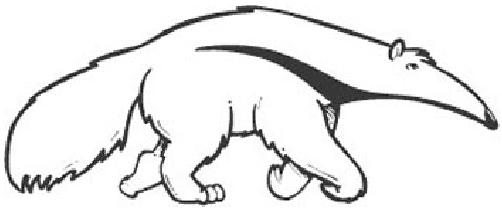
Machine Learning and Data Mining

Linear regression

(adapted from) Prof. Alexander Ihler

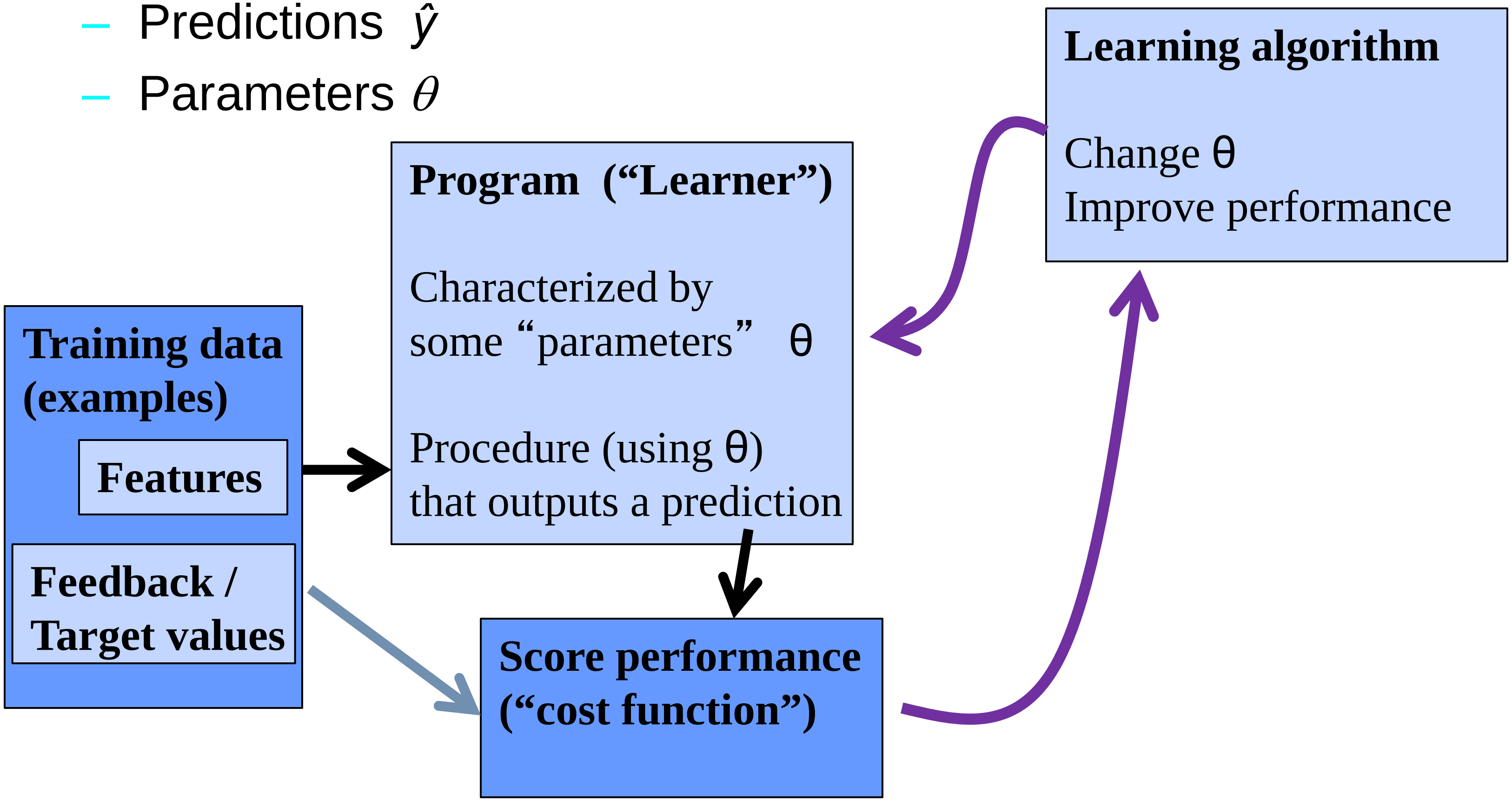


BREN:ICS INFORMATION AND COMPUTER SCIENCES

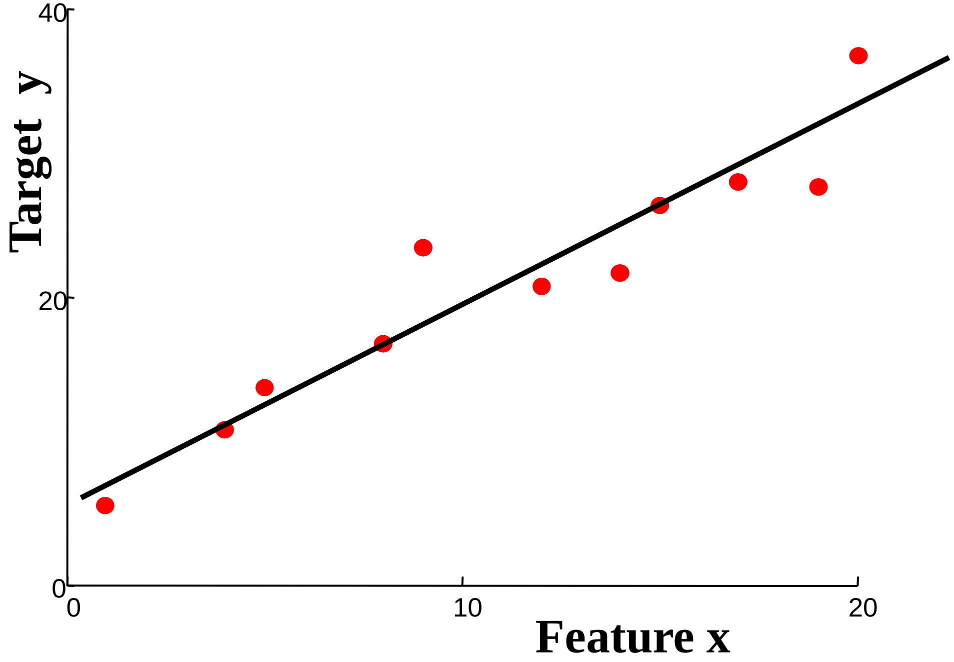
UNIVERSITY of CALIFORNIA IRVINE

Supervised learning

- Notation
  - Features x
  - Targets y
  - Predictions $\hat{y}$
  - Parameters θ



Linear regression



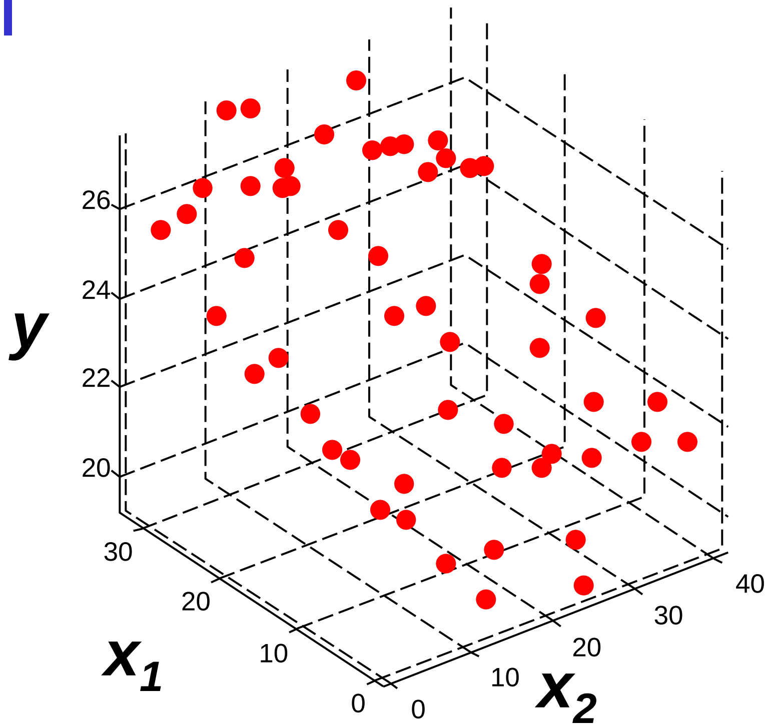
"Predictor":
Evaluate line:

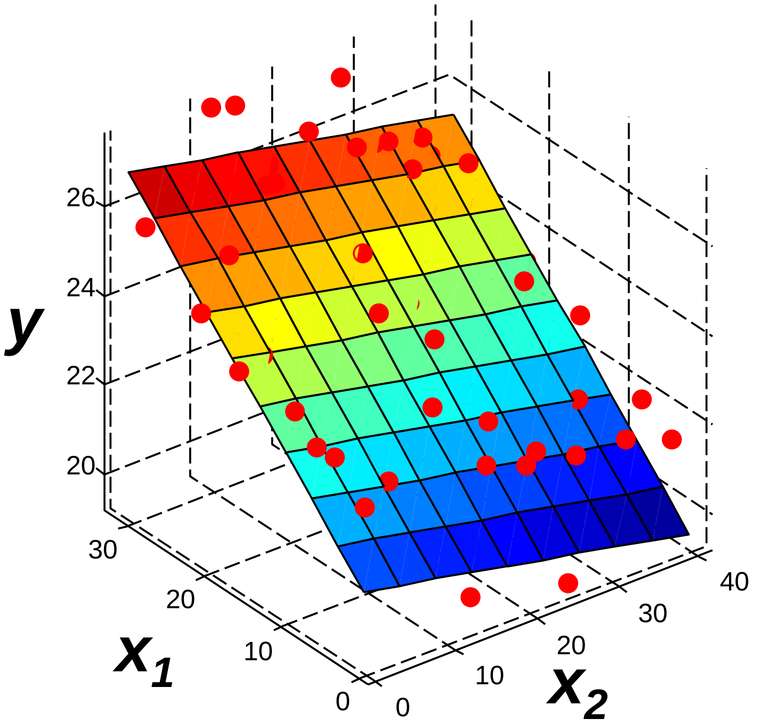
$$r = \theta_0 + \theta_1 x_1$$

return r

- Define form of function f(x) explicitly
- Find a good f(x) within that family

More dimensions?





$$\hat{y}(x) = \underline{\theta} \cdot \underline{x}^T$$

$$\underline{\theta} = [\theta_0 \ \ \theta_1 \ \ \theta_2]$$
$$\underline{x} = [1 \ x_1 \ \ x_2]$$

Notation

$$\hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$$

Define "feature" $x_0 = 1$ (constant)

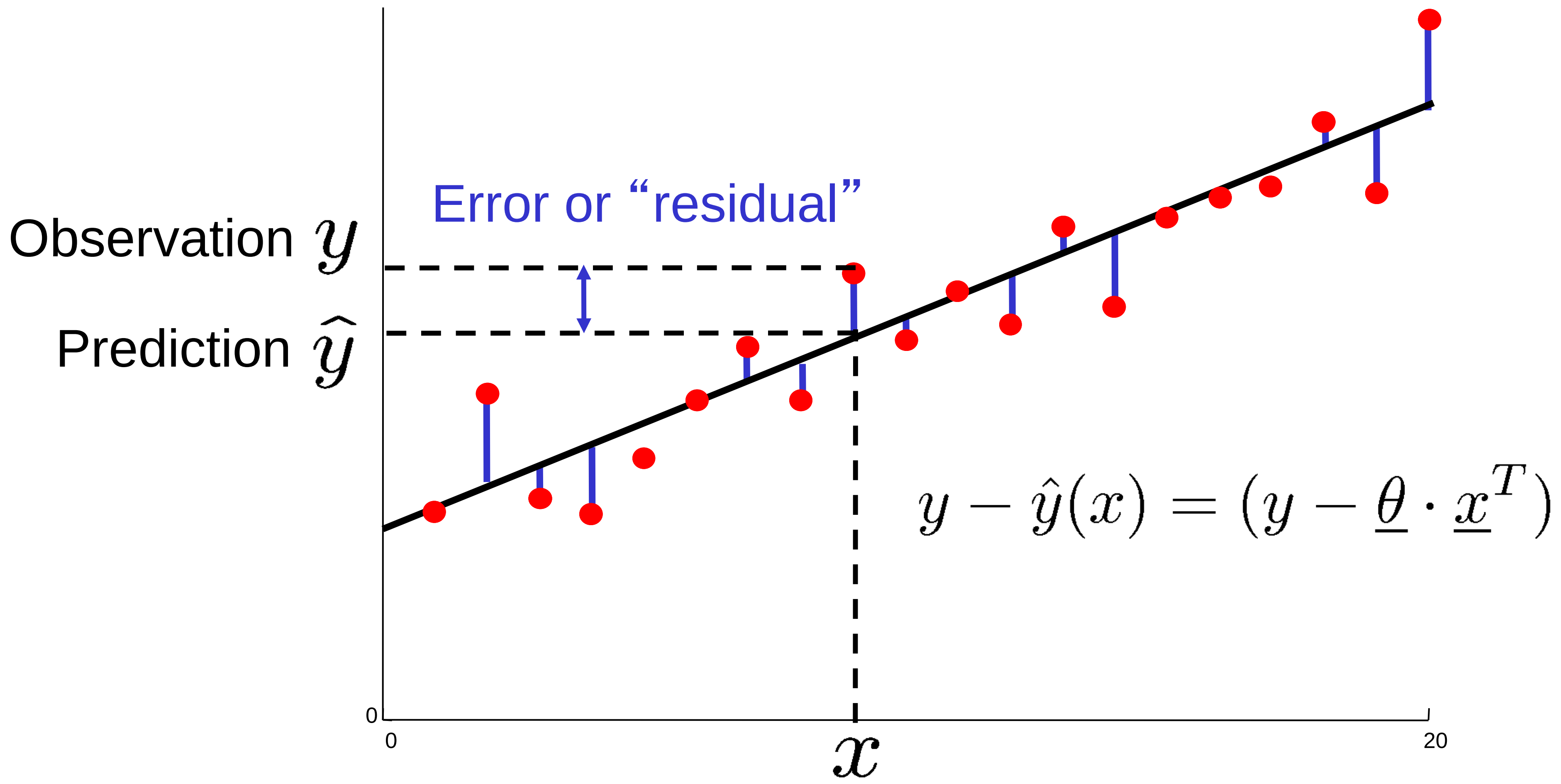
Then

$$\hat{y}(x) = \theta\, x^T$$

$$\underline{\theta} = [\theta_0, \ldots, \theta_n]$$

$$\underline{x} = [1, x_1, \ldots, x_n]$$

Measuring error



Mean squared error

- How can we quantify the error?

$$\text{MSE},\ J(\underline{\theta}) = \frac{1}{m}\sum_j (y^{(j)} - \hat{y}(x^{(j)}))^2$$

$$= \frac{1}{m}\sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})^2$$

- Could choose something else, of course…
  - Computationally convenient (more later)
  - Measures the variance of the residuals
  - Corresponds to likelihood under Gaussian model of "noise"

$$\mathcal{N}(y\ ;\ \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2}(y-\mu)^2 \right\}$$

MSE cost function

$$\text{MSE, } J(\underline{\theta}) = \frac{1}{m}\sum_j (y^{(j)} - \hat{y}(x^{(j)}))^2$$

$$= \frac{1}{m}\sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})^2$$

- Rewrite using matrix form

$$\underline{\theta} = [\theta_0, \ldots, \theta_n]$$

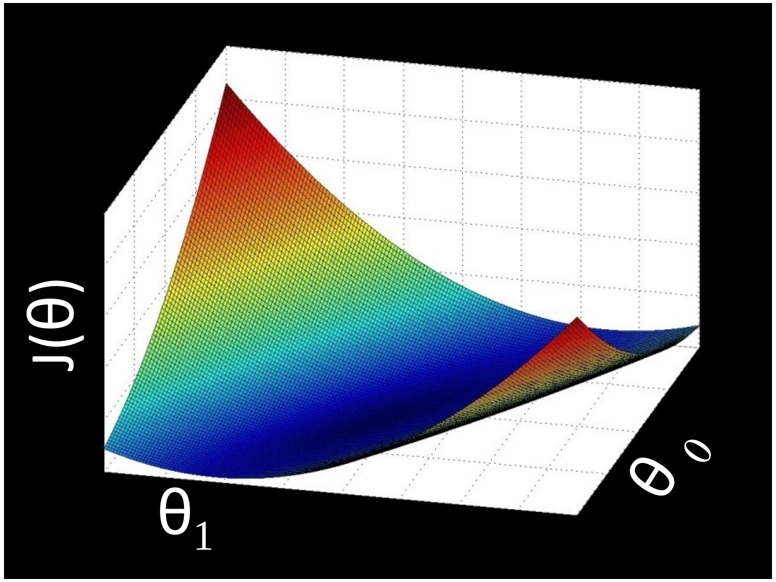
$$\underline{y} = \left[y^{(1)} \ldots, y^{(m)}\right]^T$$

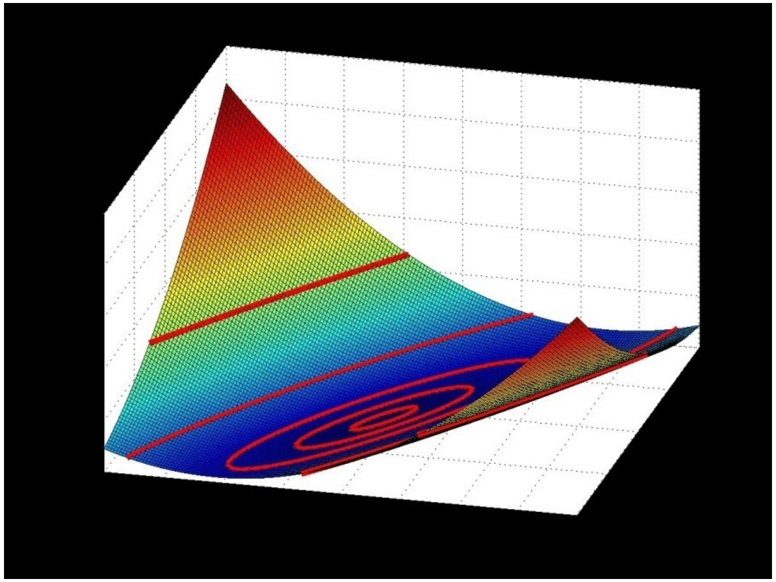
$$\underline{X} = \begin{bmatrix} x_0^{(1)} & \ldots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \ldots & x_n^{(m)} \end{bmatrix}$$

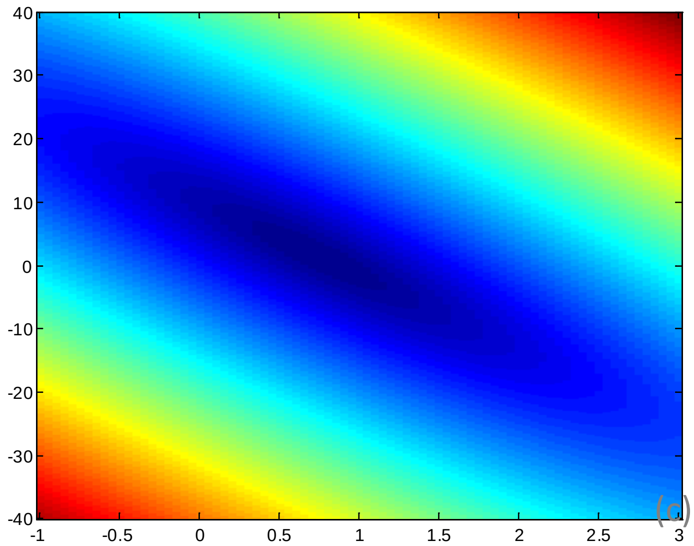
$$J(\underline{\theta}) = \frac{1}{m}(\underline{y}^T - \underline{\theta}\,\underline{X}^T) \cdot (\underline{y}^T - \underline{\theta}\,\underline{X}^T)^T$$

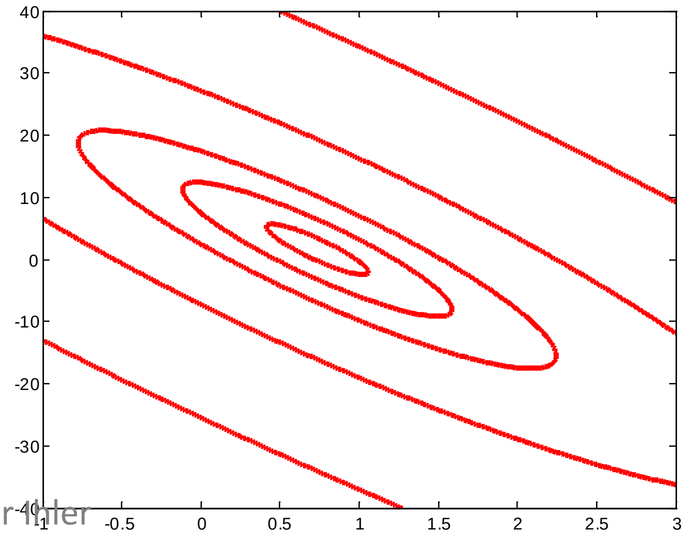
(Matlab) `>> e = y' - th*X'; J = e*e'/m;`

Visualizing the cost function



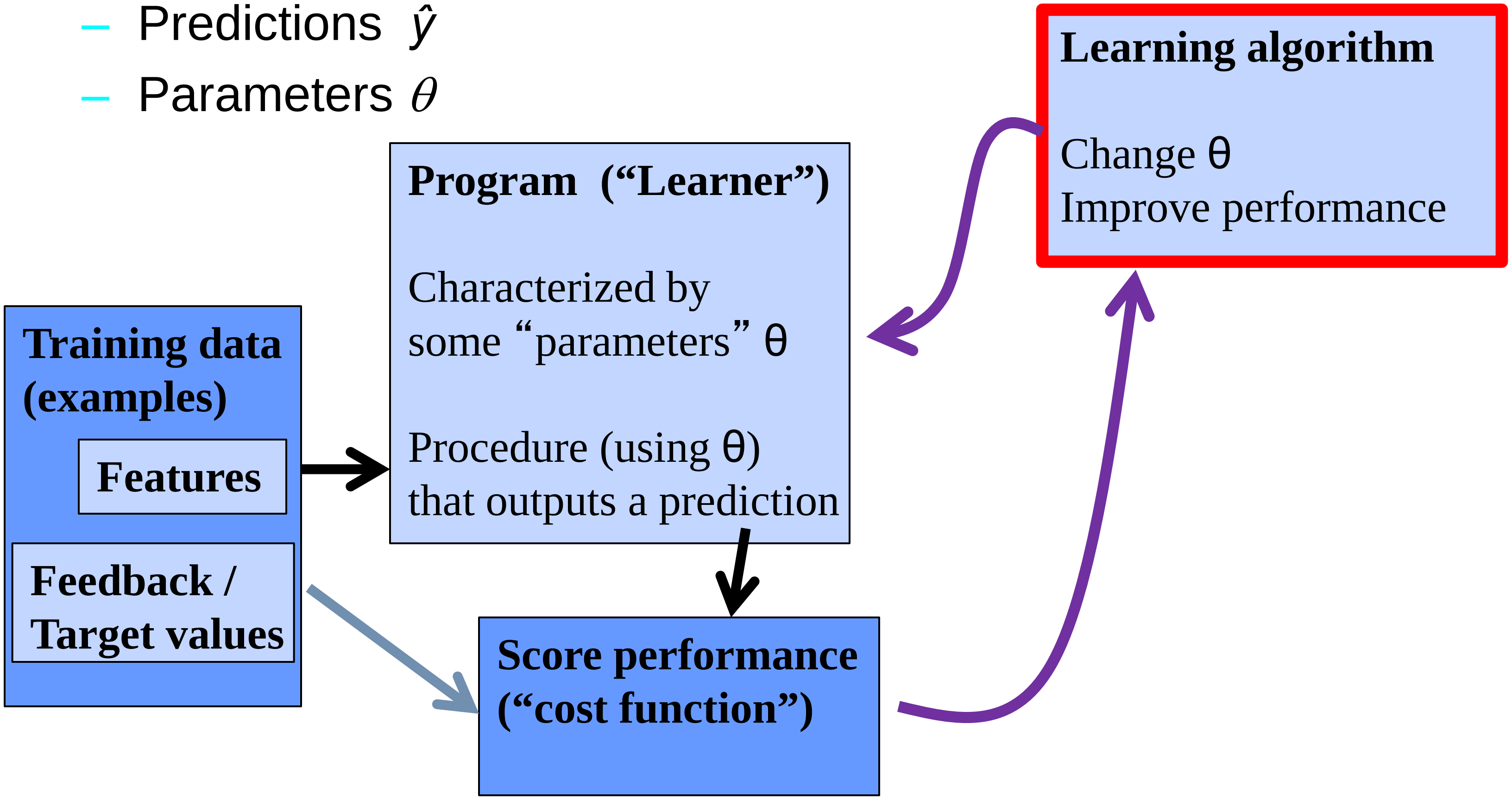






Supervised learning

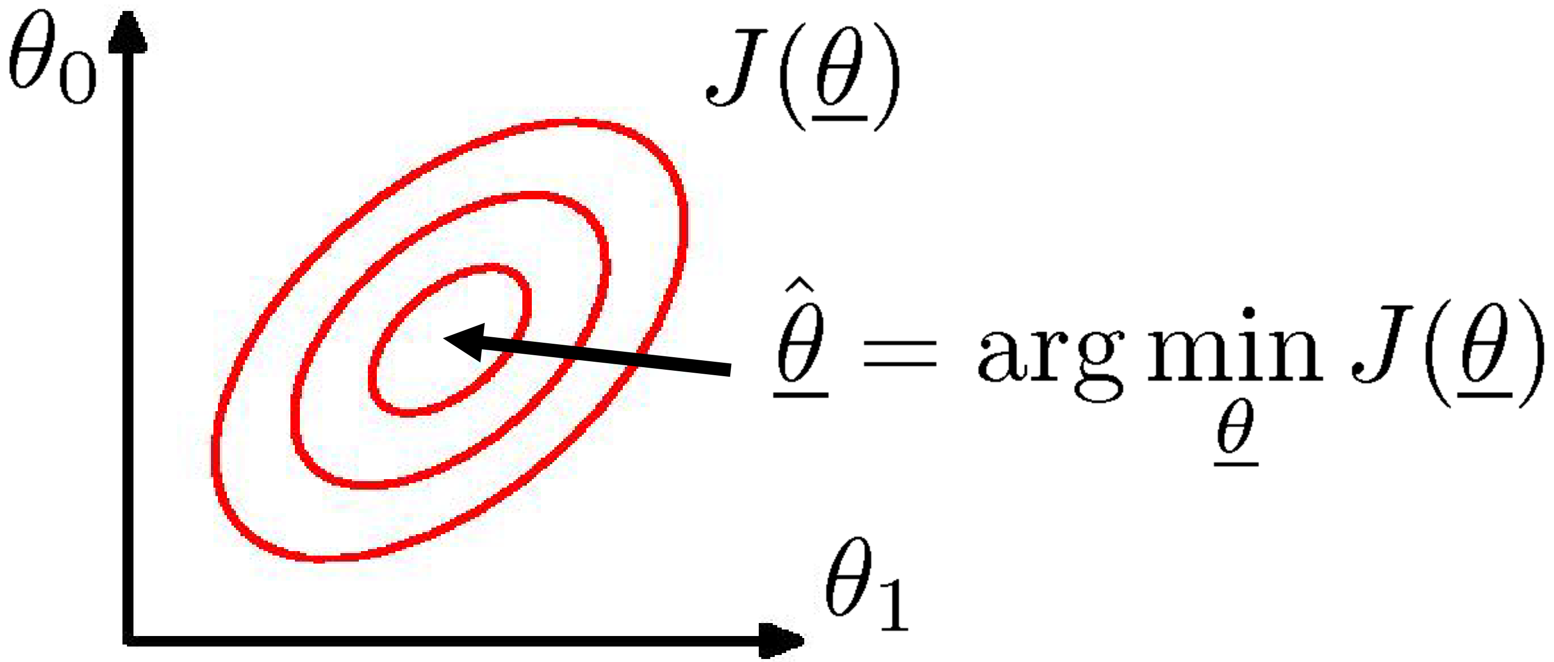
- Notation
  - Features x
  - Targets y
  - Predictions $\hat{y}$
  - Parameters θ



Finding good parameters

- Want to find parameters which minimize our error…

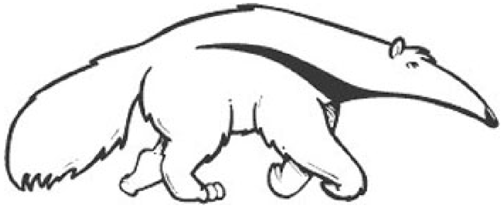
- Think of a cost "surface": error residual for that θ…



Machine Learning and Data Mining

Linear regression: direct minimization

(adapted from) Prof. Alexander Ihler



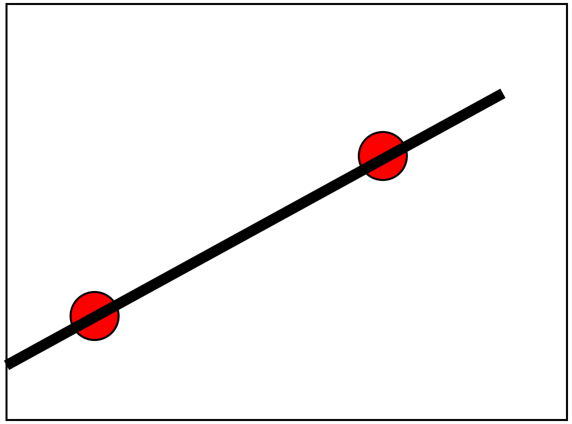
BREN:ICS
INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

MSE Minimum

- Consider a simple problem
  - One feature, two data points
  - Two unknowns: µ0, µ1
  - Two equations:

$$y^{(1)} = \theta_0 + \theta_1 x^{(1)}$$
$$y^{(2)} = \theta_0 + \theta_1 x^{(2)}$$



- Can solve this system directly:

$$\underline{y}^T = \underline{\theta}\, \underline{X}^T \qquad \Rightarrow \qquad \underline{\hat{\theta}} = y^T (\underline{X}^T)^{-1}$$

- However, most of the time, m > n
  - There may be no linear function that hits all the data exactly
  - Instead, solve directly for minimum of MSE function

SSE Minimum

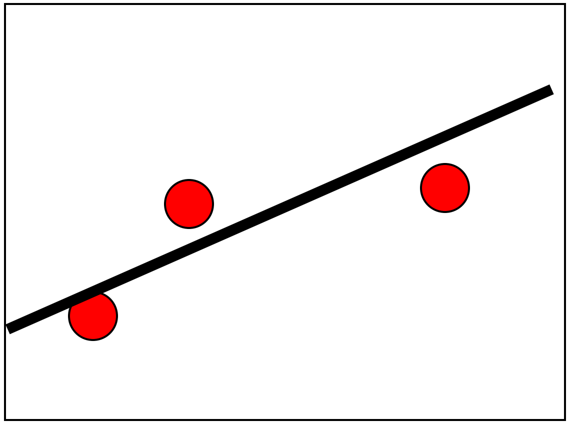
$$\nabla J(\underline{\theta}) = -(\underline{y}^T - \underline{\theta X}^T) \cdot \underline{X} \quad = \quad \underline{0}$$

- Reordering, we have

$$\underline{y}^T \underline{X} - \underline{\theta X}^T \cdot \underline{X} \quad = \quad \underline{0}$$

$$\underline{y}^T \underline{X} = \underline{\theta X}^T \cdot \underline{X}$$

$$\underline{\theta} \quad = \quad \underline{y}^T \underline{X}(\underline{X}^T \underline{X})^{-1}$$



- $X\ (X^T\ X)^{-1}$ is called the "pseudo-inverse"

- If X^T is square and independent, this is the inverse
- If m > n: overdetermined; gives minimum MSE fit

Matlab SSE

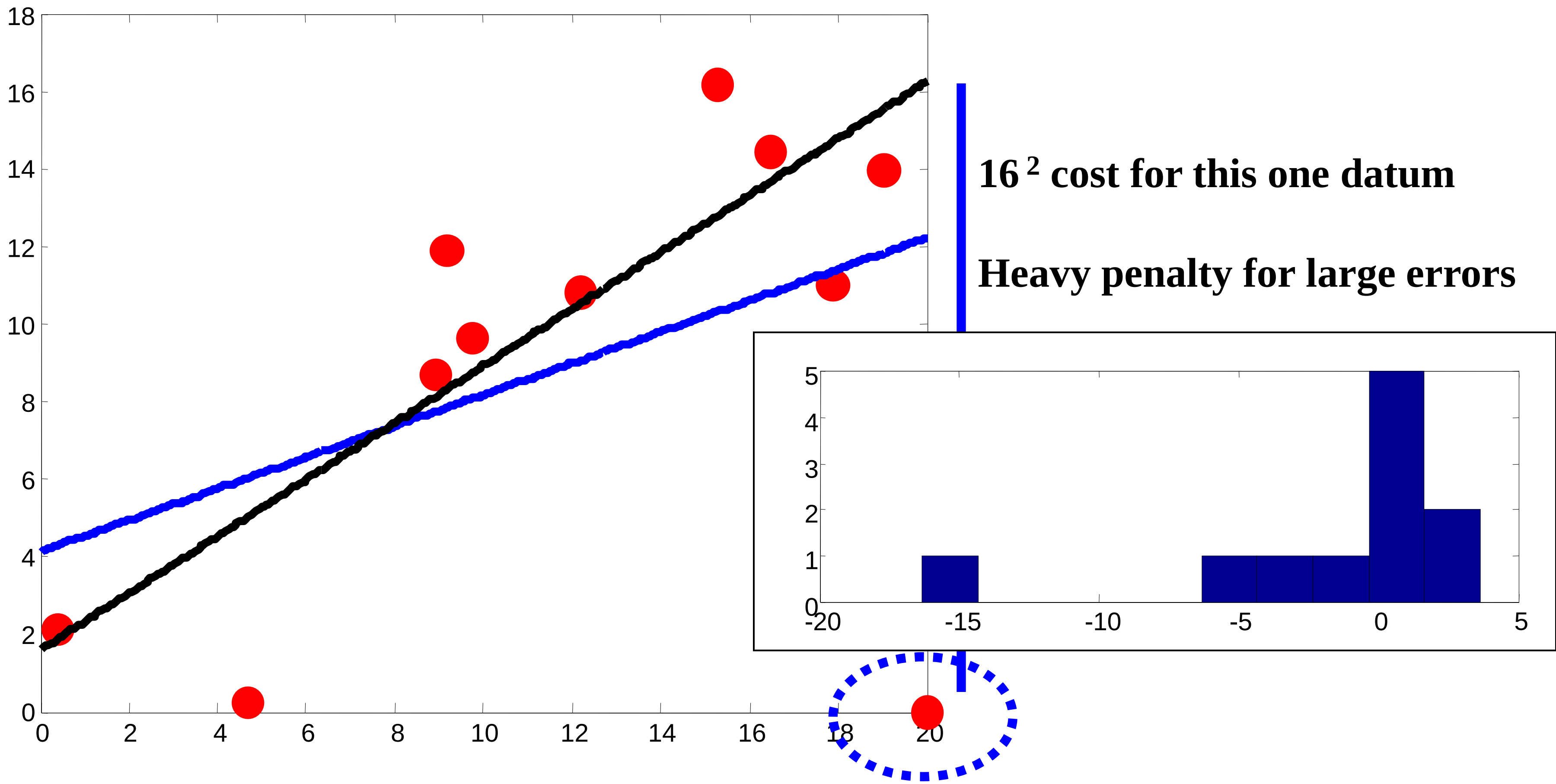
- This is easy to solve in Matlab…

$$\underline{\theta} \quad = \quad \underline{y}^T \, \underline{X} (\underline{X}^T \, \underline{X})^{-1}$$

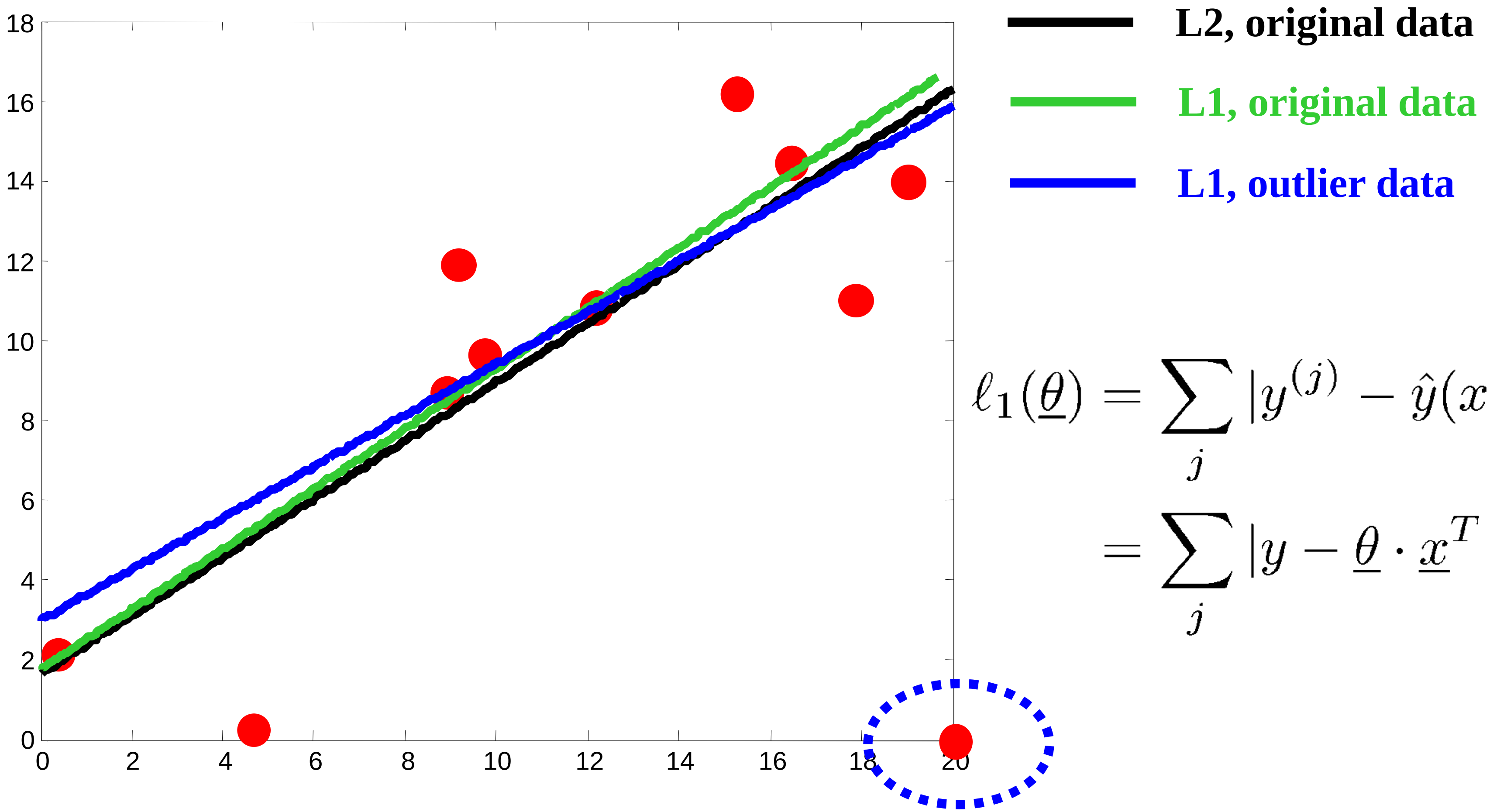
```
%  y = [y1 ; … ; ym]
%  X = [x1_0 … x1_m ; x2_0 … x2_m ; …]


% Solution 1: “manual”
   th = y’ * X * inv(X’ * X);


% Solution 2: “mrdivide”
   th = y’ / X’;     % th*X’ = y  =>  th = y/X’
```

Effects of MSE choice

- Sensitivity to outliers



L1 error



$$\ell_1(\underline{\theta}) = \sum_j \left| y^{(j)} - \hat{y}(x^{(j)}) \right|$$

$$= \sum_j \left| y - \underline{\theta} \cdot \underline{x}^T \right|$$

Cost functions for regression

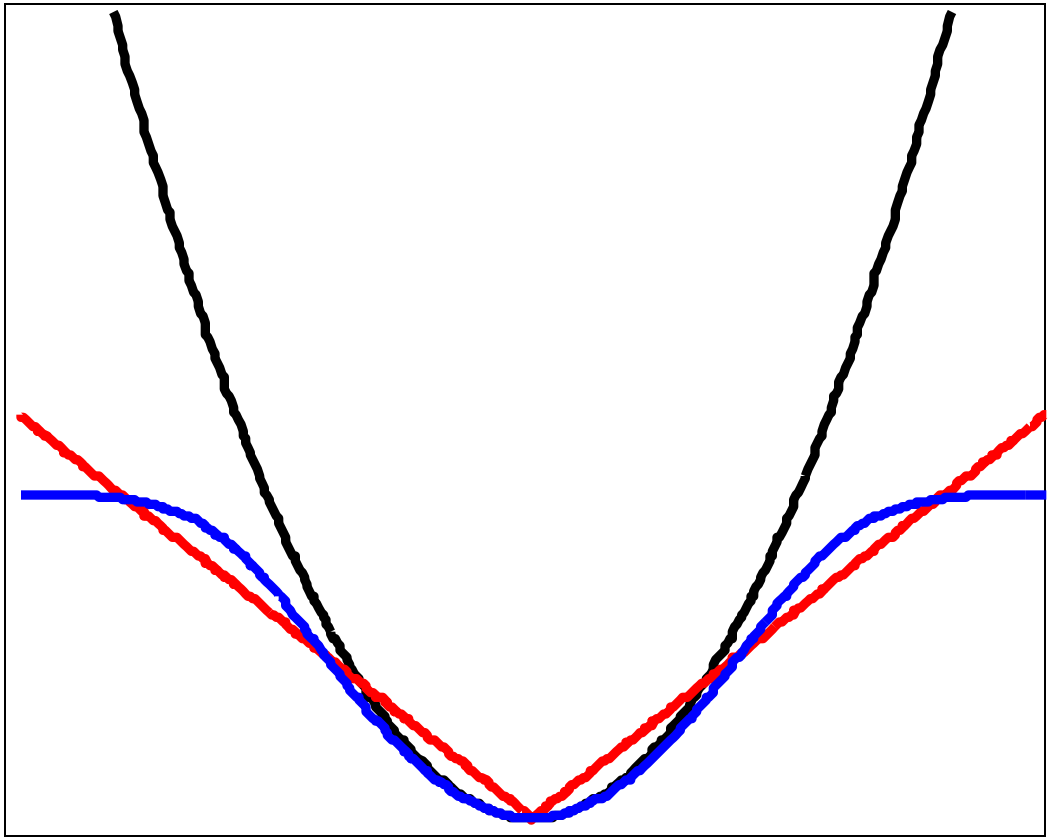
$$\ell_2 \; : \; (y - \hat{y})^2$$ **(MSE)**

$$\ell_1 \; : \; |y - \hat{y}|$$ **(MAE)**

Something else entirely…

$$c - \log(\exp(-(y - \hat{y})^2) + c)$$

(???)

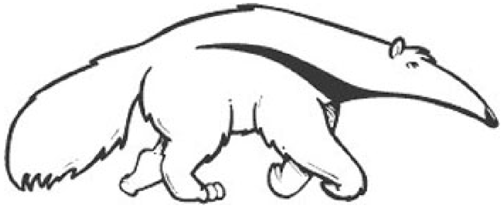


"Arbitrary" functions can't be solved in closed form…
- use gradient descent

Machine Learning and Data Mining

Linear regression: nonlinear features

(adapted from) Prof. Alexander Ihler

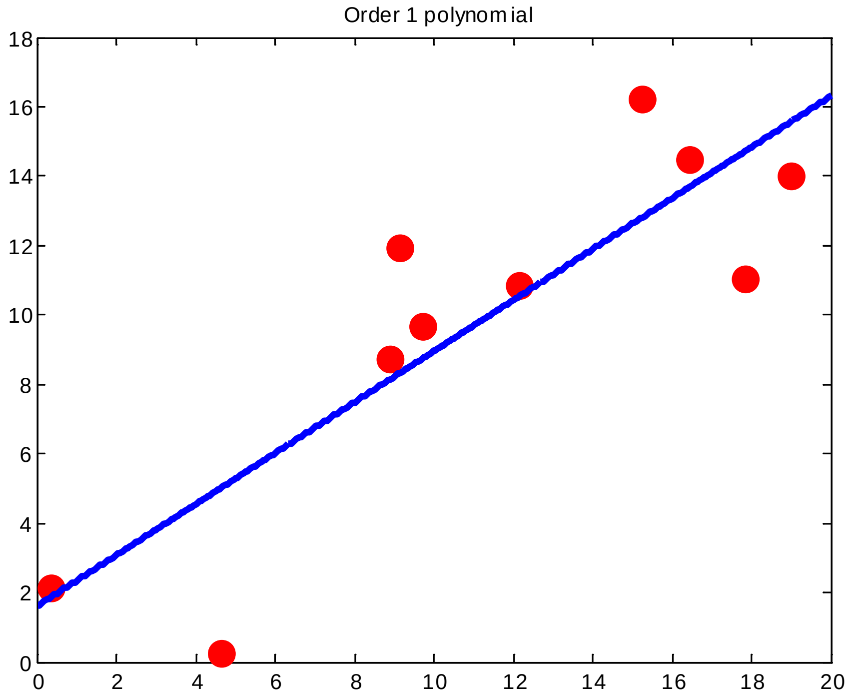


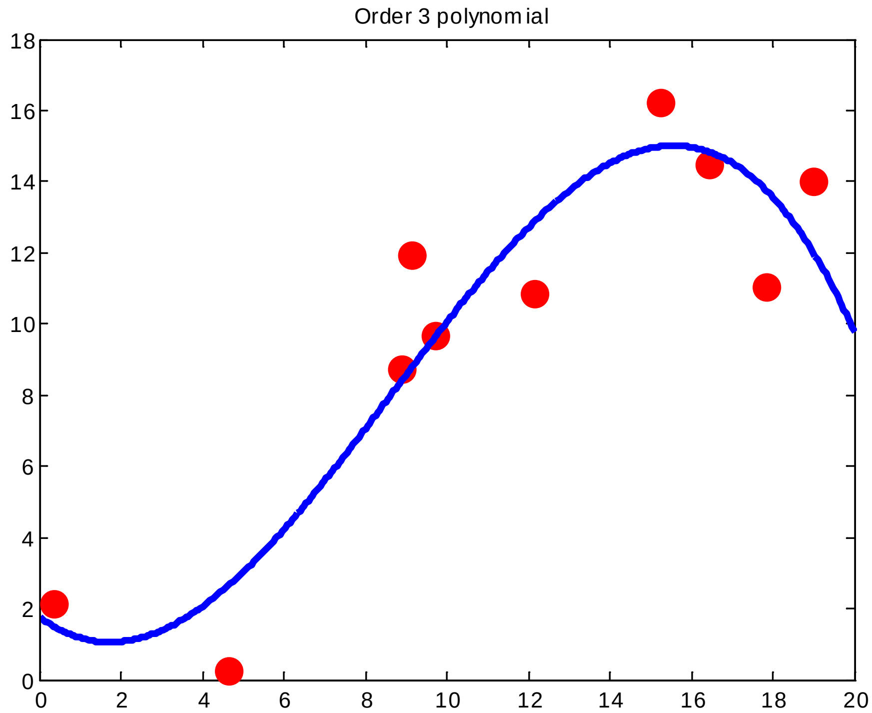
BREN:ICS INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

Nonlinear functions

- What if our hypotheses are not lines?
  - Ex: higher-order polynomials





Nonlinear functions

- Single feature x, predict target y:

$$D = \{(x^{(j)}, y^{(j)})\}$$

$$\hat{y}(x) = \theta_0 + \theta_1\, x + \theta_2\, x^2 + \theta_3\, x^3$$

$$\Downarrow \quad \text{Add features:}$$

$$D = \{([x^{(j)}, (x^{(j)})^2, (x^{(j)})^3], y^{(j)})\}$$

$$\Downarrow$$

$$\hat{y}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Linear regression in new features

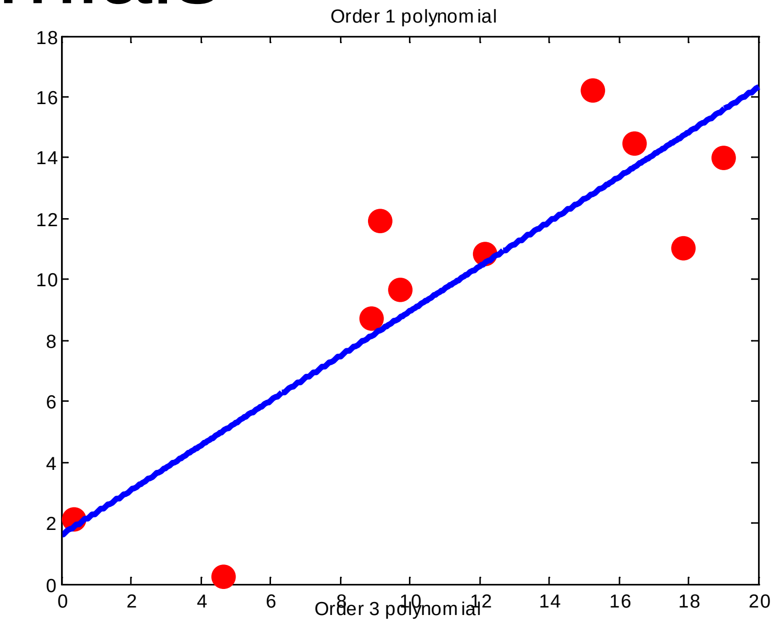
- Sometimes useful to think of “feature transform”

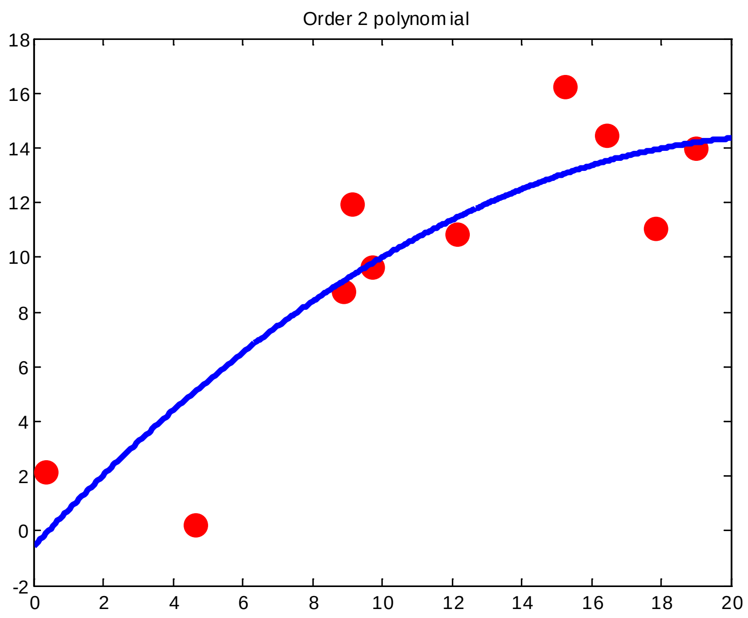
$$\Phi(x) = \left[1\,,\ x\,,\ x^2\,,\ x^3\,,\ \ldots\right]$$

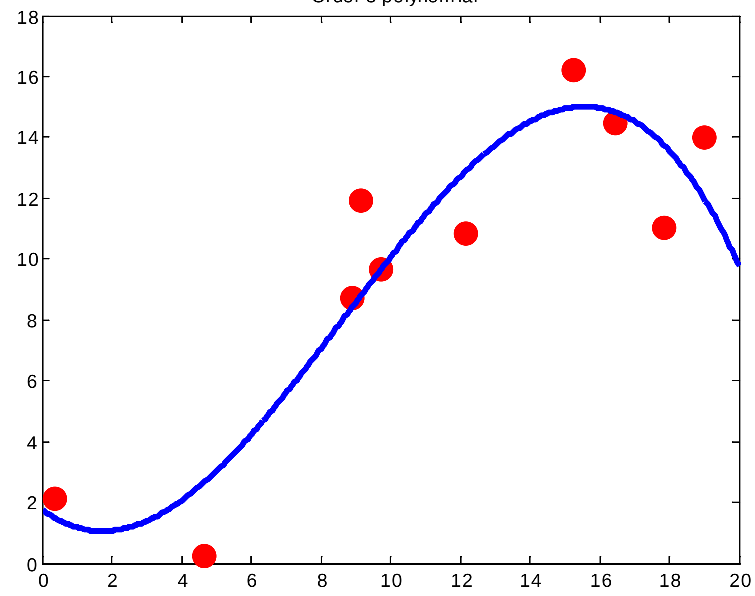
$$\hat{y}(x) = \underline{\theta} \cdot \Phi(x)$$

Higher-order polynomials

- Fit in the same way
- More “features”







Features

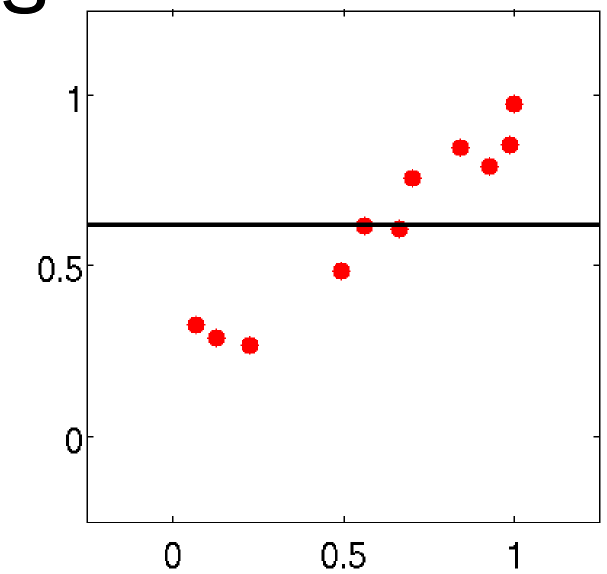
- In general, can use any features we think are useful

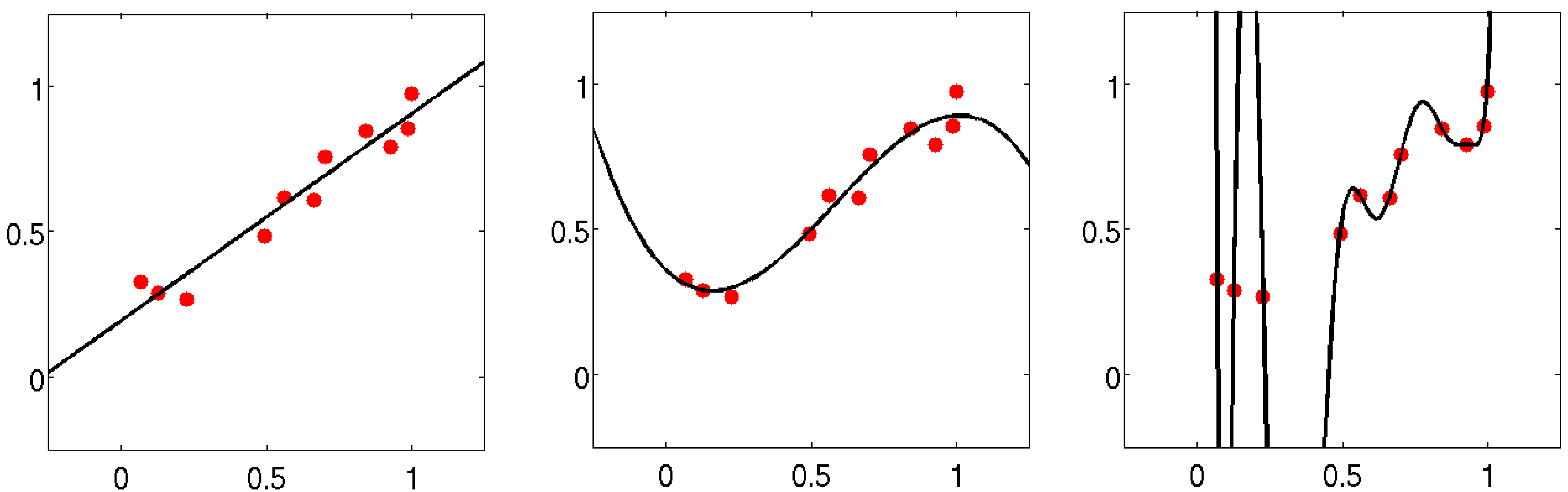
- Other information about the problem
  - Sq. footage, location, age, …
- Polynomial functions
  - Features $[1, x, x^2, x^3, \ldots]$
- Other functions
  - $1/x$, $\mathrm{sqrt}(x)$, $x_1 * x_2$, …

- "Linear regression" = linear in the parameters
  - Features we can make as complex as we want!

Higher-order polynomials

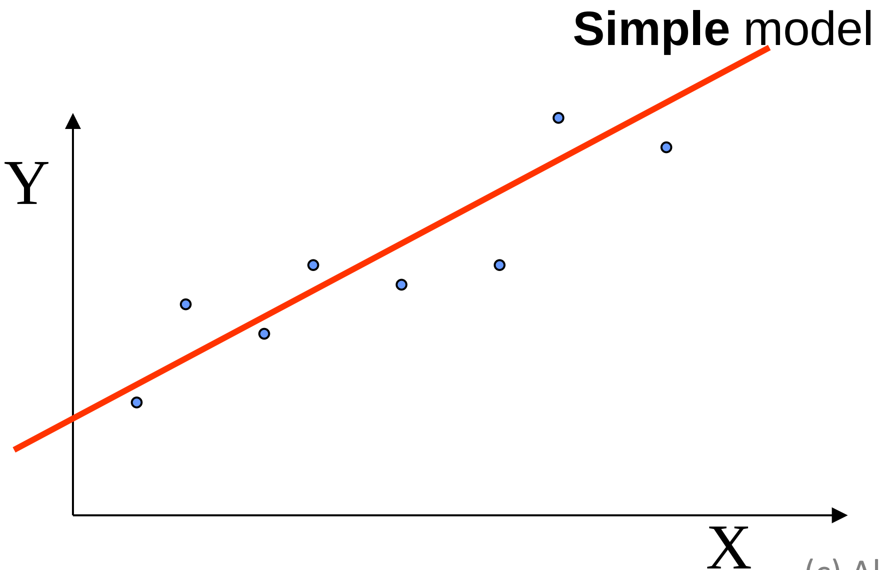
- Are more features better?
- “Nested” hypotheses
  - 2^{nd} order more general than 1^{st},
  - 3^{rd} order “ “ than 2^{nd}, …
- Fits the observed data better

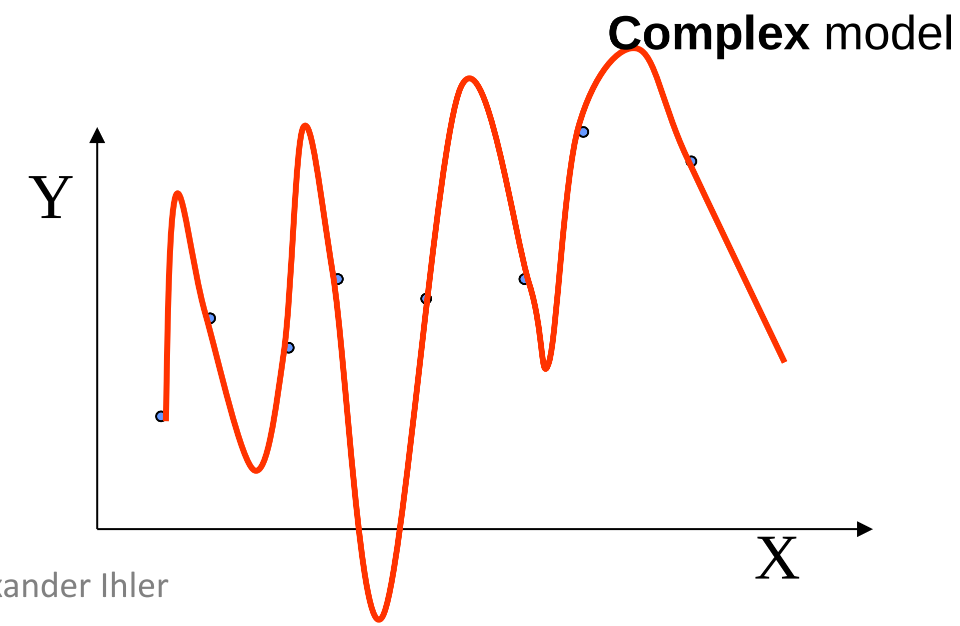




Overfitting and complexity

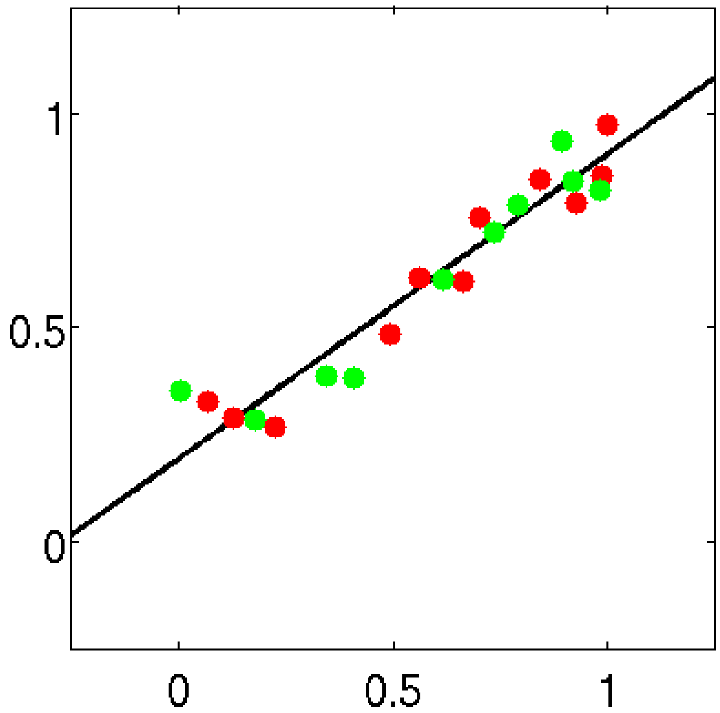
- More complex models will always fit the training data better
- But they may "overfit" the training data, learning complex relationships that are not really present

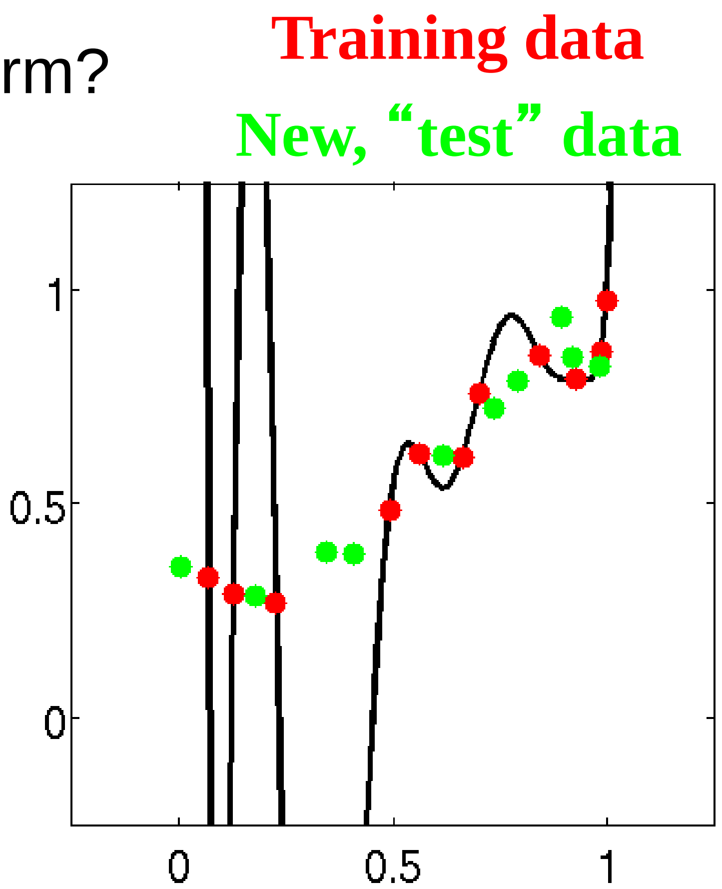




Test data

- After training the model
- Go out and get more data from the world
  - New observations (x,y)
- How well does our model perform?





Training versus test error

- Plot MSE as a function of model complexity
  - Polynomial order
- Decreases
  - More complex function fits training data better
- What about new data?
- 0^{th} to 1^{st} order
  - Error decreases
  - Underfitting
- Higher order
  - Error increases
  - Overfitting