CS 613 – Machine Learning

Introduction to Machine Learning

Some slides adapted from Matt Burlick, Drexel

adapted from material created by E. Alpaydin
Prof. Mordohai, Prof. Greenstadt, Pattern Classification (2nd Ed.),
Pattern Recognition and Machine Learning

Linear Regression

Linear Regression Example

line of "best" fit

Linear Regression Example

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

$X =$ $Y =$

$Y = mx + b$

$Y = \theta_0 + \theta_1 x_1$

θ's

x's are the input variables / "features"

y's are the output or target

augment $\theta^T x + b$ by $\theta^T x$

Linear Regression Analysis

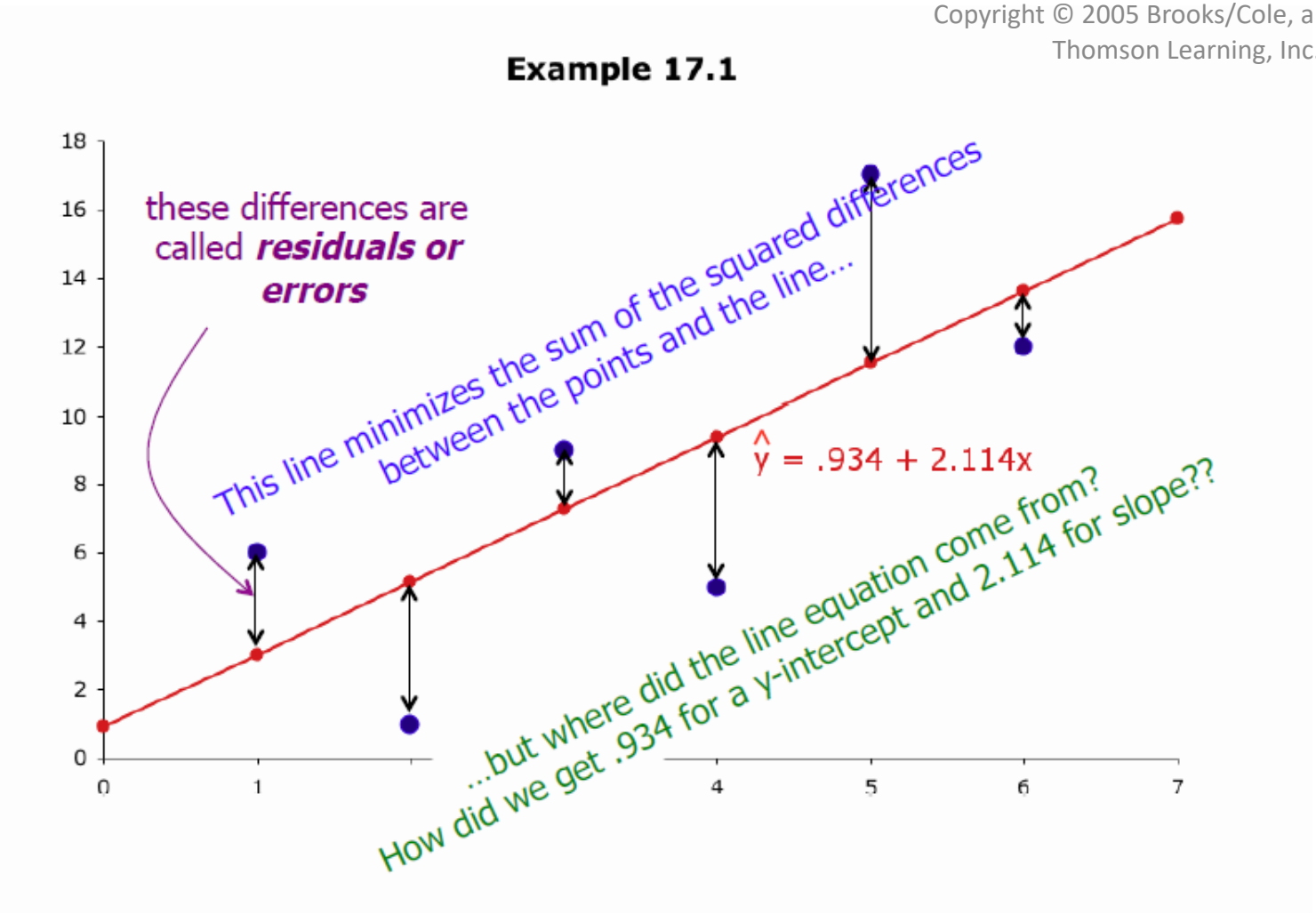
- Let's start off with the simplest version when our data has only one feature

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

- Recall we want to find our model, in this case $\theta = [\theta_0, \theta_1]^T$ such that $g(X_i) \approx Y_i \; \forall \; (X_i, Y_i)$
- To solve this problem we need to choose some error function to minimize (or some likelihood to maximize).
- One of the most common is called the *least squares*

$$J = \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2$$

Least Squares Line



Linear Regression Analysis

- So we want to value θ that minimizes the square of the error

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(Y_i - \widehat{Y}_i\right)^2$$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(Y_i - g(X_i, \theta)\right)^2$$

Least Square Estimate

- For a generally observation with D features we can write
$$g(x) = \theta_0 + \theta_1 x_1 + \cdots \theta_D x_D$$
- If we add an extra feature with a value of one to the beginning of all data instances such that $x = [1\ x]$, then we can write this equation as:
$$g(x) = x\theta$$
- We call this additional feature (or more specifically, parameter θ_0), the *bias* (just to add more confusion!)
- So now we want to minimize (over all observations $(X_i, Y_i) \in (X, Y)$
$$J = \sum_{i=1}^{N} (Y_i - X_i\theta)^2$$

Least Square Estimate

- So now we want to minimize:

$$J = \sum_{i=1}^{N} (Y_i - X_i\theta)^2$$

- Which we can write in matrix form as

$$J = (Y - X\theta)^T (Y - X\theta)$$

- How can we find the minimum of this?
  - Take the derivative with respect to θ, set it equal to zero, and solve for θ!

$$\frac{\partial J}{\partial \theta} = 0$$

Least Square Estimate

$$\frac{\partial}{\partial \theta}(Y - X\theta)^T(Y - X\theta) = 0$$

- Eventually we should arrive at

$$\theta = (X^T X)^{-1} X^T Y$$

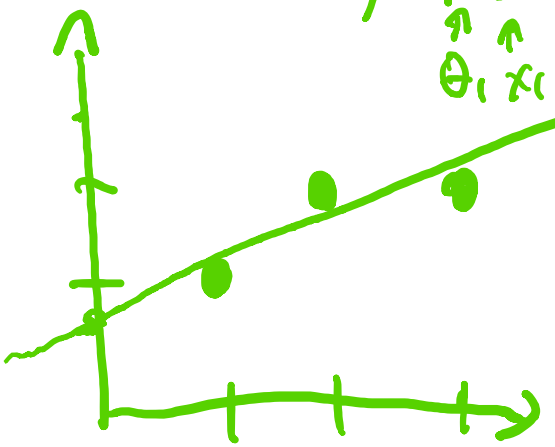
Least Square Estimate

- $J = (Y - X\theta)^T (Y - X\theta)$
- $J = (Y^T - (X\theta)^T)(Y - X\theta)$ //distribution of transpose
- $J = (Y^T - \theta^T X^T)(Y - X\theta)$ //distribution of transpose
- $J = Y^T Y - Y^T X\theta - \theta^T X^T Y + \theta^T X^T X\theta$//distribution
- $\frac{dJ}{d\theta} = -(Y^T X)^T - (X^T Y) + 2X^T X\theta = 0$ //derivative
- $\Rightarrow -X^T Y - X^T Y + 2X^T X\theta = 0$ //simplification
- $\Rightarrow -2X^T Y + 2X^T X\theta = 0$ //simplification
- $\Rightarrow X^T X\theta = X^T Y$ //algebra
- $\theta = (X^T X)^{-1} X^T Y$ //algebra

$\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

Example 1

(1,1), (2,2), (3,2)



$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \begin{bmatrix} x_i \end{bmatrix}$

$X^T X \theta = X^T y$

$y = \frac{1}{2}x + \frac{2}{3}$

$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$

$(X^T X)^{-1} X^T y$

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$

$\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$

$3\theta_0 + 6\theta_1 = 5$

$6\theta_0 + 14\theta_1 = 11$

$2\theta_1 = 1$

$\theta_1 = \frac{1}{2}$

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 11 \end{bmatrix}$

$\theta_0 = \frac{2}{3}$

Example 2

- Model:

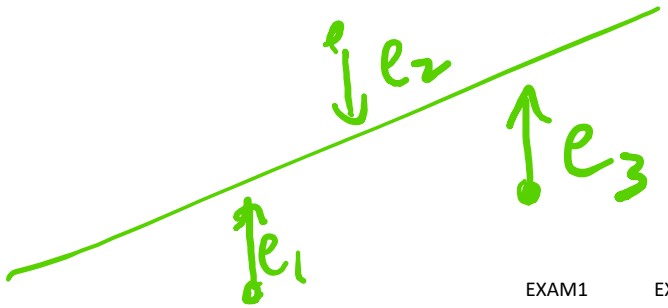
$$Final = \theta_0 + \theta_1 Exam1 + \theta_2 Exam2 + \theta_3 Exam3$$

Note: Final exam out of 200

- Testing Set: The first 8 samples
- Training Set: The rest (next 17 samples)

EXAM1	EXAM2	EXAM3	FINAL
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
93	89	96	192
78	75	68	147
81	90	93	183
88	92	86	177
78	83	77	159
82	86	90	177
86	82	89	175
78	83	85	175
76	83	71	149
96	93	95	192

Example



EXAM1	EXAM2	EXAM3	FINAL
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
93	89	96	192
78	75	68	147
81	90	93	183
88	92	86	177
78	83	77	159
82	86	90	177
86	82	89	175
78	83	85	175
76	83	71	149
96	93	95	192

- Steps
  1. Add an addition feature with value 1 to the training data
  2. Compute the weights $\theta = (X^T X)^{-1} X^T Y$
  3. Add an addition feature with value 1 to the testing data
  4. Compute predicted values as $x\theta$
  5. Compute the average error
     - Since our approach was based on the squared error we should take the mean of the squared error, called the *mean squared error (MSE)*
     - Then we can always take the square root of that, called the *root mean squared error* (RMSE)

$$\sqrt{\frac{e_1^2 + e_2^2 + e_3^2}{\#\text{ of test samples}}}$$

Example

- Training

$$\theta = \begin{bmatrix} -9.9596 \\ 0.3821 \\ 0.5660 \\ 1.1900 \end{bmatrix}$$

  - $FinalExam = -9.9596 + 0.3821 * Exam1 + 0.5660 * Exam2 + 1.1900 * Exam3$
- Testing
  - Project each point
  - Compute the average error
    - MSE: 7.9566
    - RMSE: $\sqrt{7.9566} = 2.8207$

Ytest =	$\widehat{Ytest}$ =
152	152.4652
185	186.0558
180	182.6552
196	201.1921
142	138.5914
101	101.7790
149	149.9209
115	111.0962

Gradient Ascent/Descent

- The solution to linear regression that we just provided, $\theta = (X^T X)^{-1} X^T Y$, is called the *closed-form* solution to the problem.
  - We are able to use mathematics to come up with a direct solution to the minima/maxima problem.
- However, for some problems a closed-form solution/equation may not exist and/or if our matrix is large it may become infeasible to compute inverse
  - Or inverse may not exist in some cases

Gradient Ascent/Descent

- Another approach to solving a minima/maxima problem is to compute the *gradient* of the equation with regards to each parameter in order to move our current parameter's value in that direction
  - And iteratively do this until we converge to a minima/maxima
- This approach is called ***gradient descent*** and generalizes nicely to lots of applications where we need to find the values of parameters to minimize or maximize some function

Least Squares Gradient Descent

- Let's first compute the gradient with respect to a single parameter, θ_i and a single observation, (x, y) and then *vectorize* our solution to update all parameters simultaneously.
- This approach, where we just use one observation at a time is called *iterative,* or *online* gradient learning.
- Returning to our least square linear regression problem, for a single observation the error is:

$$\mathrm{J} = (y - x\theta)^2$$

$\frac{\partial}{\partial \theta_1}$ ~~$\theta_0 + \theta_1 x_1 + \theta_2 x_2 \ldots \theta_n x_n$~~

Derive the update rule

$$\frac{\partial J}{\partial \theta_1} = (y - X\theta)^2 \Rightarrow 2(y - X\theta)(0 - x_1)$$

$$-2x_1(y - X\theta)$$

$$\frac{\partial J}{\partial \theta_1} = 2x_1(X\theta - y)$$

$$\frac{\partial J}{\partial \theta_2} = 2x_2(X\theta - y)$$

$$\frac{\partial J}{\partial \theta_j} = 2x_j(X\theta - y)$$

$$\frac{\partial J}{\partial \theta} = \begin{bmatrix} 2x_0(X\theta - y) \\ 2x_1(X\theta - y) \\ 2x_2(X\theta - y) \\ \vdots \end{bmatrix}$$

Least Squares Gradient Descent

$$J = (y - x\theta)^2$$

- Now let's take the gradient of this with respect to one of the parameters, θ_i

$$\frac{\partial J}{\partial \theta_i} = -2x_i y + 2x_i x\theta = 2x_i(x\theta - y)$$

- We can then vectorize this to get all the gradients at once for this observation:

$$\frac{\partial J}{\partial \theta} = 2x^T(x\theta - y) \Rightarrow 2x^T x\theta - 2x^T y = 0$$

$$2x^T x\theta = 2x^T y$$

$$\theta = (x^T x)^{-1} x^T y$$

- And update your parameters as:

$$\theta = \theta - 2x^T(x\theta - y)$$

Example 2 (again)

- Model:

$$Final = \theta_0 + \theta_1 Exam1 + \theta_2 Exam2 + \theta_3 Exam3$$

Note: Final exam out of 200

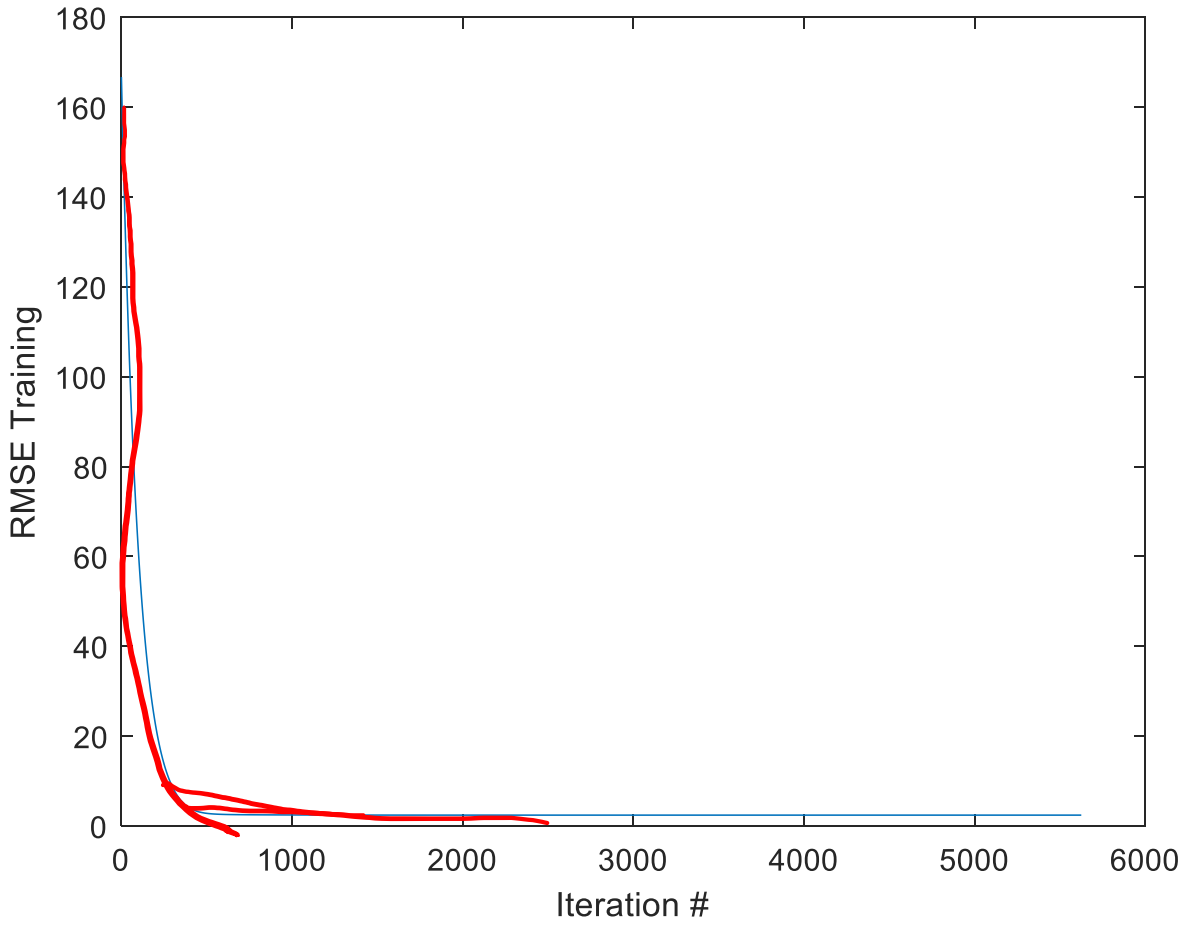
- Testing Set: The first 8 samples
- Training Set: The rest (next 17 samples)
- Let's do this with full-batch gradient descent
- Settings:
  - Standardize data using training data
  - Seed the rng to zero
  - Initialize each parameter to some random number
  - Use batch gradient descent
  - $\eta = 0.01$
  - Terminate when change in training RMSE < 2^{-32}

EXAM1	EXAM2	EXAM3	FINAL
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142
53	46	55	101
69	74	77	149
47	56	60	115
87	79	90	175
79	70	88	164
69	70	73	141
70	65	74	141
93	95	91	184
79	80	73	152
70	73	78	148
93	89	96	192
78	75	68	147
81	90	93	183
88	92	86	177
78	83	77	159
82	86	90	177
86	82	89	175
78	83	85	175
76	83	71	149
96	93	95	192

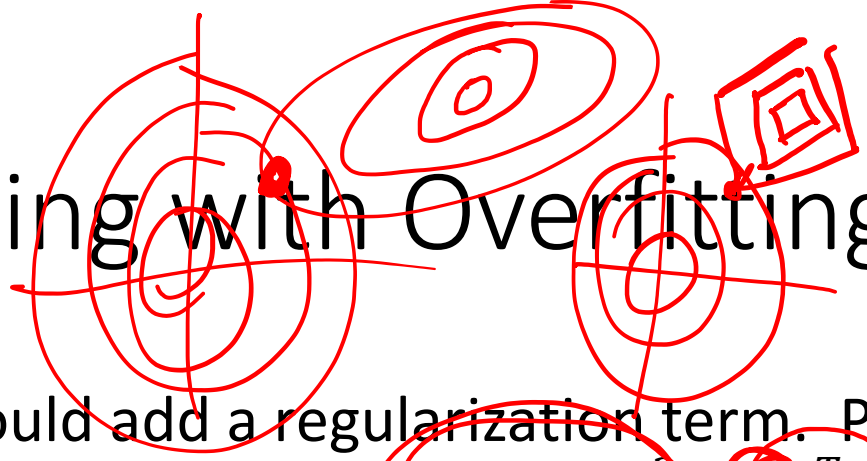
Approaches Compared

- Closed Form Linear Regression
  - Model: $\theta = [166.5294, 3.1218, 4.9823, 10.9629]^T$
  - RMSE: 2.8207
- Gradient Descent
  - Model: $\theta = [166.5294, 3.1231, 4.9815, 10.9623]^T$
  - RMSE Testing: 2.8207
  - Number of iterations: 5625

Approaches Compared



Dealing with Overfitting



- We could add a regularization term. Perhaps:

$$J = (Y - X\theta)^2 + \lambda\theta^T\theta$$

  - Therefore the larger θ is, the more it costs us
  - λ is a blending term to say how much this cost should contribute to the overall cost.
  - Then we can re-compute the closed form and/or the gradient using this equation.
  - Of course we'd have to somehow decide on λ! ☹
- If we're doing gradient descent, then we could try to halt the iterative training process by using a third data set, called a *validation set*
  - On each iteration we'll compute the validation error.
  - We'll stop when the validation error increases.

Beyond Linear Regression

- What if we want to find the parameters for a quadratic equation like $y = ax^2 + bx + c$?
- Imagine our observation just has a single feature, ie. $x = [x_1]$.
- We can write our quadratic equation as

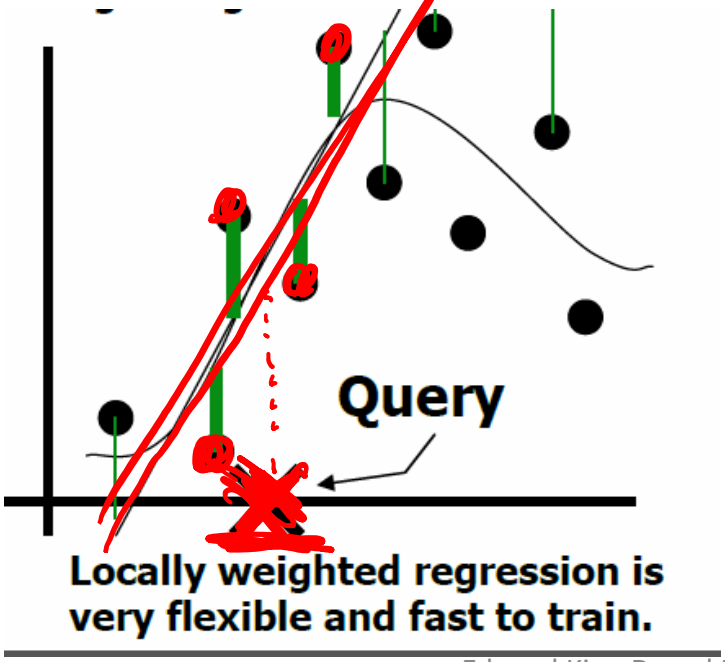
$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$$

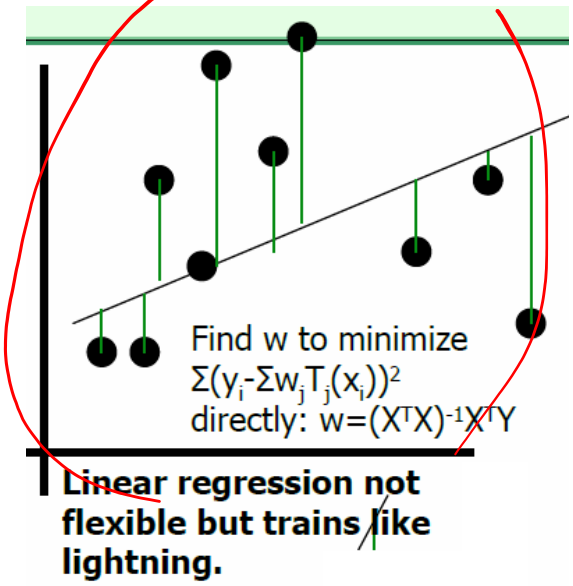
- If we re-write x to be $[1, x_1, x_1^2]$ then we can write this equation as $y = x\theta$ and we're right back at linear regression!

$(X^T X)^{-1} X^T y$

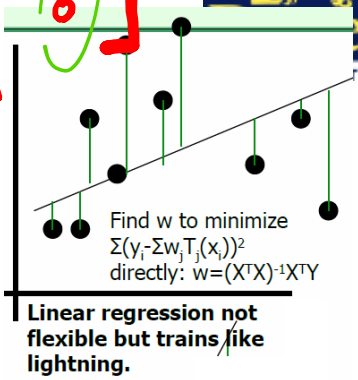
Locally Weighted Regression

- What we just did was *globally* linear regression
  - Since it's linear, it's not very flexible
- Locally weighted regression helps!



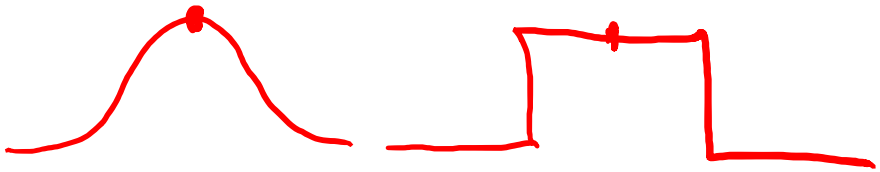


Locally Weighted Regression



- We need to make a few choices:

1. How many neighbors to allow to influence a test point?
   - All of them?



2. A way to give larger weights to closer points
   - A common way is to use the Gaussian similarity metric:

$$\beta(a,b) = e^{\frac{-d(a,b)}{k^2}}$$

   - Where $d(a,b)$ is the distance between a and b and k is some parameter affecting the drop-off rate.

$\sqrt{(a_1-b_1)^2+(a_2-b_2)^2\ldots(a_n-b_n)^2}$

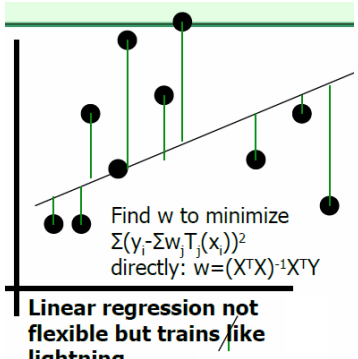
$|a_1-b_1|+|a_2-b_2|\ldots$

Locally Weighted Regression

- Then we need to find a local model:
  - Find θ that minimizes the locally weighted sum of squared residuals relative to sample x:

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{k \in training\ set} \beta(x, X_k)(Y_k - X_k\theta)^2$$

  - Where $\beta(x, X_k)$ is the **weight** of training point k to our query/test point $query$



Locally Weighted Regression

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{k \in training\ set} \beta(x, X_k)(Y_k - X_k\theta)^2$$

- To solve this, we can first put this equation in matrix format by creating a diagonal matrix of the weights:

$$W = Diag\big(\beta(x, X_1), \ldots, \beta(x, X_N)\big)$$

- Then we can write this as:

$$\theta = argmin_\theta\big((Y - X\theta)^T W (Y - X\theta)\big)$$

- Again taking the derivative, setting equal to zero, and solving for θ we get:

$$\theta = (X^T W X)^{-1} X^T W Y$$

Local Regression

- Let's use rows [1,3, 5, 6] for training and the rest for testing
- First I'll standardize the feature data and add a bias feature:
  - $X_{train} = \begin{bmatrix} 1 & -1.2402 \\ 1 & -0.3382 \\ 1 & 0.5637 \\ 1 & 1.0147 \end{bmatrix}$ $Y_{train} = \begin{bmatrix} 6 \\ 9 \\ 17 \\ 12 \end{bmatrix}$
  - $X_{test} = \begin{bmatrix} 1 & -0.7892 \\ 1 & 0.1127 \end{bmatrix}$ $Y_{test} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$

Data Points:

x	y
1	6
2	1
3	9
4	5
5	17
6	12

Local Regression

$$X_{train} = \begin{bmatrix} 1 & -1.2402 \\ 1 & -0.3382 \\ 1 & 0.5637 \\ 1 & 1.0147 \end{bmatrix} \quad Y_{train} = \begin{bmatrix} 6 \\ 9 \\ 17 \\ 12 \end{bmatrix}$$

$$X_{test} = \begin{bmatrix} 1 & -0.7892 \\ 1 & 0.1127 \end{bmatrix} \quad Y_{test} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

- Let's compute a local model for the first test sample using Gaussian similarity, a squared distance function, all the training data, and a fall-off factor of $k = 1$
- $x = [1, -0.7892]$
- $W = Diag\left(\begin{bmatrix} e^{-0.2034} \\ e^{-0.2034} \\ e^{-1.8305} \\ e^{-3.2542} \end{bmatrix}\right) = \begin{bmatrix} 0.8160 & 0 & 0 & 0 \\ 0 & 0.8160 & 0 & 0 \\ 0 & 0 & 0.1603 & 0 \\ 0 & 0 & 0 & 0.0386 \end{bmatrix}$
- $\theta = \left(X_{train}^T W X_{train}\right)^{-1} X_{train}^T W Y_{train} = \begin{bmatrix} 11.3051 \\ 4.5491 \end{bmatrix}$
- $\hat{y} = [1, -0.7892] \cdot \theta = 7.7148$
- $SE = (y - \hat{y}) = (1 - 7.718)^2 = 45.0884$

Local Regression

$$X_{train} = \begin{bmatrix} 1 & -1.2402 \\ 1 & -0.3382 \\ 1 & 0.5637 \\ 1 & 1.0147 \end{bmatrix} \quad Y_{train} = \begin{bmatrix} 6 \\ 9 \\ 17 \\ 12 \end{bmatrix}$$

$$X_{test} = \begin{bmatrix} 1 & -0.7892 \\ 1 & 0.1127 \end{bmatrix} \quad Y_{test} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

- Let's compute a local model for the **second** test sample
- $x = [1, 0.1127]$
- $W = \begin{bmatrix} 0.1603 & 0 & 0 & 0 \\ 0 & 0.8160 & 0 & 0 \\ 0 & 0 & 0.8160 & 0 \\ 0 & 0 & 0 & 0.4433 \end{bmatrix}$
- $\theta = \left(X_{train}^T W X_{train}\right)^{-1} X_{train}^T W Y_{train} = \begin{bmatrix} 11.46 \\ 4.3155 \end{bmatrix}$
- $\hat{y} = [1\ , 0.1127] \cdot \theta = 11.9466$
- $SE = 48.2552$
- RMSE (over these two test samples)$= 6.8317$