Regression

Le Song

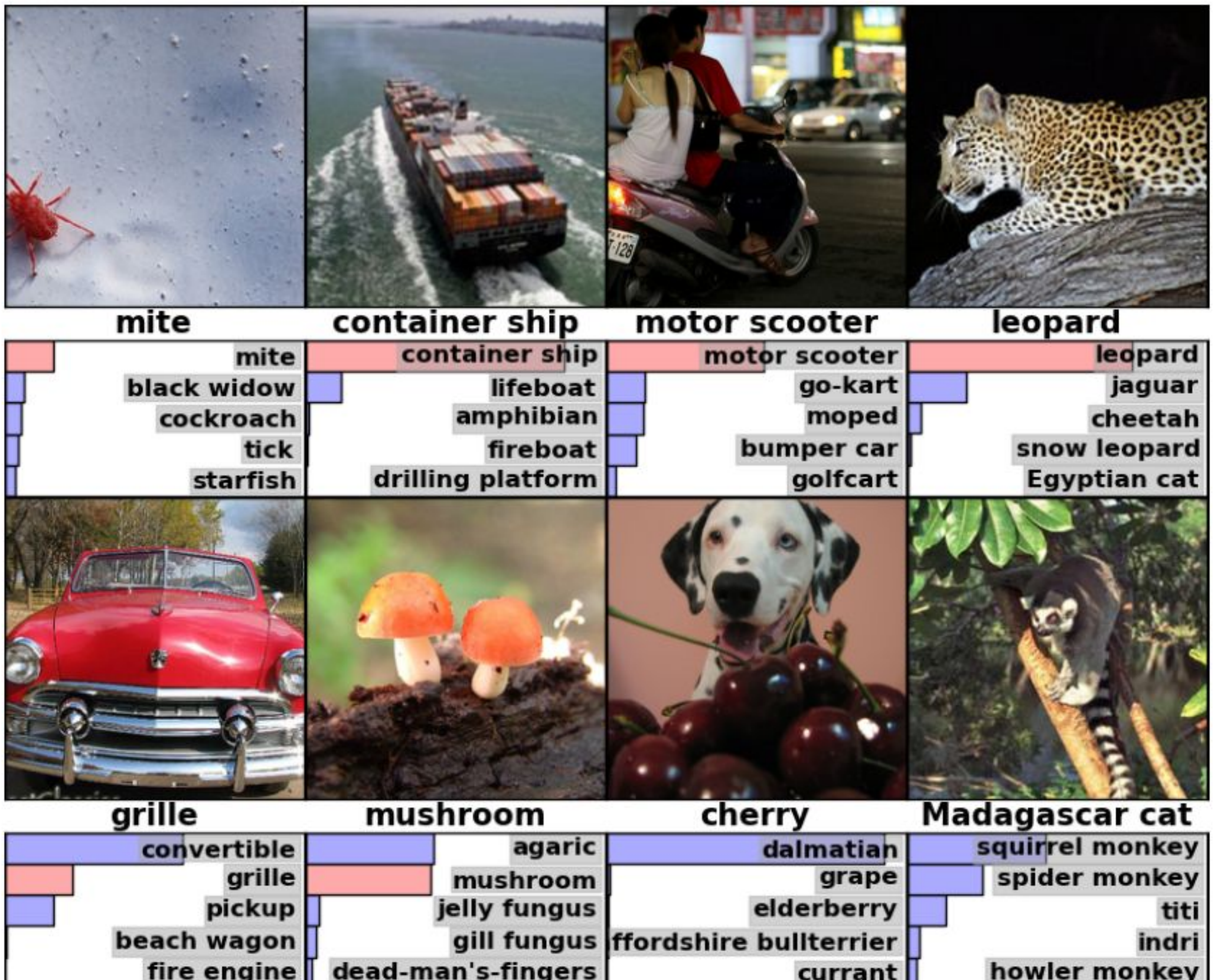
Machine Learning

CSE/ISYE 6740, Fall 2019

ImageNet

Image classification with 1.3M color images and 1000 classes

Need large scale nonparametric methods

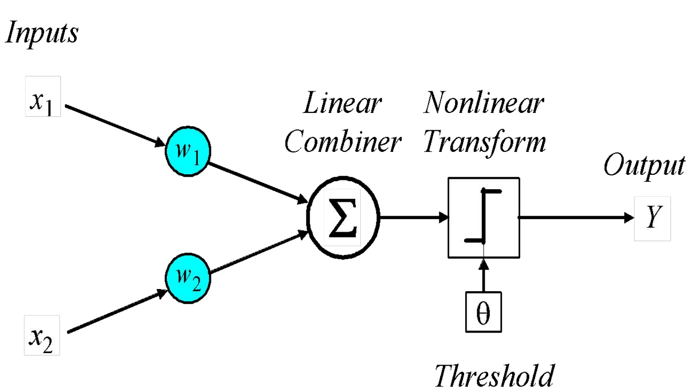


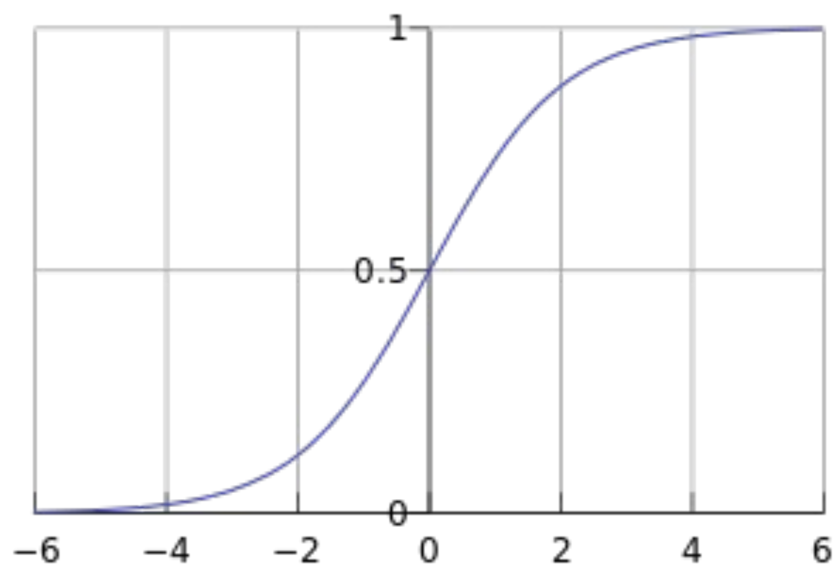
Logistic regression is a neuron

- Assume that the posterior distribution $p(y=1|x)$ take a particular form

$$p(y=1|x,w)=\frac{1}{1+\exp(-w^\top x)}$$

- Logistic function (or sigmoid function) $\sigma(u)=\frac{1}{1+\exp(-u)}$
  - $\frac{\partial\sigma(u)}{\partial u}=\sigma(u)\big(1-\sigma(u)\big)$





Maximum likelihood learning

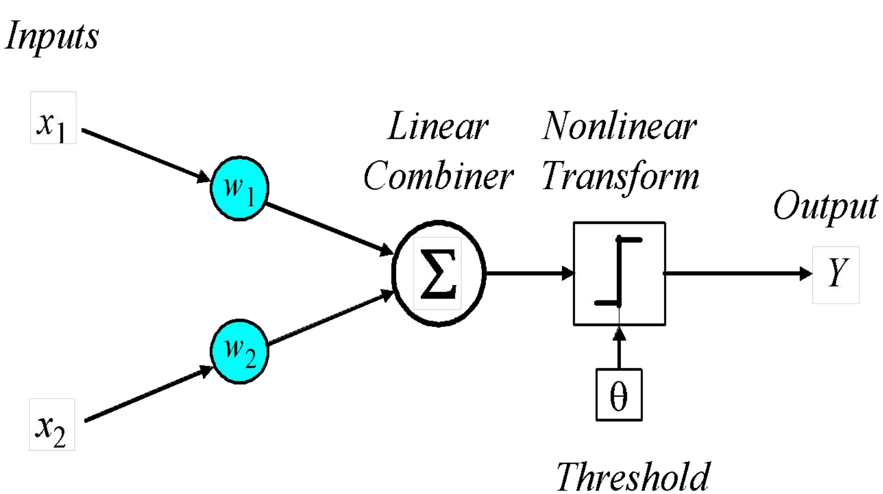
- $$l(w) := \log \prod_{i=1}^{m} P\left(y^i \middle| x^i, w\right)$$
$$= \sum_i (y^i - 1)\, w^\top x^i - \log(1 + \exp\left(-w^\top x^i\right))$$

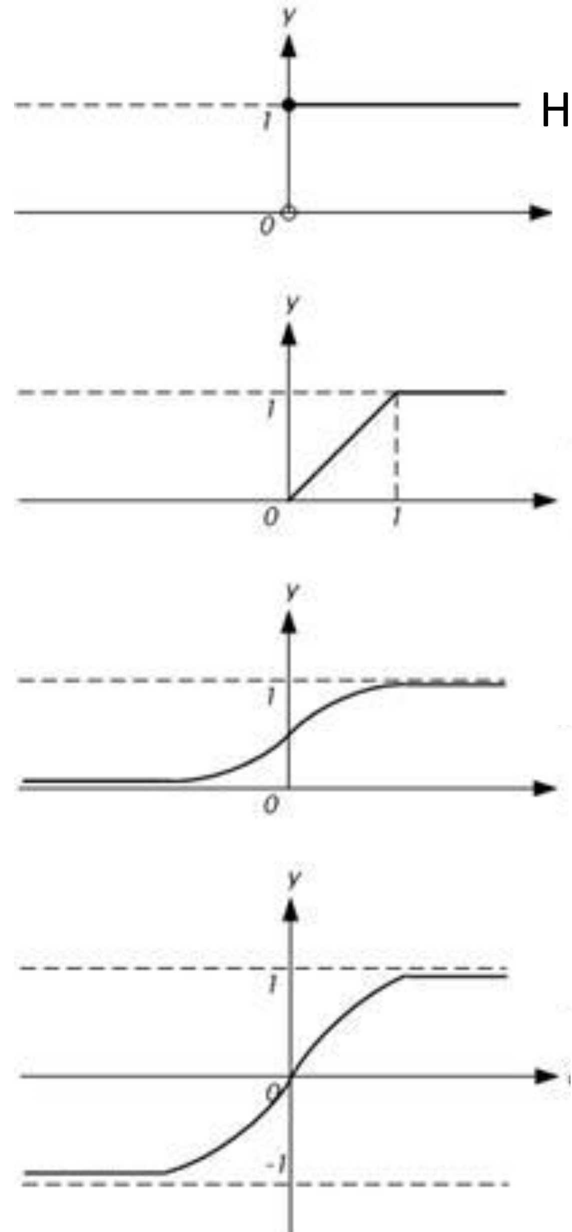
- Gradient
$$\frac{\partial l(w)}{\partial w} = \sum_i (y^i - 1)\, x^i + \frac{\exp\left(-w^\top x^i\right) x^i}{1 + \exp(-w^\top x)}$$

- Setting it to 0 does not lead to closed form solution

Other nonlinear neurons

- Use different nonlinear transformations $Y = f(u)$
- Before that, perform weighted combination of inputs $u = w^{\mathsf{T}} x$





Learning with square loss

- Find w, such that the conditional probability of the labels are close to the actual labels (may be values other than 0 or 1)

$$\min_{w} l(w) := \sum_{i}^{n} \left(y^i - P\left(y = 1 \middle| x^i, w\right)\right)^2 = \sum_{i}^{n} \left(y^i - \sigma\left(w^\top x^i\right)\right)^2$$

- Not a convex objective function

- Use gradient decent to find a local optimum

The gradient of $l(w)$

- $$l(w) := \sum_i^n \left(y^i - P\left(y = 1 \middle| x^i, w\right)\right)^2 = \sum_i^n \left(y^i - \sigma\left(w^\top x^i\right)\right)^2$$

- Let $u^i = w^\top x^i$

- For sigmoid function: $\frac{\partial \sigma(u)}{\partial u} = \sigma(u)\left(1 - \sigma(u)\right)$

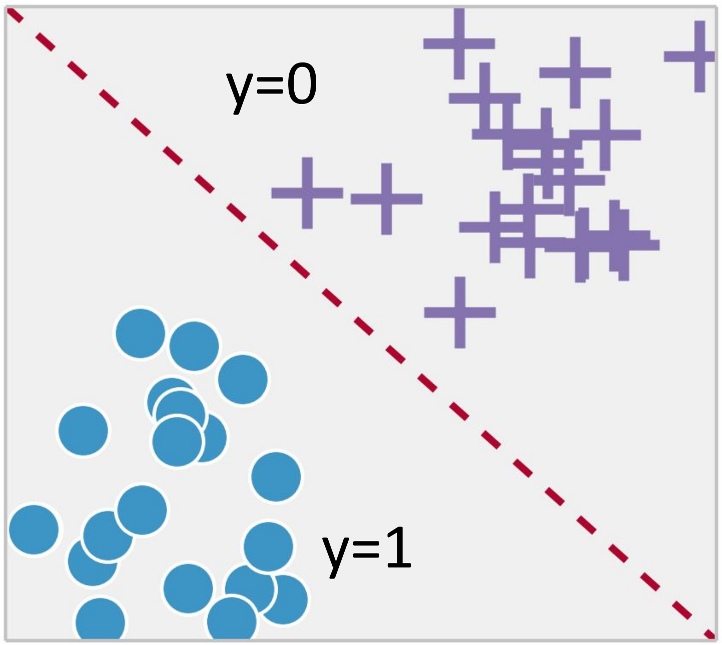
- Gradient

$$\frac{\partial l(w)}{\partial w} = \sum_i 2\left(y^i - \sigma\left(u^i\right)\right) \sigma\left(u^i\right) \left(1 - \sigma\left(u^i\right)\right) x^i$$

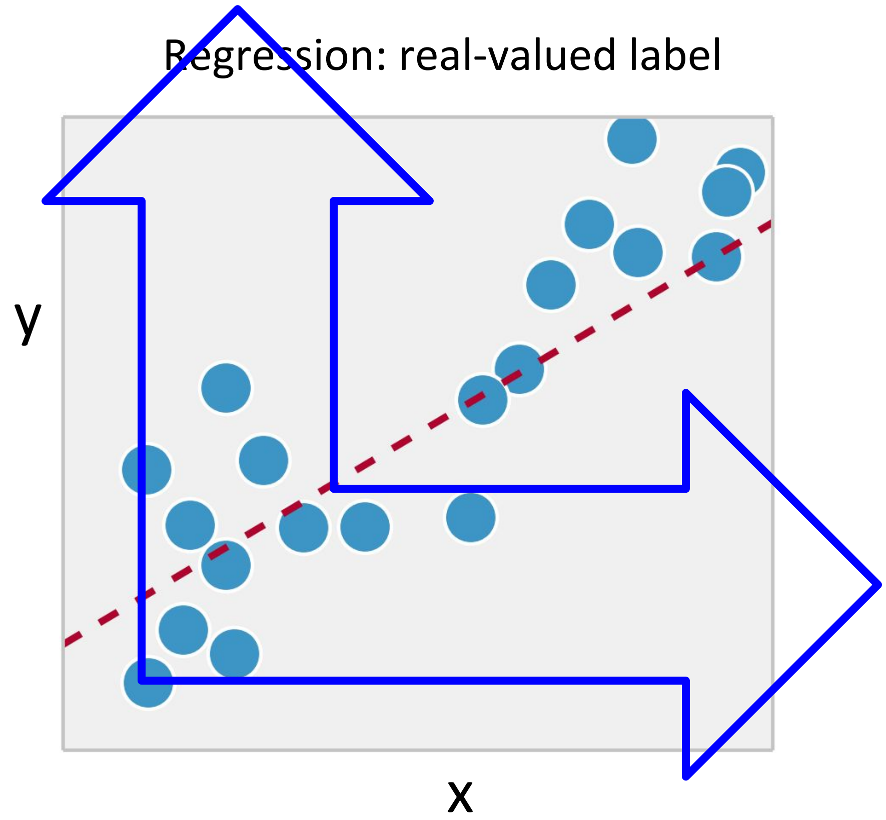
Regression

Classification v.s. Regression

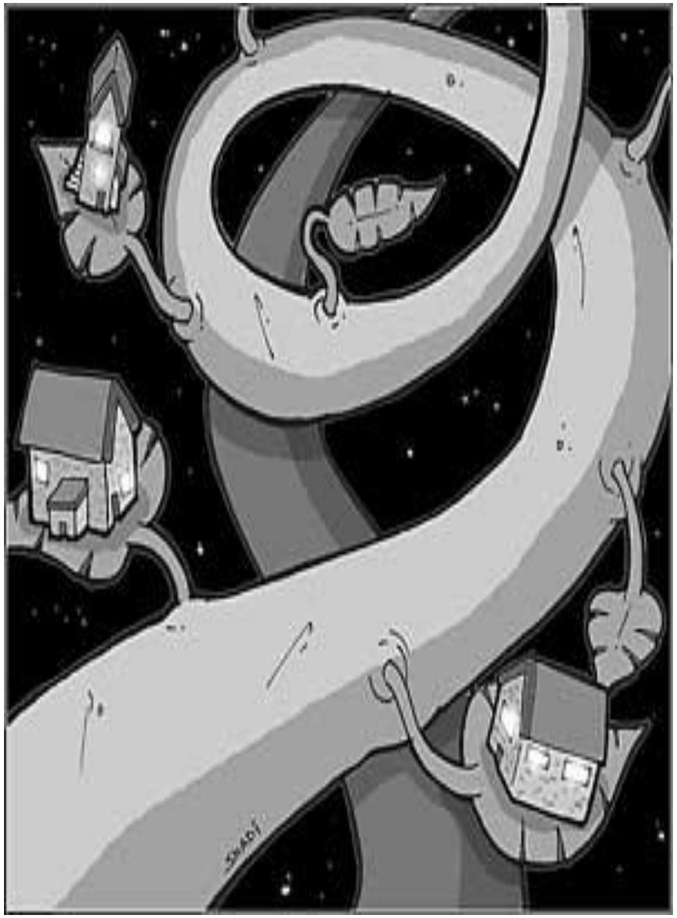
Classification: discrete label



Regression: real-valued label



Machine learning for apartment hunting



- Suppose you are to move to Atlanta
- And you want to find the **most reasonably priced** apartment satisfying your **needs:**

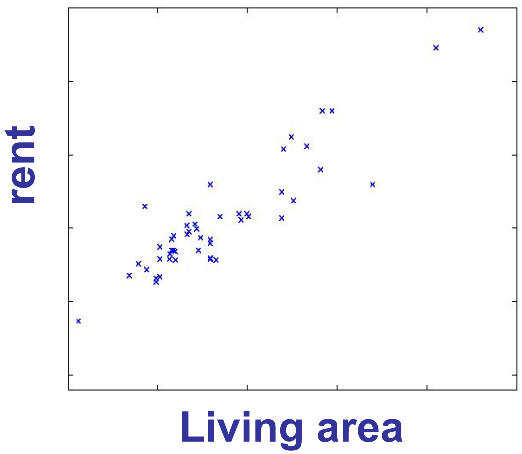
  square-ft., # of bedroom, distance to campus …

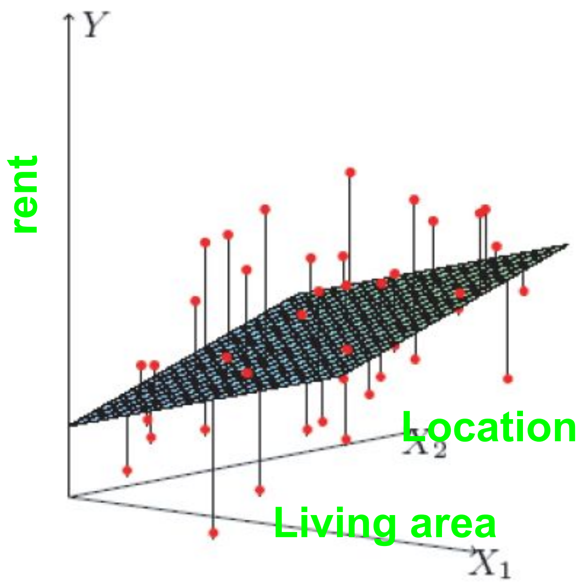
Living area (ft^2)	# bedroom	Rent ($)
230	1	600
506	2	1000
433	2	1100
109	1	500
…		
150	1	?
270	1.5	?

$$f\left(\underbrace{\text{square-ft., \# bedroom, distance}}_{\text{input: feature } x}\right) = \underbrace{\text{rent}}_{\text{output: real-valued label } y}$$

target function

The problem of regression





$$f: x \mapsto y$$

target function, input, output

- Features (input):
  - Living area, distance to campus, # bedroom ...
  - Denote as a vector $x = (x_1, x_2, \ldots, x_n)^{\top}$
- Real-valued label (output):
  - Rent
  - Denoted as y
- Training set:
  - $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$
- Testing set:
  - $(x^{(m+1)}, ?), (x^{(m+2)}, ?), \ldots, (x^{(m+M)}, ?)$

Linear Regression Model

- Assume y is a linear function of x plus noise ϵ

$$y = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n + \epsilon$$

  - ϵ is a random variable with $\mathbb{E}\epsilon = 0$ and $\mathrm{E}\epsilon^2 = \sigma^2 < \infty$
  - ϵ is independent of x

- Using the notation

augmented with an additional dimension

$$\theta \leftarrow \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_n \end{bmatrix} \quad \text{and} \quad x \leftarrow \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

- The linear model can be written as

$$\boxed{y = x^\top \theta + \epsilon}$$

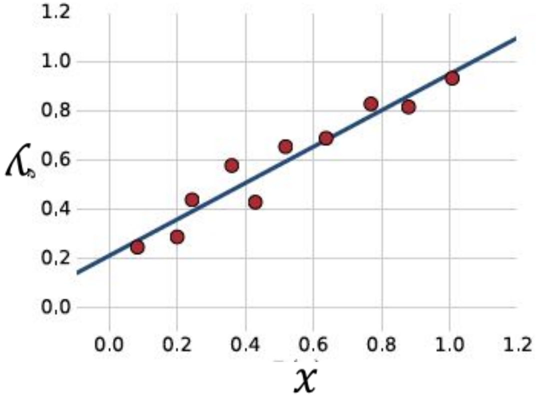
Least mean square method

- $$y = x^{\top}\theta + \epsilon$$

- Given m data points $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, how to estimate θ?

- Find θ which minimizes the mean square error

$$\min_{\theta} L(\theta) := \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - {x^{(i)}}^{\top}\theta\right)^2$$



- Our usual trick: set $\frac{\partial L(\theta)}{\partial \theta} = 0$ and find the solution θ

Matrix version

- Using the notation:

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & \dots & x^{(m)} \\ | & | & | \end{bmatrix} \text{ and } Y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

  - each $x^{(i)}$ is a feature vector
  - X is a $n \times m$ matrix
  - Y is a $m \times 1$ vector
- The mean square error is

$$L(\theta) = \frac{1}{m} \| X^\top \theta - Y \|^2$$

- The minimizer satisfies

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m} XY + \frac{2}{m} XX^\top \theta = 0 \Leftrightarrow \quad XX^\top \theta = XY$$

Analytical Solution

- When $XX^\top$ is invertible, the solution is unique

$$\boxed{\hat{\theta} = (XX^\top)^{-1}XY}$$

If d > m, not invertible

- The analytical solution is unbiased:

$$\begin{aligned}\hat{\theta} &= (XX^\top)^{-1}XY \\ &= (XX^\top)^{-1}X(X^\top\theta^* + \epsilon) \\ &= \theta^* + (XX^\top)^{-1}X\epsilon\end{aligned}$$

$$\Rightarrow \mathbb{E}[\hat{\theta}] = \theta^* + (XX^\top)^{-1}X\mathbb{E}[\epsilon] = \theta^*$$

Computation Cost

- The matrix inversion in $\hat{\theta} = (XX^\top)^{-1}XY$ can be very expensive to compute.

- Matrix Multiplication $XX^\top$: $O(mn^2)$
- Matrix Inversion $(XX^\top)^{-1}$: $O(n^3)$

- Overall computational cost: $O(mn^2)$, given $m \gg n$

Gradient Descent

-

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} - \alpha_t \frac{\partial L(\hat{\theta}^t)}{\partial \hat{\theta}^t}$$

- Step size $\alpha_t > 0$. (fixed or line search)
- For LMS, $\frac{\partial L(\hat{\theta}^t)}{\partial \hat{\theta}^t} = -\frac{2}{m} XY + \frac{2}{m} XX^\top \hat{\theta}^t$.
- Computation cost:
  - Matrix Vector Multiplication $X^\top \hat{\theta}^t$: $O(mn)$
  - Matrix Vector Multiplication $X(X^\top \hat{\theta}^t)$ and XY: $O(mn)$
  - Overall computational cost per iteration: $O(mn)$
  - Better than $O(mn^2)$, but it may runs many iterations?

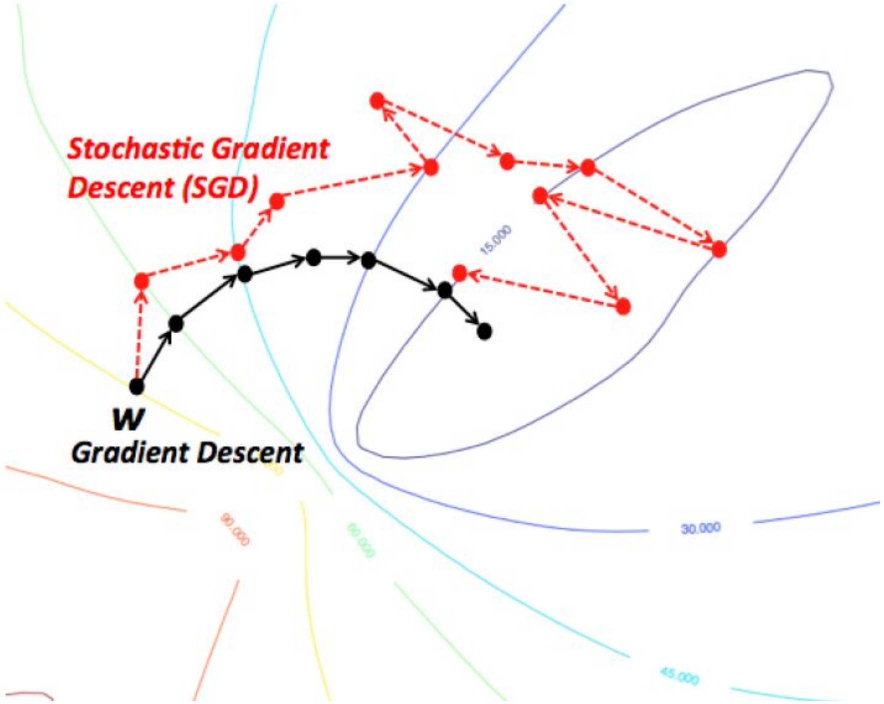
Stochastic Gradient Descent

- What if n is also too large?

- $L(\theta) := \frac{1}{m}\sum_{i=1}^{m} l_i(\theta)$
  - For Least Mean Square, $l_i(\theta) := \left(y^{(i)} - {x^{(i)}}^\top \theta\right)^2$.

- Stochastic gradient descent: use one data point each time
  - Randomly sample i from $1, \ldots, m$ with equal probability
  - Perform a gradient step $\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} - \beta_t \frac{\partial l_i(\hat{\theta}^t)}{\partial \hat{\theta}^t}$
  - For Least Mean Square, $\frac{\partial l_i(\hat{\theta}^t)}{\partial \hat{\theta}^t} = \left(y^{(i)} - {\hat{\theta}^t}{}^\top x^{(i)}\right) x^{(i)}$

Stochastic Gradient Descent



A recap:

- Stochastic gradient update rule

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} - \beta_t \frac{\partial l_i(\hat{\theta}^{t})}{\partial \hat{\theta}^{t}}$$

  - Pros: on-line, low per-step cost
  - Cons: coordinate, maybe slow-converging
- Gradient descent

$$\hat{\theta}^{t+1} \leftarrow \hat{\theta}^{t} - \alpha_t \frac{\partial L(\hat{\theta}^{t})}{\partial \hat{\theta}^{t}}$$

  - Pros: fast-converging, easy to implement
  - Cons: need to read all data
- Solve normal equations

$$(XX^\top)\hat{\theta} = XY$$

  - Pros: a single-shot algorithm! Easiest to implement.

Geometric Interpretation of LMS

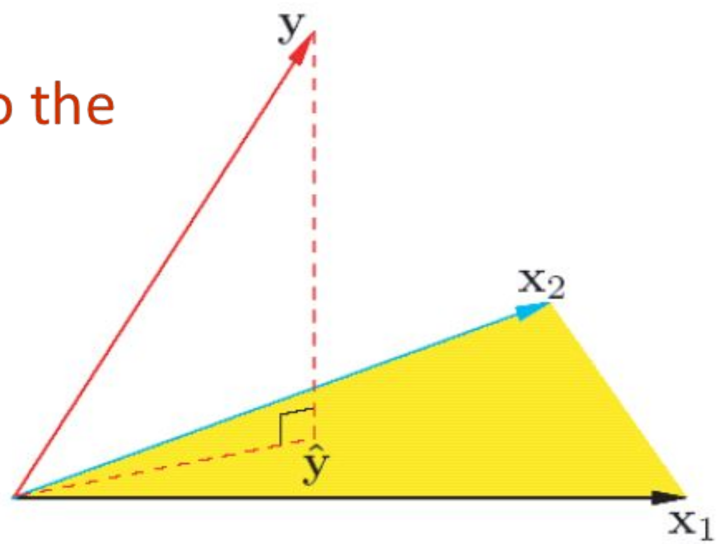
- The predictions on the training data are:
$$\hat{y} = X^\top\theta = X^\top(XX^\top)^{-1}Xy$$

- Look at residual $\hat{y} - y$

$$\hat{y} - y = (X^\top(XX^\top)^{-1}X^\top - I)y$$

$$X(\hat{y} - y) = X(X^\top(XX^\top)^{-1}X^\top - I)y = 0$$

- $\hat{y}$ is the orthogonal projection of y into the space spanned by the columns of X



Probabilistic Interpretation of LMS

- Assume y is a linear in x plus noise ϵ

$$y = \theta^\top x + \epsilon$$

- Assume ϵ follows a Gaussian $\epsilon \sim \mathcal{N}(0, \sigma^2)$

$$p\left(y^{(i)} \middle| x^{(i)}; \theta\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right)$$

- By independence assumption, likelihood is

$$L(\theta) = \prod_{i=1}^{m} p\left(y^{(i)} \middle| x^{(i)}; \theta\right) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^m \exp\left(-\frac{\sum_{i=1}^{m}\left(y^{(i)} - \theta^\top x^{(i)}\right)^2}{2\sigma^2}\right)$$

Probabilistic Interpretation of LMS, cont.

- The log-likelihood is:

$$\log L(\theta) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^\top x^{(i)} \right)^2$$

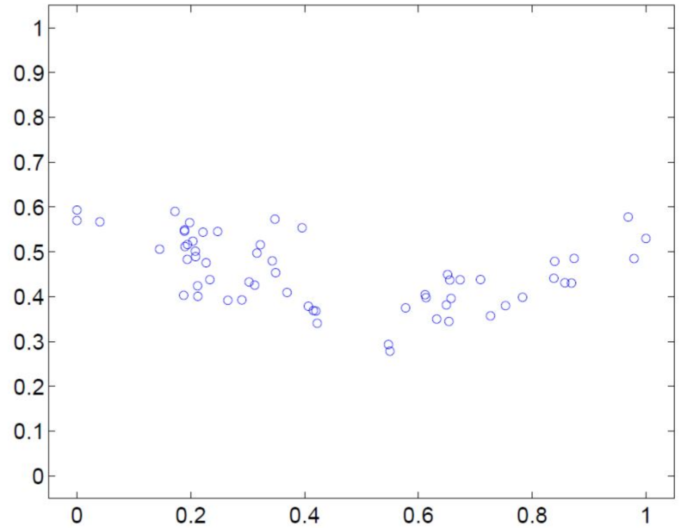
- Do you recognize the last term?

$$LMS\colon \quad \frac{1}{m} \sum_{i}^{m} \left(y^{(i)} - \theta^\top x^{(i)} \right)^2$$

- Thus under independence assumption and Gaussian noise

Nonlinear regression

•



- Want to fit a polynomial regression model

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n + \epsilon$$

- Let $\tilde{x} = (1, x, x^2, \ldots, x^n)^\top$ and $\theta = (\theta_0, \theta_1, \theta_2, \ldots, \theta_n)^\top$

Least mean square method

- Given m data points, find θ that minimizes the mean square error

$$\hat{\theta} = argmin_\theta \, L(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top \tilde{x}^i\right)^2$$

- Our usual trick: set gradient to 0 and find parameter

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}\sum_{i=1}^{m}\left(y^i - \theta^\top \tilde{x}^i\right)\tilde{x}^i = 0$$

$$\Leftrightarrow -\frac{2}{m}\sum_{i=1}^{m} y^i \tilde{x}^i + \frac{2}{m}\sum_{i=1}^{m} \tilde{x}^i {\tilde{x}^i}^\top \theta = 0$$

Matrix version of the gradient

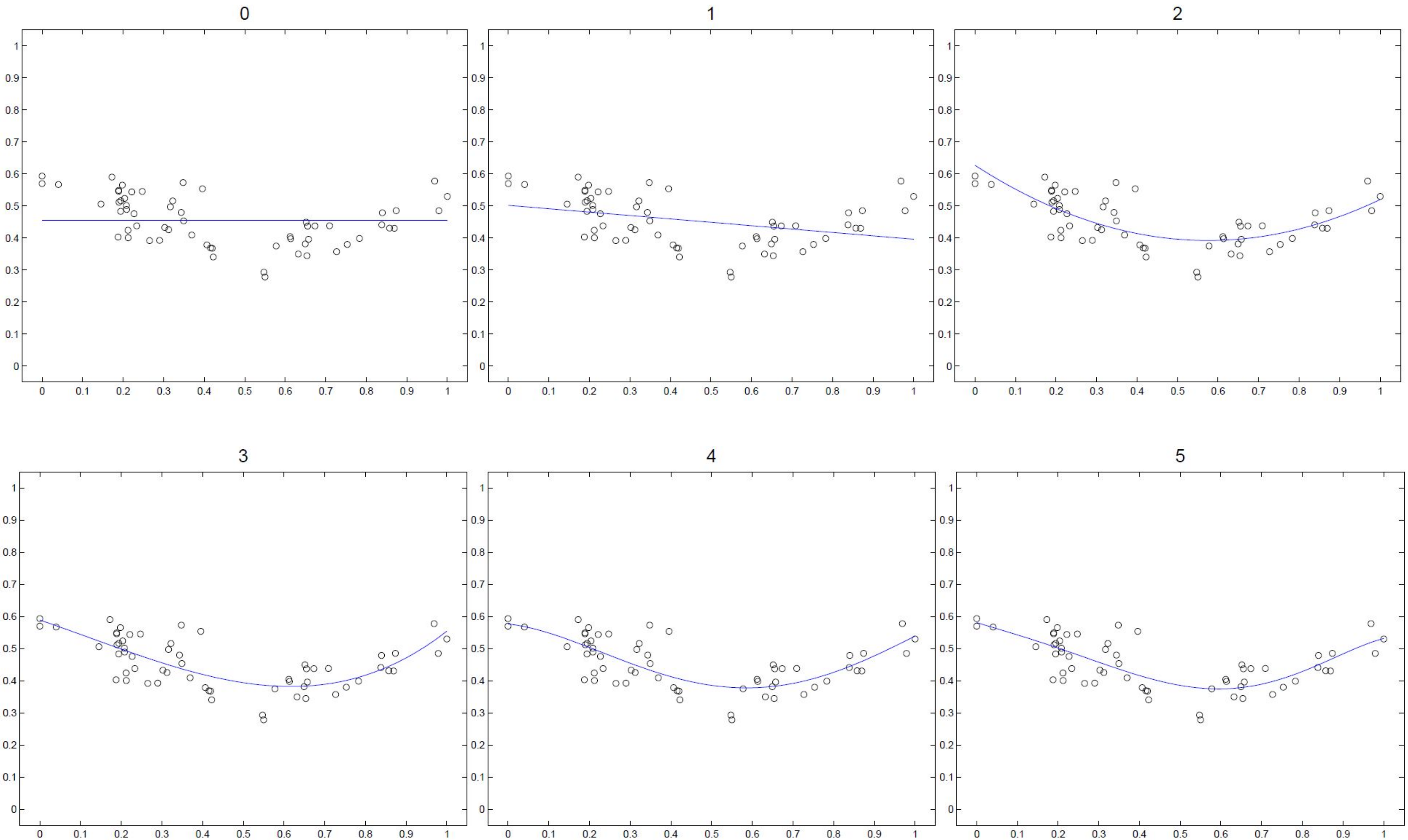
- Define $\tilde{X} = \left(\tilde{x}^{(1)}, \tilde{x}^{(2)}, \dots \tilde{x}^{(m)}\right), y = \left(y^{(1)}, y^{(2)}, \dots, y^{(m)}\right)^{\top}$, gradient becomes

$$\frac{\partial L(\theta)}{\partial \theta} = -\frac{2}{m}\tilde{X}y + \frac{2}{m}\tilde{X}\tilde{X}^{\top}\theta = 0$$
$$\Rightarrow \hat{\theta} = \left(\tilde{X}\tilde{X}^{\top}\right)^{-1}\tilde{X}y$$

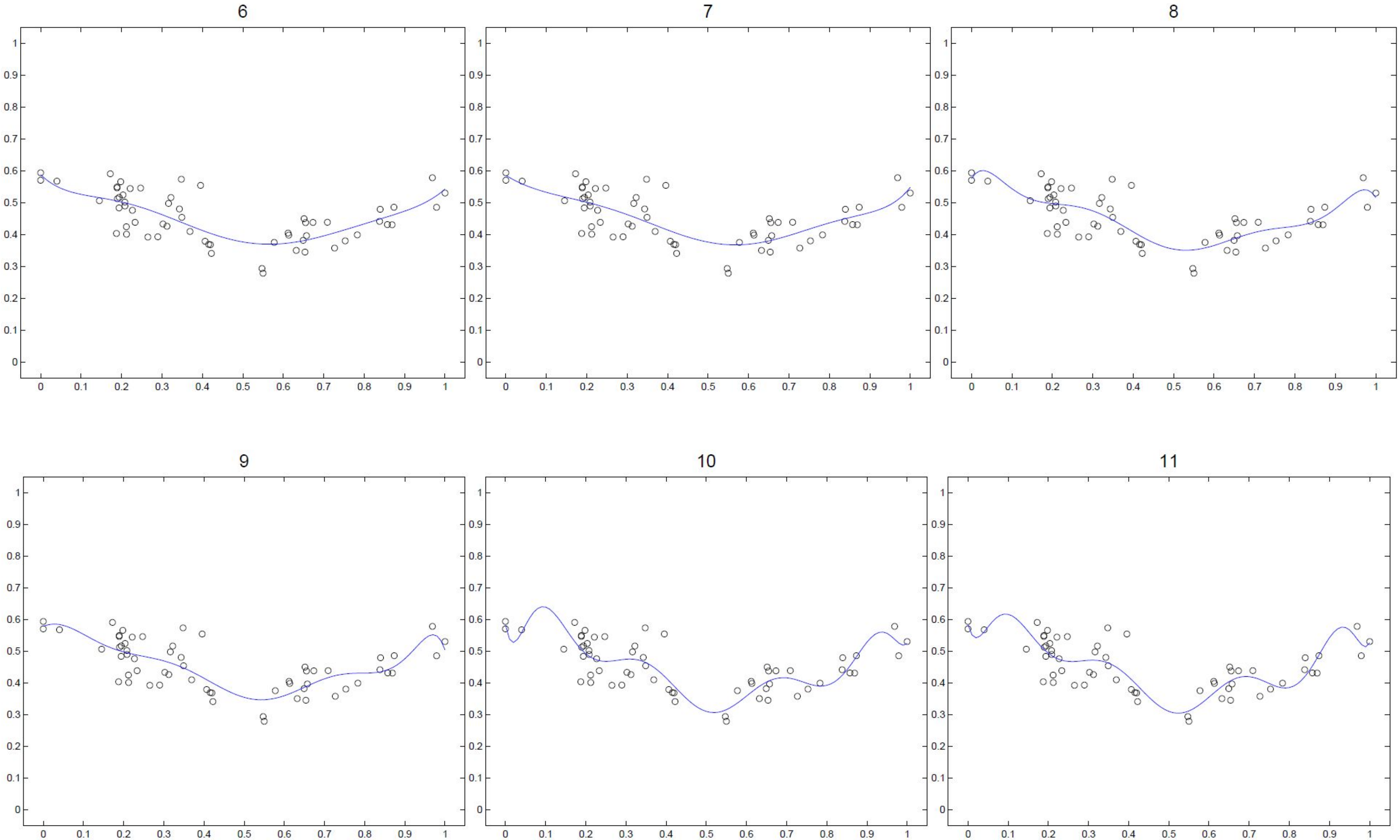
- Note that $\tilde{x} = (1, x, x^2, \dots, x^n)^{\top}$

- If we choose a different maximal degree n for the polynomial, the solution will be different.

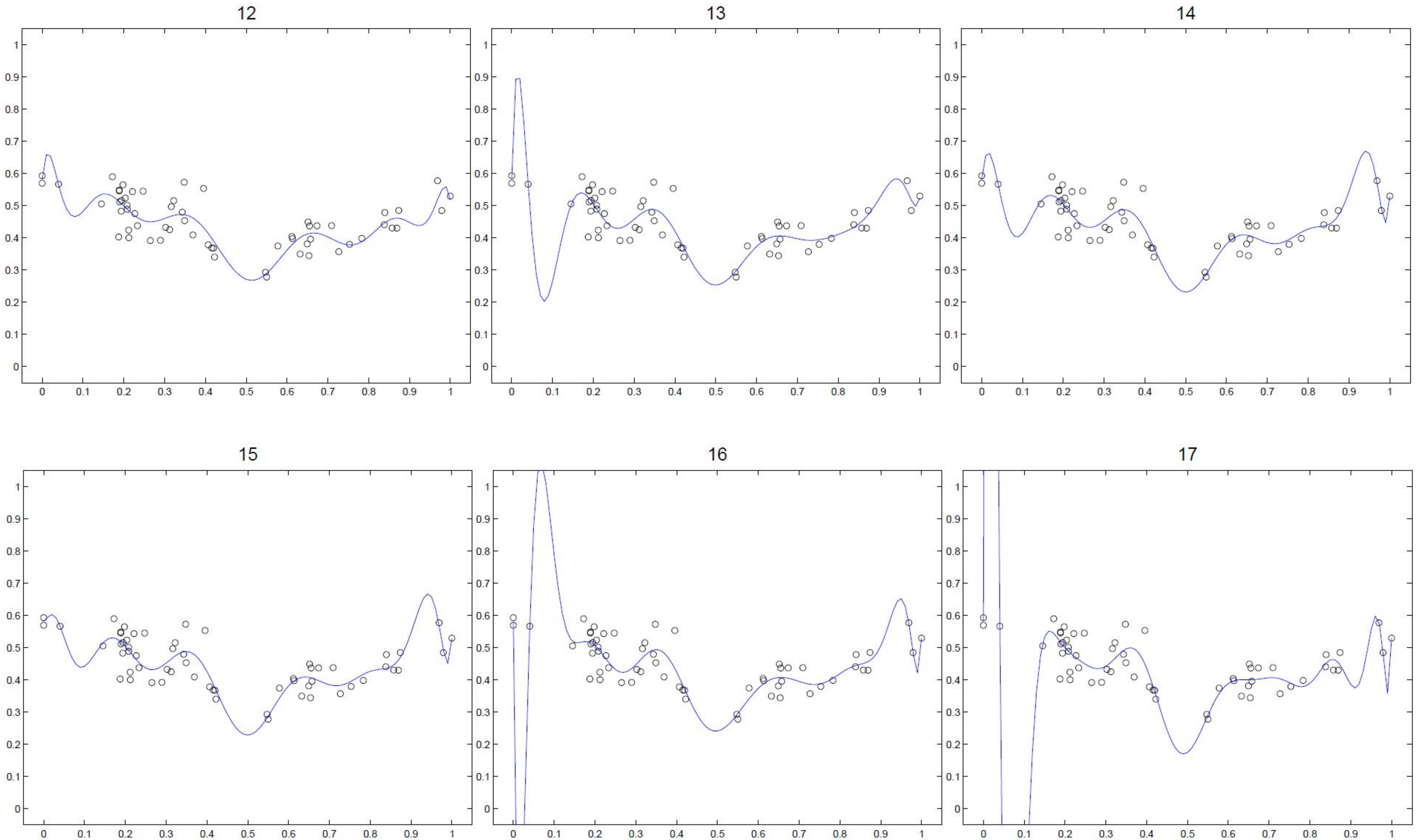
Increasing the maximal degree



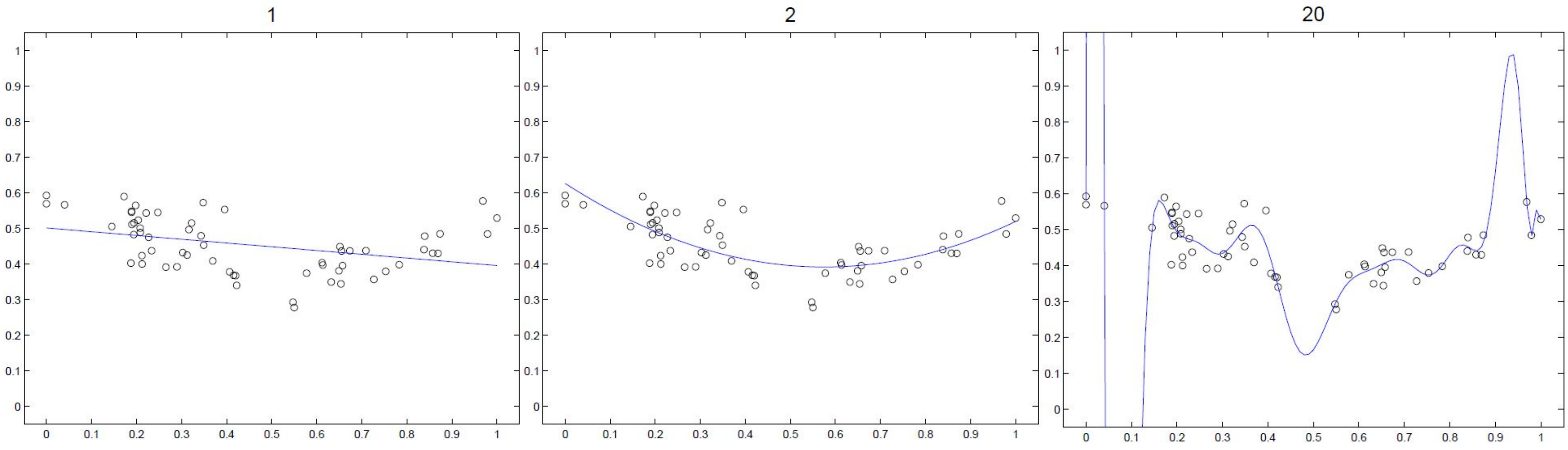
Increasing the maximal degree



Increasing the maximal degree



Which one is better?



- Can we increase the maximal polynomial degree to very large, such that the curve passes through all training points?

- The optimization does not prevent us from doing that