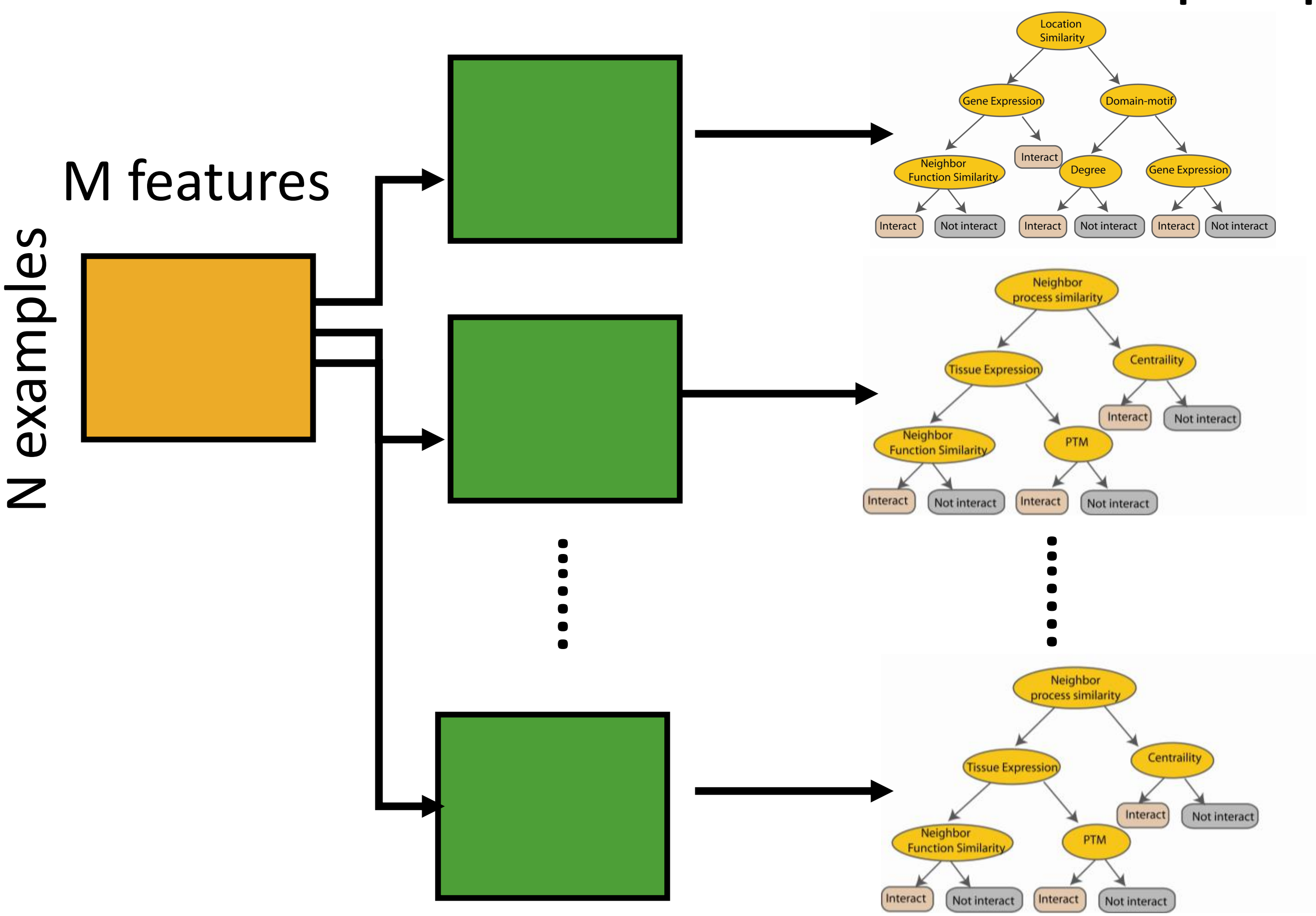
Random Forest Classifier

At each node in choosing the split feature choose only among *m*<*M* features

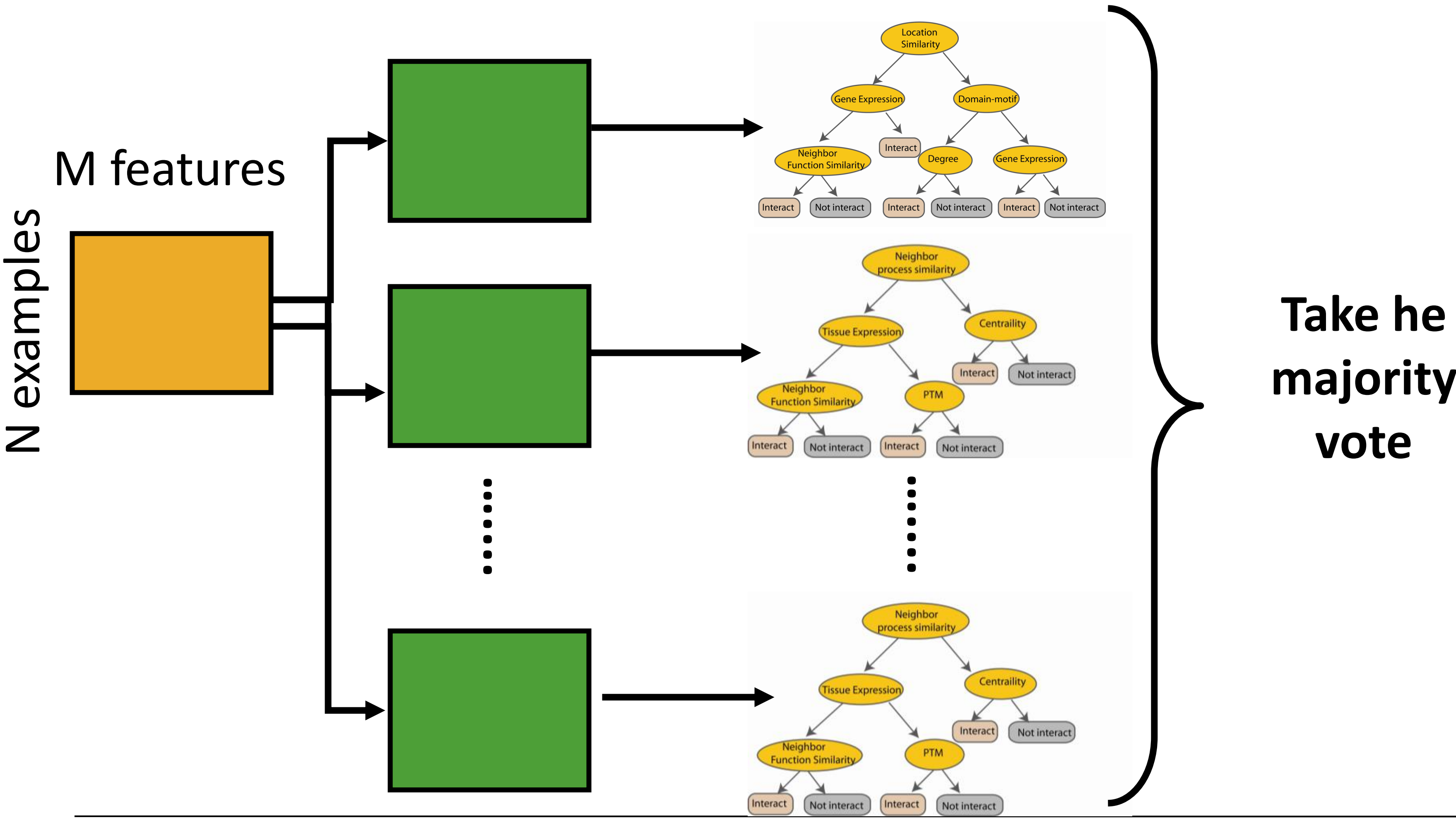


Random Forest Classifier

Create decision tree from each bootstrap sample



Random Forest Classifier



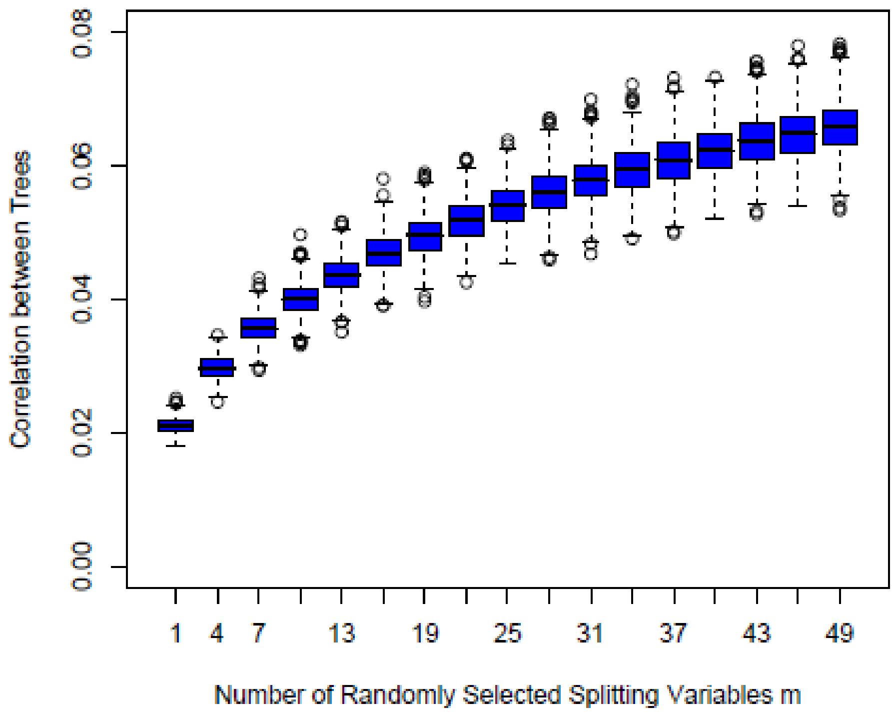
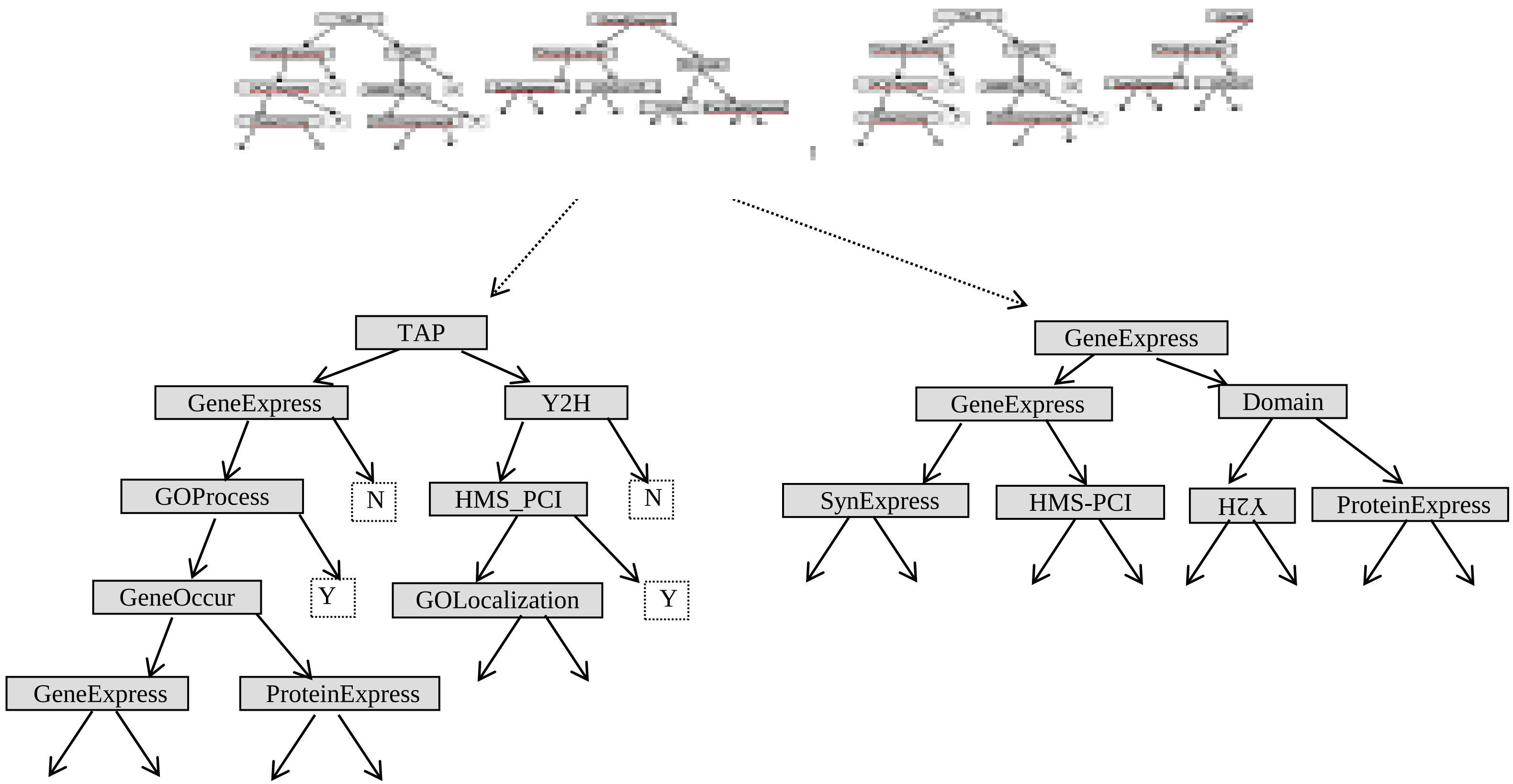


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

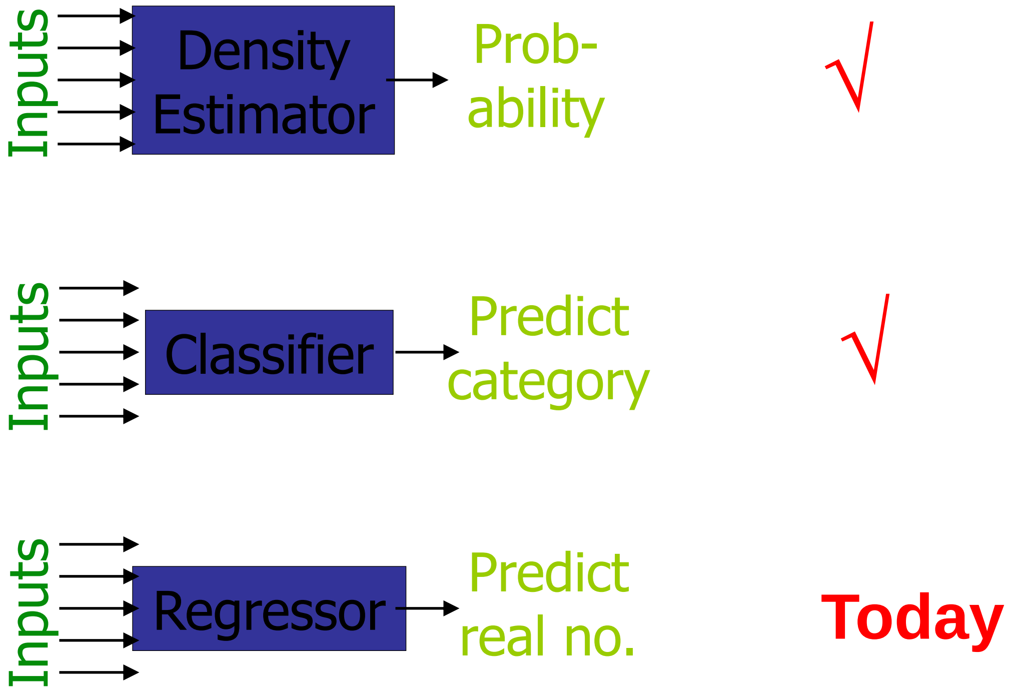
Random forest for biology



10-701
Machine Learning

Regression

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 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

?

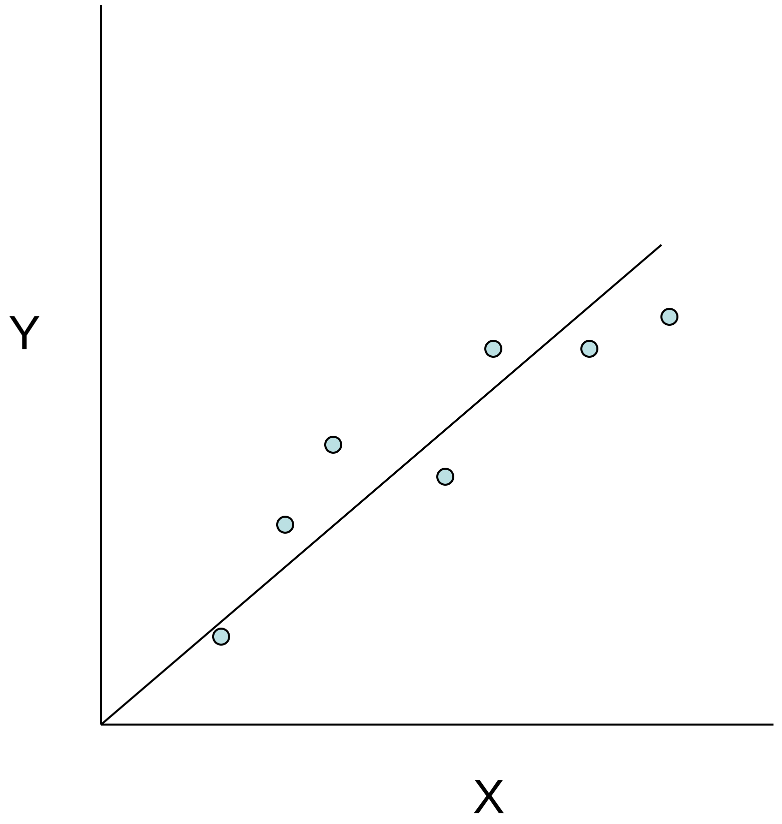
Linear regression

- 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

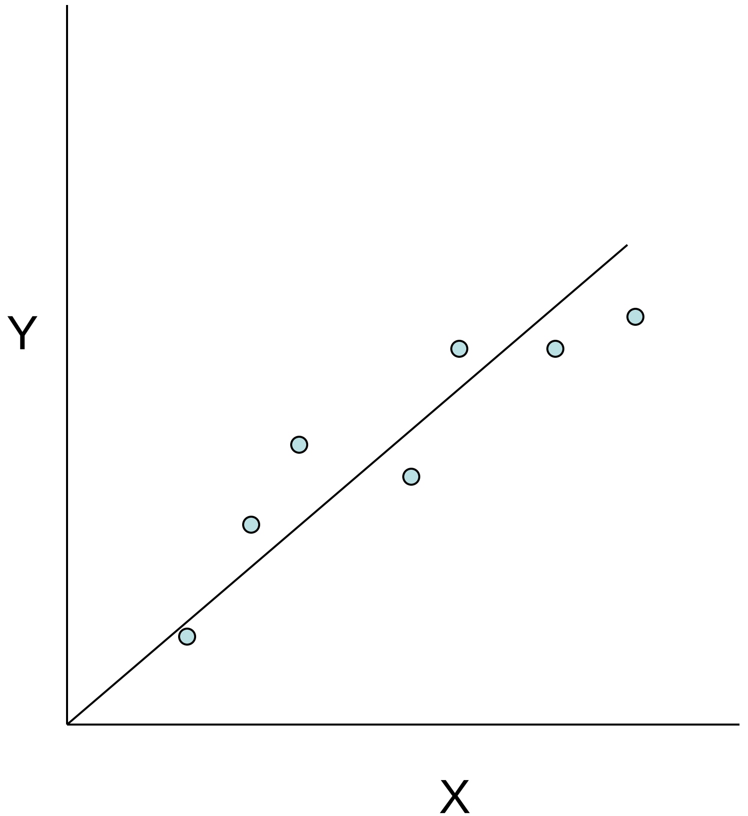
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are trying to predict

Observed values

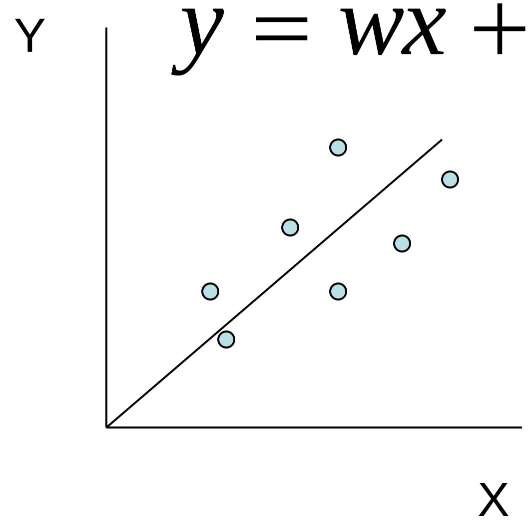
$$y = wx+\varepsilon$$

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
- 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:

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

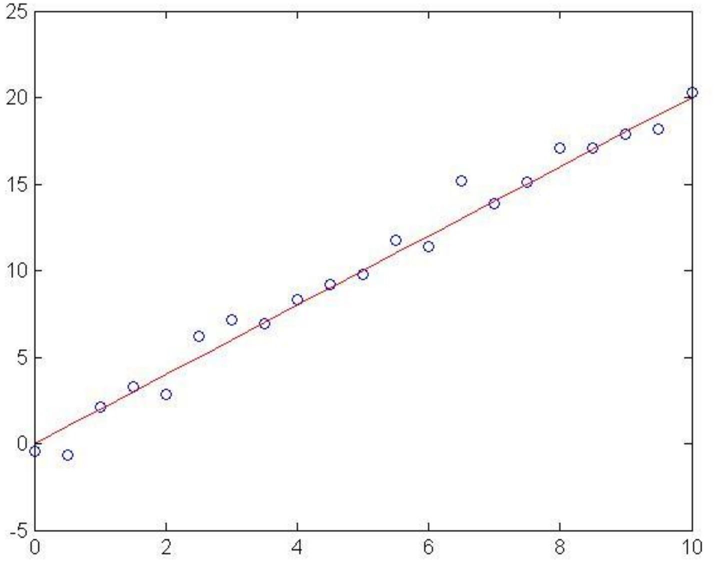
$$2\sum_i x_i(y_i - wx_i) = 0 \Rightarrow$$

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

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

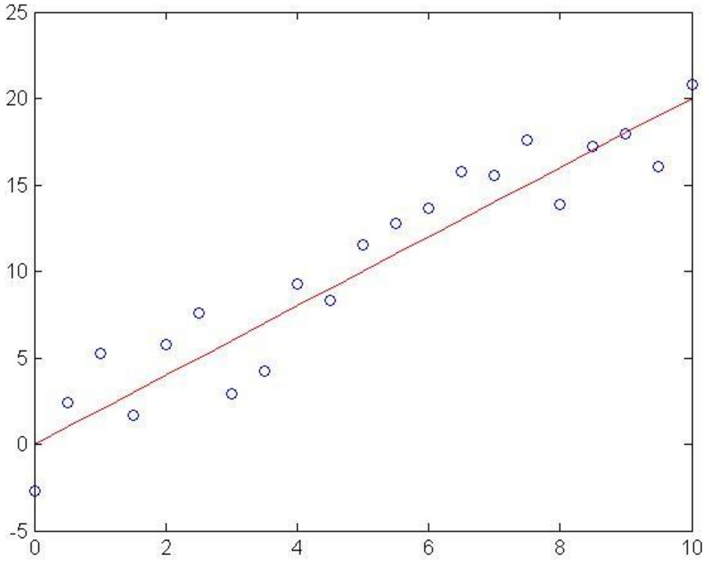
Regression example

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



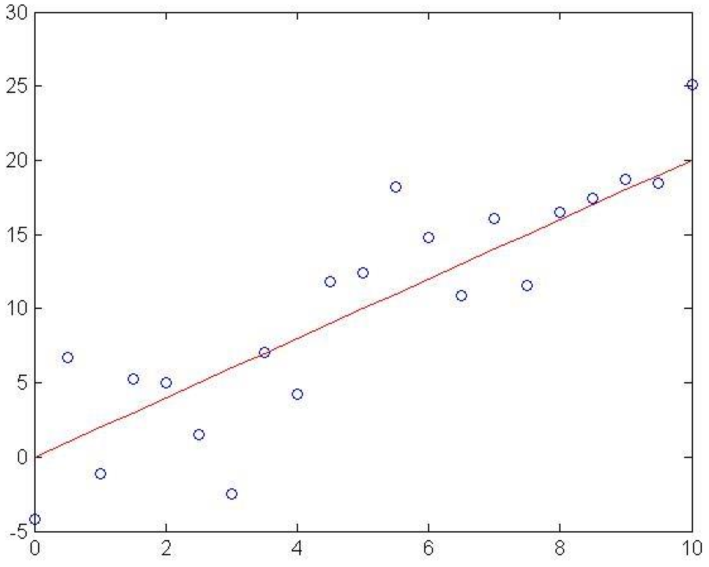
Regression example

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



Regression example

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

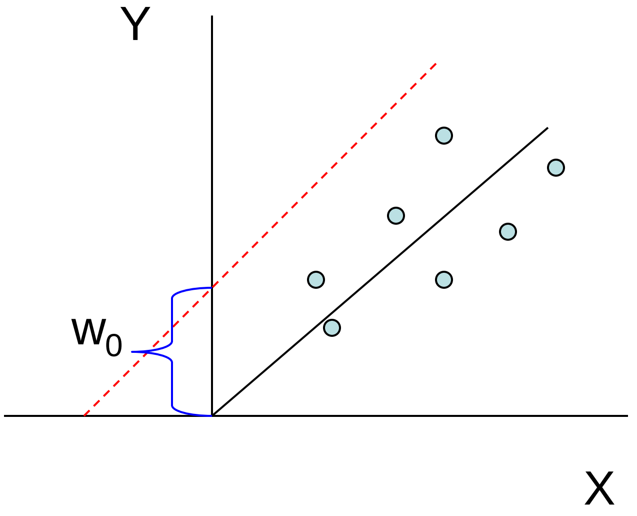


Bias term

- 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



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

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

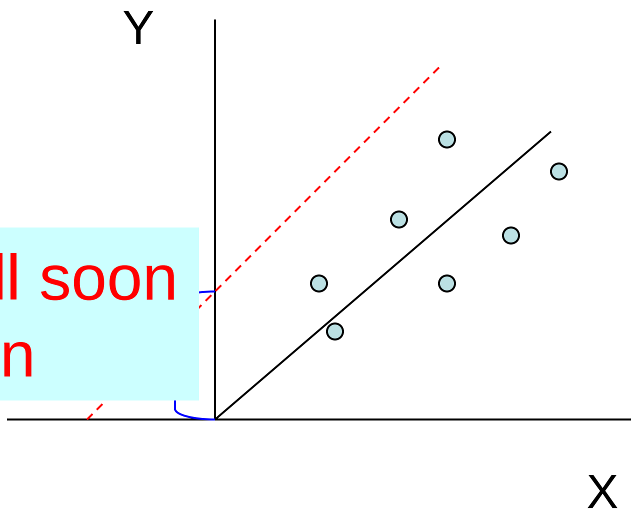
Bias term

- 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



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

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

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:

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

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 Goo

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

Not all functions can be approximated using the input values directly

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

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

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

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

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

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

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:

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

Then deriving w we get:

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

LMS for general linear regression problem

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

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

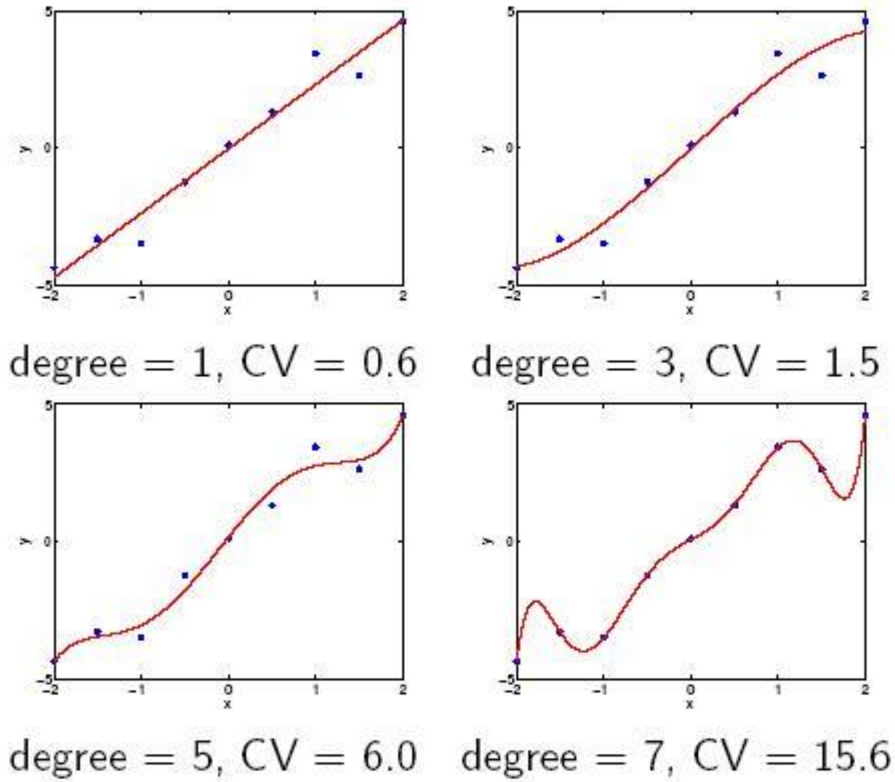
k+1 entries vector

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

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

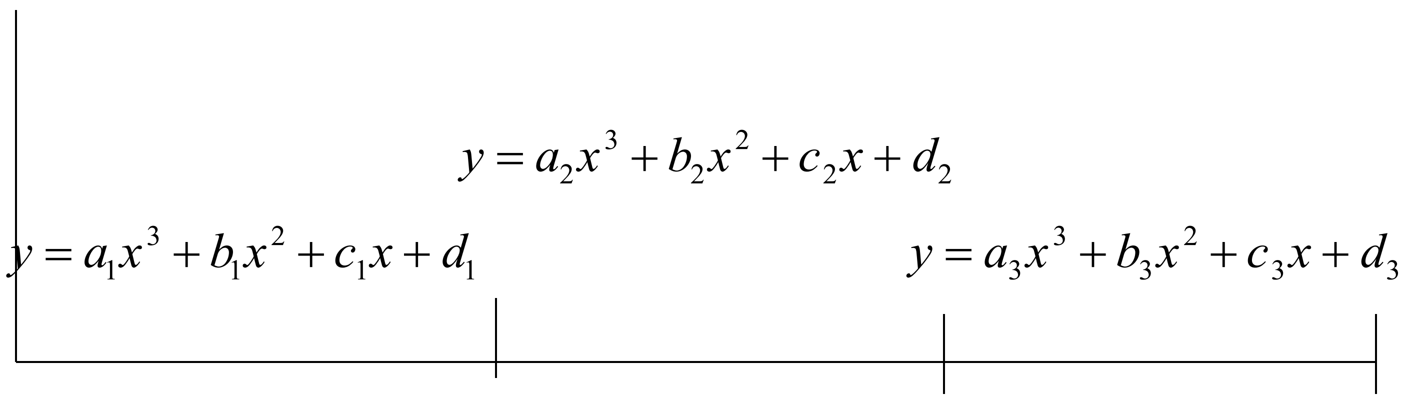
$$\mathrm{w} = (\Phi^T \Phi)^{-1} \Phi^T \mathrm{y}$$

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)

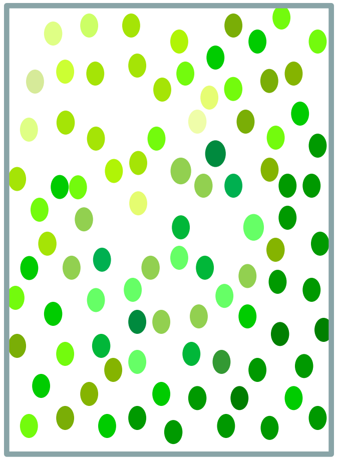


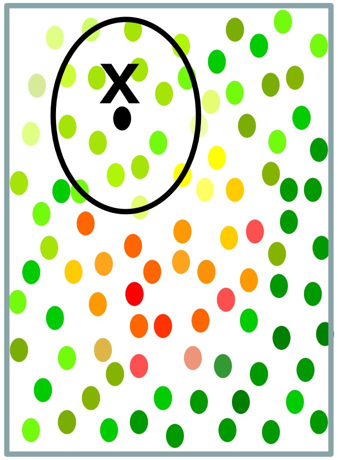
LOCAL, KERNEL REGRESSION

Local Kernel Regression

- What is the temperature in the room?

at location x?





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

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

Average

"Local" Average

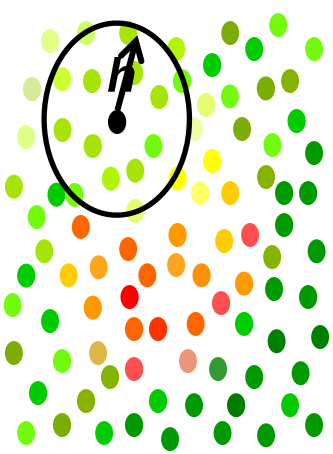
Local Average Regression

#pts in h ball around X

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

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

Sum of Ys in h ball around X



Recall: NN classifier with majority vote

Here we use Average instead

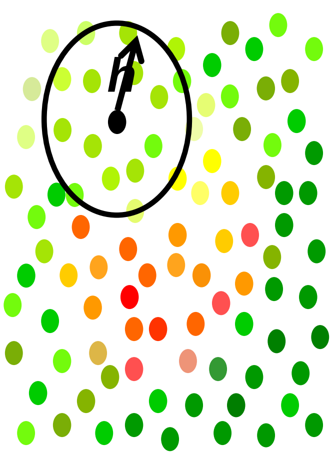
Nadaraya-Watson Kernel Regression

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

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

boxcar kernel :

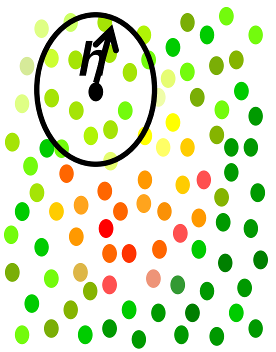
$$K\left(\frac{X-X_i}{h}\right) = 1_{|X-X_i|\leq h}$$



Local Kernel Regression

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

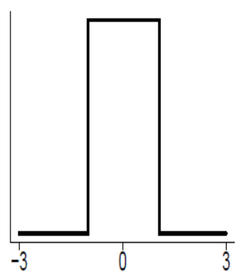
$$\widehat{f}_n(X) = \sum_{i=1}^{n} w_i Y_i \qquad\qquad w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$



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

boxcar kernel :

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



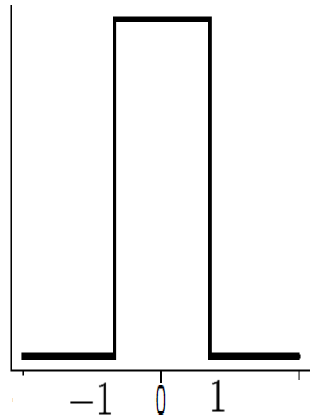
Kernels

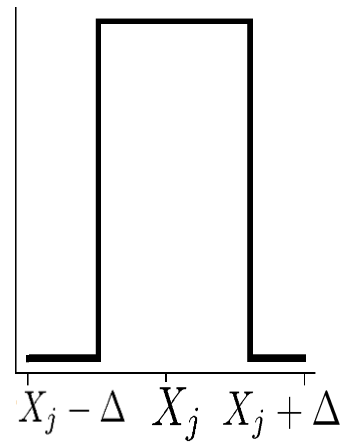
$$\begin{aligned} K(x) &\geq 0, \\ \int K(x)dx &= 1 \end{aligned}$$

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

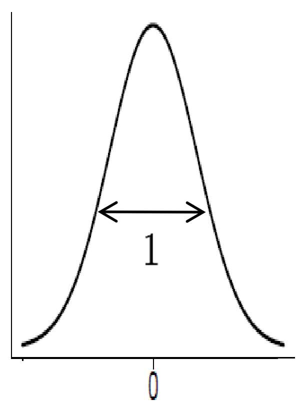
$$K(x) = \frac{1}{2}I(x),$$

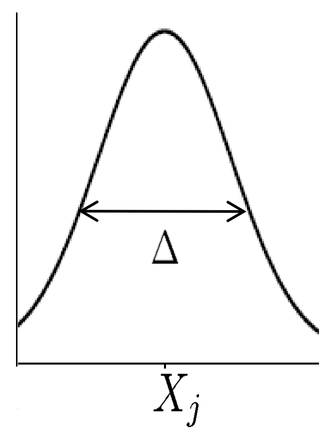




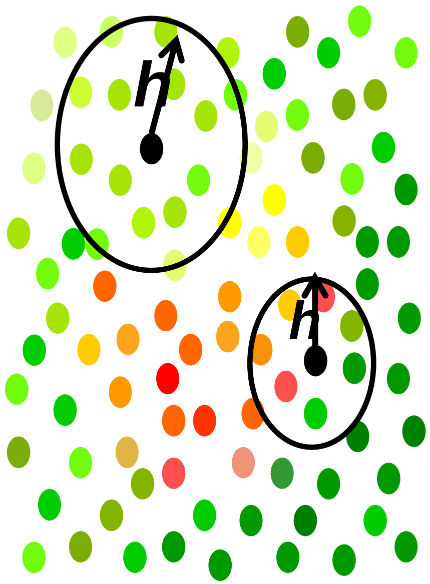
Gaussian kernel :

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





Spatially adaptive regression



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

Local polynomials, splines, wavelets, regression trees …

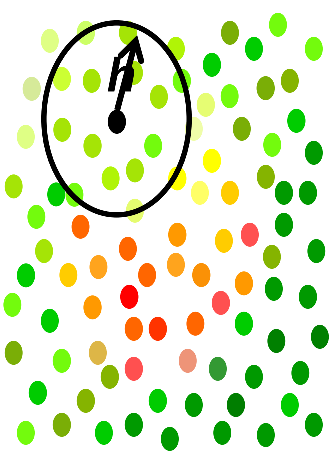
Local Average Regression

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

#pts in h ball around X

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

Sum of Ys in h ball around X



Recall: NN classifier with majority vote

Here we use Average instead

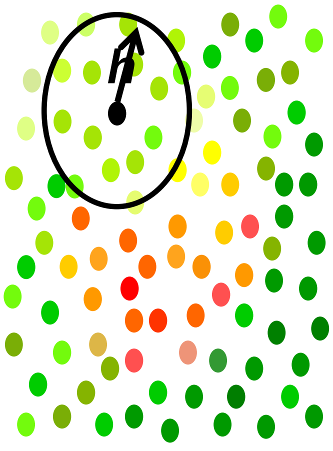
Nadaraya-Watson Kernel Regression

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

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

boxcar kernel :

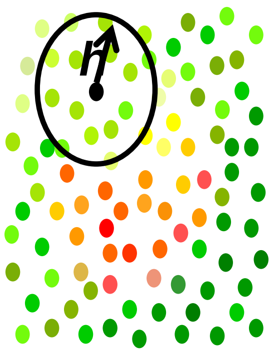
$$K\left(\frac{X-X_i}{h}\right) = 1_{|X-X_i|\leq h}$$



Local Kernel Regression

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

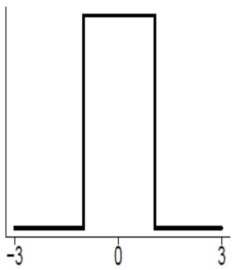
$$\widehat{f}_n(X) = \sum_{i=1}^{n} w_i Y_i \qquad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$



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

boxcar kernel :

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



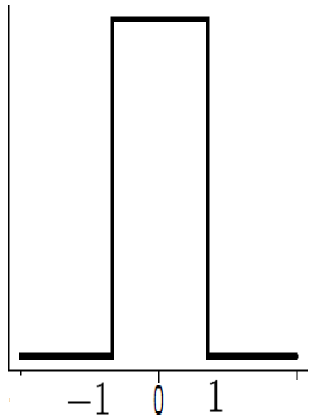
Kernels

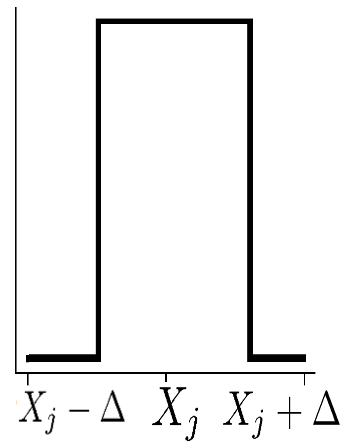
$$
\begin{aligned}
K(x) &\geq 0, \\
\int K(x)dx &= 1
\end{aligned}
$$

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

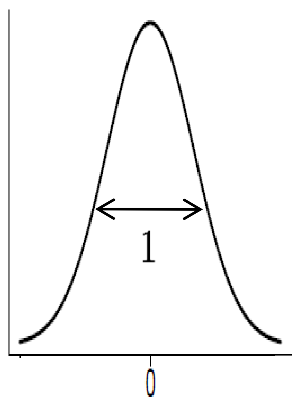
$$K(x) = \frac{1}{2}I(x),$$

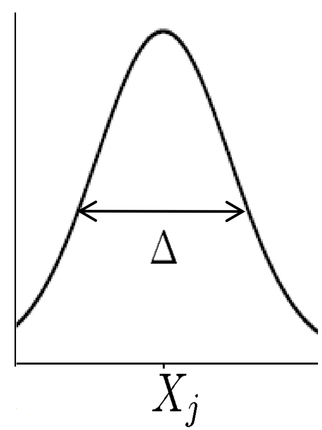




Gaussian kernel :

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





Choice of kernel bandwidth h

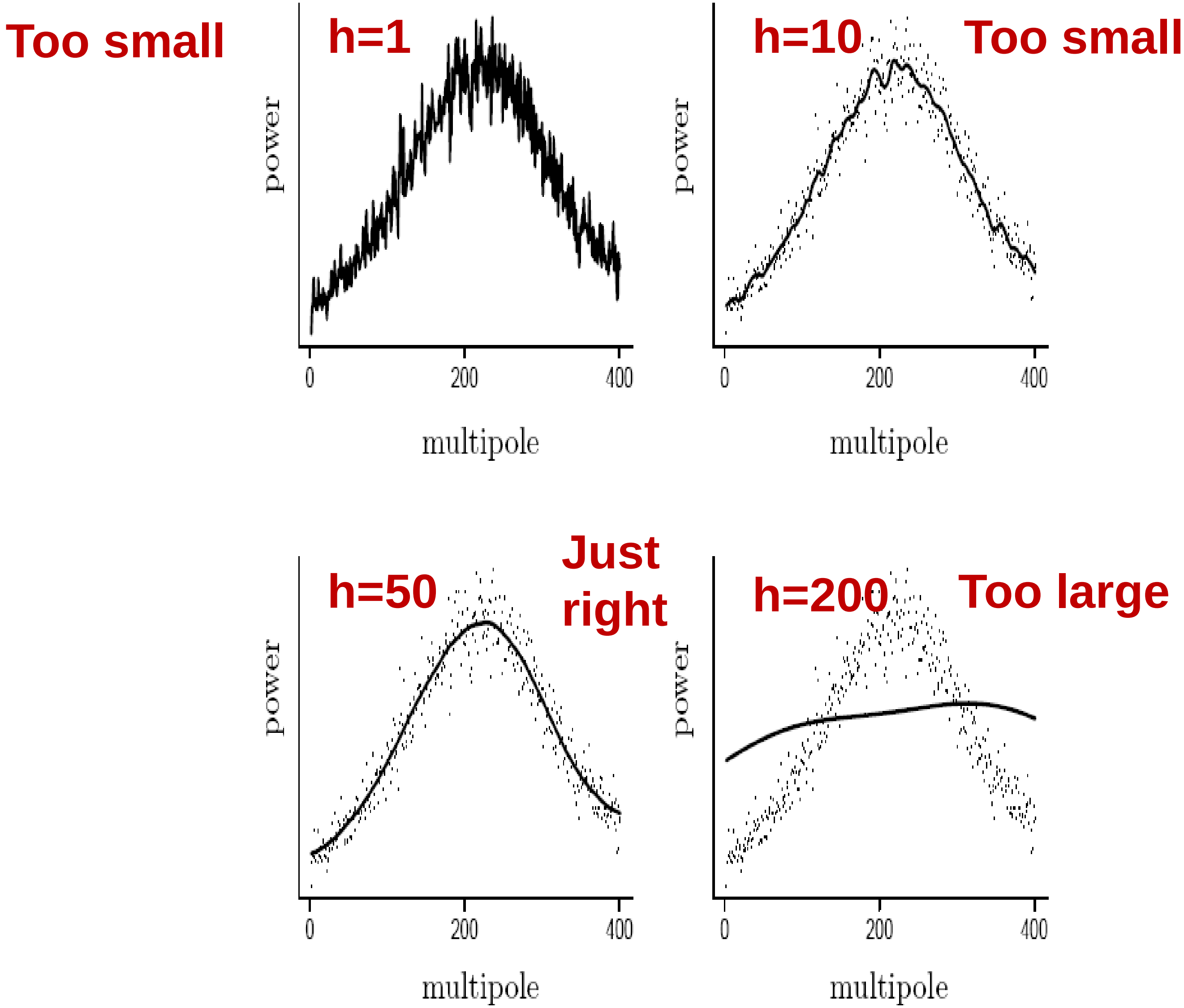
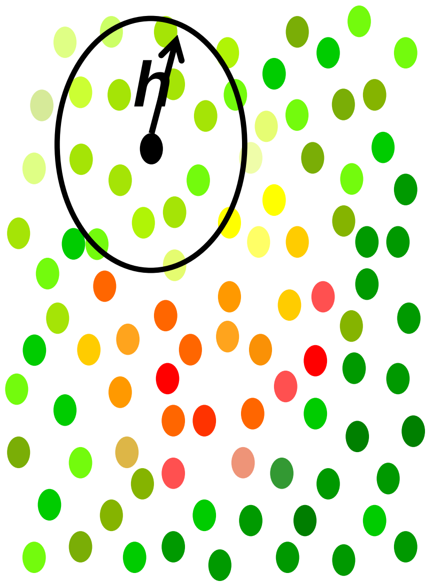


Image Source: Larry's book – All of Nonparametric Statistics

Choice of Bandwidth



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

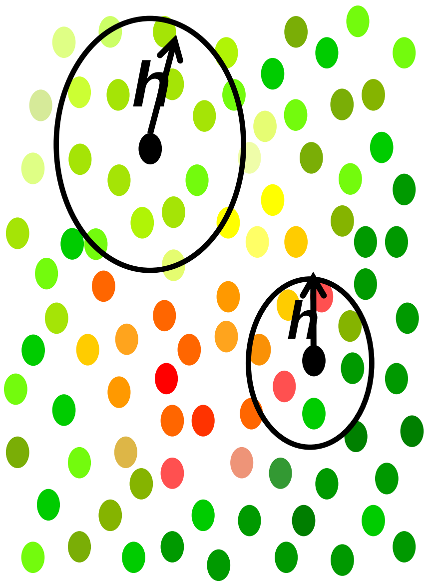
Should depend on smoothness of function
(determines bias)

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 …

Important points

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