2019
怪兽
学堂

Regression

虾米

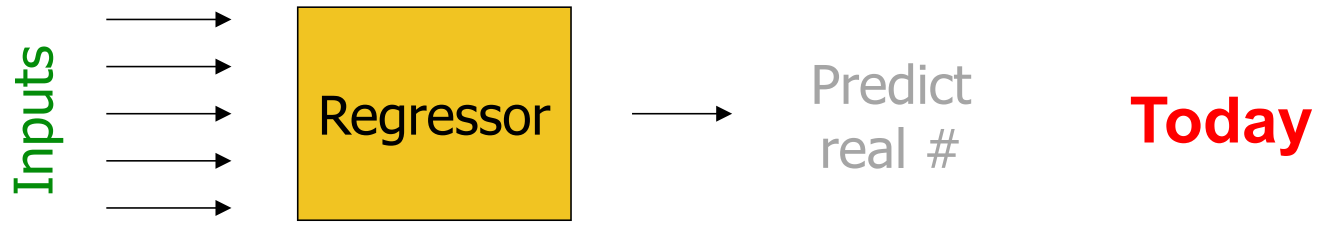
时间：2019年3月

Outline

- Regression
- Linear regression
  - As optimization ➔ Gradient descent
- Overfitting and bias-variance

What is regression?

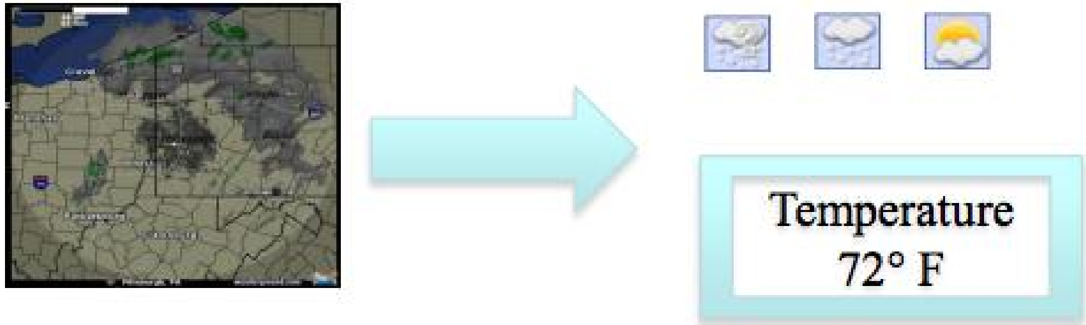
Where we are



Regression examples

Stock market

Weather prediction



Predict the temperature at any given location

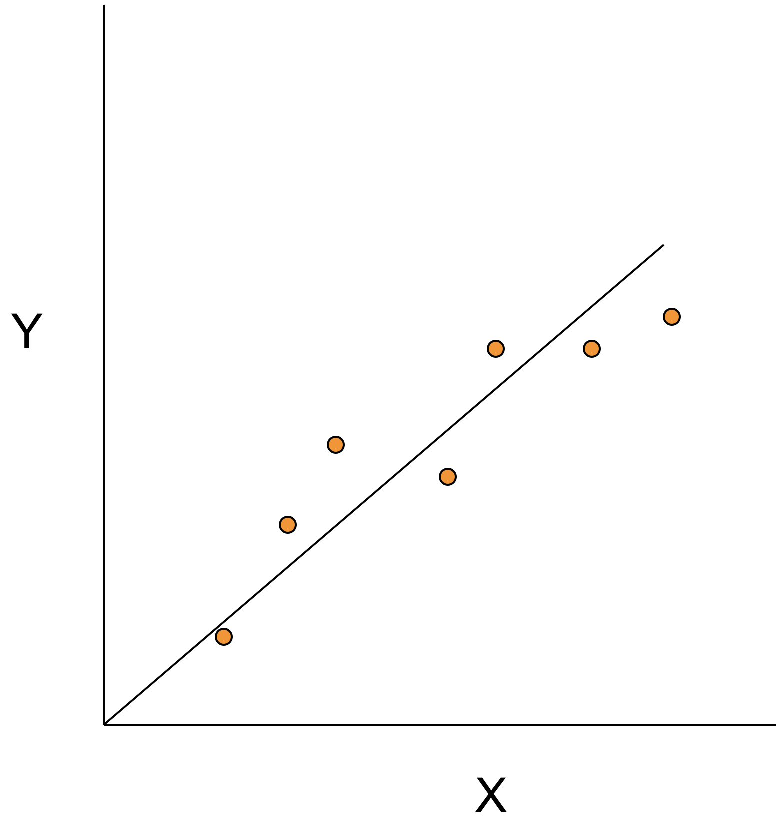
Linear regression

- 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

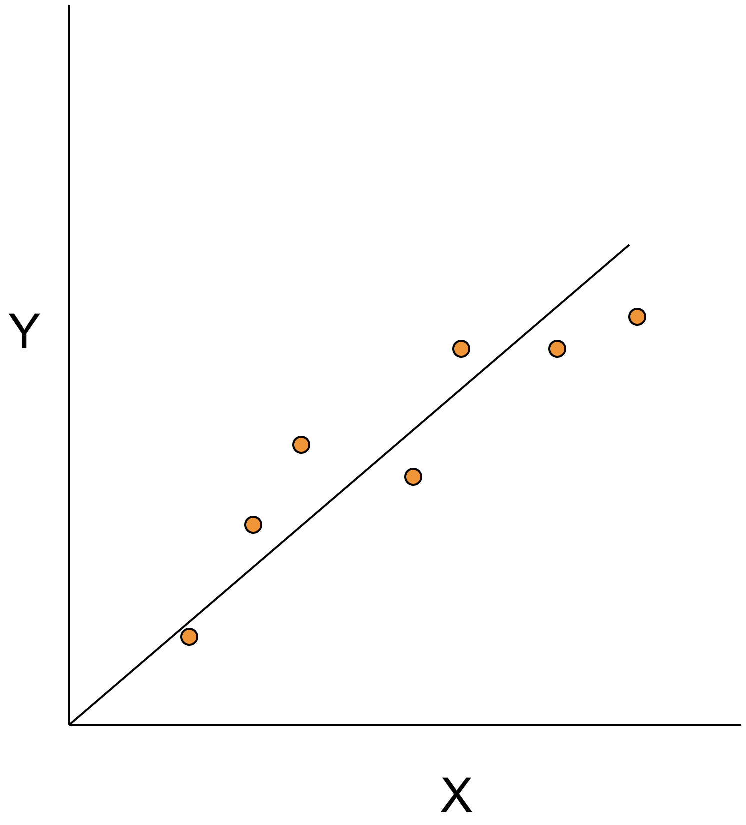
- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

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

- Our goal is to estimate *w* from a training data of $<x_i,y_i>$ pairs
- Optimization goal: minimize squared error (least squares):

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

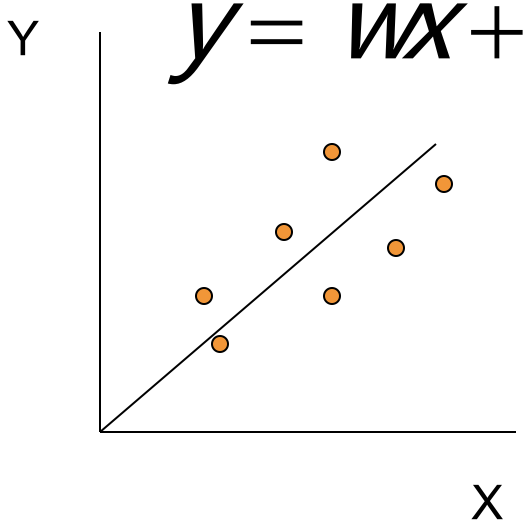
- Why least squares?

  - minimizes squared distance between measurements and predicted line

  - has a nice probabilistic interpretation

  - the math is pretty

$$y = wx + \varepsilon$$



Solving linear regression

- To optimize:
- We just take the derivative w.r.t. to w ….

prediction

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

Solving linear regression

- To optimize – closed form:
- We just take the derivative w.r.t. to w and set to 0:

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

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

$$\sum_i x_i y_i = \sum_i w x_i^2 \quad \Rightarrow$$

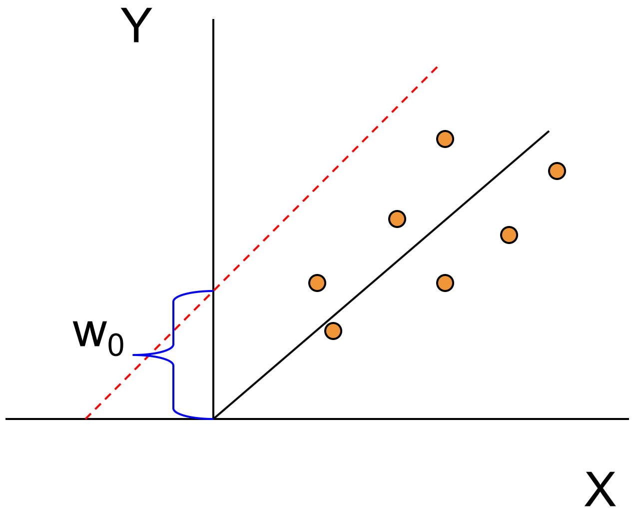
$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

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

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

- Can use least squares to determine w_0 , w_1



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

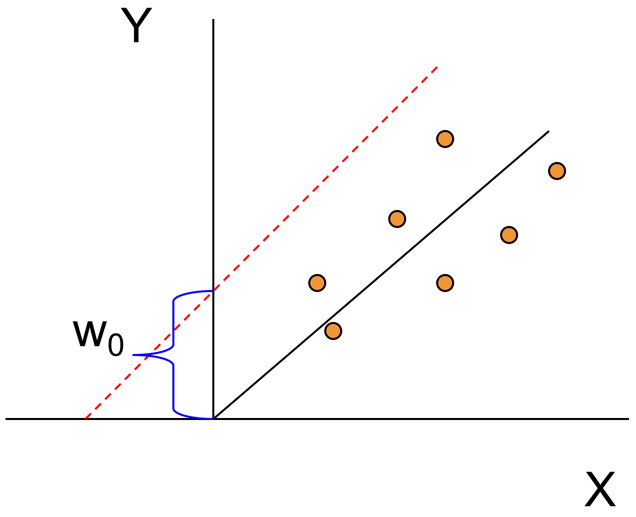
Simpler solution is coming soon…

Bias term

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

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

- Can use least squares to determine w_0 , w_1



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

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:

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

Microsoft’s stock price

Google’s stock price

Yahoo’s stock price

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:

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

Not all functions can be approximated by a line/hyperplane…

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

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

Non-Linear basis function

- So far we only used the observed values $x_1, x_2, \cdots$
- However, linear regression can be applied in the same way to **functions** of these values
  - Eg: to add a term $w\, x_1 x_2$ add a new variable $z = x_1 x_2$ so each example becomes: $x_1, x_2, \cdots. z$
- 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 multi-variate linear regression problem

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

Non-Linear basis function

- How can we use this to add an intercept term?

Add a new "variable" $z=1$ and weight w_0

Non-linear basis functions

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

  - Polynomial: $\phi_j(x) = x^j$ for $j=0 \cdots n$
  - Gaussian: $$\phi_j(x) = \frac{(x-\mu_j)}{2\sigma_j^2}$$
  - Sigmoid: $$\phi_j(x) = \frac{1}{1+\exp(-s_j x)}$$
  - Logs: $$\phi_j(x) = \log(x+1)$$

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

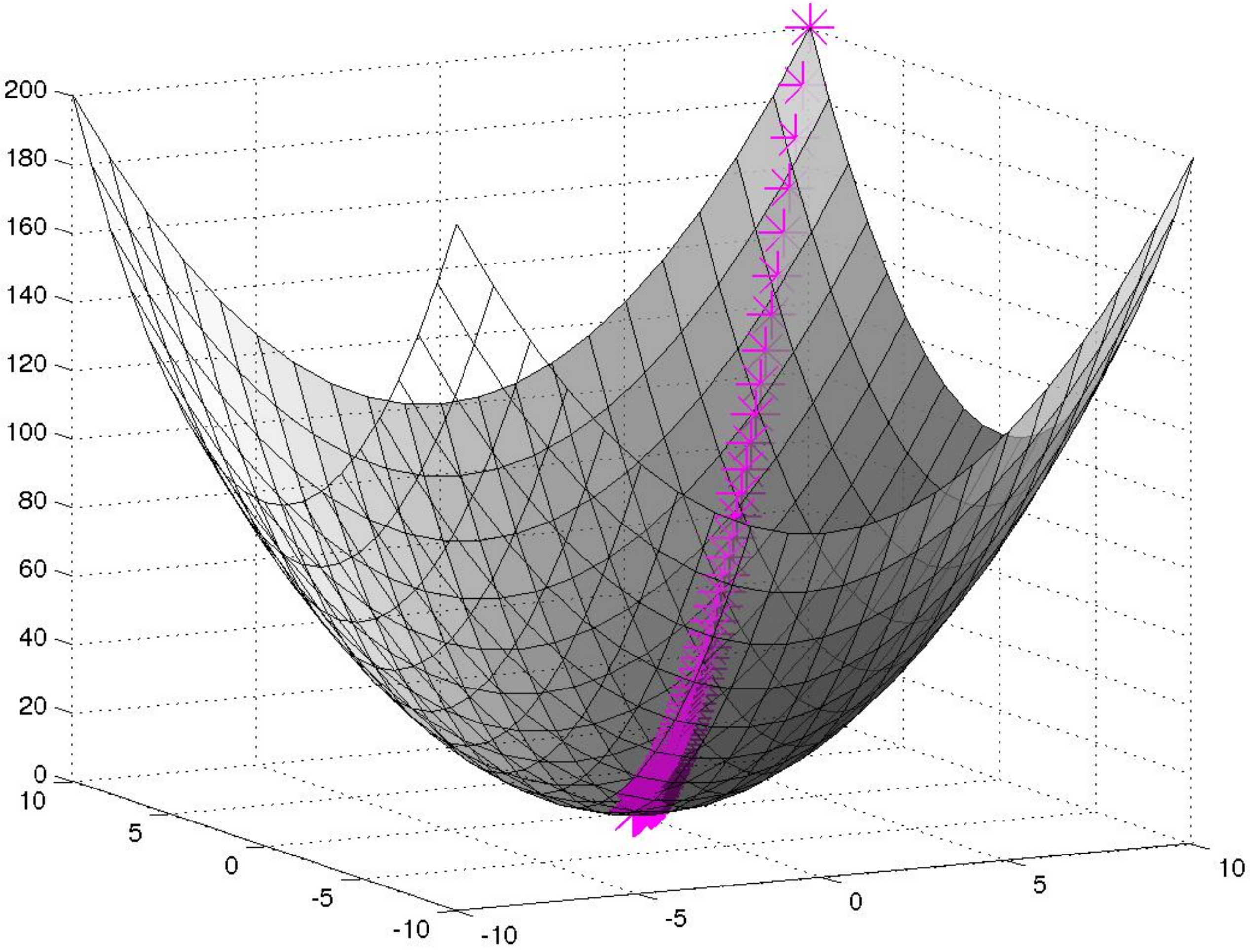
$$y = \sum_{j=0}^{n} w_j \phi_j(x)$$

- Where $\phi_j(\mathbf{x})$ can be either x_j for multivariate regression or one of the non-linear basis functions we defined
- … and $\phi_0(\mathbf{x})=1$ for the intercept term

Learning/Optimizing Multivariate Least Squares

Approach 1: Gradient Descent

Gradient descent



Gradient Descent for Linear Regression

Goal: minimize the following loss function:

predict with: $\hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$

$$J_{\mathbf{X},\mathbf{y}}(\mathbf{w}) = \underbrace{\sum_i}_{\text{sum over } n \text{ examples}} \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \underbrace{\sum_j}_{\text{sum over } k+1 \text{ basis vectors}} w_j \phi_j(\mathbf{x}^i)\right)^2$$

Gradient Descent for Linear Regression

Goal: minimize the following loss function:

$$\text{predict with: } \hat{y}^i = \sum_j^n w_j \phi_j(\mathbf{x}^i)$$

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \sum_i \left(y^i - \hat{y}^i\right)^2 = \sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i)\right)^2$$

$$\frac{\partial}{\partial w_j} J(\mathbf{w}) = \frac{\partial}{\partial w_j} \sum_i \left(y^i - \hat{y}^i\right)^2$$

$$= 2\sum_i \left(y^i - \hat{y}^i\right) \frac{\partial}{\partial w_j} \hat{y}^i$$

$$= 2\sum_i \left(y^i - \hat{y}^i\right) \frac{\partial}{\partial w_j} \sum_j w_j \phi_j(\mathbf{x}^i)$$

$$= 2\sum_i \left(y^i - \hat{y}^i\right) \phi_j(\mathbf{x}^i)$$

Gradient Descent for Linear Regression

Learning algorithm:

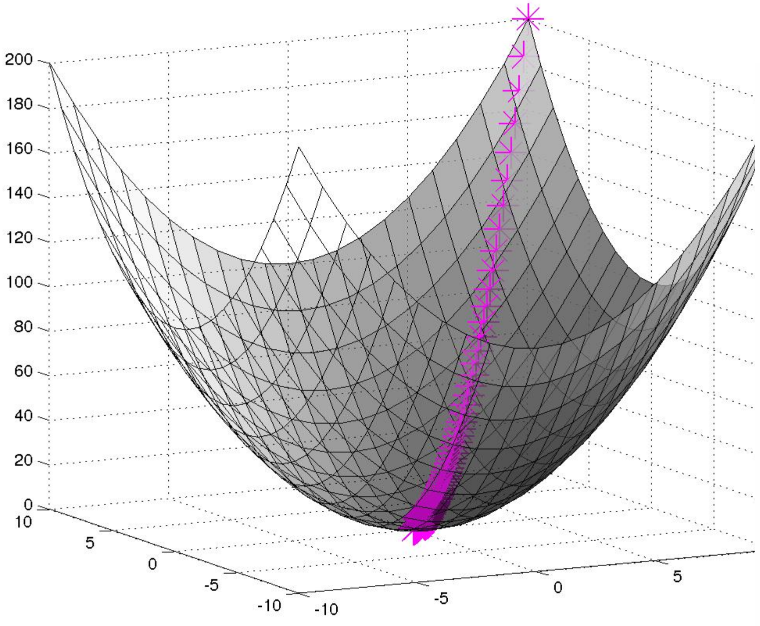
- Initialize weights **w=0**
- For t=1,… until convergence:
  - Predict for each example $\boldsymbol{x}^i$ using **w:** $$\hat{y}^i = \sum_{j=0}^{k} w_j \phi_j(\mathbf{x}^i)$$
  - Compute gradient of loss: $$\frac{\partial}{\partial w_j} J(\mathbf{w}) = 2\sum_i \left(y^i - \hat{y}^i\right) \phi_j(\mathbf{x}^i)$$
  - This is a vector **g**
  - Update: **w = w** – λ**g**
  - λ is the learning rate.

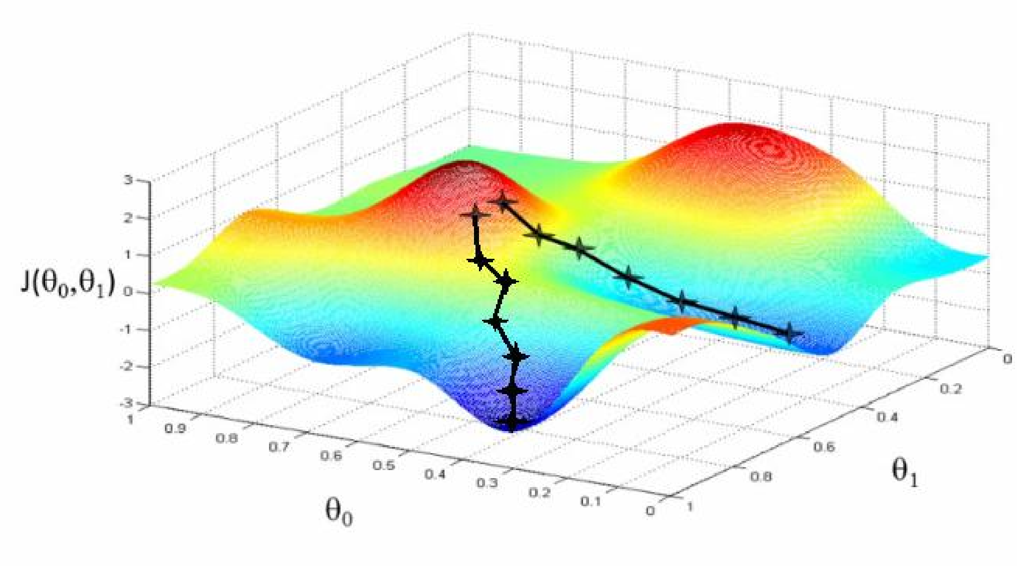
Gradient Descent for Linear Regression

- We can use any of the tricks we used for logistic regression:
  - stochastic gradient descent (if the data is too big to put in memory)
  - regularization
  - ...

Linear regression is a *convex* optimization problem

so again gradient descent will reach a *global* optimum





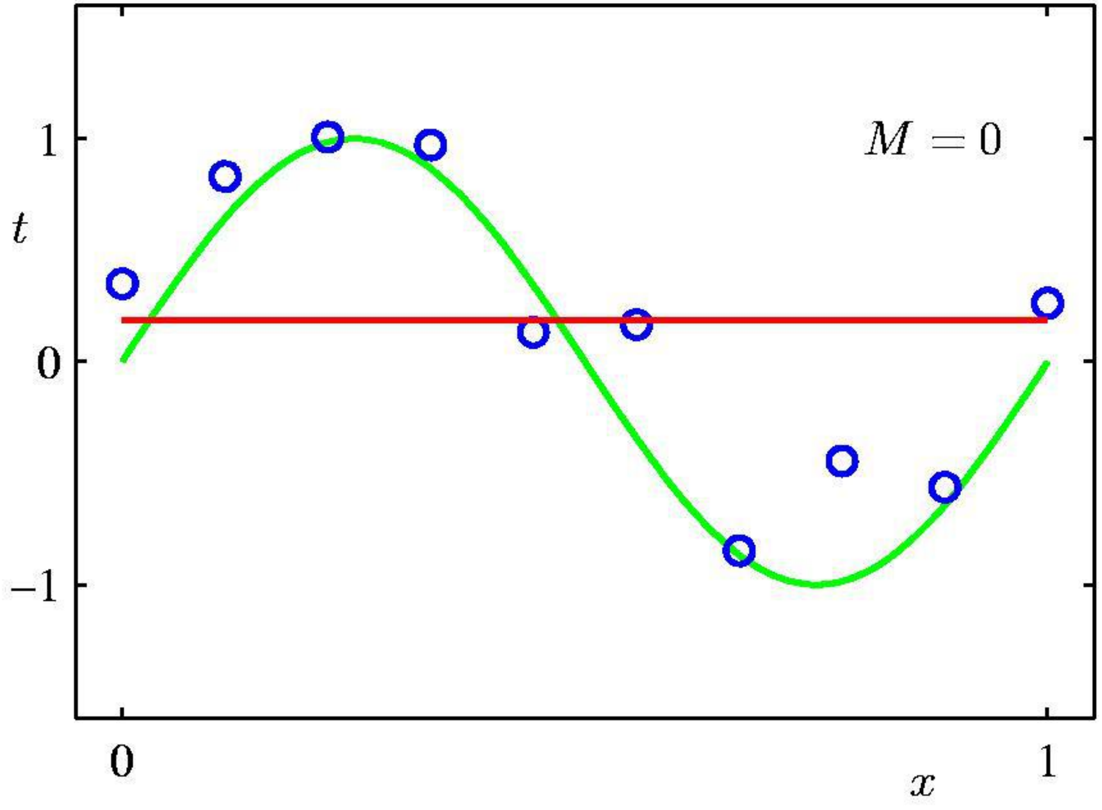
proof: differentiate again to get the second derivative

Regression and Overfitting

An example: polynomial basis vectors on a small dataset

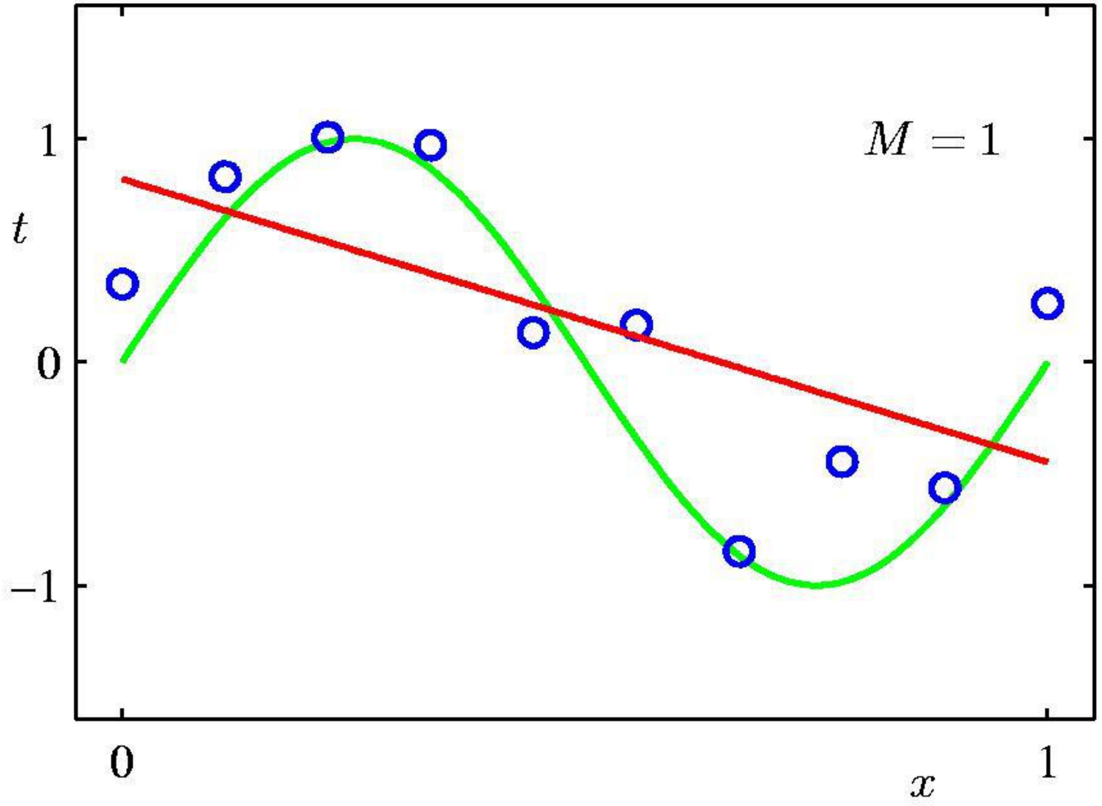
- From Bishop Ch 1

0th Order Polynomial

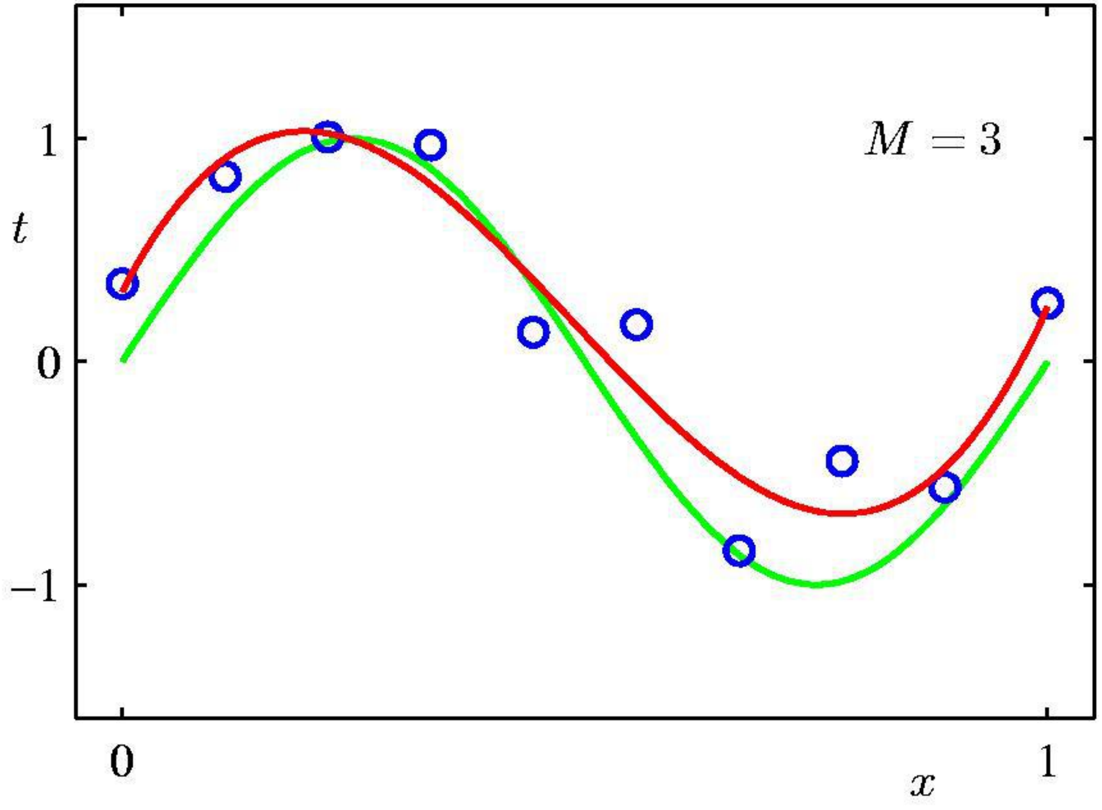


n=10

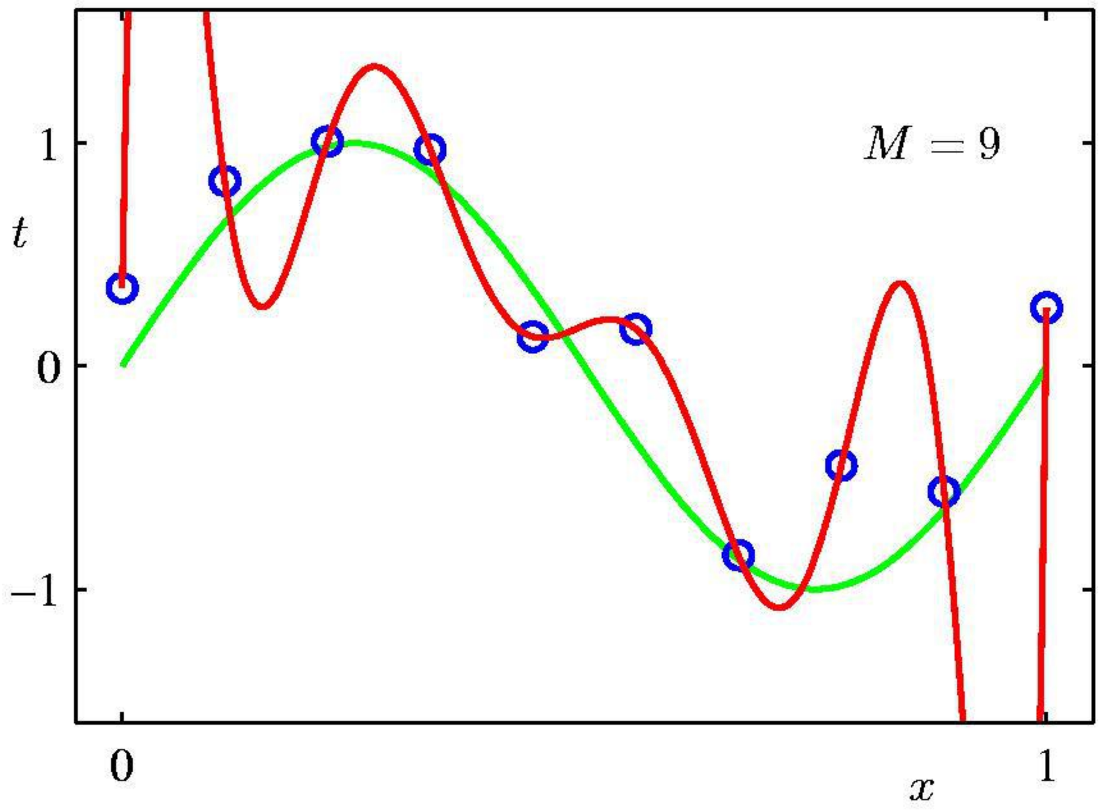
1^{st} Order Polynomial



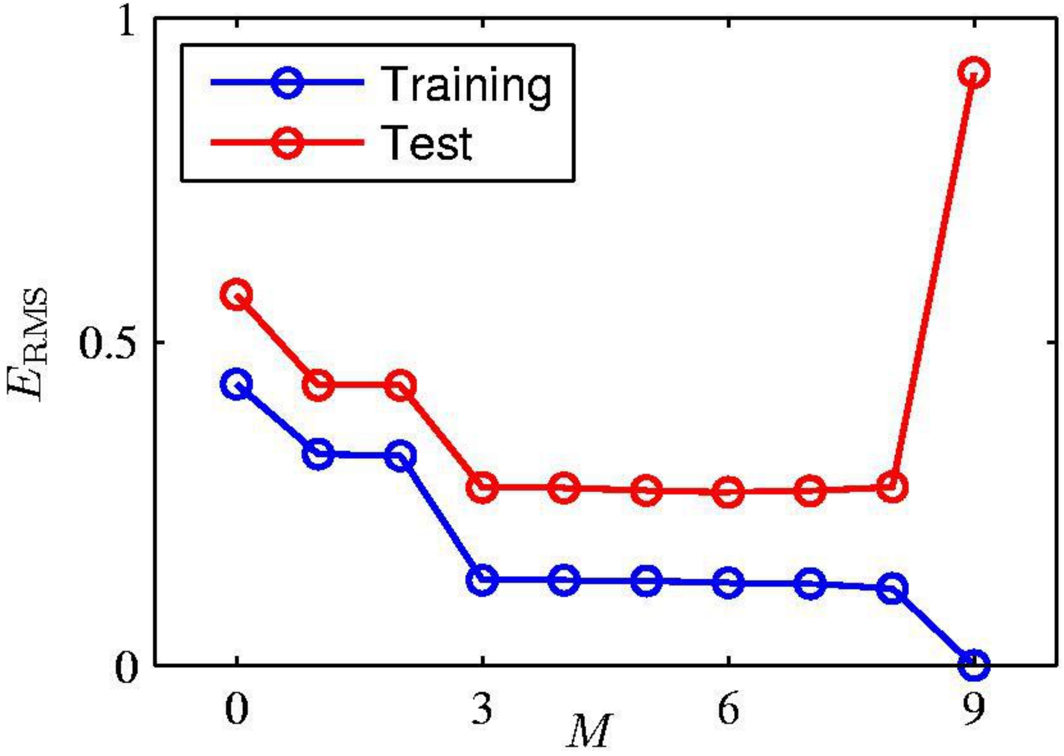
3rd Order Polynomial



9th Order Polynomial



Over-fitting



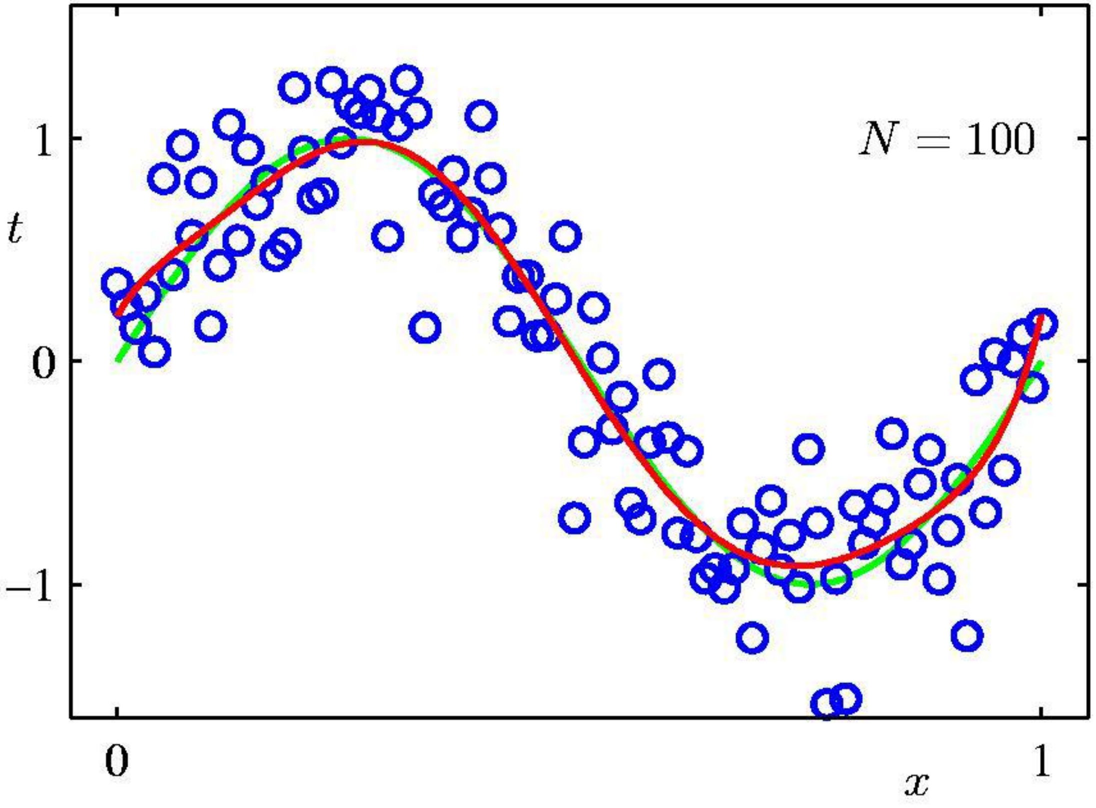
Root-Mean-Square (RMS) Error: $E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$

Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Data Set Size:

9th Order Polynomial $N = 100$



Regularization

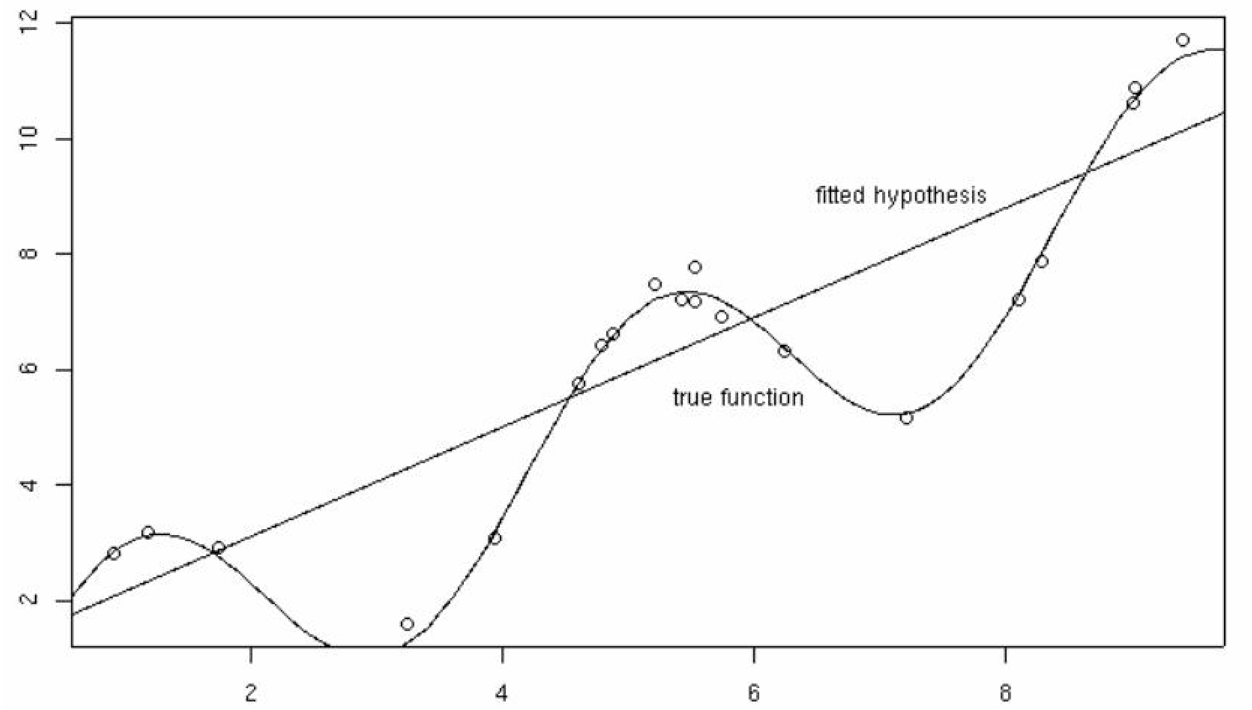
Penalize large coefficient values

$$J_{\mathbf{x},\mathbf{y}}(\mathbf{w}) = \frac{1}{2}\sum_i \left(y^i - \sum_j w_j \phi_j(\mathbf{x}^i) \right)^2 - \underline{\frac{\lambda}{2}\|\mathbf{w}\|^2}$$

Understanding Overfitting: Bias-Variance

Example

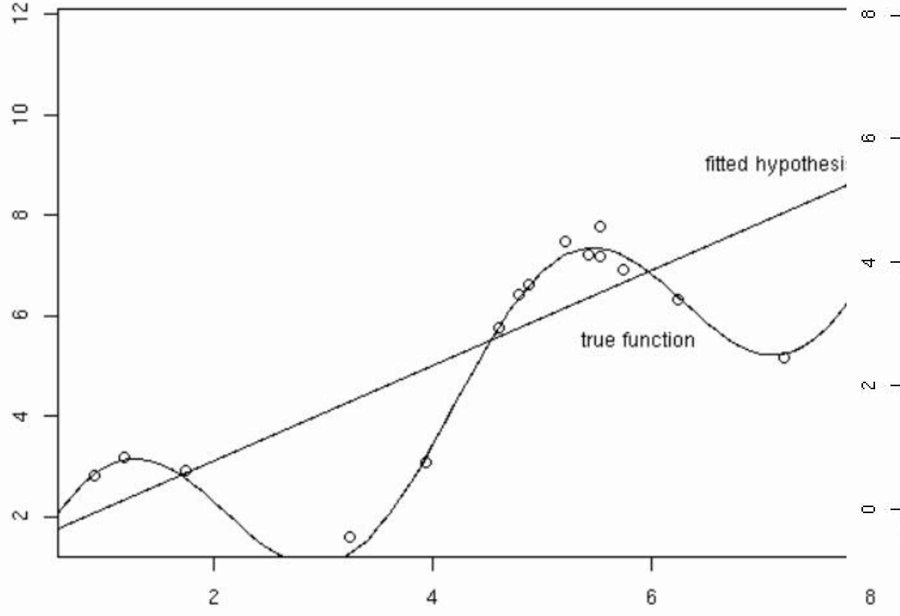
Tom Dietterich, Oregon St



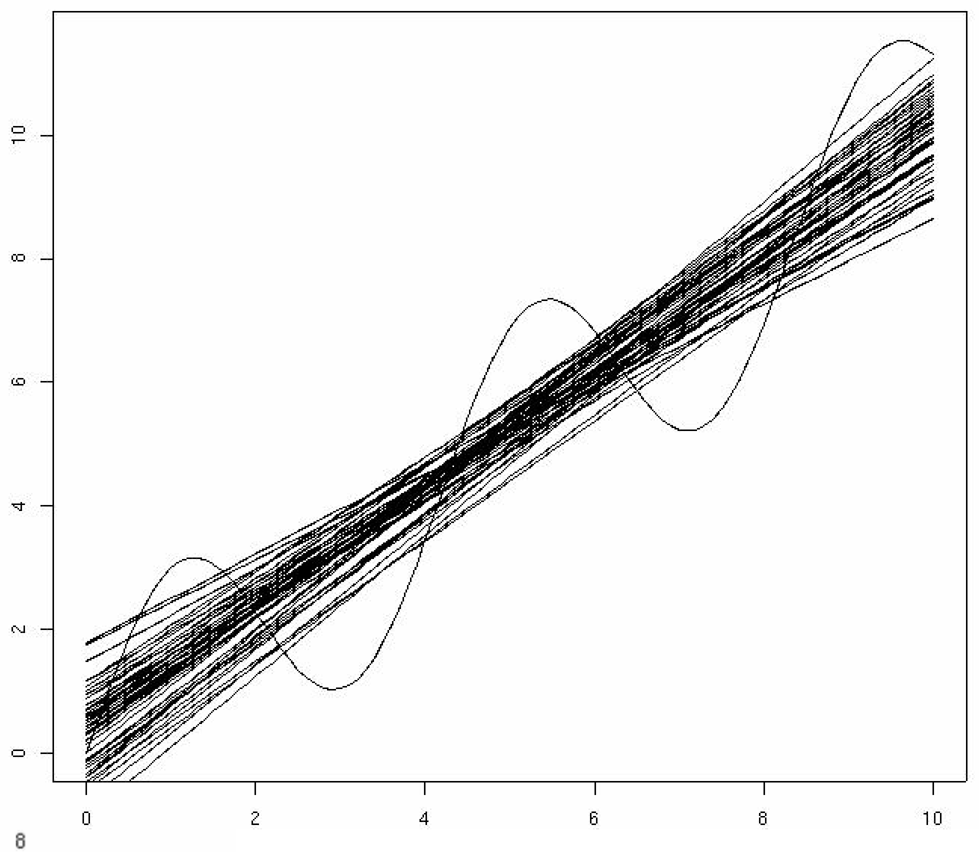
$$y = x + 2\sin(1.5x) + N(0,0.2)$$

Example

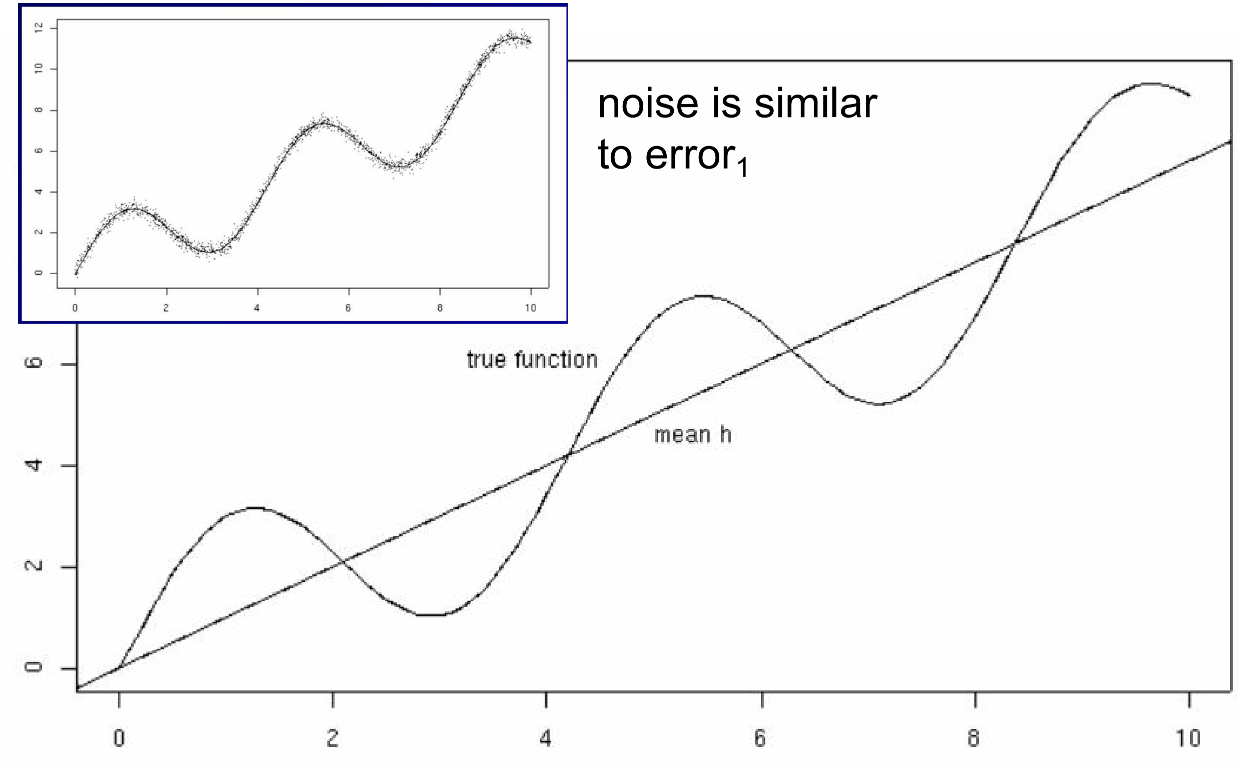
Tom Dietterich, Oregon St



y = x + 2 sin(1.5x) + N(0,0.2)



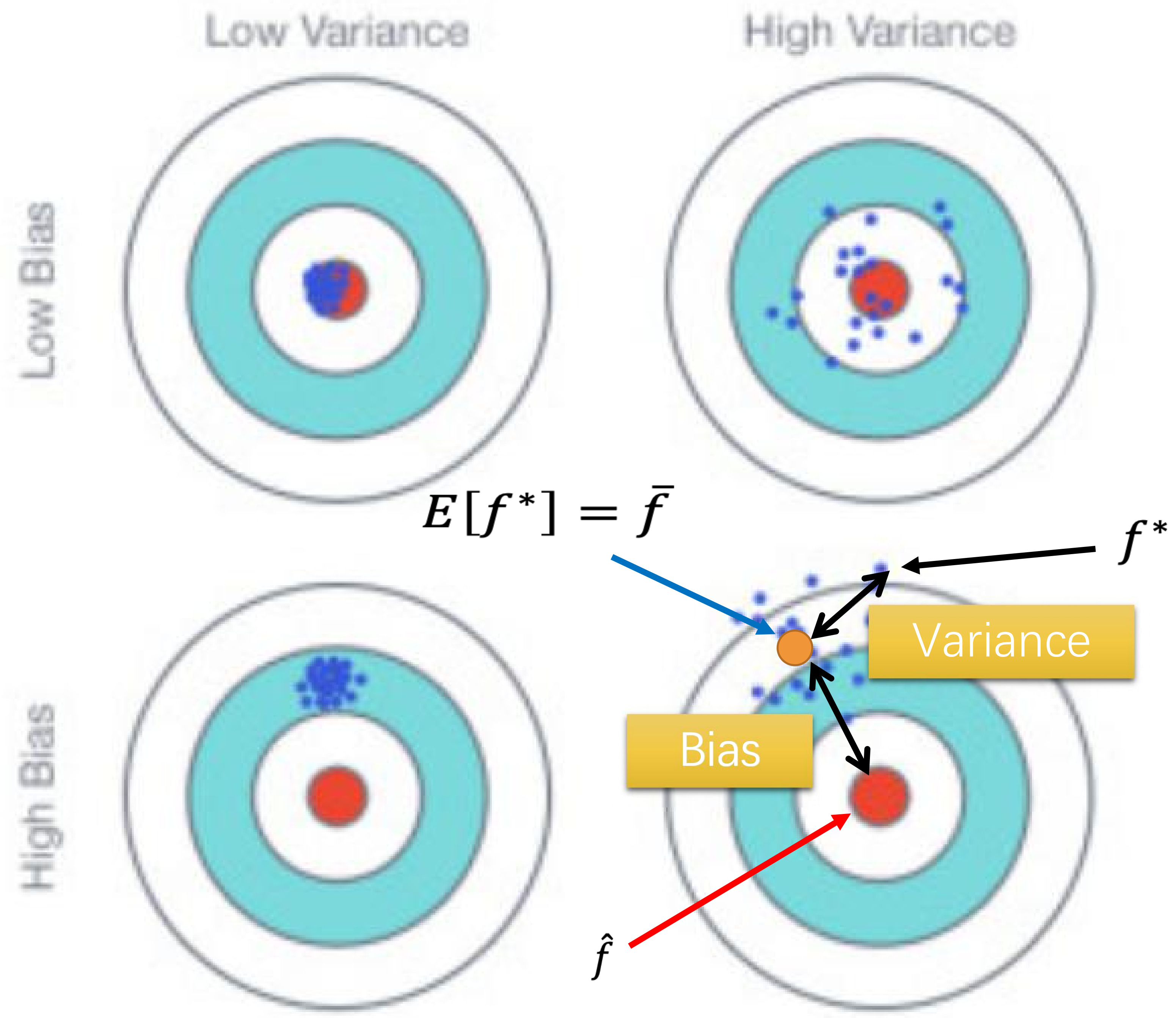
Same experiment, repeated:
with 50 samples of 20 points each



noise is similar to error_1

The true function *f* can't be fit perfectly with hypotheses from our class *H* (lines) ➔ Error_1

Fix: *more* expressive set of hypotheses *H*


Low Variance
High Variance
Low Bias
High Bias
$E[f^*] = \bar{f}$
f^*
Variance
Bias
$\hat{f}$

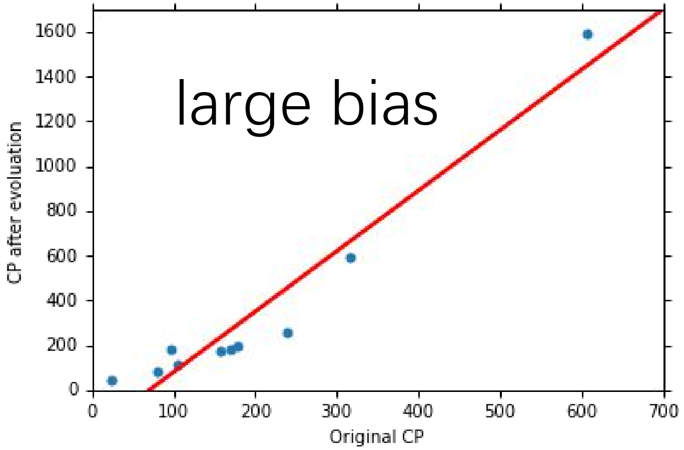
What to do with large bias?

- Diagnosis:
  - If your model cannot even fit the training examples, then you have large bias

Underfitting

- For bias, redesign your model:
  - Add more features as input
  - A more complex model

Overfitting



What to do with large variance?

- Diagnosis:
  - If you can fit the training data, but large error on testing data, then you probably have large variance

Overfitting

- For variance
  - More data: Very effective, but not always practical
  - Regularization

2019
怪兽
学堂

THANKS