Lecture 18: Least squares continued

Thursday, October 29, 2015 9:30 AM

Admin:

"Least Squares" Fitting & Applications to Data Analysis

Reading:
Meyer 4.6
5.13-14
Strang 3.3

Problem: System of equations

$$A\vec{x} = \vec{b}$$

but $\vec{b} \notin R(A)$!

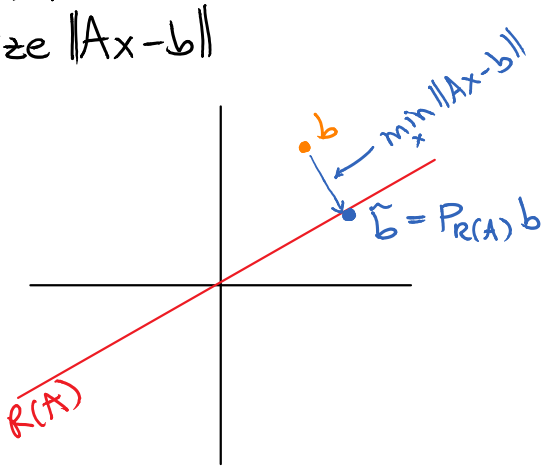
$\Rightarrow$ Find x to minimize $\|Ax-b\|$

Algebraically:

$$A = \sum_i \lambda_i \vec{u}_i \vec{v}_i^T$$

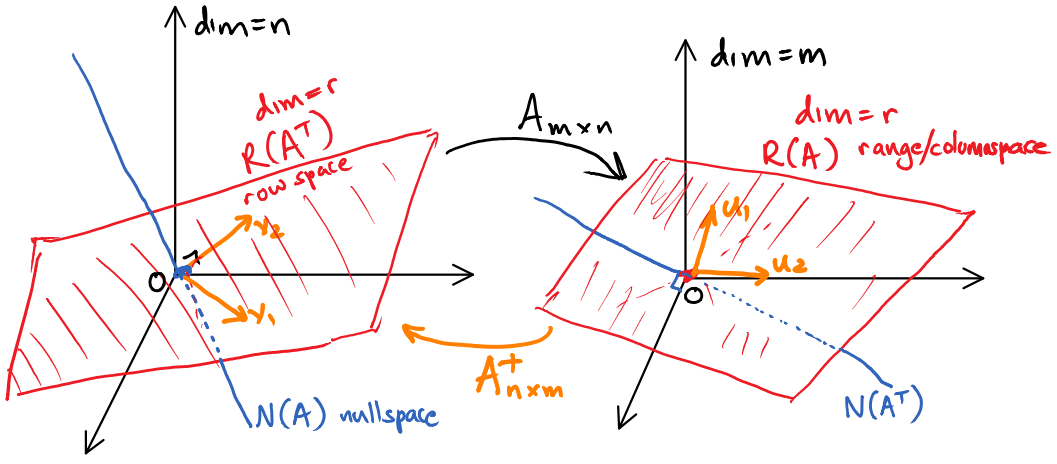
$$A^+ = \sum_{i:\lambda_i>0} \frac{1}{\lambda_i} \vec{v}_i \vec{u}_i^T$$

(psendo-inverse)



$$\Rightarrow x = A^+ b$$

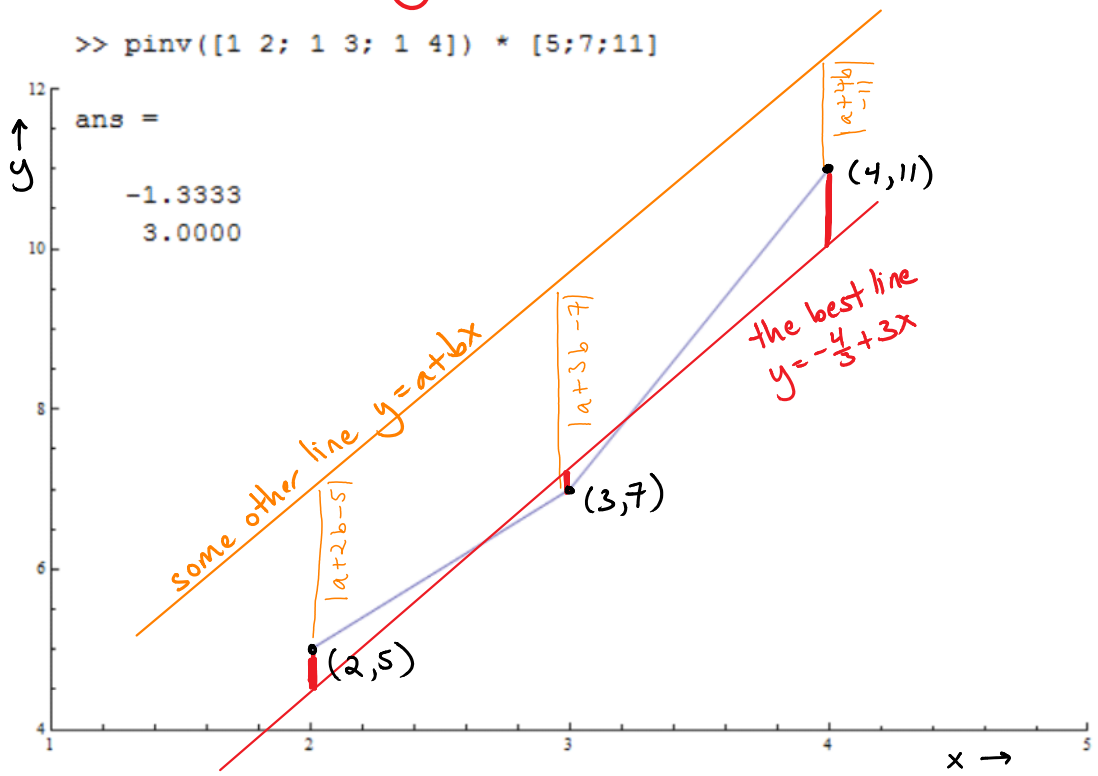
minimizes $\|Ax-b\|$



Example: Linear regression

Example: Linear regression

```
>> pinv([1 2; 1 3; 1 4]) * [5;7;11]

ans =

   -1.3333
    3.0000
```



Setting: m data points

$$\left(x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_k^{(1)}, y^{(1)}\right)$$
$$\left(x_1^{(2)}, x_2^{(2)}, \dots, x_k^{(2)}, y^{(2)}\right)$$
$$\vdots$$
$$\left(x_1^{(m)}, x_2^{(m)}, \dots, x_k^{(m)}, y^{(m)}\right)$$

components known exactly, e.g., date/time

component that we want to predict

Goal: Find the best linear predictor for y,

$$a_0, a_1, a_2, \dots, a_k \in \mathbb{R}$$

to minimize total squared error.

$$\sum_{j=1}^{m} \left| a_0 + a_1 x_1^{(j)} + \dots + a_k x_k^{(j)} - y^{(j)} \right|^2$$

Answer:

$$\begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_k^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_k^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & \cdots & x_k^{(3)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \end{pmatrix}$$

$$\begin{pmatrix} 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_k^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & \cdots & x_k^{(3)} \\ & \vdots & & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_k^{(m)} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{pmatrix}$$

A $\qquad$ $\vec{y}$

The least-squares solution is $A^+ \vec{y}$.

(A^+: pseudoinverse)

Four ways to find x to minimize $\|Ax - b\|$:

① Compute pseudo-inverse A^+, set $x = A^+ b$.

Pros: Easy to remember, one line pinv(A)*b in Matlab

Cons: Slow for large A, numerically unstable

② Find x manually, with calculus:

$$E = \text{squared error} = \left\| \begin{pmatrix} 1 & a_1 \\ 1 & a_2 \\ 1 & a_3 \\ & \vdots \\ 1 & a_m \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \right\|^2$$

$$= \sum_{j=1}^{m} (x_1 + a_j x_2 - b_j)^2$$

$$\frac{\partial}{\partial x_1} = 2 \sum_j (x_1 + a_j x_2 - b_j) = 0 \quad \Rightarrow m x_1 + \left(\sum_j a_j\right) x_2 = \left(\sum_j b_j\right)$$

$$\frac{\partial E}{\partial x_2} = 2 \sum_j a_j (x_1 + a_j x_2 - b_j) = 0 \quad \Rightarrow \left(\sum_j a_j\right) x_1 + \left(\sum_j a_j^2\right) x_2 = \sum_j a_j b_j$$

Let $\bar{a} = \frac{1}{m} \sum a_j$ (average value of a_j's)

$\bar{b} = \frac{1}{m} \sum_j b_j$ (average of b_j's)

First equation

$\Rightarrow$ intercept $x_1 = \bar{b} - \bar{a} x_2$

(this is 0 if the data is centered so $\bar{a} = \bar{b} = 0$)

Second equation

$$\Rightarrow x_2 = \frac{\sum_j a_j b_j - m \cdot \bar{a} \cdot \bar{b}}{\sum a_j^2 - m(\bar{a})^2}$$

$\left(= \frac{\mathrm{Cov}(A,B)}{\mathrm{Var}(A)} \quad \text{if } A \text{ and } B \text{ are random variables with } \Pr[(A,B)=(a_j,b_j)] = \frac{1}{m}. \right)$

Pros: None (okay, it gives an easy closed-form solution)

③ Solve $A^TAx = A^Tb$

Theorem: Let A be an $m \times n$ real matrix with rank n. Then for any b,

- The equation $A^TAx = A^Tb$ is feasible (i.e., it has a solution x)
- The unique solution is

$$(A^TA)^{-1}A^Tb = A^+b$$

(A^+: pseudoinverse)

Proof: There is a solution because

- $R(A^TA) = R(A^T)$ (shown above)
- $A^Tb \in R(A^T)$

$\Rightarrow A^Tb \in R(A^TA)$ ✓

The solution is unique because $N(A) = \{\vec{0}\}$.

(because $\mathrm{rank}(A) = \dim R(A^T) = n$, by assumption, and by rank-nullity $\dim R(A^T) + \dim N(A) = n$)

So why is $(A^TA)^{-1}A^Tb = A^+b$, as claimed? There are several ways of seeing it.

Algebraically: Using the SVD of A, $A = \sum_i \lambda_i \vec{u}_i \vec{v}_i^T$,

$$A^+ = \sum_{i:\lambda_i>0} \frac{1}{\lambda_i} v_i u_i^T$$

$$(A^TA)^{-1}A^T = \left[\sum_i \lambda_i^2 v_i v_i^T\right]^{-1}\left(\sum_j \lambda_j v_j u_j^T\right)$$

$$= \left(\sum_i \frac{1}{\lambda_i^2} v_i v_i^T\right)\left(\sum_j \lambda_j v_j u_j^T\right)$$

* since $\mathrm{rank}(A) = n$,

* since rank(A) = n, $\lambda_1, \ldots, \lambda_n$ are all > 0

$$= \sum_i \frac{1}{\lambda_i} v_i u_i^T$$

$$= A^+ \quad \checkmark$$

More geometrically:

$x = A^+ b$ solves $Ax = P_{R(A)} b$

$$\Rightarrow A^T A x = A^T P_{R(A)} b$$

$$= A^T b \quad \text{since } A^T = A^T P_{R(A)}$$

(in general $A = P_{R(A)} A P_{R(A^T)}$)

□

Corollary: For any $m \times n$ real matrix A, of rank n,

$$A^+ = (A^T A)^{-1} A^T$$

Proof: Since we just showed $A^+ b = (A^T A)^{-1} A^T b$ for any b. □

Pros: Fast

Cons: condition number of $A^T A$ = (condition number of A)^2

Method ④ Solve $\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$

Pros: Maintains sparsity of A.

[Problem 4.6.9, p.237 of Meyer]

⑤ Latest research has focused on finding fast randomized approximation algorithms, based on dimension reduction à la Johnson-Lindenstrauss Lemma, eg.,

[Clarkson Woodruff 2012 http://arxiv.org/abs/1207.6365]

[Nelson & Nguyen 2012 http://arxiv.org/abs/1211.1002]

Evaluating the quality of a fit:

Of course there is a lot of statistics to consider when running

Of course there is a lot of statistics to consider when running linear regressions.

The most basic statistic is the "R^2 value", also known as the "coefficient of determination."

Definition: Given data points $(x_1, y_1), \ldots, (x_m, y_m)$, and the least-squared-error fit line $y = \alpha + \beta x$,

$$R^2 = 1 - \frac{\sum_j (y_j - \alpha - \beta x_j)^2}{\sum_j (y_j - \bar{y})^2}$$

where $\bar{y} = \frac{1}{m}\sum_j y_j$

$$= 1 - \frac{\text{(total squared error from best-fit line)}}{\text{(total squared error from horizontal line } \bar{y})}$$

Observe: $0 \le R^2 \le 1$

and higher is better

- Collinear data points will have $R^2 = 1$ (since best-fit line will fit exactly)
- But what counts as a "good fit" depends very much on the field (eg., macroeconomics)

Interpretation of R^2: Explained variance

If the data were distributed randomly about the mean $\bar{y}$, the observed variance is $\mathbb{E}[Y - \mathbb{E}(Y)]^2 = \frac{1}{m}\sum_{j=1}^{m} (y_j - \bar{y})^2$.

For data distributed randomly off the line $y = \alpha + \beta x$, ie., $y_j = \alpha + \beta x_j + e_j$, the remaining variance is

$$\frac{1}{m}\sum_j e_j^2 = \frac{1}{m}\sum_j (y_j - \alpha - \beta x_j)^2.$$

$\Rightarrow R^2$ measures "how much of the observed variance is explained by the observed x_j data points.

Probabilistic interpretation: Why least squares?

Choosing $\vec{x}$ to minimize $\|A\vec{x} - \vec{y}\|$ is geometrically natural, but why is this natural in data analysis?

One reason (of several): In a natural probabilistic model,

least-squares regression finds the maximum likelihood estimate of $\vec{\alpha}$.

Model: Assume that the y-values are generated via

$$y^{(i)} = \vec{\alpha} \cdot \vec{x}^{(i)} + \varepsilon^{(i)}$$

where $\varepsilon^{(i)} \sim N(0, \sigma^2)$ — independent, identically distributed, Gaussian random noise.

$$\Rightarrow p_{\vec{\alpha}}(y^{(i)} \mid x^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[\frac{-1}{2\sigma^2}(y^{(i)} - \alpha \cdot x^{(i)})^2\right]$$

↑ probability density function of $y^{(i)}$ for model parameters $\vec{\alpha}$

$$\Rightarrow p_{\vec{\alpha}}(y^{(1)}, \ldots, y^{(m)} \mid x^{(1)} \ldots x^{(m)}) = \frac{1}{(2\pi\sigma^2)^{m/2}} \exp\left[\frac{-1}{2\sigma^2} \sum_{i=1}^{m} (y^{(i)} - \alpha \cdot x^{(i)})^2\right]$$

$\Rightarrow$ Setting $\vec{\alpha}$ to minimize $\sum_i (y^{(i)} - \alpha \cdot x^{(i)})^2$ maximizes the likelihood of the observed data.

(**Notice**: This holds for any value of the variance σ^2, and σ need not be known to choose α.)

Linear regression with weights:

If errors between observations are correlated, or if some observations are more reliable than others, then minimizing the weighted sum of squared errors is better than unweighted.

Example: Say

$$y_1 = \alpha x_1 + \varepsilon_1, \quad \varepsilon_1 \sim N(\overset{\mu}{0}, \overset{\sigma^2}{1}) \leftarrow \text{more reliable}$$

$$y_2 = \alpha x_2 + \varepsilon_2, \quad \varepsilon_2 \sim N(0, 2) \leftarrow \text{less reliable}$$

$$\Rightarrow p_\alpha[y_1, y_2 \mid x_1, x_2] = \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}(y_1 - \alpha x_1)^2} \cdot \frac{1}{\sqrt{2\pi \cdot 2}} e^{\frac{-1}{2 \cdot 2}(y_2 - \alpha x_2)^2}$$

$$\propto \exp\left[\frac{-1}{2} \cdot \left\{(y_1 - \alpha x_1)^2 + \frac{1}{2}(y_2 - \alpha x_2)^2\right\}\right]$$

$\Rightarrow$ The max-likelihood α minimizes the weighted sum

$$S = \frac{1}{\sigma_1^2}(y_1 - \alpha x_1)^2 + \frac{1}{\sigma_2^2}(y_2 - \alpha x_2)^2$$

$$-\frac{1}{2}\frac{\partial S}{\partial \alpha} = \frac{x_1}{\sigma_1^2}(y_1 - \alpha x_1) + \frac{x_2}{\sigma_2^2}(y_2 - \alpha x_2) = 0$$

$$\Rightarrow \alpha = \frac{\frac{x_1 y_1}{\sigma_1^2} + \frac{x_2 y_2}{\sigma_2^2}}{\frac{x_1^2}{\sigma_1^2} + \frac{x_2^2}{\sigma_2^2}}$$

More generally: To choose $\vec{\alpha}$ to minimize

$$\sum_{j=1}^{m} w_j (y_j - x_j \cdot \alpha)^2,$$

observe $\| W(y - A\alpha) \|^2$ where $W = \begin{pmatrix} w_1 & & 0 \\ & w_2 & \\ & & \ddots \\ 0 & & w_m \end{pmatrix}$

$$A = \begin{pmatrix} — & x_1 & — \\ — & x_2 & — \\ & \vdots & \\ — & x_m & — \end{pmatrix}$$

This is standard least-squares for the equation

$$WA\alpha = Wy$$

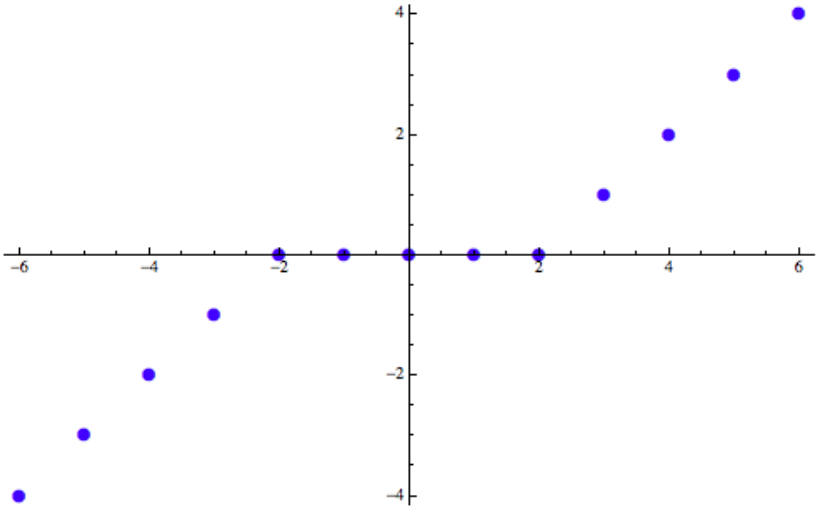
$\Rightarrow$ best α satisfies

$$A^T W^T W A \alpha = A^T W^T W y, \text{ etc.}$$

Practical application: Locally weighted linear regression

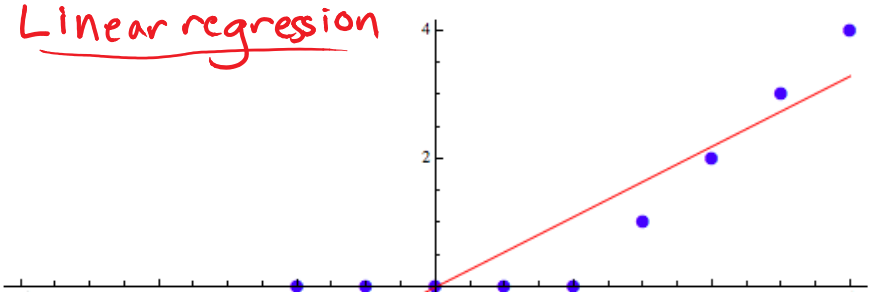
Example:

```
data = {
    {-6, -4},
    {-5, -3},
    {-4, -2},
    {-3, -1},
    {-2, 0},
    {-1, 0},
    {0, 0},
    {1, 0},
    {2, 0},
    {3, 1},
    {4, 2},
    {5, 3},
    {6, 4}
   };
m = Length[data];

dataplot = ListPlot[data]
```




```
X = Table[{1, data[[j, 1]]}, {j, 1, m}];
X
y = data[All, 2];
ymean = 1/m Plus @@ y;

linear = PseudoInverse[X].y
plotlinear = Plot[{1, x}.linear, {x, -6, 6}]
Show[dataplot, plotlinear]
```

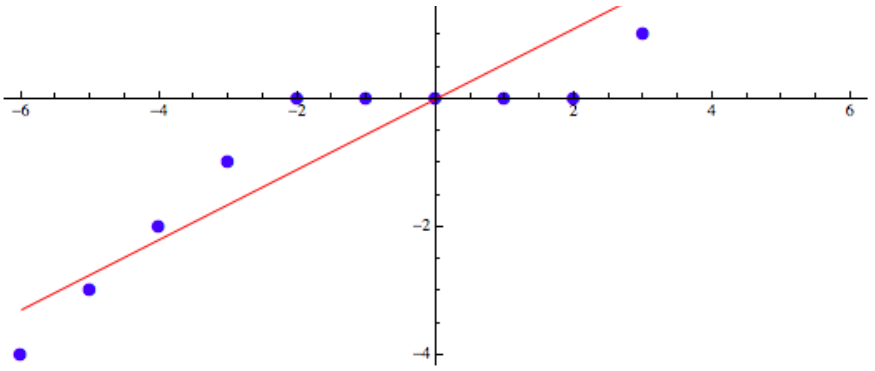
Linear regression



```
plotlinear = Plot[{1, x}.linear, {x, -6, 6}]
Show[dataplot, plotlinear]
```

```
{{1, -6}, {1, -5}, {1, -4}, {1, -3}, {1, -2},
 {1, -1}, {1, 0}, {1, 1}, {1, 2}, {1, 3}, {
```

$$\left\{0, \frac{50}{91}\right\}$$




```
"R^2 value:"
R2linear = 1 - Σ_{j=1}^{m} (y[[j]] - X[[j]].linear)^2 / Σ_{j=1}^{m} (y[[j]] - ymean)^2 // N
```

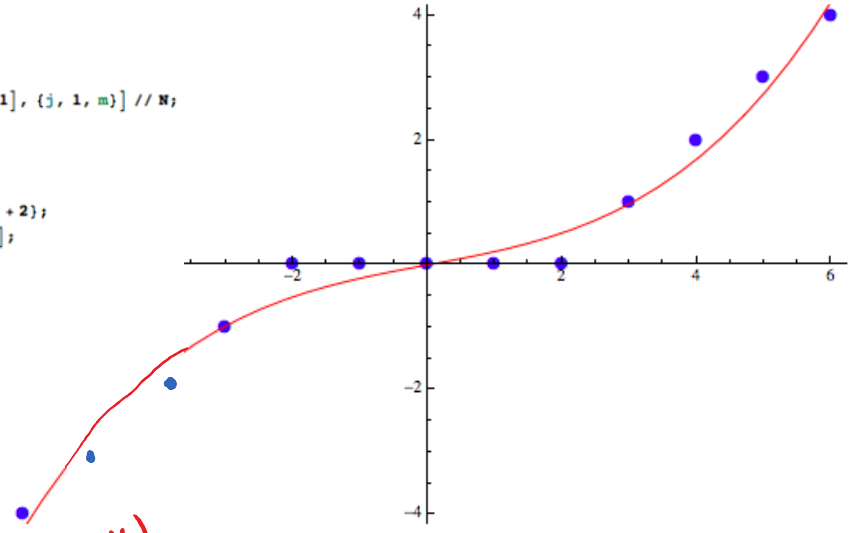
```
R^2 value:

0.915751
```

Cubic regression $1, x, x^2, x^3$

```
HigherOrderRegression[data, 3]
```

```
α= {0, 0.1998, 0, 0.013986}
R2= 0.982884
```

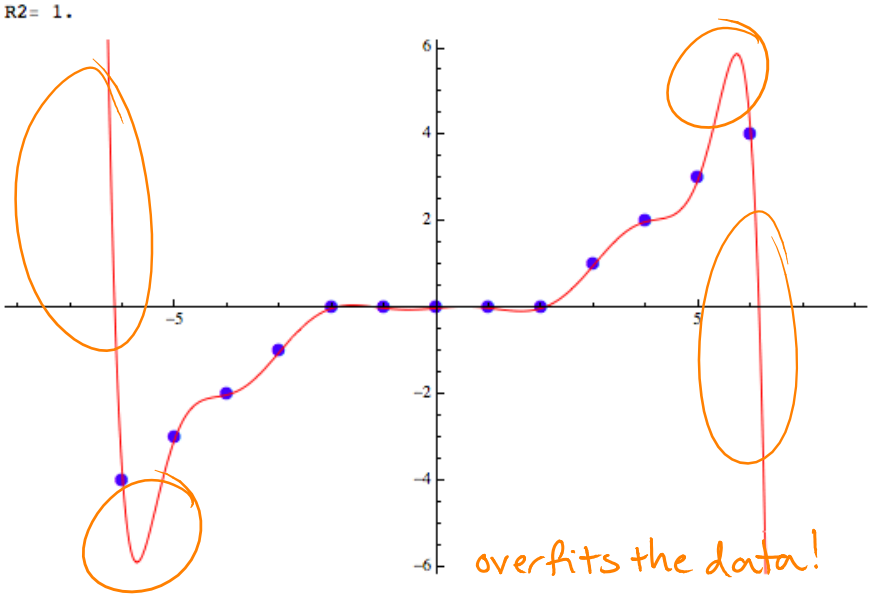
```
HigherOrderRegression[data_, order_] :=
  Module[{m, y, ymean, X, α, plot, min, max, R2},
   m = Length[data];
   y = data[[All, 2]];
   X = Table[Prepend[Table[data[[j, 1]]^k, {k, 1, order}], 1], {j, 1, m}] // N;

   α = PseudoInverse[X].y;
   Print["α= ", α // Chop];

   {min, max} = {Min[data[[All, 1]] - 2, Max[data[[All, 1]] + 2};
   plot = Plot[Table[x^k, {k, 0, order}].α, {x, min, max}];

   ymean = 1/m Plus @@ y;
   R2 = 1 - Σ_{j=1}^{m} (y[[j]] - X[[j]].α)^2 / Σ_{j=1}^{m} (y[[j]] - ymean)^2 // N;
   Print["R2= ", R2];
   Show[dataplot, plot]
  ];
```



11th order $(1, x, x^2, \ldots, x^{11})$

```
Show[HigherOrderRegression[data, 11], PlotRange → {{-8, 8}, {-6, 6}}]
```

```
α= {1.70782×10^-10, 0.101443, -2.42844×10^-10,
  -0.137535, 0, 0.03917, 0, -0.00318122, 0, 0.000104718, 0, -1.2025×10^-6}
R2= 1.
```



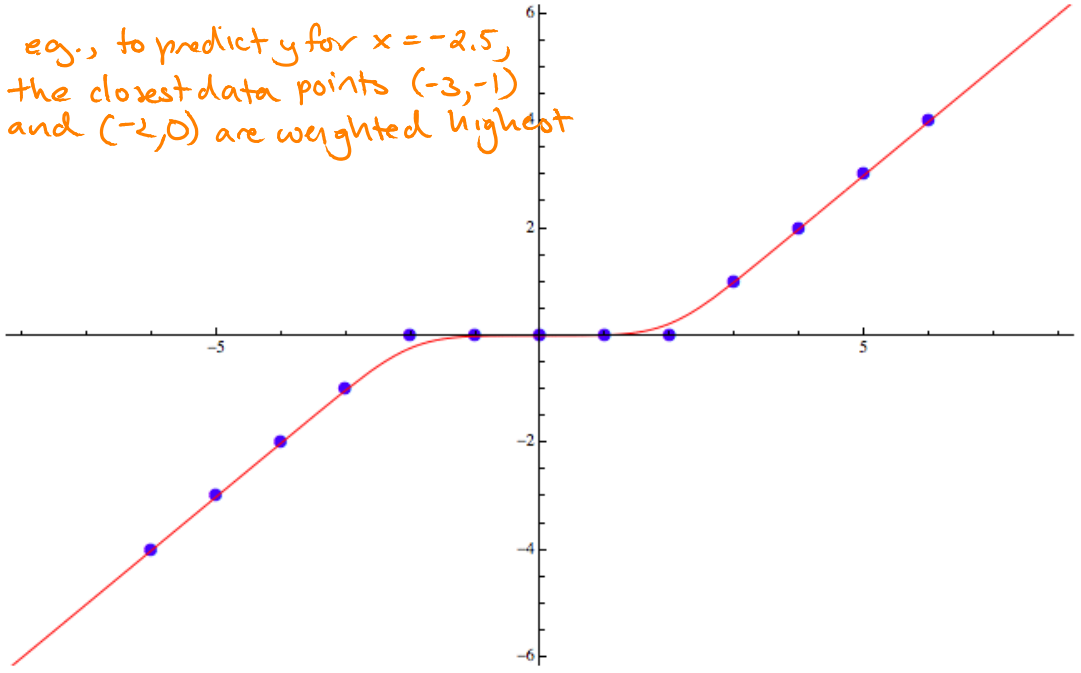
overfits the data!

Locally weighted linear regression

— to predict y at point x,
run weighted linear regression
giving higher weights to data points close to x!

eg, $w_j = e^{-\frac{1}{2\tau^2}(x_j - x)^2}$

```
m = Length[data];
y = data[[All, 2]];
X = Table[{1, data[[j, 1]]}, {j, 1, m}] // N[#, 32] &;

τ = 1;

weightedlinearplot = Plot[
   W = DiagonalMatrix[Table[Exp[-1/(2 τ^2) (data[[j, 1]] - x)^2], {j, 1, m}]];
   α = PseudoInverse[W.X].W.y;
   {1, x}.α,
   {x, -10, 10}];
Show[dataplot, weightedlinearplot, PlotRange → {{-8, 8}, {-6, 6}}]
```



Note: • After running weighted linear regression at x, we don't keep the line — just the prediction at x.

- This is slower than running linear regression just once.
- Full "training set" must be kept to make predictions.
  (known as a "non-parametric" learning algorithm)