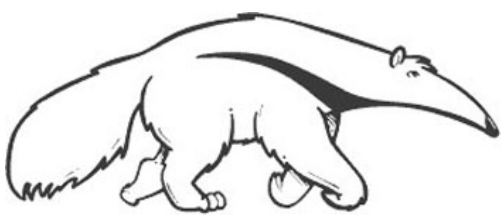
Machine Learning and Data Mining

Linear regression

Prof. Alexander Ihler

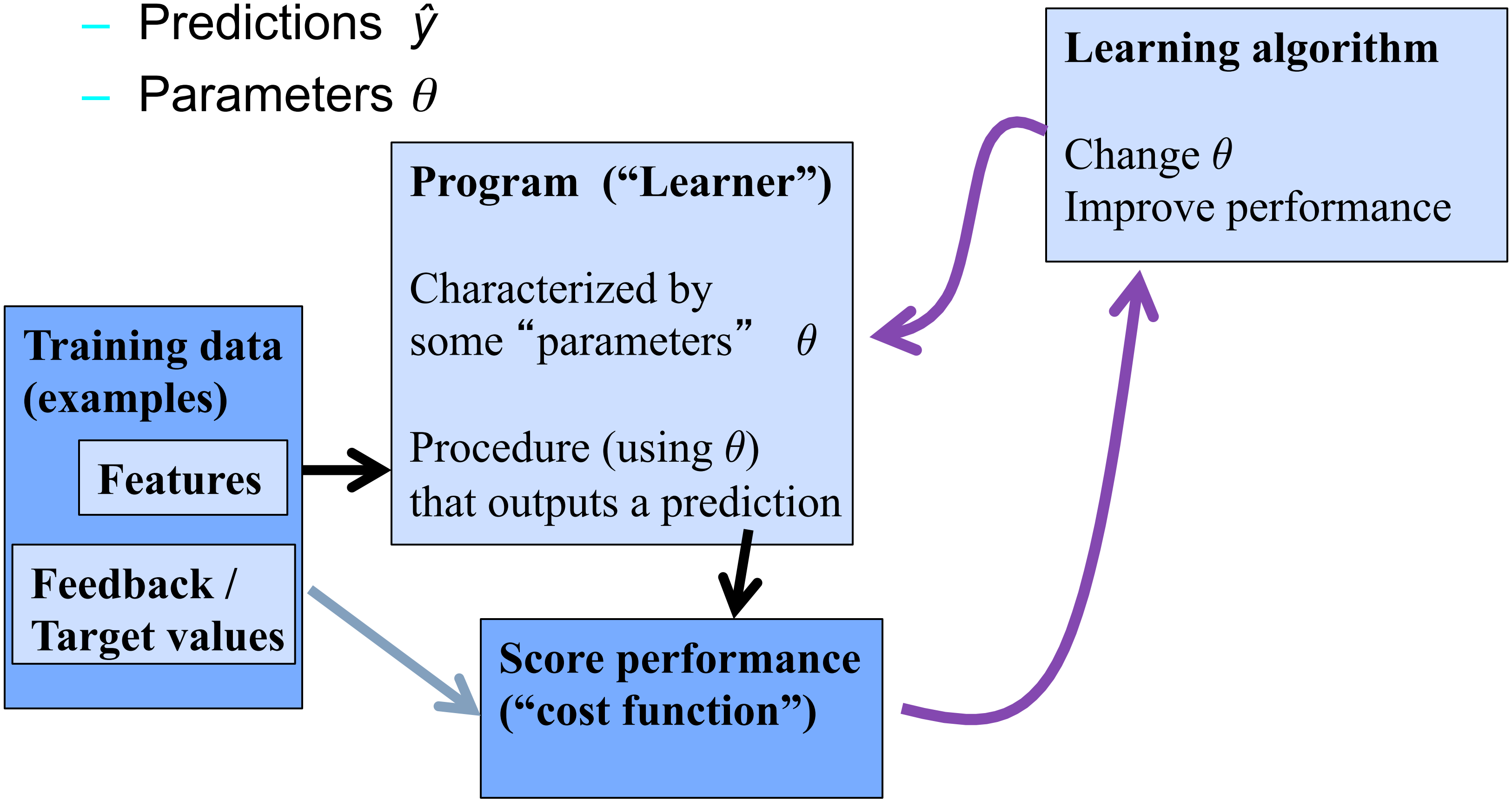


BREN:ICS
INFORMATION AND COMPUTER SCIENCES

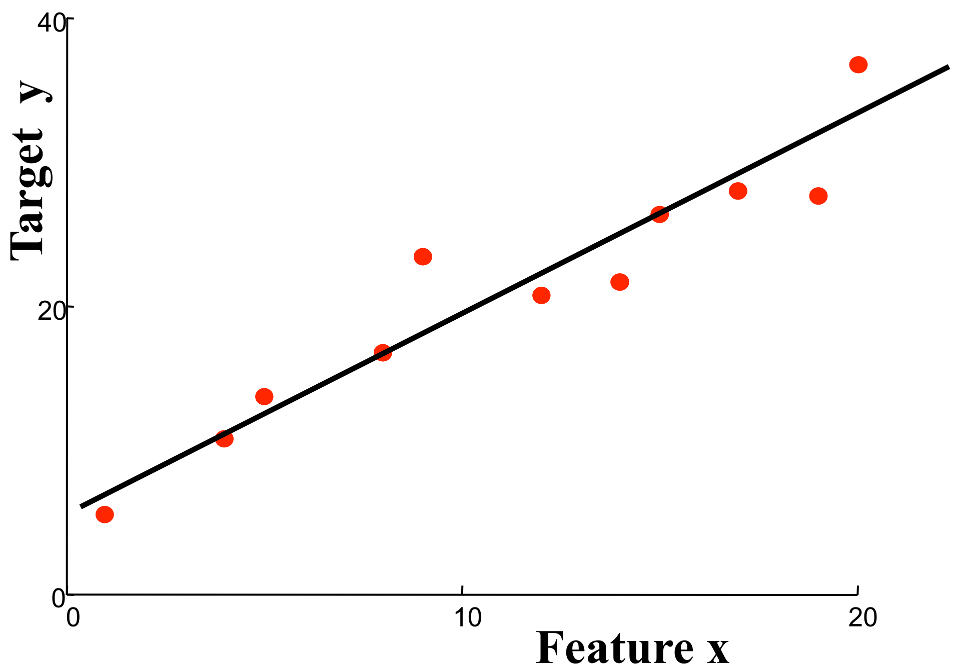
UNIVERSITY of CALIFORNIA IRVINE

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

Notation

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

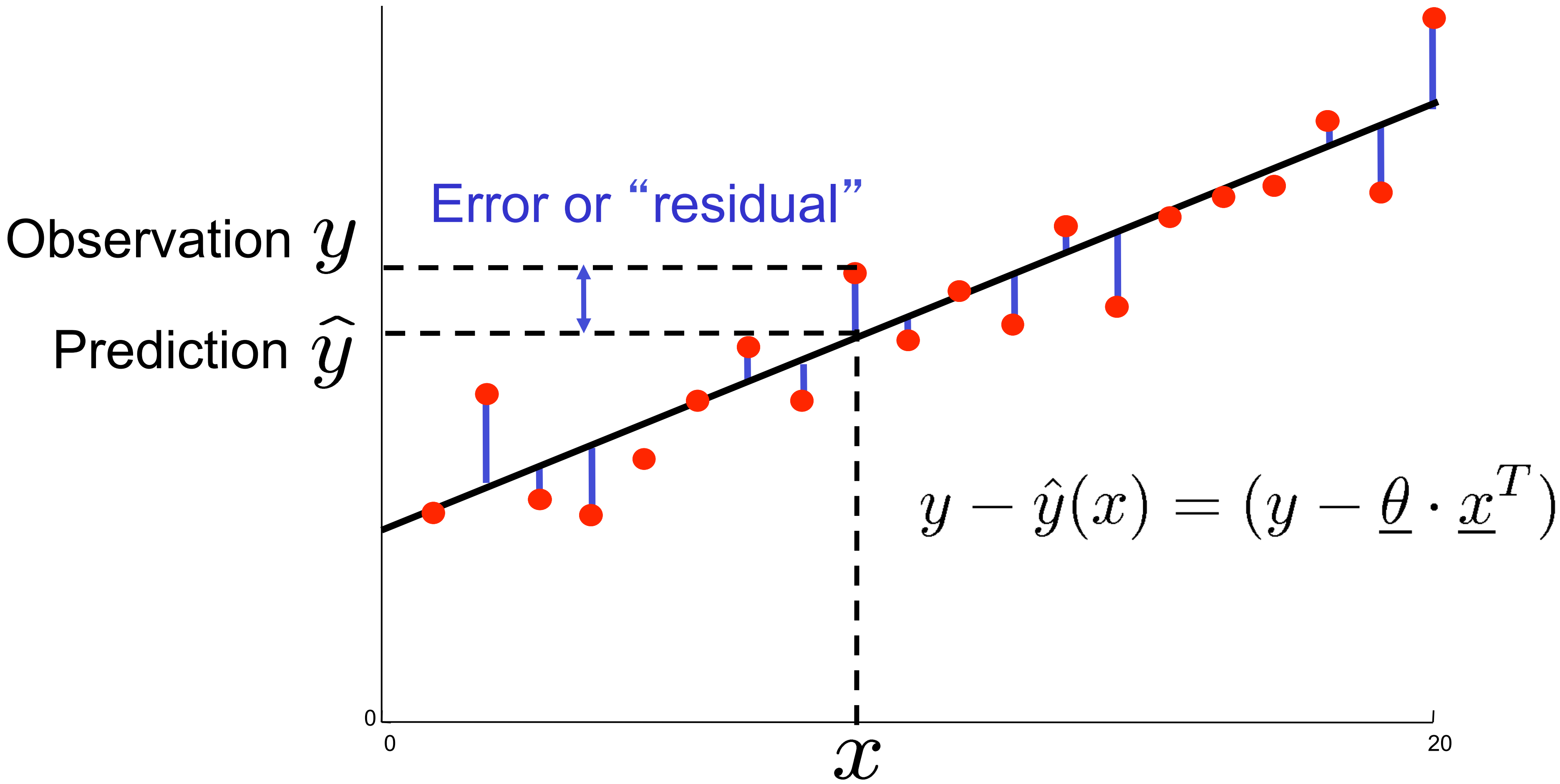
Define "feature" $x_0 = 1$ (constant)
Then

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

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

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

Measuring error



Mean squared error

- How can we quantify the error?

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

- 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

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

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

- Rewrite using matrix form

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

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

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

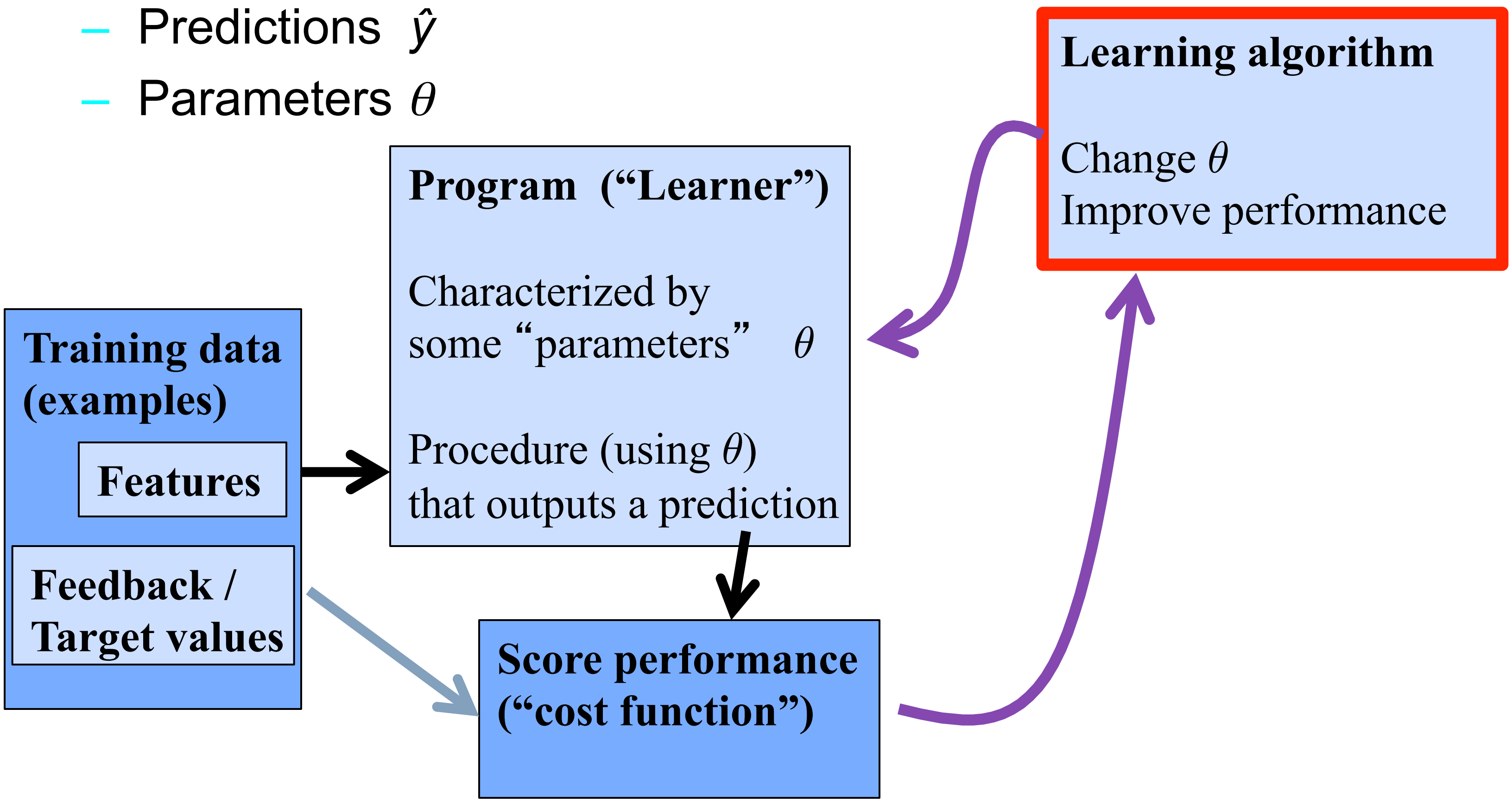
$$J(\underline{\theta}) = \frac{1}{m}(\underline{y} - \underline{\theta}\,\underline{X}^T) \cdot (\underline{y} - \underline{\theta}\,\underline{X}^T)^T$$

(Matlab)

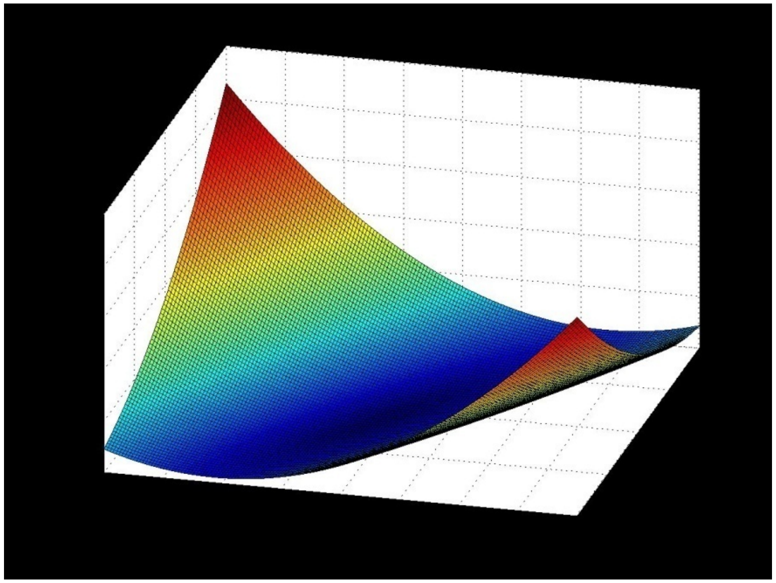
```
>> e = y' - th*X';    J = e*e'/m;
```

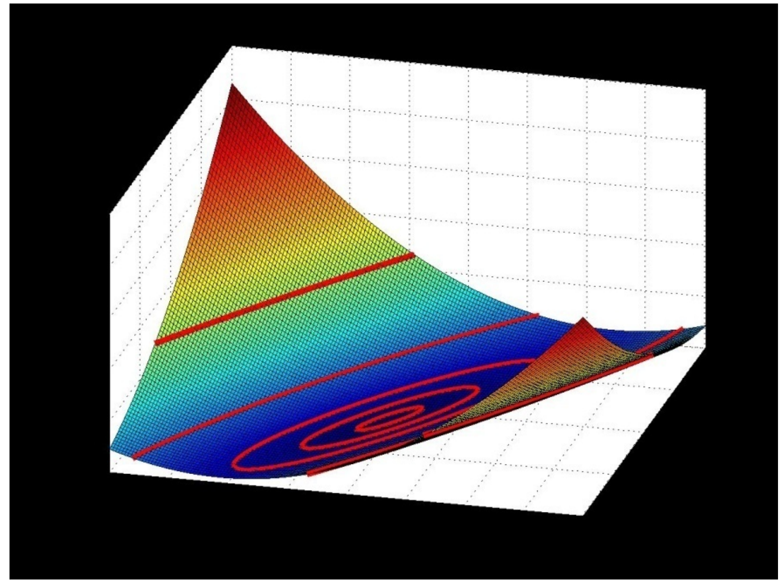
Supervised learning

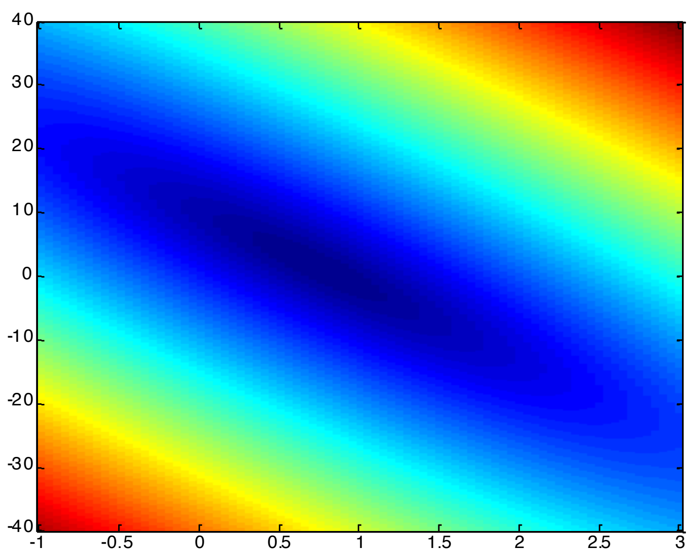
- Notation
  - Features x
  - Targets y
  - Predictions $\hat{y}$
  - Parameters θ

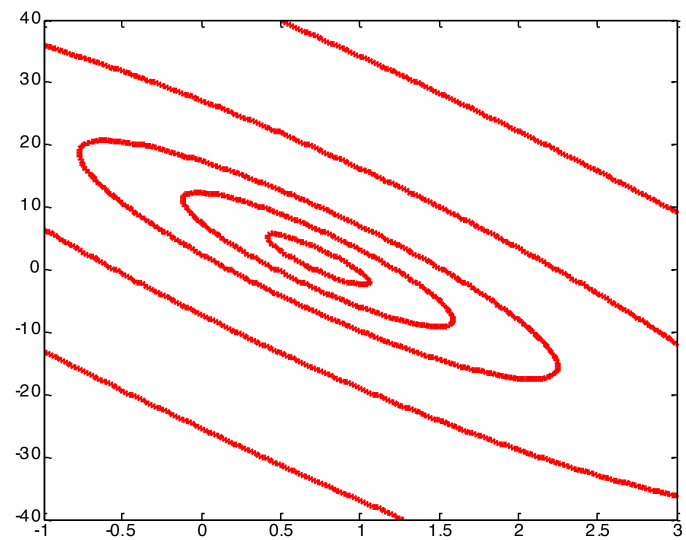


Visualizing the cost function



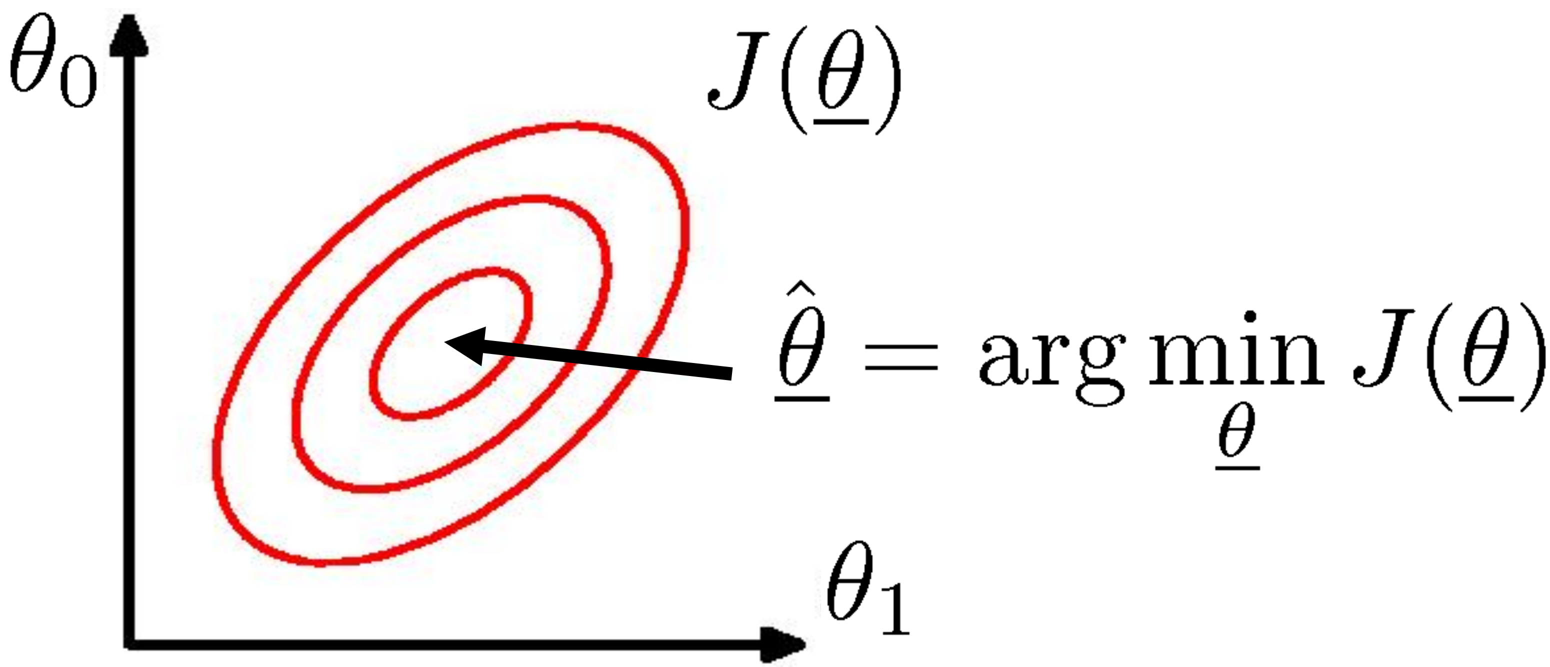






Finding good parameters

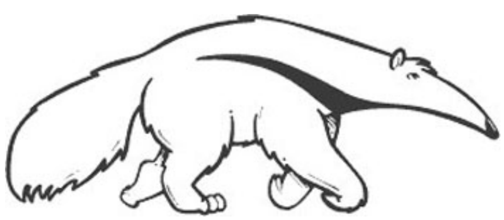
- Want to find parameters which minimize our error…
- Think of a cost "surface": error residual for that θ…



Machine Learning and Data Mining

Linear regression: Gradient descent & stochastic gradient descent

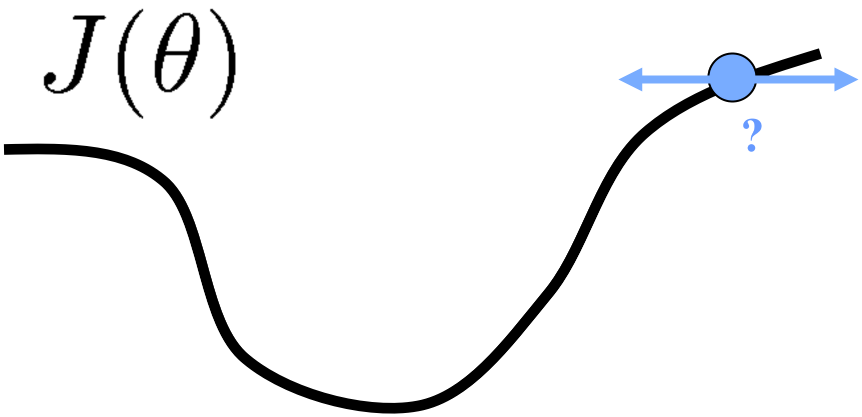
Prof. Alexander Ihler



BREN:ICS
INFORMATION AND COMPUTER SCIENCES

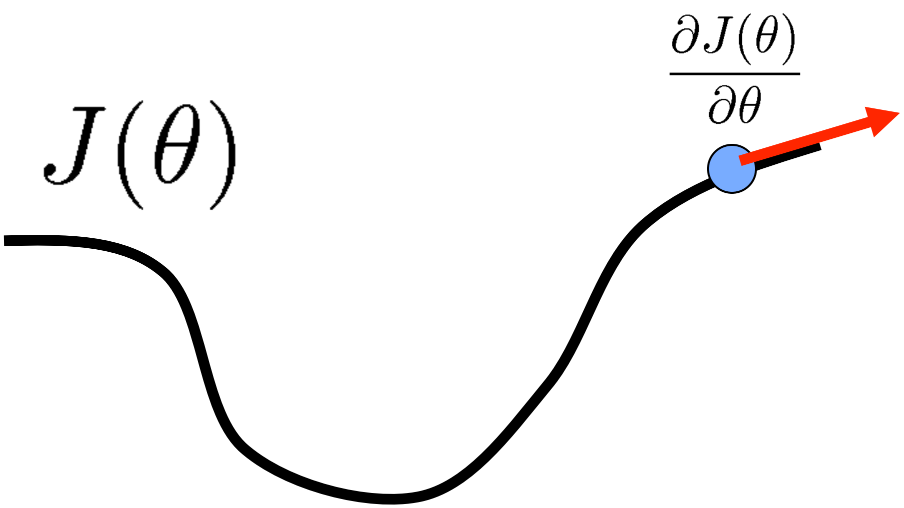
UNIVERSITY of CALIFORNIA IRVINE

Gradient descent



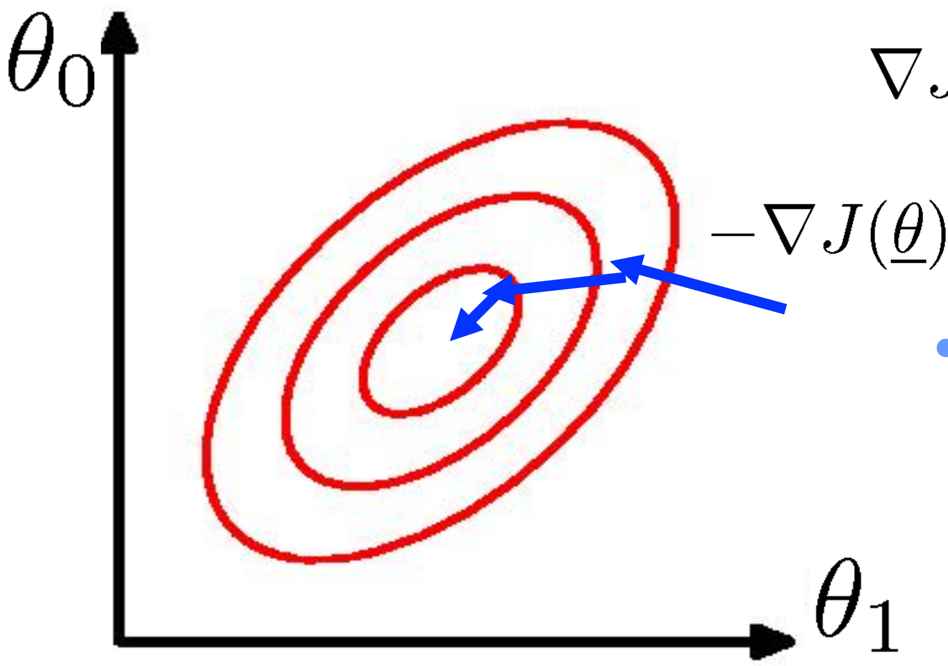
- How to change θ to improve J(θ)?
- Choose a direction in which J(θ) is decreasing

Gradient descent



- How to change θ to improve J(θ)?
- Choose a direction in which J(θ) is decreasing
- Derivative $\frac{\partial J(\theta)}{\partial \theta}$
- Positive => increasing
- Negative => decreasing

Gradient descent in more dimensions



- Gradient vector

$$\nabla J(\underline{\theta}) = \left[\frac{\partial J(\underline{\theta})}{\partial \theta_0} \quad \frac{\partial J(\underline{\theta})}{\partial \theta_1} \quad \ldots \right]$$

- Indicates direction of steepest ascent

  (negative = steepest descent)

Gradient descent

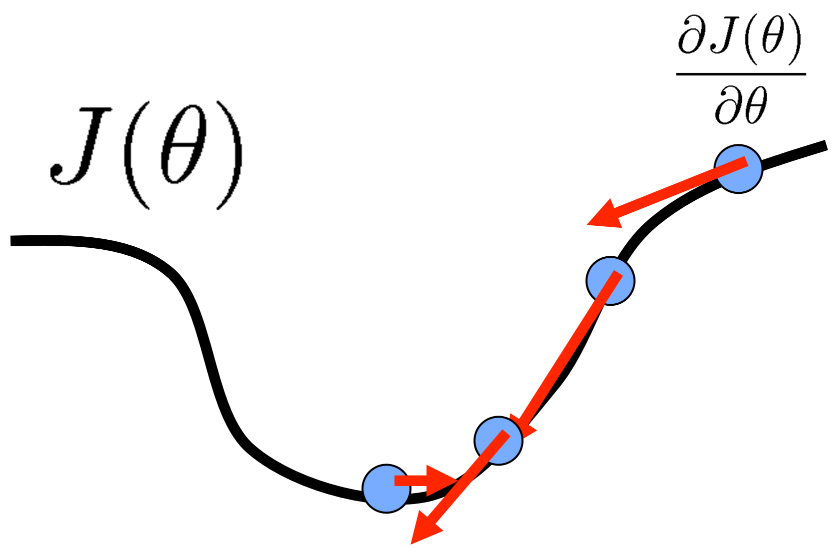
- Initialization
- Step size
  - Can change as a function of iteration
- Gradient direction
- Stopping condition

`Initialize` θ

`Do {`

$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta)$

`} while` $(\alpha \|\nabla J\| > \epsilon)$



Gradient for the MSE

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

- ∇ J = ?

$$J(\underline{\theta}) = \frac{1}{m}\sum_j (\overbrace{y^{(j)} - \theta_0 \underline{x}_0^{(j)} - \theta_1 \underline{x}_1^{(j)} - \ldots}^{e_j(\theta)})^2$$

$$\frac{\partial J}{\partial \theta_0} = \frac{\partial}{\partial \theta_0}\frac{1}{m}\sum_j (\ e_j(\theta)\ )^2$$

$$= \frac{1}{m}\sum_j \ \frac{\partial}{\partial \theta_0}(\ e_j(\theta)\ )^2$$

$$= \frac{1}{m}\sum_j \ 2 e_j(\theta)\ \frac{\partial}{\partial \theta_0} e_j(\theta)$$

$$\frac{\partial}{\partial \theta_0} e_j(\theta) = \cancelto{0}{\frac{\partial}{\partial \theta_0} y^{(j)}} - \frac{\partial}{\partial \theta_0}\theta_0 x_0^{(j)} - \cancelto{0}{\frac{\partial}{\partial \theta_0}\theta_1 x_1^{(j)}} - \ldots$$

$$= -x_0^{(j)}$$

Gradient for the MSE

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

- ∇ J = ?

$$J(\underline{\theta}) = \frac{1}{m} \sum_j (\overbrace{y^{(j)} - \theta_0 \underline{x}_0^{(j)} - \theta_1 \underline{x}_1^{(j)} - \ldots}^{e_j(\theta)})^2$$

$$\nabla J(\underline{\theta}) = \left[\quad \frac{\partial J}{\partial \theta_0} \qquad \frac{\partial J}{\partial \theta_1} \qquad \ldots \right]$$

$$= \left[\frac{1}{m} \sum_j -e_j(\theta) x_0^{(j)} \quad \frac{1}{m} \sum_j -e_j(\theta) x_1^{(j)} \qquad \ldots \right]$$

Gradient descent

- Initialization
- Step size
  - Can change as a function of iteration
- Gradient direction
- Stopping condition

Initialize θ

Do {

$\theta \leftarrow \theta$ - $\alpha\ \nabla_\theta$ J(θ)

} while (α||∇J|| > ϵ)

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

$$\nabla J(\underline{\theta}) = -\frac{2}{m}\sum_j \left(y^{(j)} \underbrace{- \underline{\theta}\cdot\underline{x}^{(j)^T}}_{\text{Error magnitude \& direction for datum j}}\right)\cdot \left[\underbrace{x_0^{(j)}\, x_1^{(j)}}_{\text{Sensitivity to each } \theta_i} \ldots\right]$$

Derivative of SSE

$$\nabla J(\underline{\theta}) = -\frac{2}{m}\sum_j (y^{(j)} \underbrace{- \underline{\theta} \cdot}_{\text{Error magnitude \& direction for datum j}} \underline{x}^{(j)T}) \cdot \underbrace{[x_0^{(j)} x_1^{(j)}}_{\text{Sensitivity to each } \theta_i} \ldots]$$

- Rewrite using matrix form

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

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

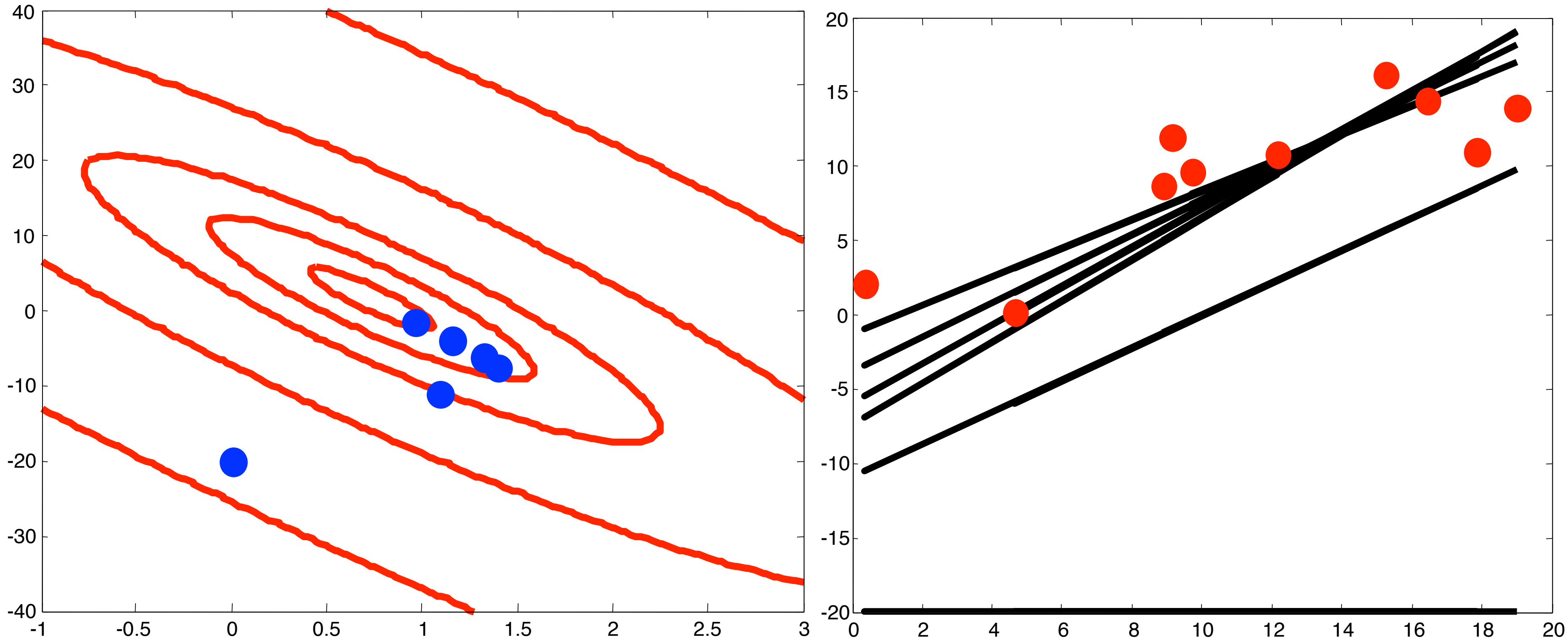
$$\underline{X} = \begin{bmatrix} x_0^{(1)} & \ldots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_0^{(m)} & \ldots & x_n^{(m)} \end{bmatrix}$$

$$\nabla J(\underline{\theta}) = -\frac{2}{m}(\underline{y}^T - \underline{\theta X}^T) \cdot \underline{X}$$

(Matlab)

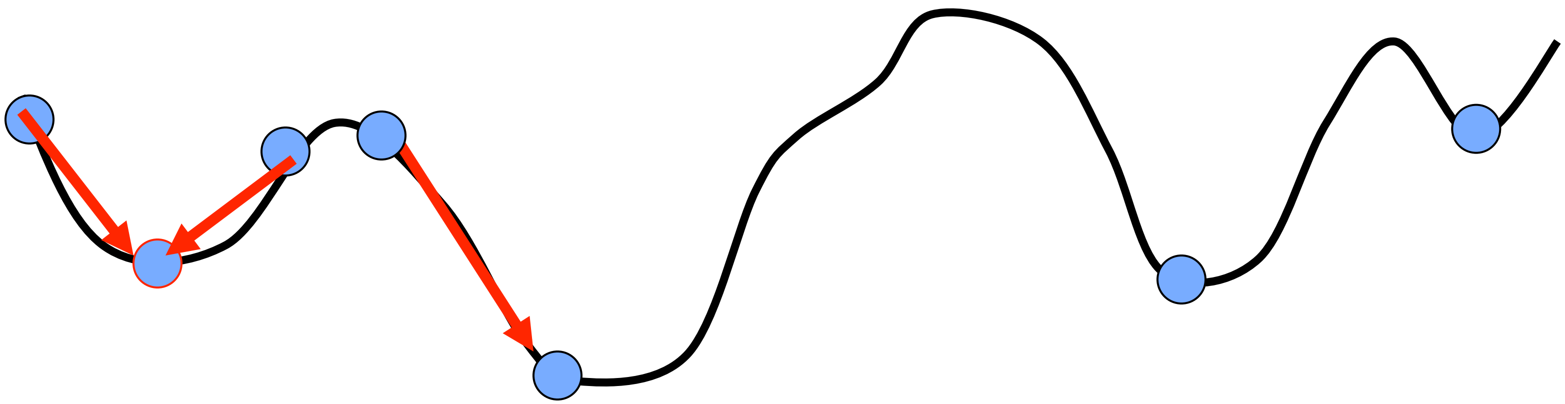
```
>> e = y' - th*X'; DJ = -e*X*2/m; th=th - al*DJ;
```

Gradient descent on cost function



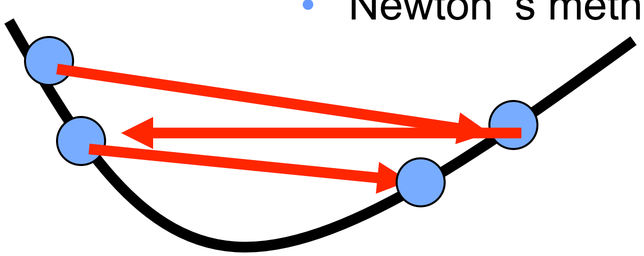
Comments on gradient descent

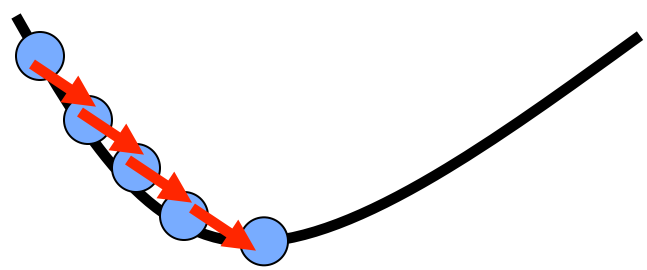
- Very general algorithm
  - we'll see it many times
- Local minima
  - Sensitive to starting point



Comments on gradient descent

- Very general algorithm
  - we'll see it many times
- Local minima
  - Sensitive to starting point
- Step size
  - Too large? Too small? Automatic ways to choose?
  - May want step size to decrease with iteration
  - Common choices:
    - Fixed
    - Linear: C/(iteration)
    - Newton's method   (we'll return to this…)





Stochastic / Online Gradient Descent

- MSE

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

- Gradient

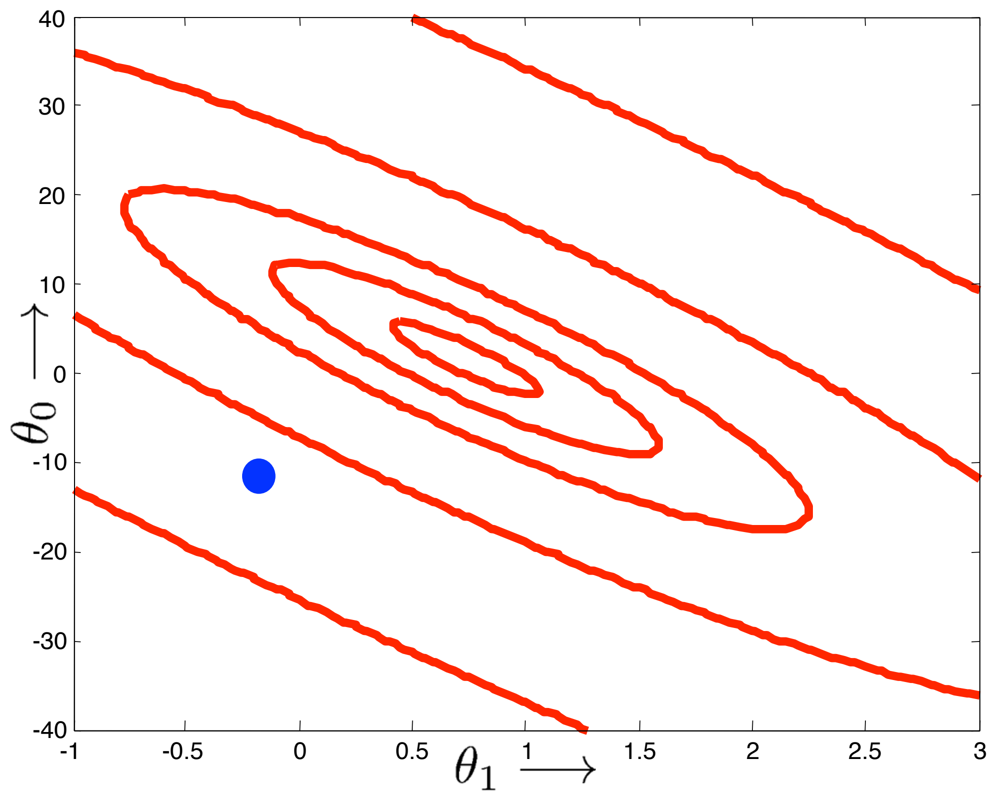
$$\nabla J(\underline{\theta}) = \frac{1}{m} \sum_j \nabla J_j(\underline{\theta}) \qquad \nabla J_j(\underline{\theta}) = (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)\,T}) \cdot [x_0^{(j)} x_1^{(j)} \ldots]$$

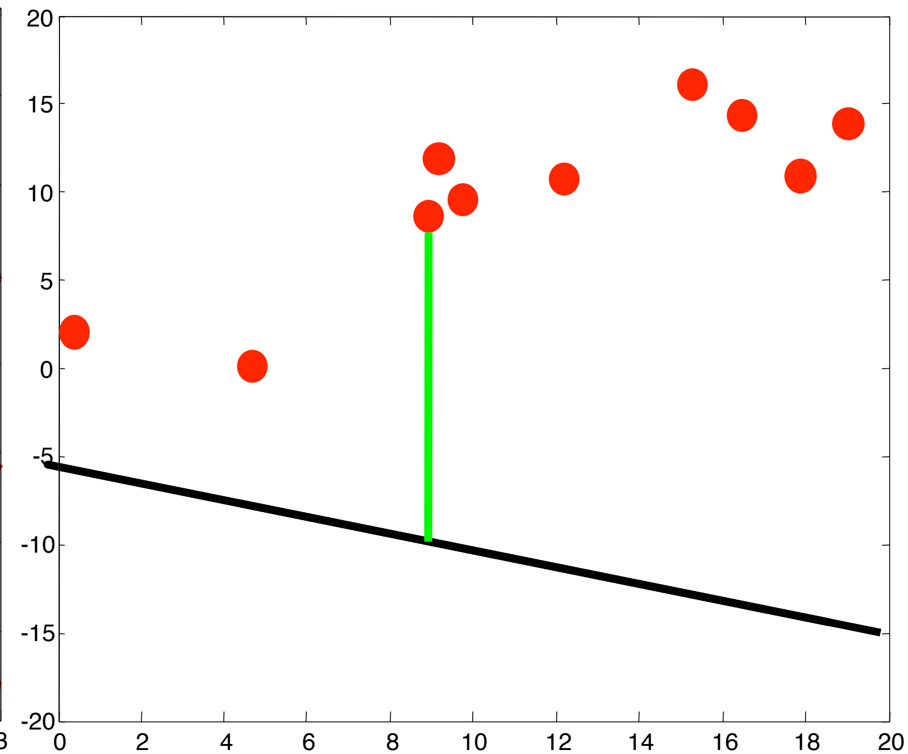
- Stochastic (or “online”) gradient descent:
  - Use updates based on individual datum j, chosen at random
  - At optima, $\mathbb{E}\left[\nabla J_j(\underline{\theta})\right] = \nabla J(\underline{\theta}) = 0$
    (average over the data)

Online gradient descent

- Update based on each datum at a time
  - Find residual and the gradient of its part of the error & update

```
Initialize θ
Do {
  for j=1:m
    θ ← θ - α ∇_θ J_j(θ)
} while (not done)
```

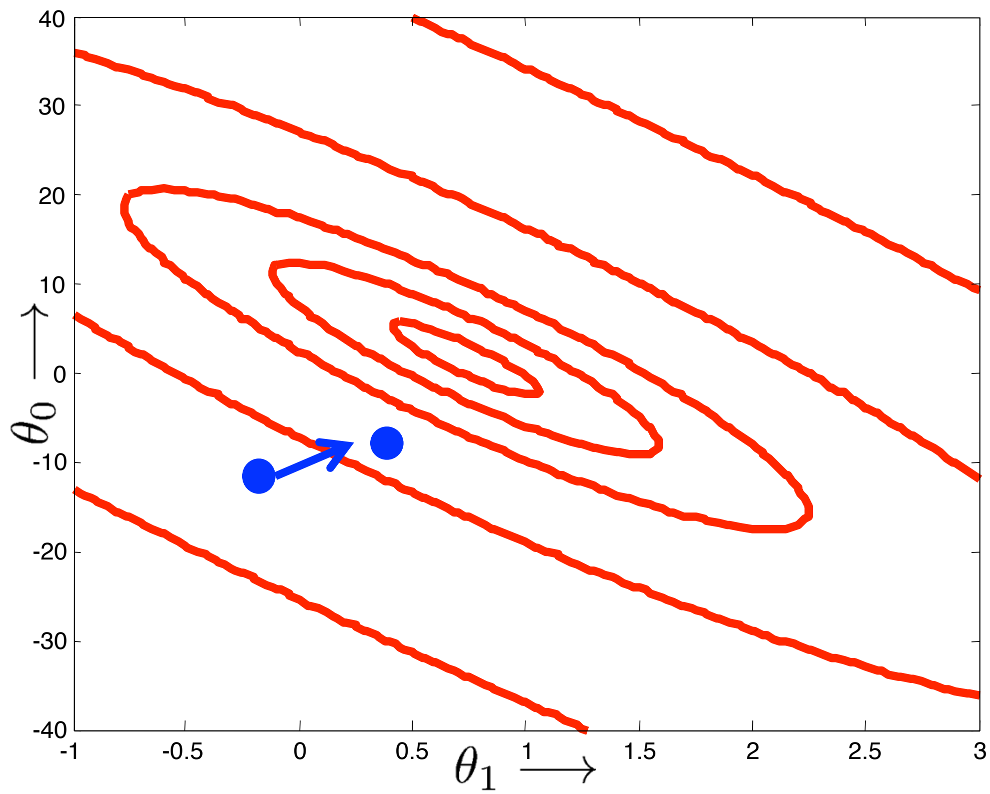


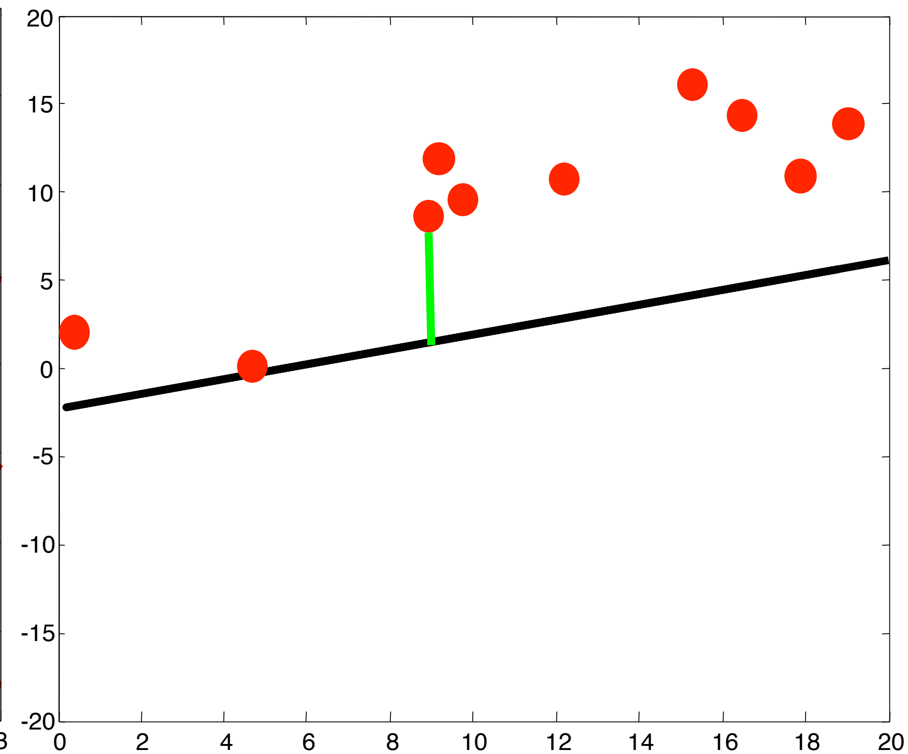


Online gradient descent

- Update based on each datum at a time
  - Find residual and the gradient of its part of the error & update

```
Initialize θ
Do {
  for j=1:m
    θ ← θ - α ∇_θ J_j(θ)
} while (not done)
```

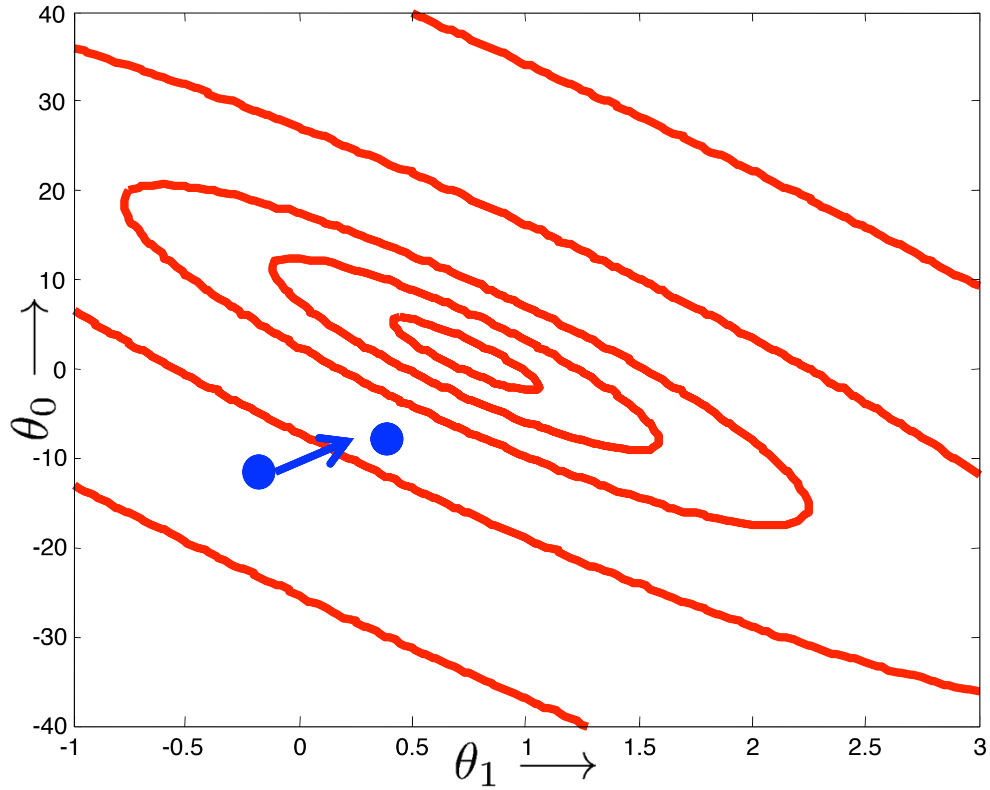


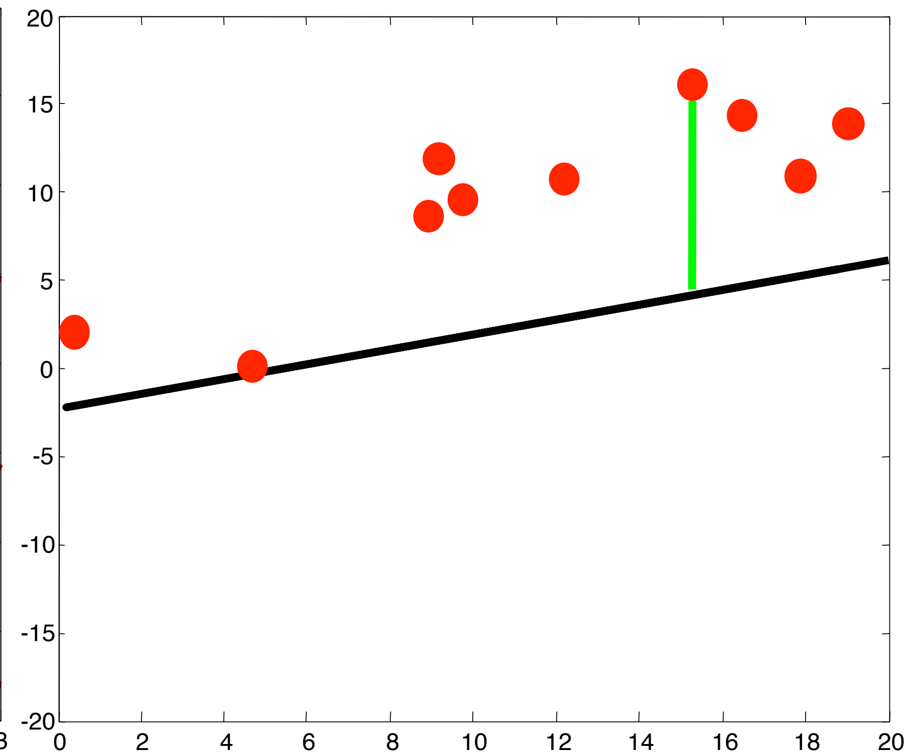


Online gradient descent

- Update based on each datum at a time
  - Find residual and the gradient of its part of the error & update

```
Initialize θ
Do {
 for j=1:m
```
$\theta \leftarrow \theta - \alpha \nabla_\theta J_j(\theta)$
```
} while (not done)
```

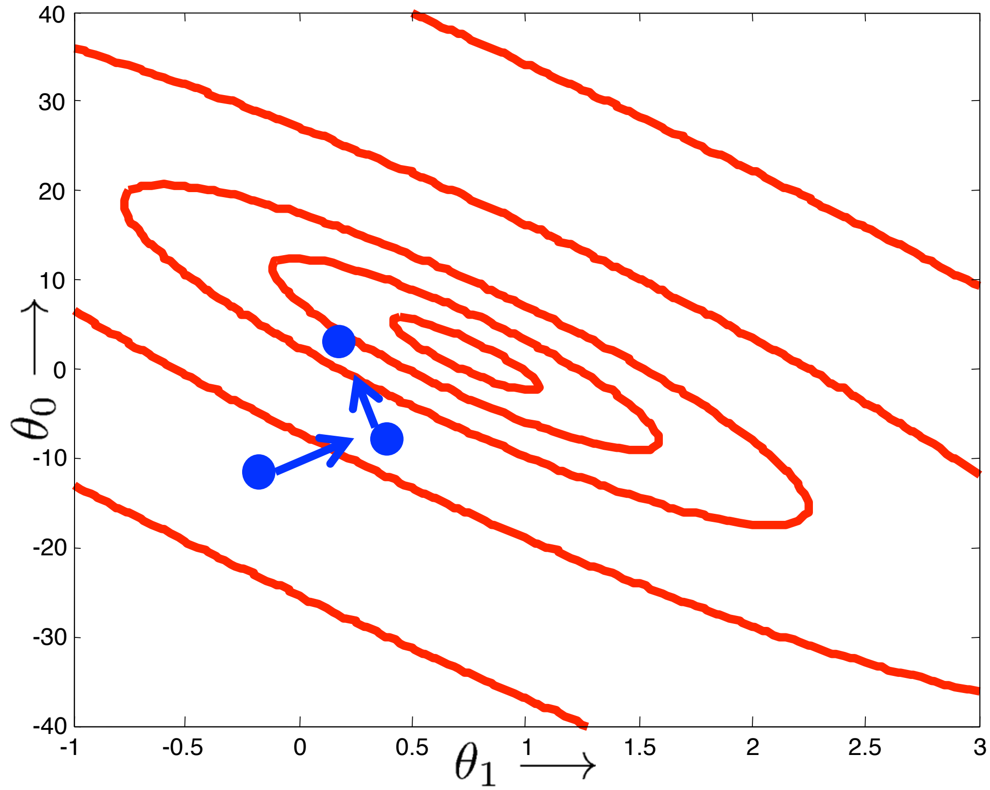


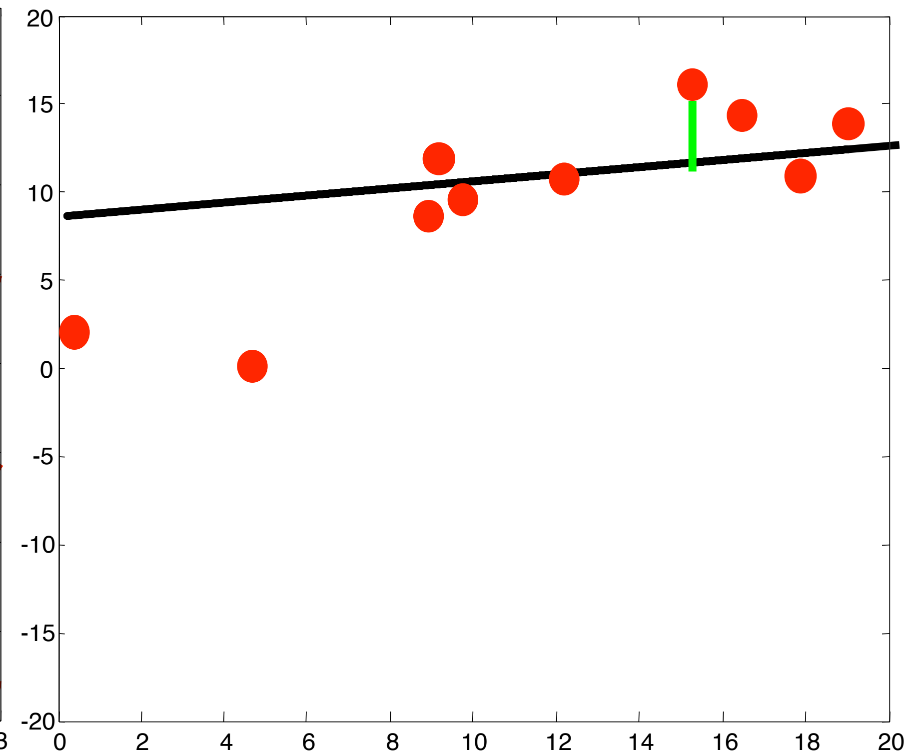


Online gradient descent

- Update based on each datum at a time
  - Find residual and the gradient of its part of the error & update

```
Initialize θ
Do {
 for j=1:m
   θ ← θ - α ∇_θ J_j(θ)
} while (not done)
```

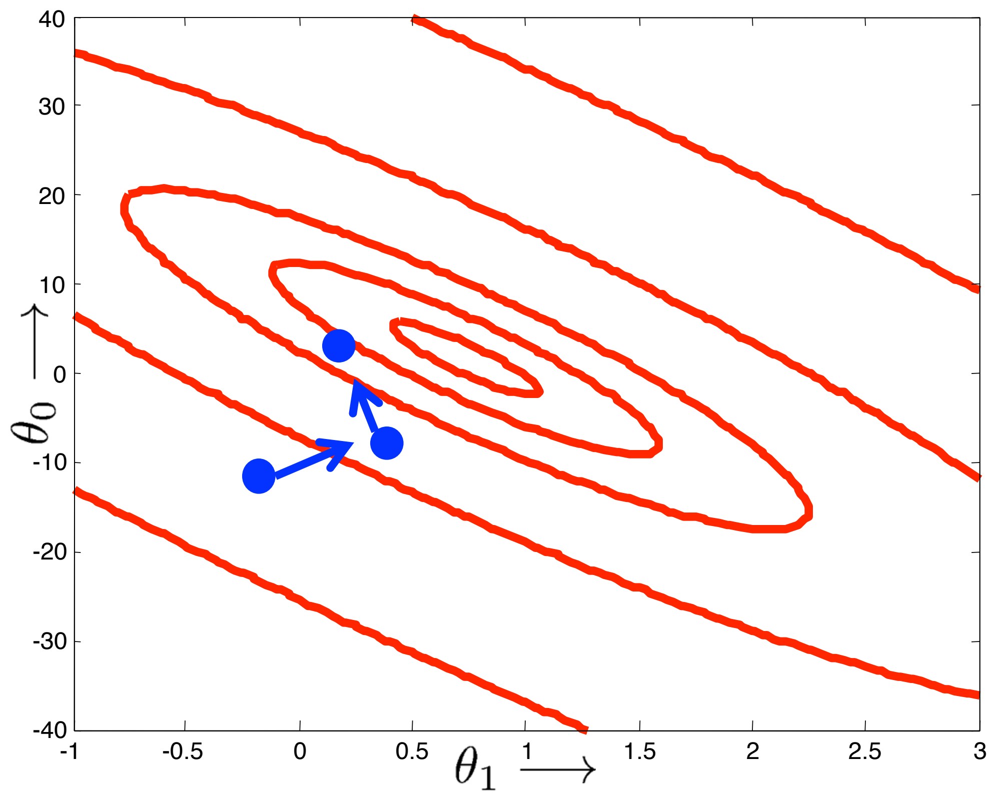


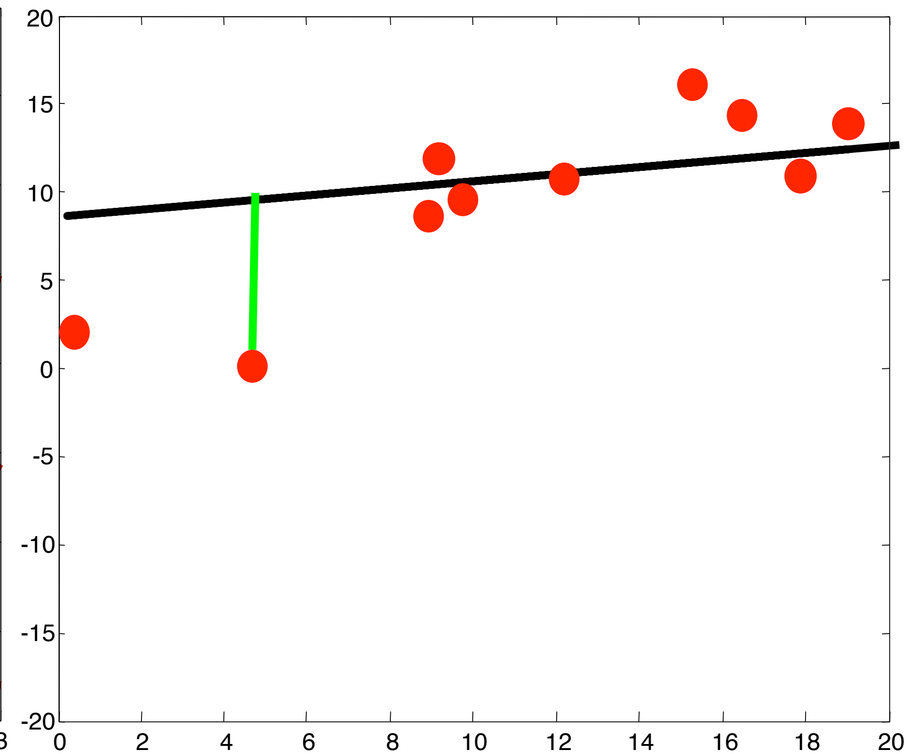


Online gradient descent

- Update based on each datum at a time
  - Find residual and the gradient of its part of the error & update

```
Initialize θ
Do {
 for j=1:m
   θ ← θ - α ∇θ Jj(θ)
} while (not done)
```

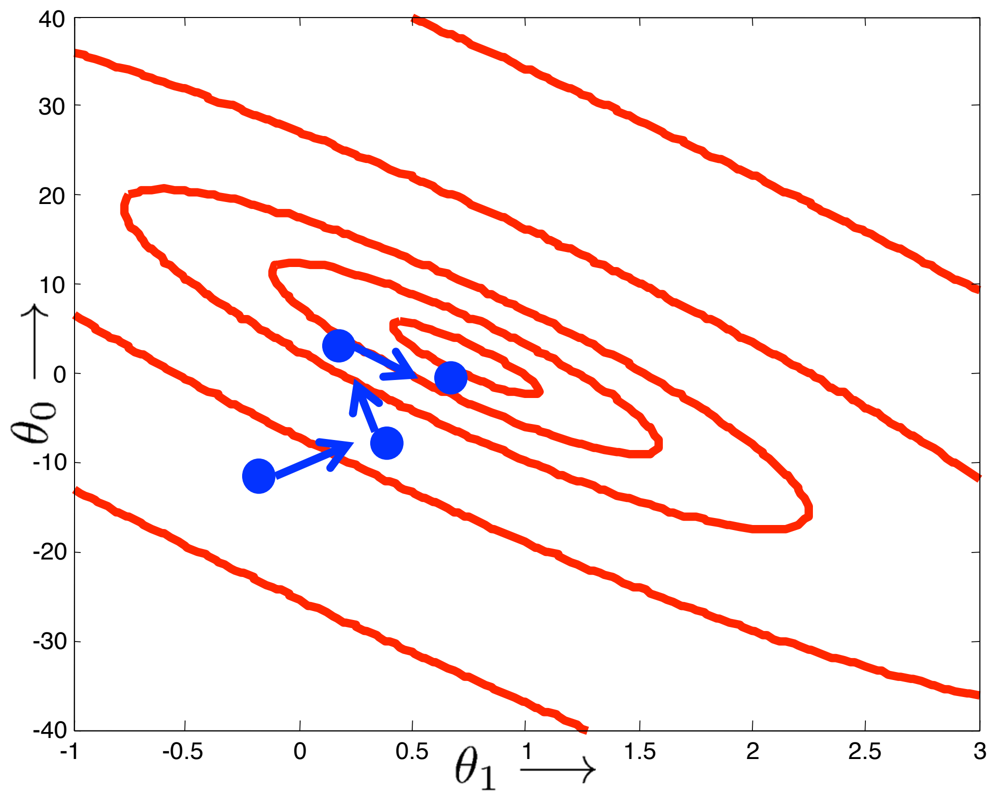


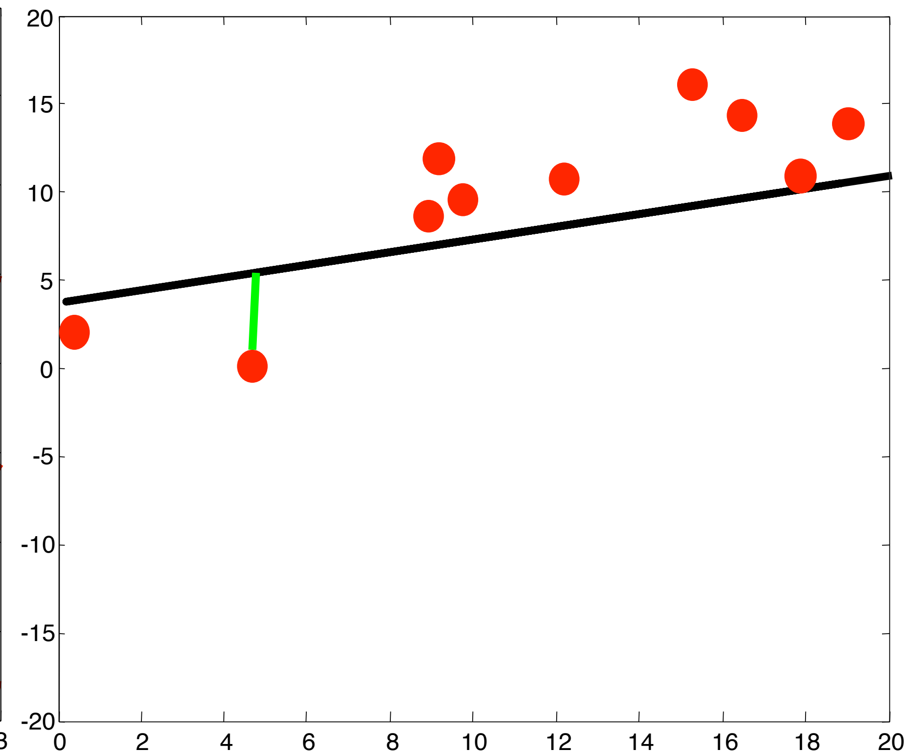


Online gradient descent

- Update based on each datum at a time
  - Find residual and the gradient of its part of the error & update

```
Initialize θ
Do {
  for j=1:m
    θ ← θ - α ∇θ Jj(θ)
} while (not done)
```





Online gradient descent

$$J_j(\underline{\theta}) = (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)^T})^2$$

$$\nabla J_j(\underline{\theta}) = -2(y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)^T}) \cdot [x_0^{(j)} x_1^{(j)} \ldots]$$

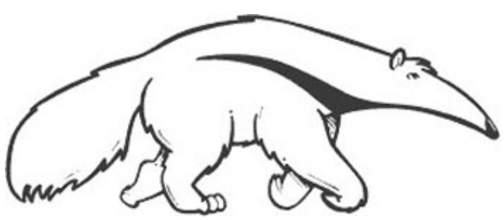
```
Initialize θ
Do {
 for j=1:m
   θ ← θ - α ∇θ Jj(θ)
} while (not converged)
```

- Benefits
  - Lots of data = many more updates per pass
  - Computationally faster
- Drawbacks
  - No longer strictly "descent"
  - Stopping conditions may be harder to evaluate
    (Can use "running estimates" of J(.), etc.)
- Related: mini-batch updates, etc.

Machine Learning and Data Mining

Linear regression: direct minimization

Prof. Alexander Ihler



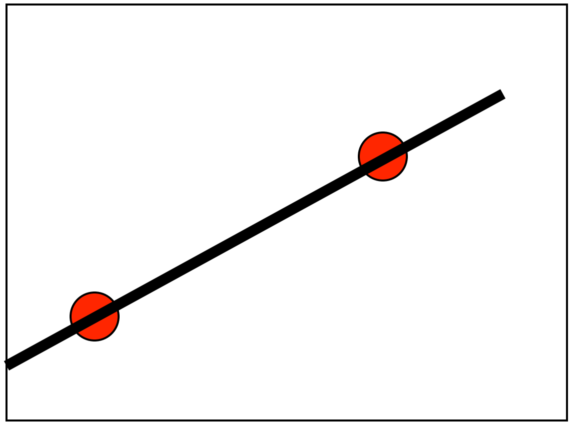
BREN:ICS
INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

MSE Minimum

- Consider a simple problem
  - One feature, two data points
  - Two unknowns: \theta_0, \theta_1
  - Two equations:

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



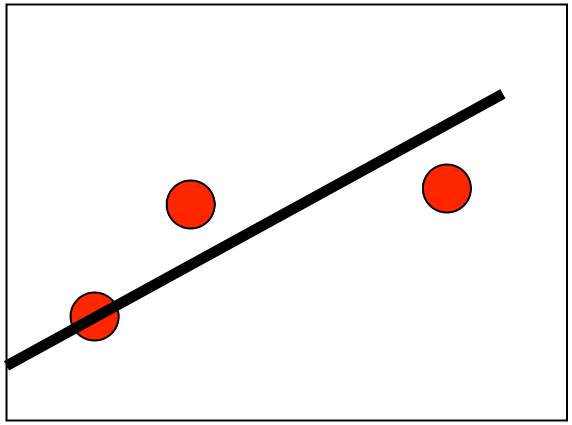
- Can solve this system directly:

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

- 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…

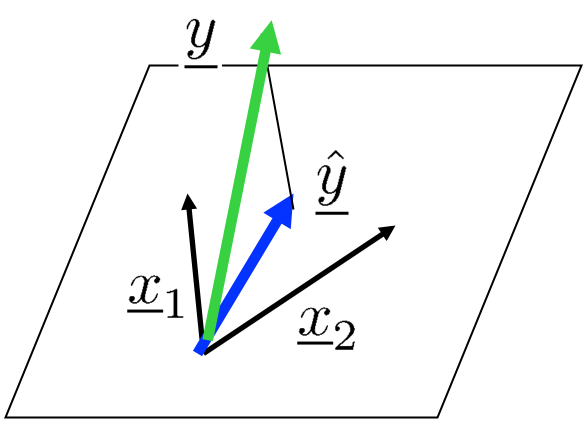
$$\underline{\theta} \quad = \quad \underline{y}^T\,\underline{X}(\underline{X}^T\,\underline{X})^{-1}$$

```
%  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'
```

Normal equations

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

- Interpretation:
  - $(y - \theta X) = (y - yhat)$ is the vector of errors in each example
  - X are the features we have to work with for each example
  - Dot product = 0: orthogonal

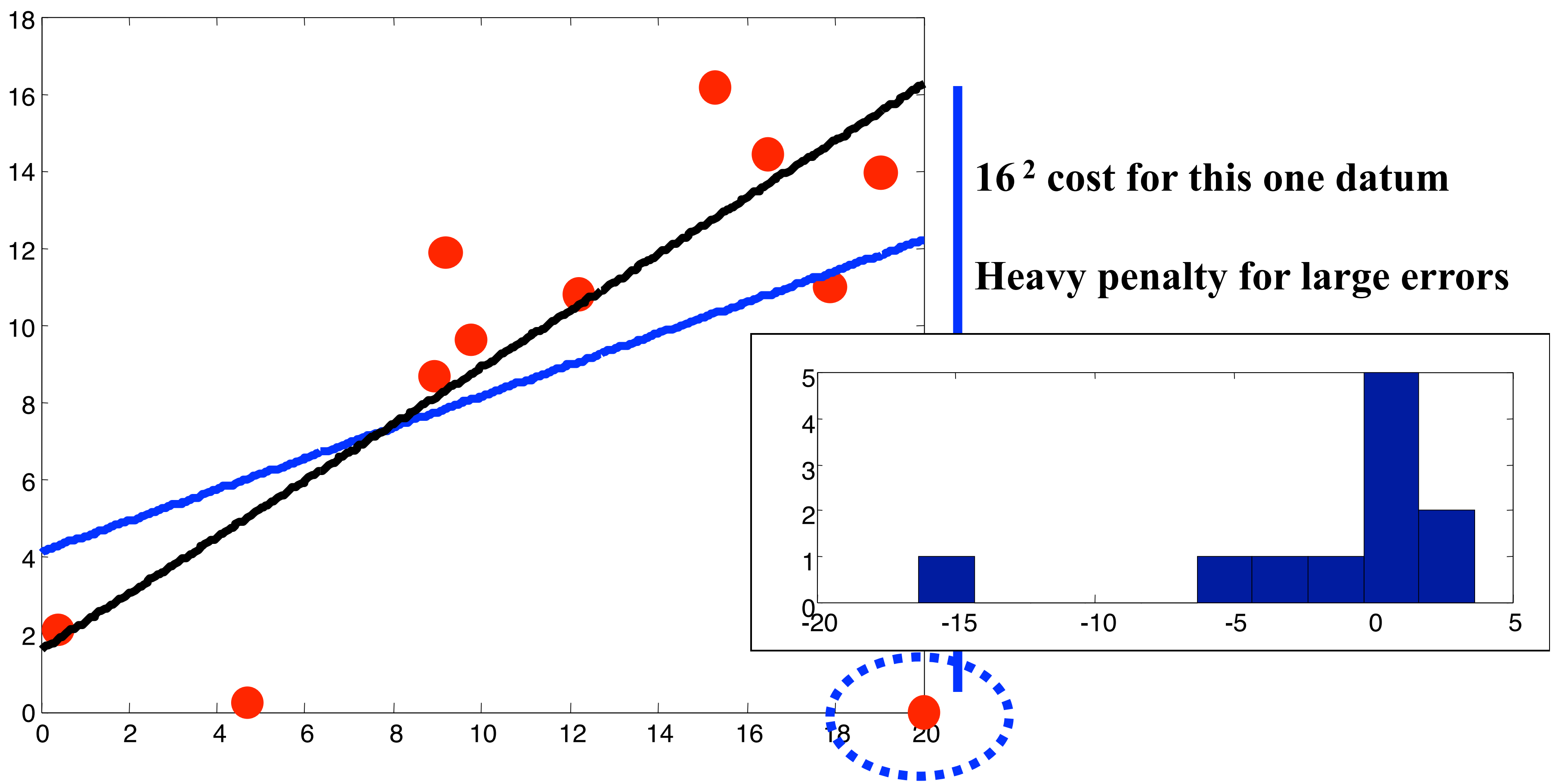


$$\underline{y}^T = [y^{(1)} \ldots y^{(m)}]$$

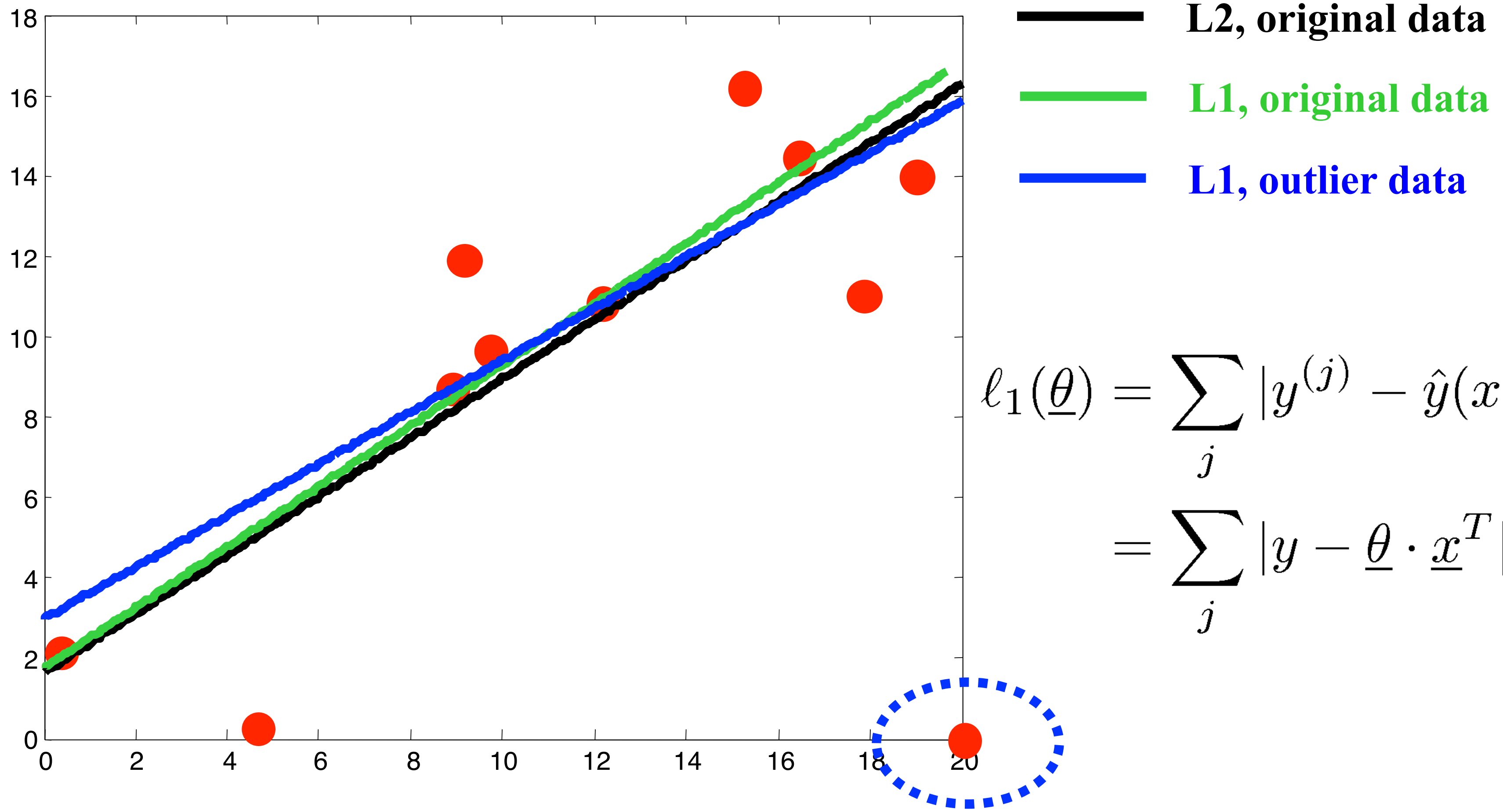
$$\underline{x}_i = [x_i^{(1)} \ldots x_i^{(m)}]$$

Effects of MSE choice

- Sensitivity to outliers



L1 error



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

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

Cost functions for regression

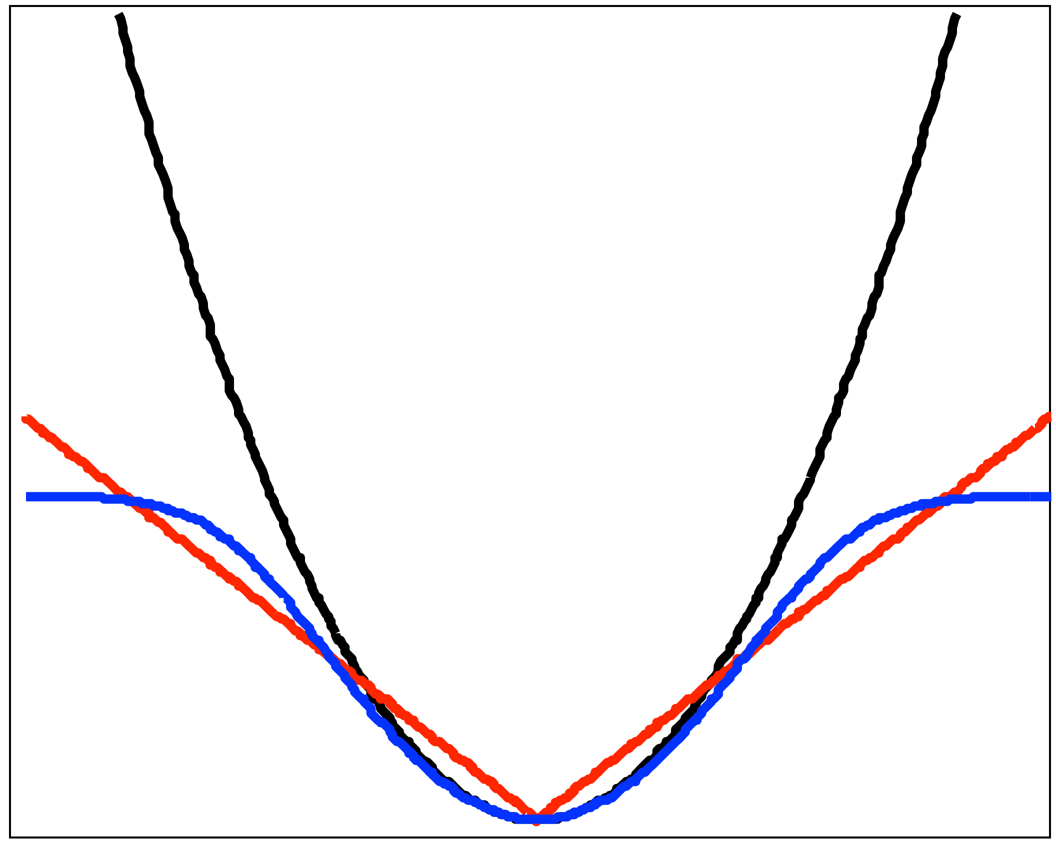
$\ell_2 \;:\; (y - \hat{y})^2$ **(MSE)**

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

Something else entirely…

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

(???)



$\leftarrow (y - \hat{y}) \rightarrow$

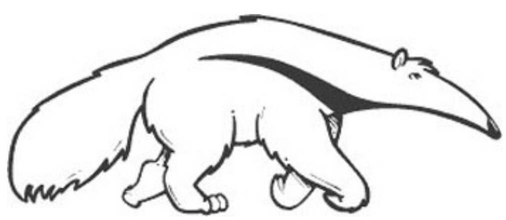
"Arbitrary" functions can't be solved in closed form…

- use gradient descent

Machine Learning and Data Mining

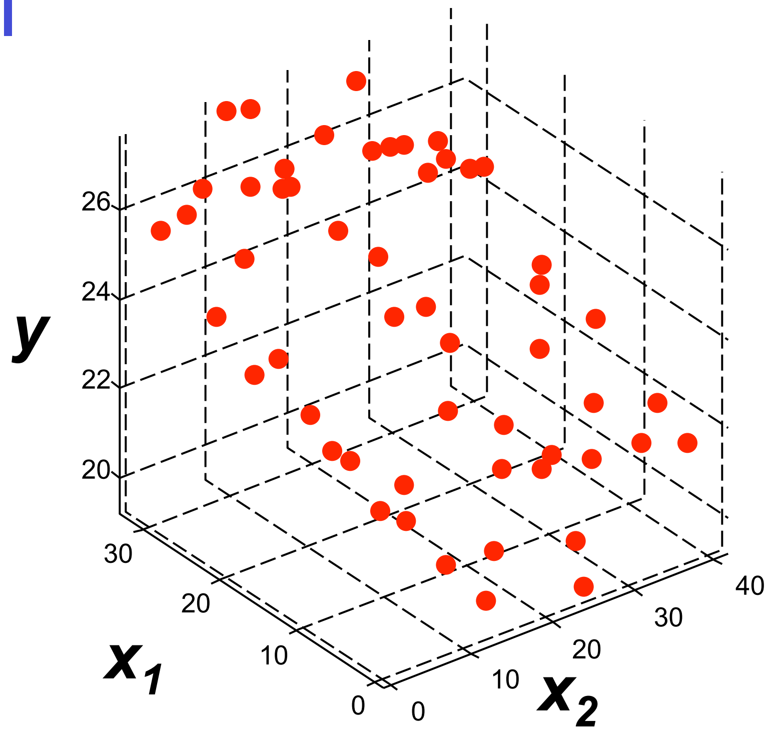
Linear regression: nonlinear features

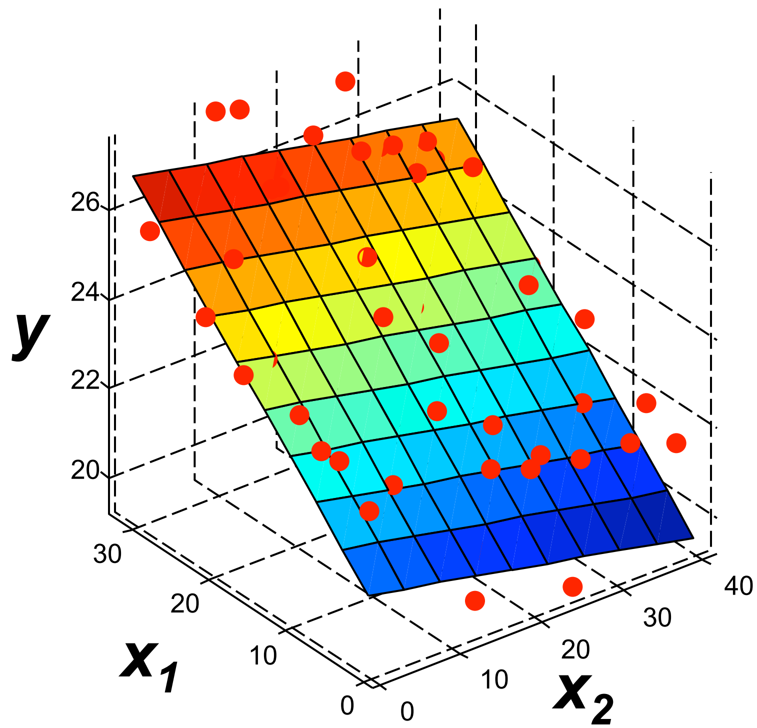
Prof. Alexander Ihler



BREN:ICS
INFORMATION AND COMPUTER SCIENCES
UNIVERSITY of CALIFORNIA IRVINE

More dimensions?





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

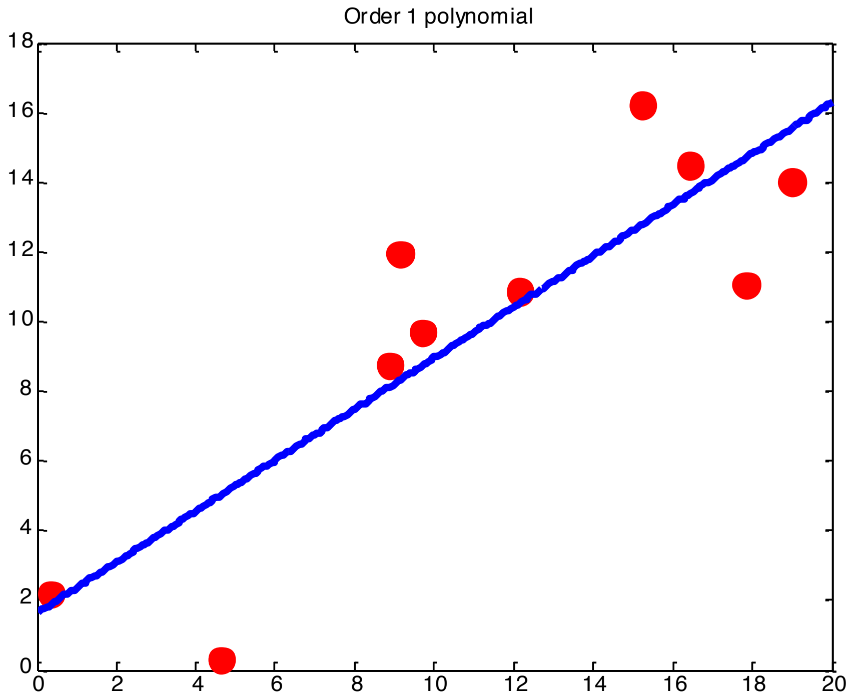
$$\hat{y}(x) = \underline{\theta} \cdot \underline{x}^T$$

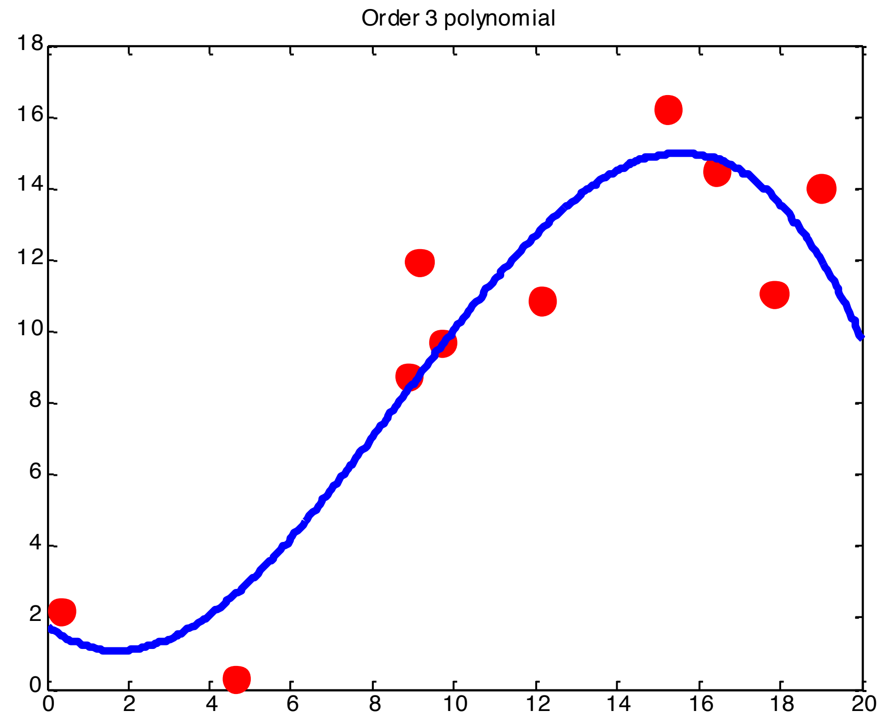
$$\underline{\theta} = [\theta_0 \;\; \theta_1 \;\; \theta_2]$$

$$\underline{x} = [1 \;\; x_1 \;\; x_2]$$

Nonlinear functions

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





Nonlinear functions

- Single feature x, predict target y:

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

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

$\Downarrow$ Add features:

$\Downarrow$

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

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

Linear regression in new features

- The new predictor is a linear regression
  - Now *nonlinear* in x, but *linear* in the parameters θ

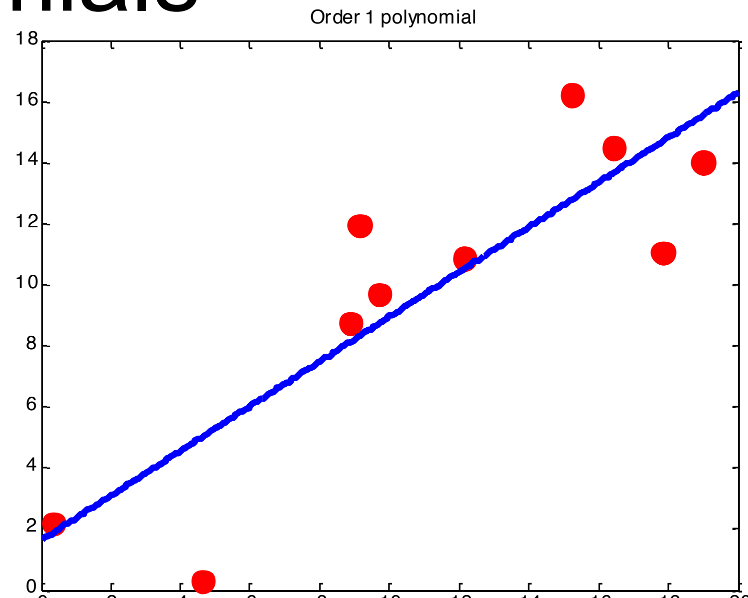
- Sometimes useful to think of "feature transform"

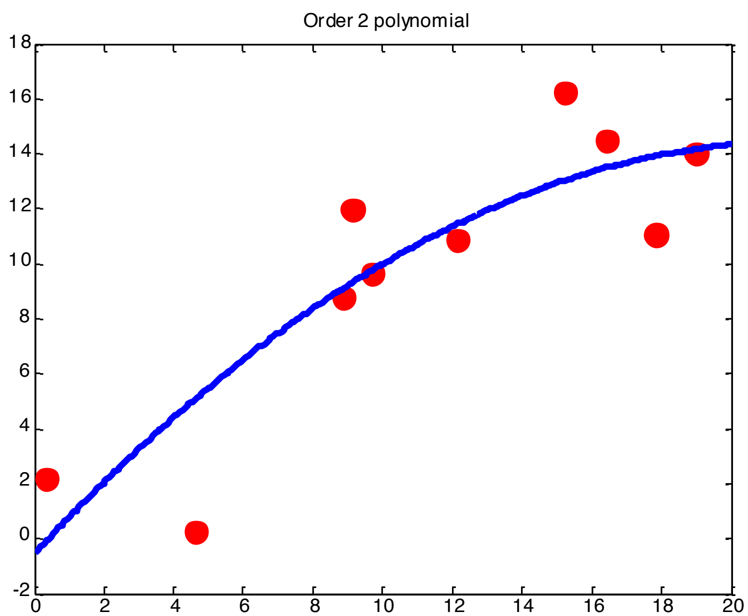
$$\Phi(x) = \left[1\,,\, x\,,\, x^2\,,\, x^3\,,\, \ldots\right]$$

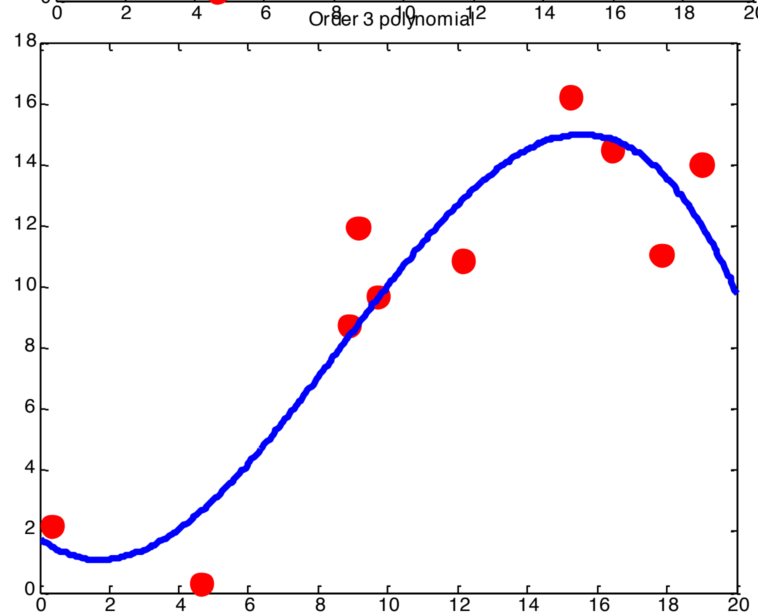
$$\hat{y}(x) = \underline{\theta} \cdot \Phi(x)$$

Higher-order polynomials

- Fit in the same way
- More “features”







Features

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

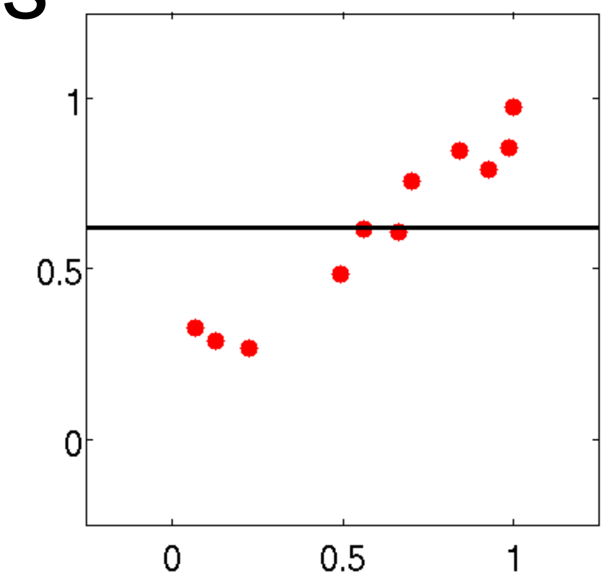
- Other information about the problem
  - Sq. footage, location, age, …
- Polynomial functions
  - Features $[1, x, x^2, x^3, \ldots]$
- Other functions
  - $1/x$, $\mathrm{sqrt}(x)$, $x_1 * x_2$, …

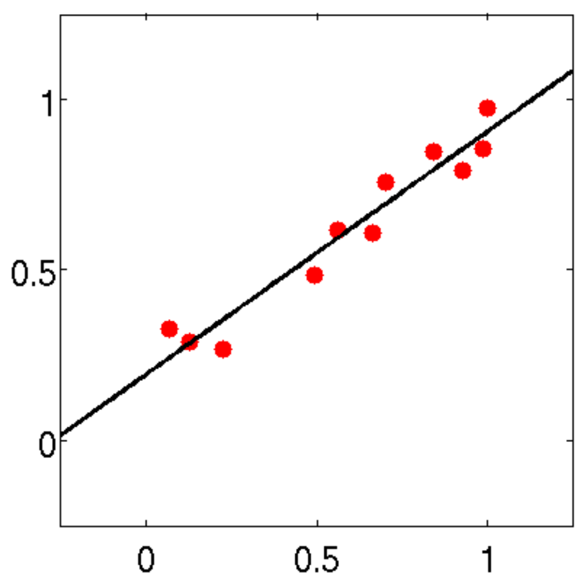
- "Linear regression" = linear in the parameters
  - Features we can make as complex as we want!

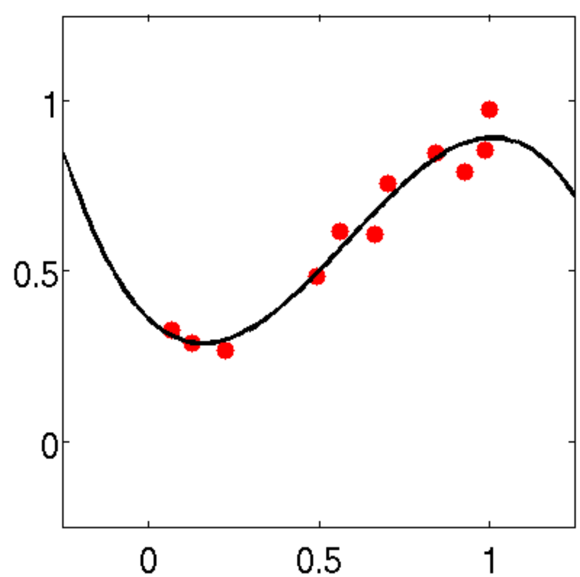
Higher-order polynomials

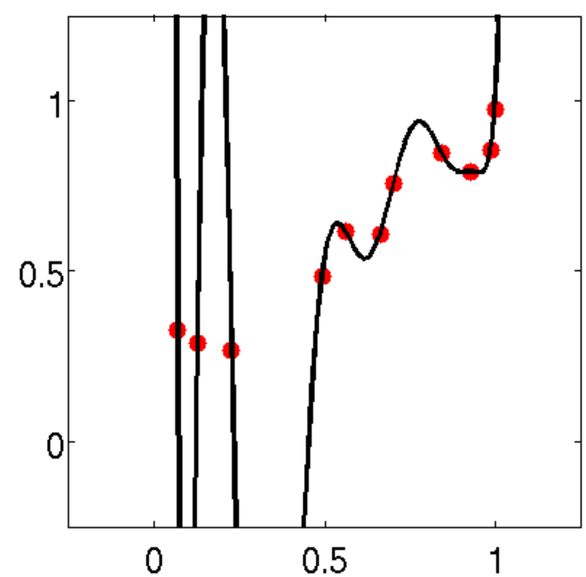
- Are more features better?

- "Nested" hypotheses
  - 2nd order more general than 1st,
  - 3rd order " " than 2nd, …
- Fits the observed data better



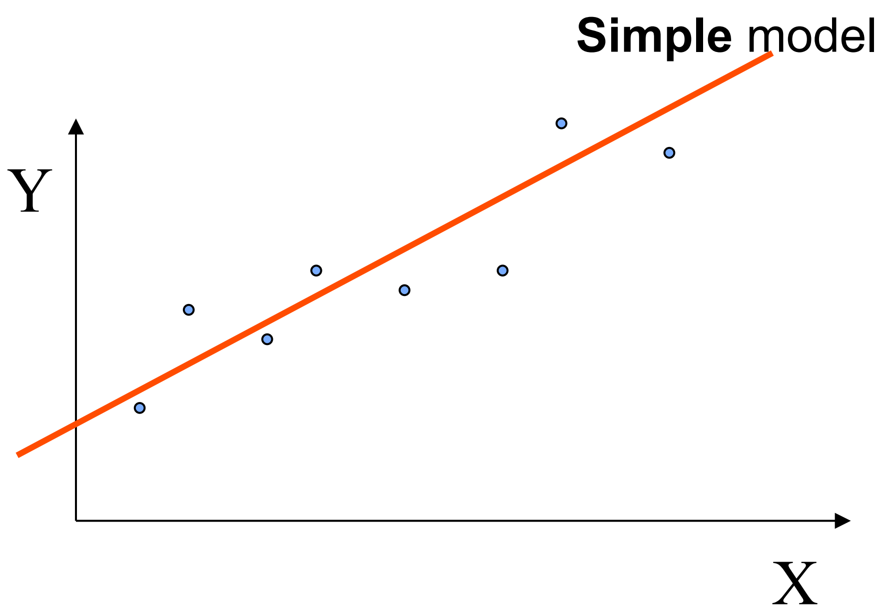


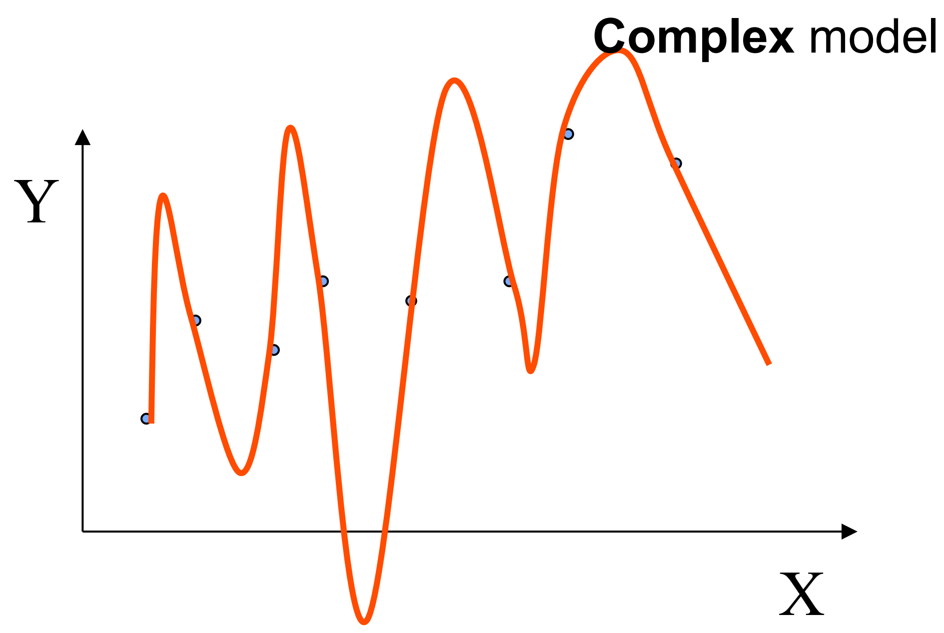




Overfitting and complexity

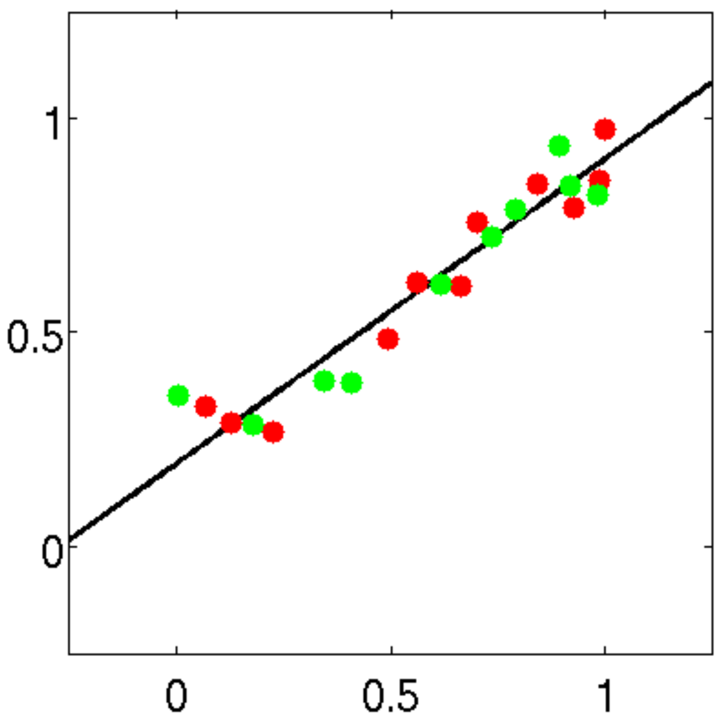
- More complex models will always fit the training data better
- But they may "overfit" the training data, learning complex relationships that are not really present

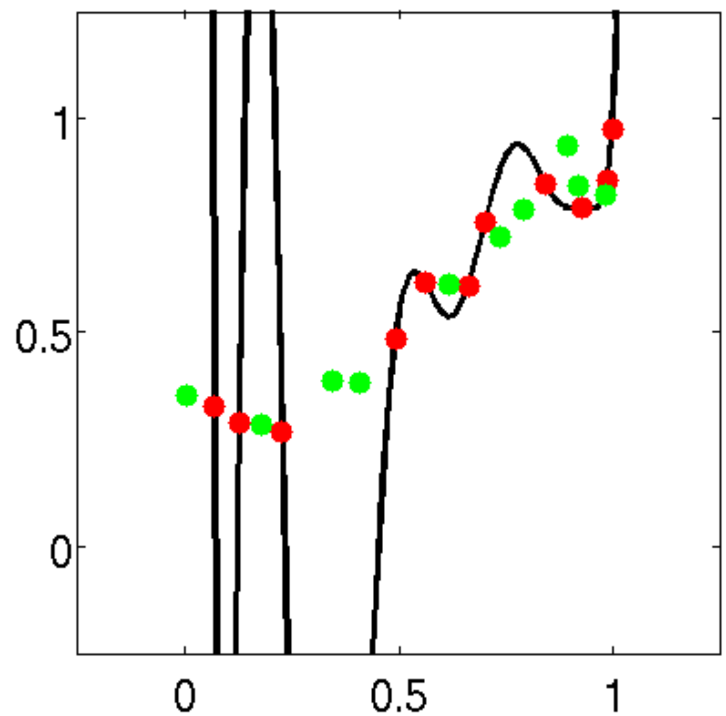




Test data

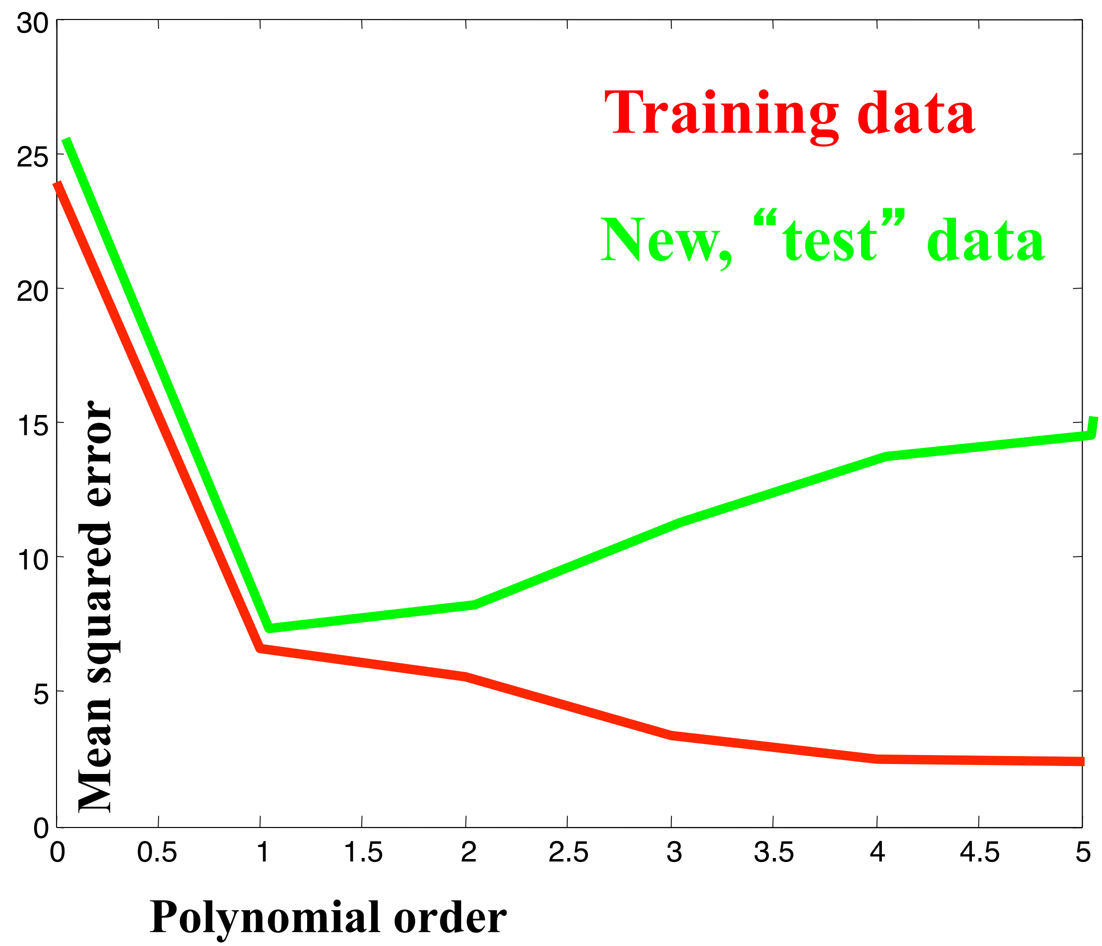
- After training the model
- Go out and get more data from the world
  - New observations (x,y)
- How well does our model perform?





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

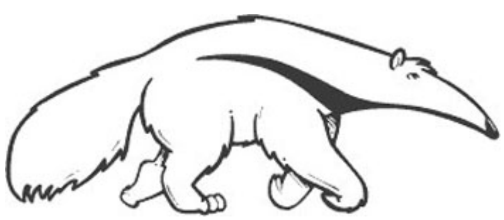


+

Machine Learning and Data Mining

Linear regression: bias and variance

Prof. Alexander Ihler



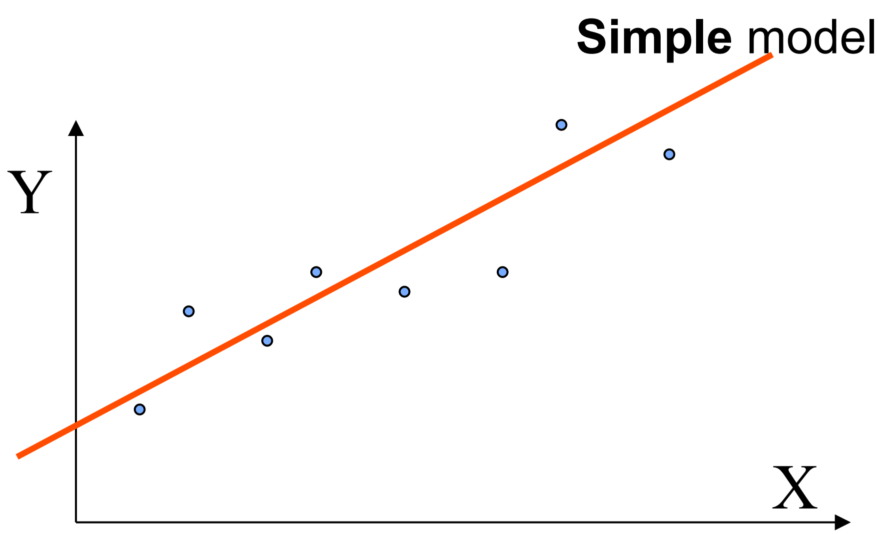
BREN:ICS
INFORMATION AND COMPUTER SCIENCES

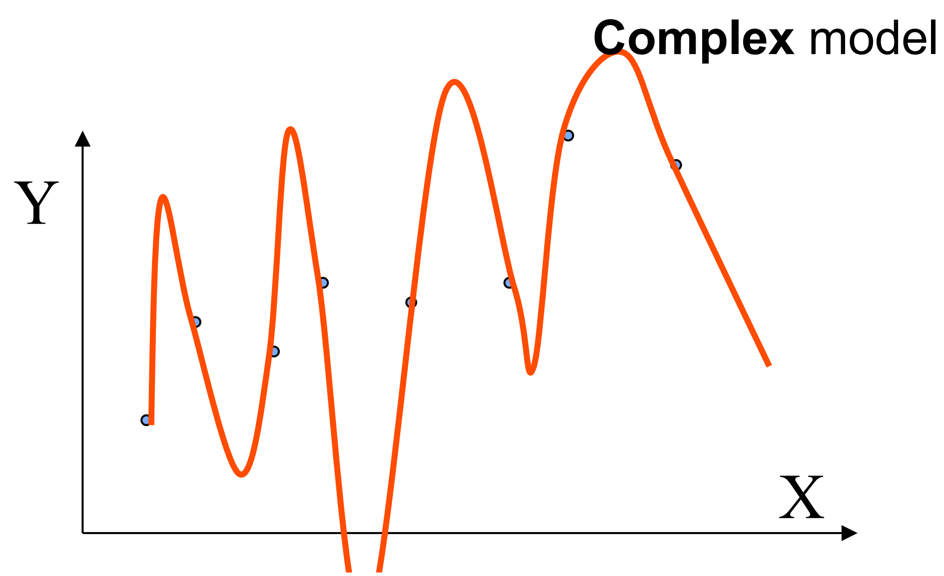
UNIVERSITY of CALIFORNIA IRVINE

Inductive bias

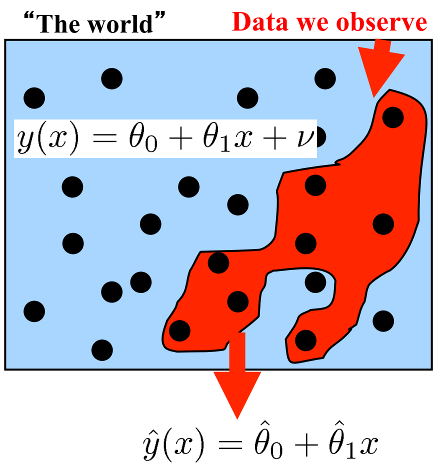
- The assumptions needed to predict examples we haven’t seen
- Makes us “prefer” one model over another
- Polynomial functions; smooth functions; etc

- Some bias is necessary for learning!

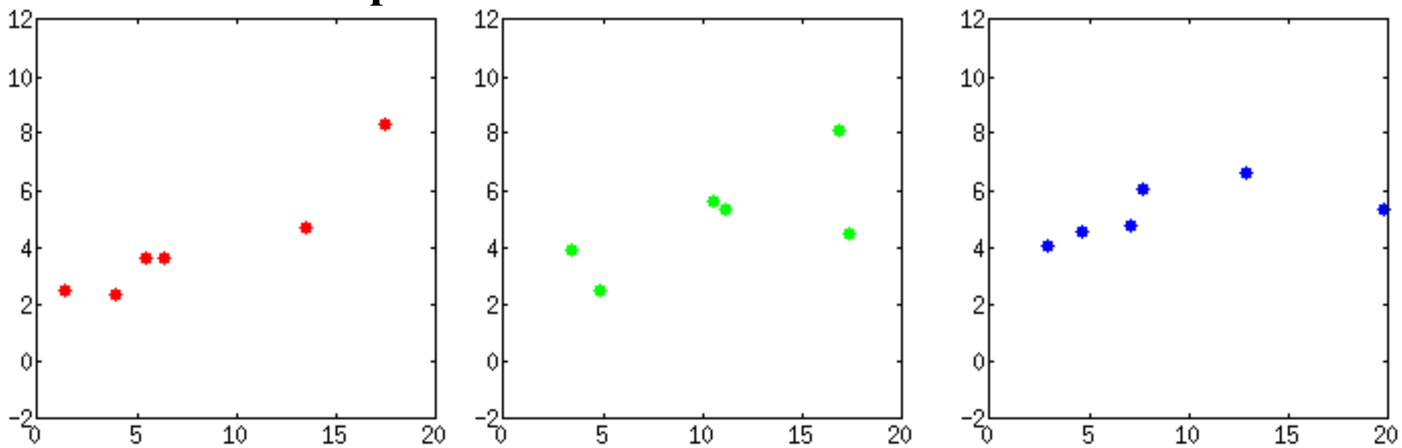




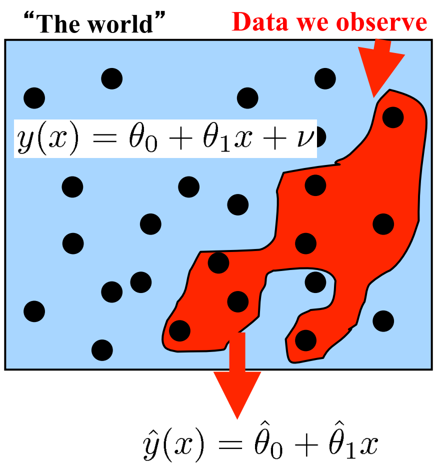
Bias & variance



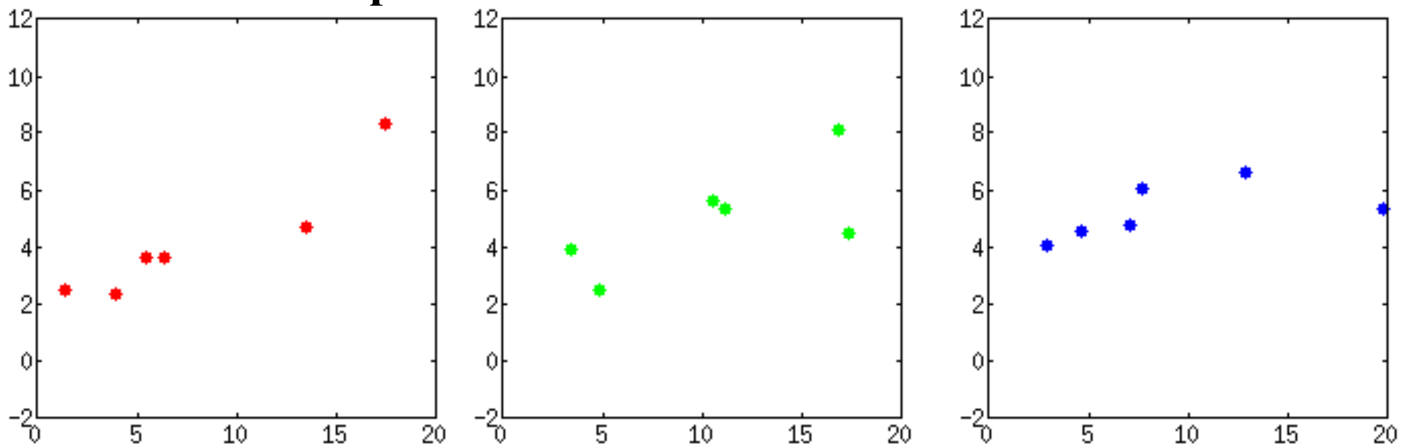
Three different possible data sets:



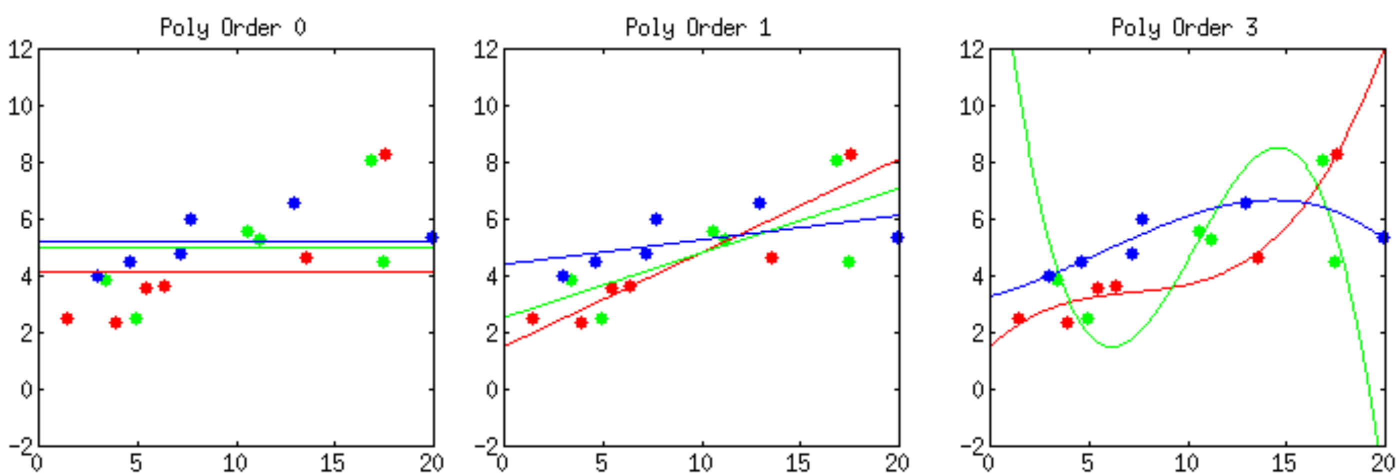
Bias & variance



Three different possible data sets:



Each would give different predictors for any polynomial degree:



Detecting overfitting

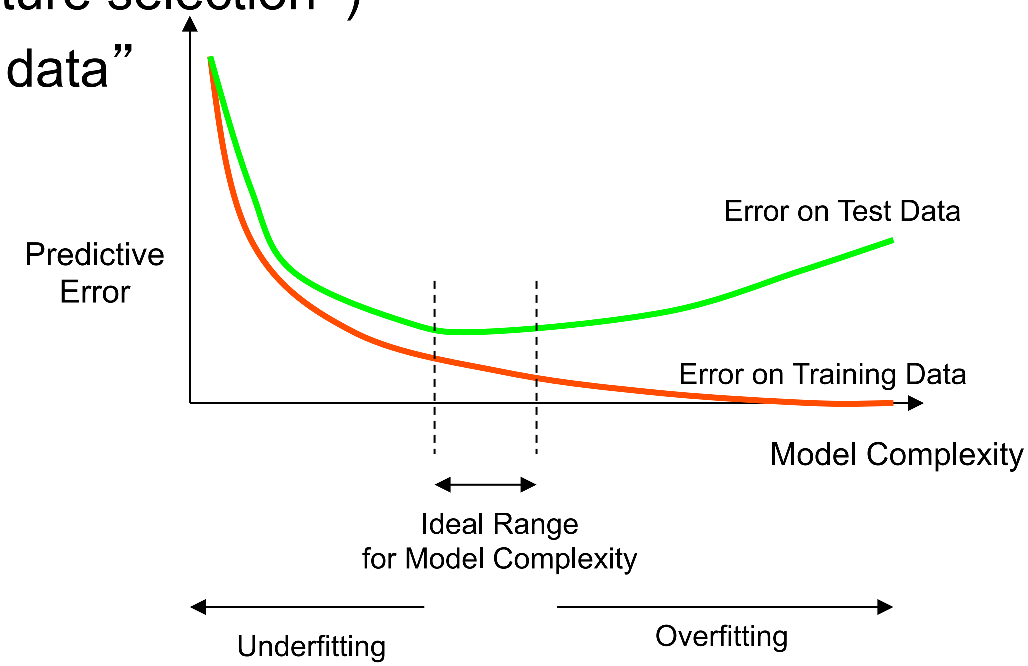
- Overfitting effect
  - Do better on training data than on future data
  - Need to choose the “right” complexity

- One solution: “Hold-out” data
- Separate our data into two sets
  - Training
  - Test
- Learn only on training data
- Use test data to estimate generalization quality
  - Model selection

- All good competitions use this formulation
  - Often multiple splits: one by judges, then another by you

What to do about under/overfitting?

- Ways to increase complexity?
  - Add features, parameters
  - We'll see more…
- Ways to decrease complexity?
  - Remove features ("feature selection")
  - "Fail to fully memorize data"
    - Partial training
    - Regularization

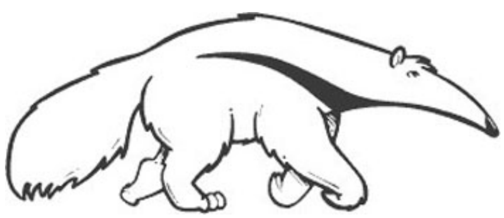


+

Machine Learning and Data Mining

Linear regression: regularization

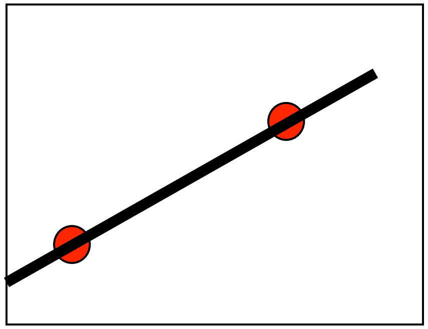
Prof. Alexander Ihler

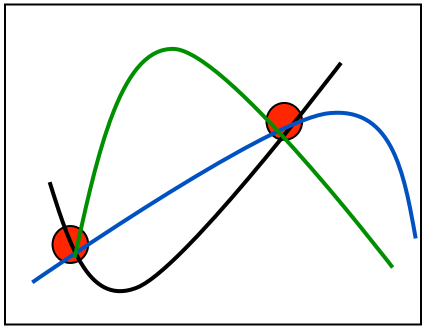


BREN:ICS
INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

Linear regression





- Linear model, two data
- Quadratic model, two data?
  - Infinitely many settings with zero error
  - How to choose among them?
- Higher order coefficents = 0?
  - Uses knowledge of where features came from…
- Could choose e.g. minimum magnitude:

$$\min \underline{\theta\theta}^T \quad s.t. \quad J(\underline{\theta}) = 0$$

- A type of *bias*: tells us which models to prefer

Regularization

- Can modify our cost function J to add "preference" for certain parameter values

$$J(\underline{\theta}) = \frac{1}{2}(\underline{y} - \underline{\theta}\,\underline{X}^T) \cdot (\underline{y} - \underline{\theta}\,\underline{X}^T)^T + \alpha\,\theta\theta^T$$

L_2 penalty: "Ridge regression"

- New solution (derive the same way)

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

  – Problem is now well-posed for any degree

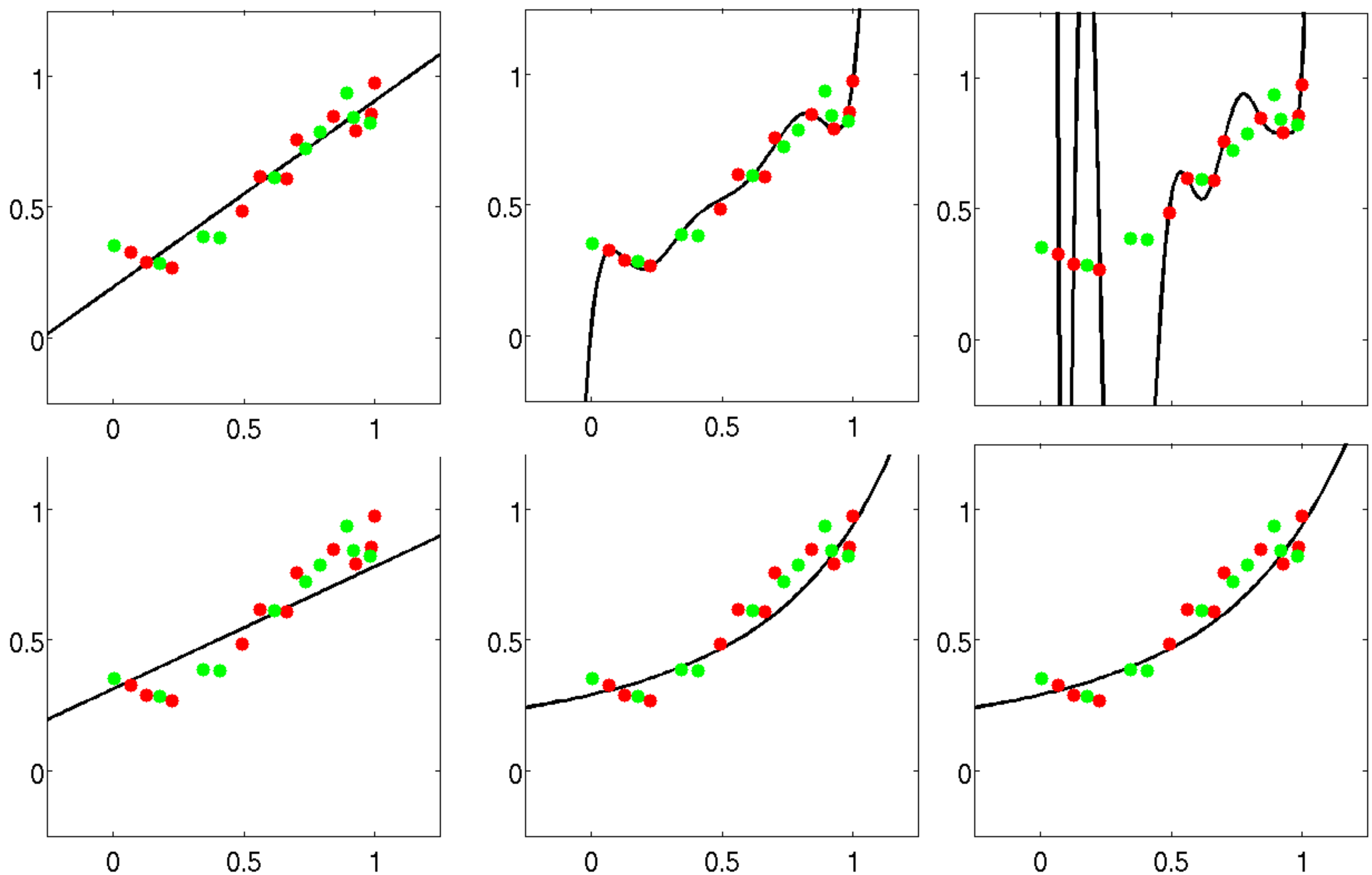
- Notes:
  – "Shrinks" the parameters toward zero
  – Alpha large: we prefer small theta to small MSE
  – Regularization term is independent of the data: paying more attention reduces our variance

Regularization

- Compare between unreg. & reg. results

Alpha =0 (Unreg)

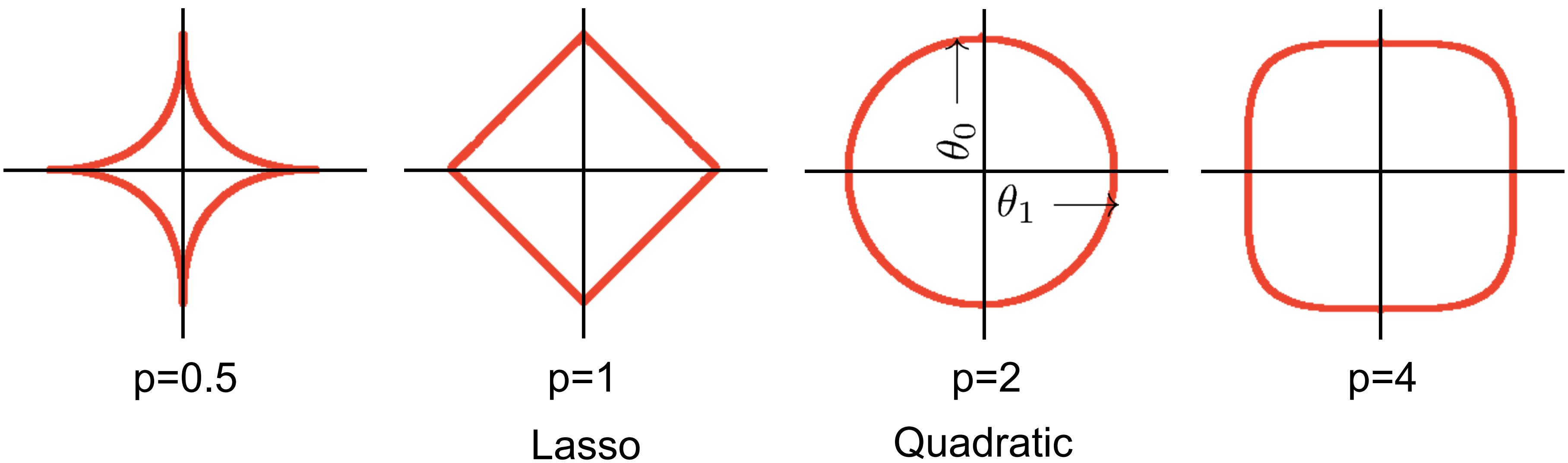
Alpha =1



Different regularization functions

- More generally, for the L_p regularizer:

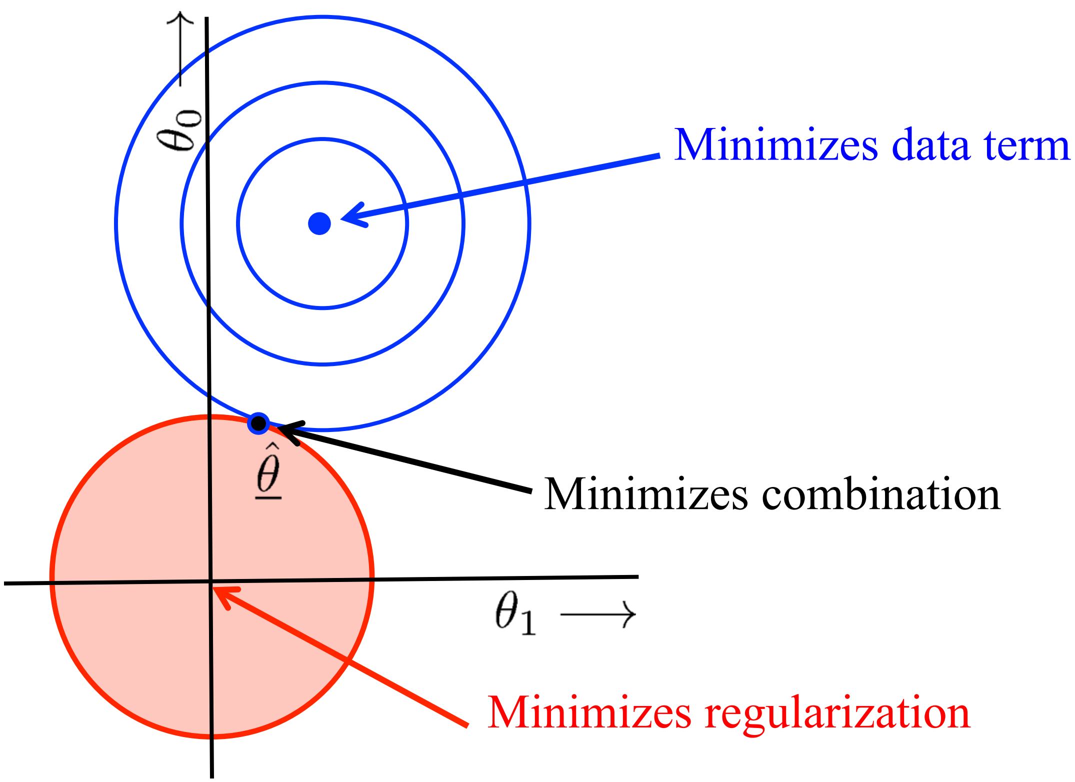
$$J(\underline{\theta}) = \frac{1}{m} \sum_j (y^{(j)} - \underline{\theta} \cdot \underline{x}^{(j)T})^2 + \alpha \left(\sum_i |\theta_i|^p \right)^{\frac{1}{p}}$$



L_0 = limit as p $\rightarrow$ 0 : "number of nonzero weights", a natural notion of complexity

Regularization: L1 vs L2

- Estimate balances data term & regularization term



Regularization: L1 vs L2

- Estimate balances data term & regularization term
- Lasso tends to generate sparser solutions than a quadratic regularizer.