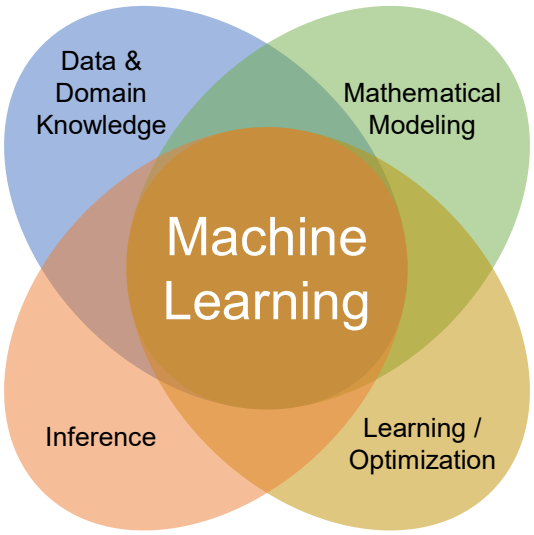
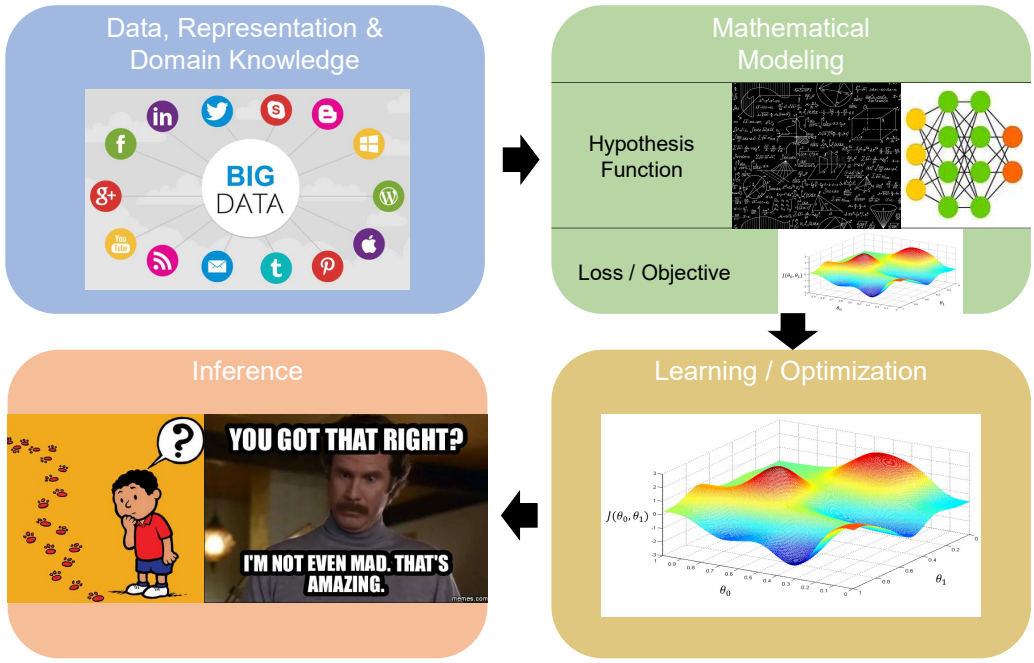
Linear Models for Regression

Kai-Lung Hua (花凱龍)

Basic Machine Learning Recipe / Pipeline





3 of 91

Data, Representation & Domain Knowledge

4 of 91

Linear Regression

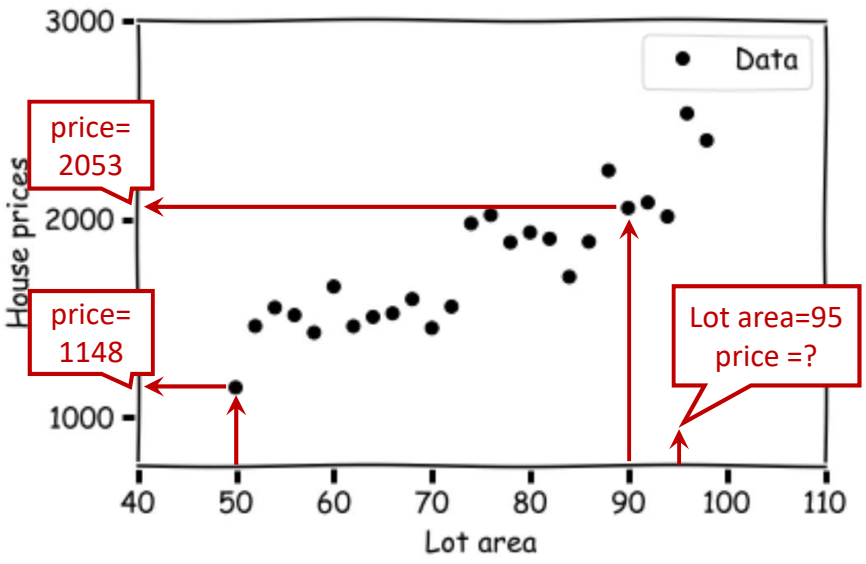
Similar to k-NN, we assume that our input data is a vector $x \in \mathbb{R}^D$.

We can express the whole dataset as a matrix $X \in \mathbb{R}^{N \times D}$

Since this is a regression task, our target variable is a continuous scalar value $y \in \mathbb{R}$.

5 of 91

Lot area	House prices
50	1148
52	1458
54	1551
56	1513
58	1425
60	1657
62	1457
64	1504
66	1522
68	1594
70	1448
72	1556
74	1975



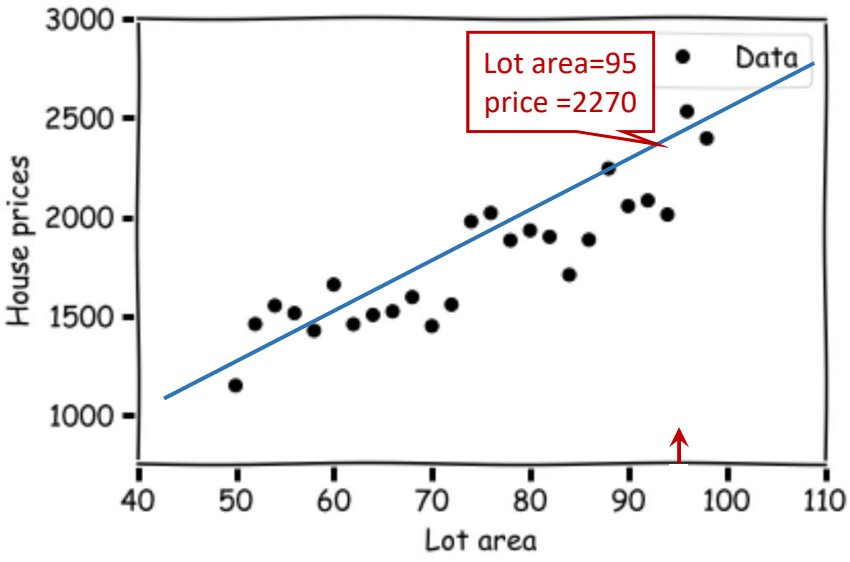
6 of 91

Mathematical Modeling

Our Hypothesis

Lot area	House prices
50	1148
52	1458
54	1551
56	1513
58	1425
60	1657
62	1457
64	1504
66	1522
68	1594
70	1448
72	1556
74	1975

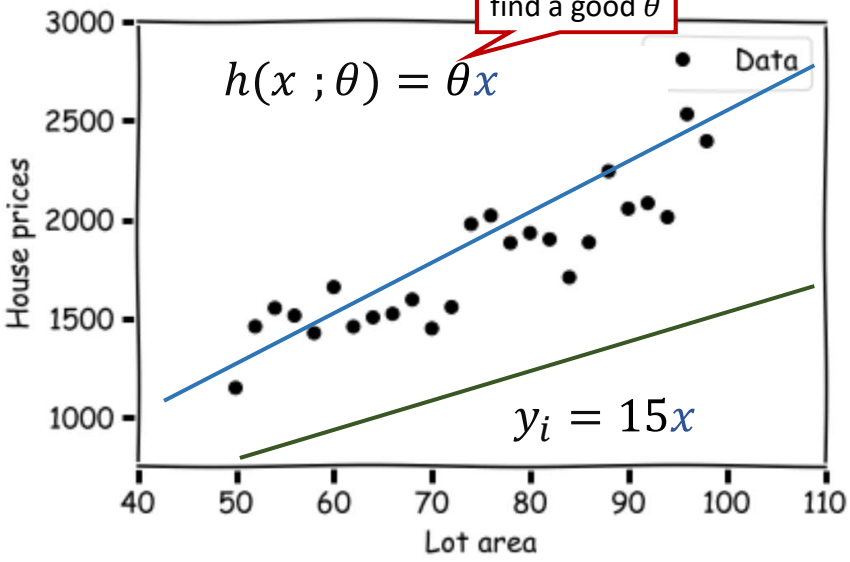
Hypothesis function



Lot area	House prices
50	1148
52	1458
54	1551
56	1513
58	1425
60	1657
62	1457
64	1504
66	1522
68	1594
70	1448
72	1556
74	1975

Hypothesis function

We need to find a good θ



9 of 91

Linear Regression

For linear regression, we assume that the input x and the output y are related with a linear function with some noise as shown in the equation below

$$y = h(x;\theta) + \epsilon$$

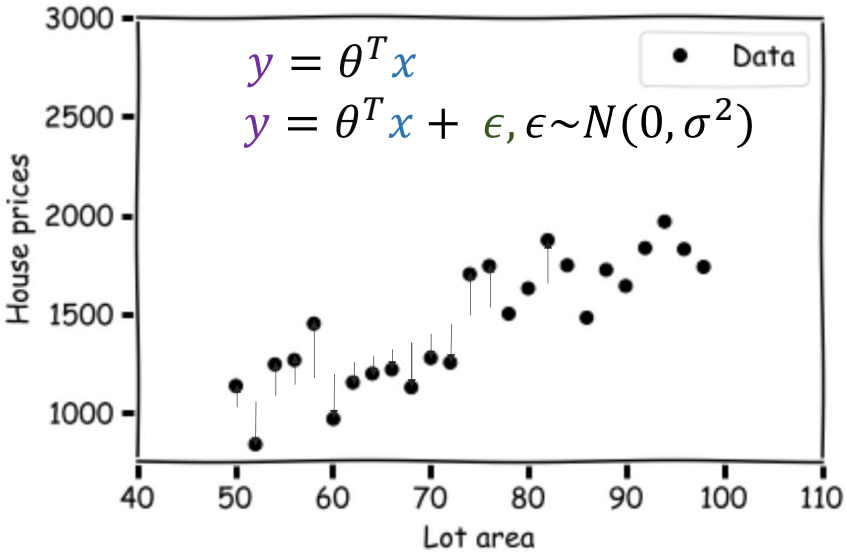
$$h(x;\theta) = \sum_{i=1}^{D} \theta^j x^j = \theta^T x$$

x_i^j **Superscript** – denotes feature dimension j
Subscript – denotes sample i

where $h(x;\theta)$ is our hypothesis function, $\theta \in \mathbb{R}^D$ are the parameters (weights) and $\epsilon \sim N(0,\sigma^2) \in \mathbb{R}^D$ represents measurement error or some other random noise.

10 of 91

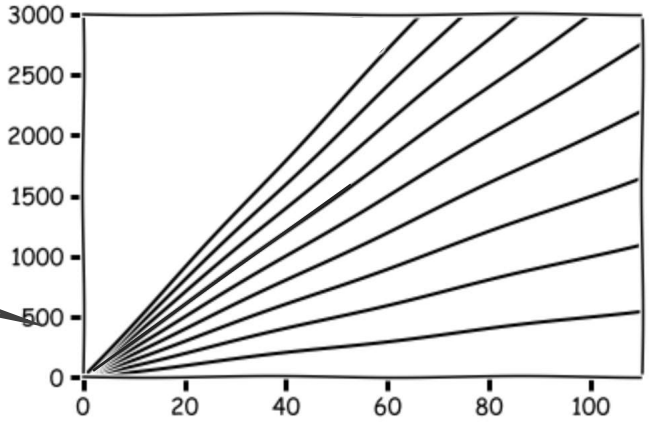
Lot area	House prices	ϵ
50	1000	-47
52	1040	-376
54	1080	193
56	1120	-293
58	1160	383
60	1200	-73
62	1240	363
64	1280	-502
66	1320	312
68	1360	473
70	1400	-3
72	1440	-433
74	1480	-43
76	1520	-609
78	1560	-419
80	1600	-423
82	1640	-325
84	1680	143
86	1720	-483
88	1760	701
90	1800	-497



Isn't a line (1D) / hyperplane $y = \theta^T x + b$?

Where did the b go?

$y = \theta^T x$



Isn't a line (1D) / hyperplane $y = \theta^T x + b$?

Where did the b go?

$$\theta = [\theta_1, \theta_2, \dots \theta_D]$$
$$x = [x_1, x_2, \dots, x_D]$$
$$b \in \mathbb{R}$$
$$\theta^T x = \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_D x_D$$
$$\theta^T x + b = b + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_D x_D$$

If we add another element $\theta_0 = b$ and $x_0 = 1$, we can express this as

$$\theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_D x_D$$
$$= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_D x_D$$
$$= b + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_D x_D$$

13 of 91

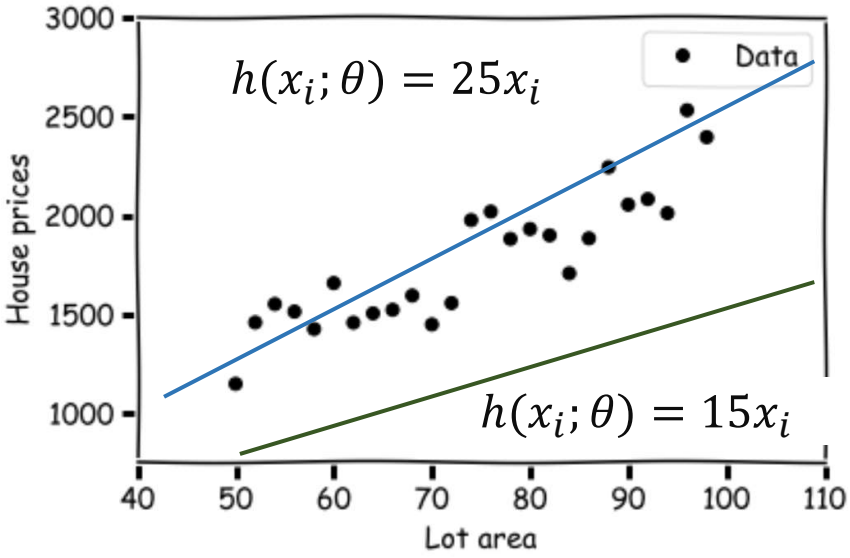
Mathematical Modeling

The Objective

14 of 91

Lot area	House prices
50	1148
52	1458
54	1551
56	1513
58	1425
60	1657
62	1457
64	1504
66	1522
68	1594
70	1448
72	1556
74	1975

Remember this?

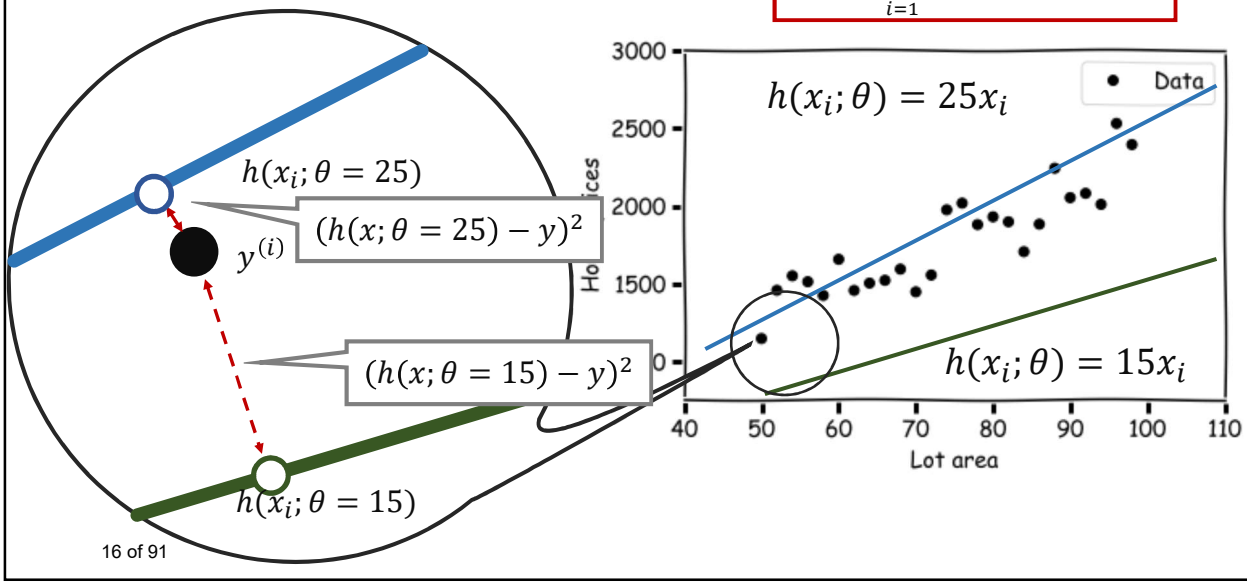


15 of 91

Objective/Loss function

How is $\theta = 25$ better than $\theta = 15$?

$$\frac{1}{2N}\sum_{i=1}^{N}(h(x_i;\theta) - y)^2$$



16 of 91

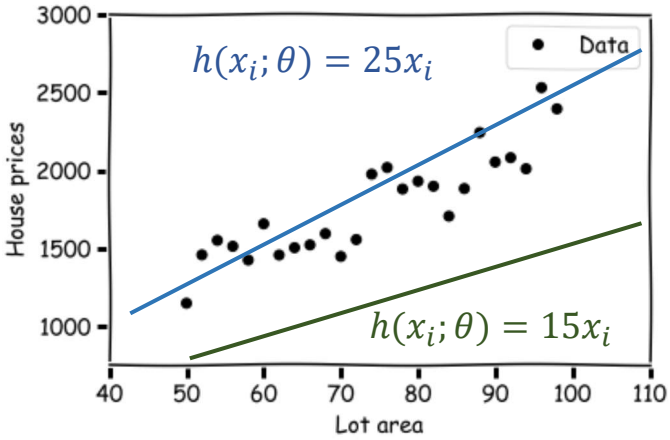
Objective/Loss function

Will explain this later

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(h(x_i;\theta) - y)^2$$

$$L(\theta = 25) = 16463$$
$$L(\theta = 15) = 237255$$

$$\underset{\theta}{\operatorname{argmin}}\, L(\theta)$$



17 of 91

Linear Regression

- Why Euclidean Loss?
  - It has nice probabilistic interpretations (Maximum Likelihood Estimation)

18 of 91

Derivations

Recall:

(random) (random / stochastic)

$$y = \theta^T x + \epsilon, \epsilon \sim N(0, \sigma^2)$$

(deterministic / "constant")

So what will be the distribution of y?

$$y \sim N(\theta^T x, \sigma^2)$$

19 of 91

Derivations MLE to Euclidean Loss

$$P(y_i|x_i;\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_i-\theta^T x_i}{\sigma^2}\right)^2}$$

$$\text{likelihood } \mathcal{L}(\theta) = P(y_1, y_2, \dots y_N|x_1, x_2, \dots, x_N;\theta) = \prod_{i=1}^{N} P(y_i|x_i;\theta)$$

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{N} \log P(y_i|x_i;\theta) = \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2\sigma^2}(y_i - \theta^T x_i)^2$$

$$\approx -\sum_{i=1}^{N} (y_i - \theta^T x_i)^2$$

$$\underset{\theta}{\text{argmax}} \log \mathcal{L}(\theta) = \underset{\theta}{\text{argmin}} - \log \mathcal{L}(\theta)$$

$$\underset{\theta}{\text{argmin}} - \log \mathcal{L}(\theta) = \underset{\theta}{\text{argmin}} \sum_{i=1}^{N} (y_i - \theta^T x_i)^2$$ Least Squares Objective

Additive constants and positive multiplicative constants do not affect the optimization problem

20 of 91

Optimization

Learning the optimal parameters for our model

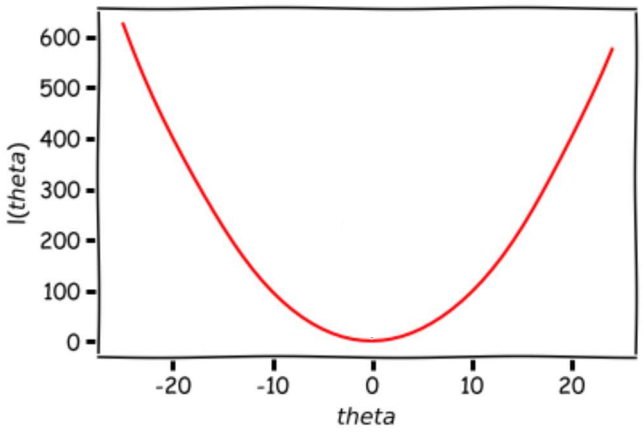
Linear Regression

Hypothesis function

$h(x_i; \theta) = \theta^T x_i$

Objective function

$$L(\theta) == \frac{1}{2N} \sum_{i=1}^{N} (h(x_i; \theta) - y)^2$$

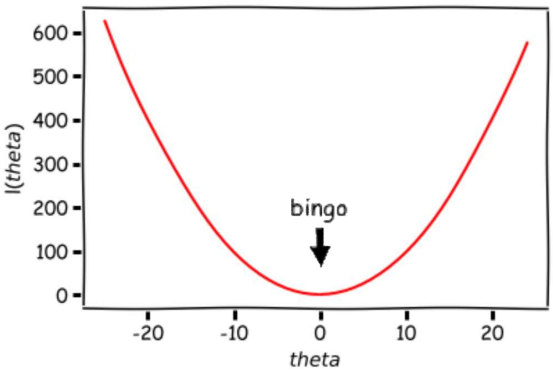


How can we get a θ that will yield a low $L(\theta)$?

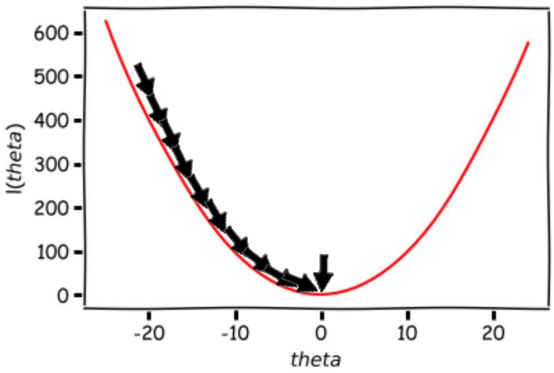
Which part of the graph yields the lowest $l(\theta)$/loss?

Two approaches to get to minima

Analytical Solution



Gradient Descent



23 of 91

Analytical Solution

Normal equation

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

Using this equation we can solve for the optimal θ directly

Problem! This is an incredibly expensive computation!

Inverse operation is approximately $O(N^3)$

$X^T X$ uses all of your data which may not fit on your memory if data is large

24 of 91

Derivation of the normal equation

From single instance to batch

$\theta^T x_i$ → $X\theta$

$(1 \times D)$ $(D \times 1)$ $(N \times D)$ $(D \times 1)$

Scalar output

Vector output
$(N \times 1)$

25 of 91

Derivation of the normal equation

Least Squares Objective

$$\sum_{i=1}^{N} (y_i - \theta^T x_i)^2 = (X\theta - y)^T (X\theta - y)$$

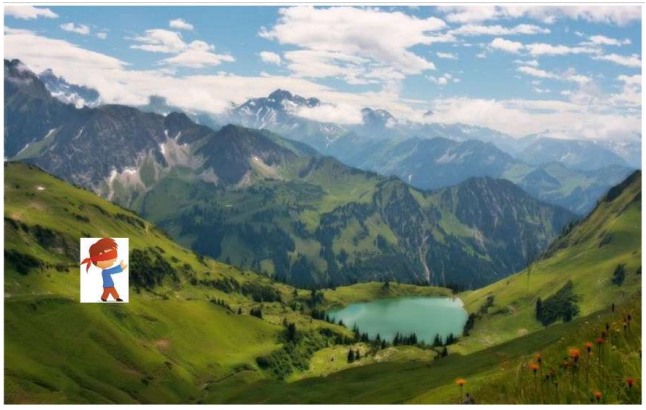
To optimize we get the derivative with respect to θ and set to 0.

$$\frac{\partial}{\partial \theta} (X\theta - y)^T (X\theta - y) = 0$$
$$2X^T(X\theta - y) = 0$$
$$X^T(X\theta - y) = 0$$
$$X^T X\theta - X^T y = 0$$
$$X^T X\theta = X^T y$$
$$\theta = (X^T X)^{-1} X^T y$$

26 of 91

Gradient Descent Solution

Idea: Start with a random guess then iteratively improve your solution by stepping towards the opposite direction of the gradient

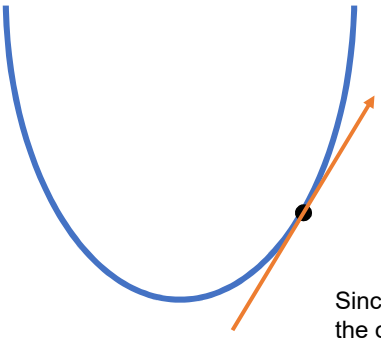


27 of 91

Image Credit: Fei-Fei Li, Andrej Karpathy, Justin Johnson - CS231N

Gradients

Why do we want to go to the opposite direction of the gradient?



Gradient - “slope of the tangent line”
- direction of steepest ascent

Since we are minimizing, we want to go to the opposite (negative) direction

28 of 91

Gradient Descent

procedure gradient_descent(X, θ_i):
initialize θ randomly
while not converged do:
$\theta := \theta - \alpha \frac{\partial}{\partial \theta} L(\theta)$
return θ

α is the learning rate, determines how large the update will be

$\frac{\partial}{\partial \theta_i} L(\theta)$ is the gradient of the loss with respect to θ

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h(x_i;\theta) - y_i)^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}(\theta^T x_i - y_i)^2$$

$$\frac{\partial}{\partial \theta} L(\theta) = \frac{\partial}{\partial \theta}\frac{1}{2}\sum_{i=1}^{n}(\theta^T x_i - y_i)^2$$

$$= 2 * \frac{1}{2}\sum_{i=1}^{n}(\theta^T x_i - y_i)\frac{\partial}{\partial \theta_j}(\theta^T x_i - y_i)$$

$$= 1\sum_{i=1}^{n}(\theta^T x_i - y_i)\, x_i$$

$$= \sum_{i=1}^{n}(\theta^T x_i - y_i)\, x_i$$

Gradient Descent

$$\underset{\theta}{\operatorname{argmin}}\, L(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(h(x_i;\theta) - y_i)^2$$

So the loss doesn't scale with the size of the batch

Gradient Descent	Stochastic GD	Mini-Batch GD
Updates θ based on gradient of the **whole dataset**	Updates θ based on gradient of **one data instance**	Updates θ based on gradient of a **subset of the whole dataset**
Runs slow if dataset is very large	Runs fast, but jittery	Runs faster than GD, and path is not as jittery as stochastic GD
1 iter = N instances N = size of whole dataset	1 iter = 1 instance	1 iter = n instances, n = size of batch

$$\theta := \theta - \alpha\underbrace{(\theta^T x_1 - y_1)\, x_1}$$

Assuming $x \geq 0$

1 If $(\theta^T x_1 - y_1)x_1 > 0,\ \theta$ will go down

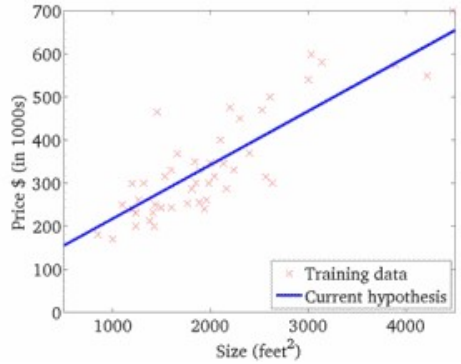
2 If $(\theta^T x_1 - y_i)x_1 = 0,\ \theta$ will remain the same

3 If $(\theta^T x_1 - y_1)x_1 < 0,\ \theta$ will go up

31 of 91

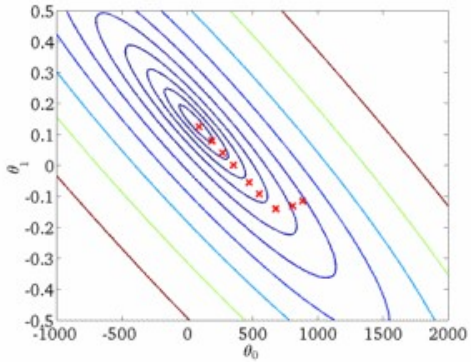
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

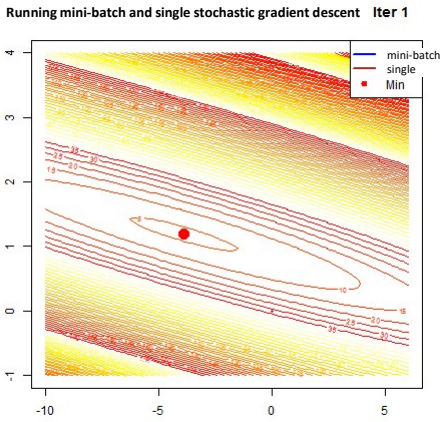


$L(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



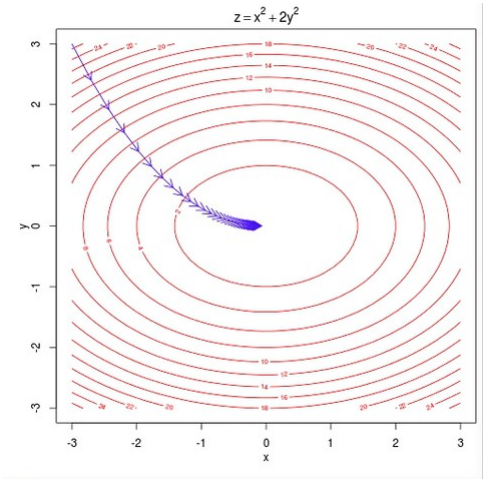
32 of 91



33 of 91

Gradient Descent

- Pros
  - It is very simple and quite effective in many ML tasks
  - Very scalable
- Cons
  - Only applies to smooth/differentiable functions
  - Can get stuck in local minimum



34 of 91

Inference

Predict the most likely values for new data

35 of 91

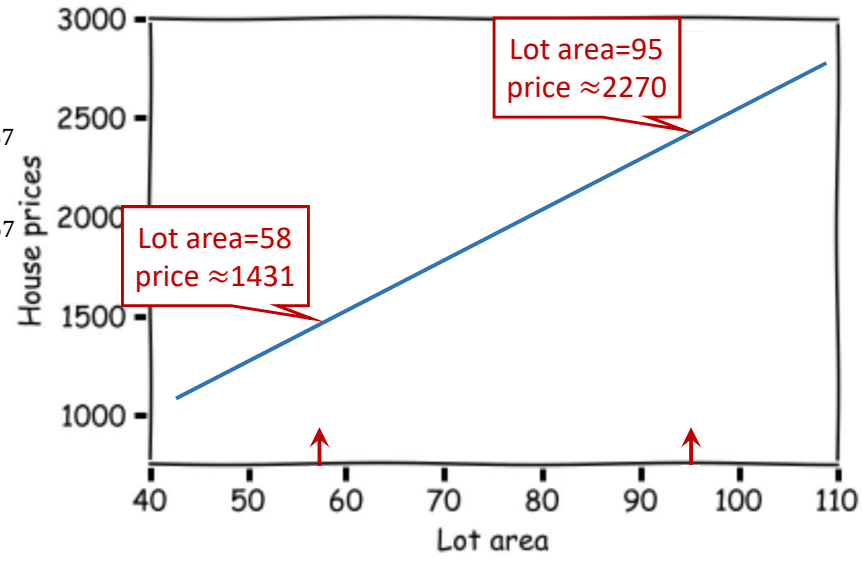
Suppose these are the learned parameters

$$\theta = \begin{bmatrix} 115.81081 \\ 22.67567 \end{bmatrix}$$

$\theta^T x_1 = 115.81081 + 95 * 22.67567$
$= 2269.99$

$\theta^T x_2 = 115.81081 + 58 * 22.67567$
$= 1430.99$

Lot area	House prices
95	?
58	?



36 of 91

Disadvantages of LR

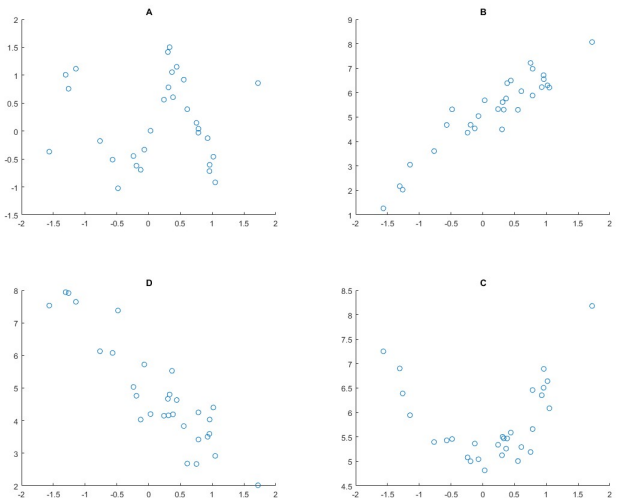
- Limited to Linear function / Linear Relationships
- Assumes data is independent and identically distributed (i.i.d.)

Advantages of LR

- Relatively fast to train
- Fast test time (just plug in to the equation)
- Do not need to store all the data just the weights (in contrast to kNN)

37 of 91

Which of these data will Linear Regression be suitable with?

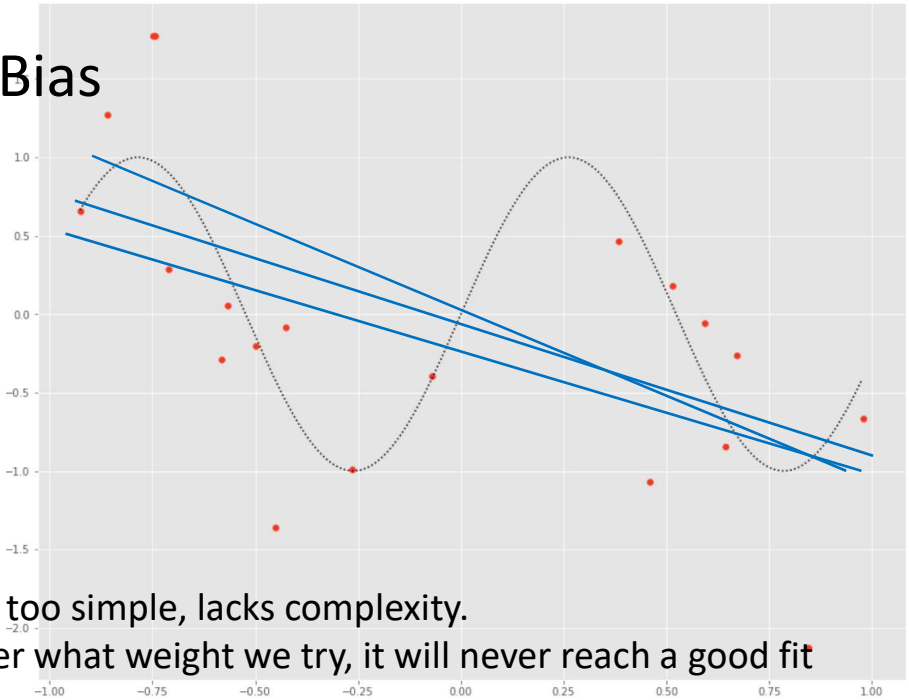


38 of 91

How do we generate non-linear regression curves / decision boundaries?

And, why do we want to do that anyway?

High Bias



Model is too simple, lacks complexity.

No matter what weight we try, it will never reach a good fit

How do we generate non-linear regression curves / decision boundaries?

Solution: Transform features

By Linear Models, we mean that the hypothesis function $h(x;\theta)$ is a linear function of the parameters θ, not the input vector x.

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

We don't define linear by the requirement that the graph is a line.

As long as these functions can be directly computed from the observed values and the **parameters are still linear in the data**, then the problem is still a Linear Regression problem.

This is related to the kernel trick that is usually used for SVM.

41 of 91

How do we generate non-linear regression curves / decision boundaries?

So far we only used the original observed values / features $[x_1, x_2, \ldots x_D]$. However, Linear Regression can be applied in the same way to **functions of these values.**

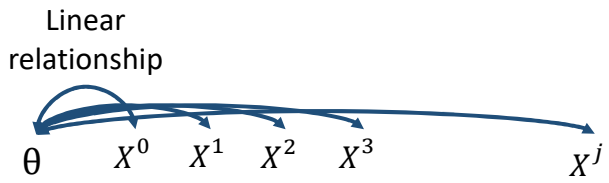
$$\phi(x) = [x_1, x_2, \ldots x_D, \quad x_1x_2, \quad x_1^2]$$

Common feature transforms $\phi(x)$:

- Polynomial
- Gaussian
- Exponential

42 of 91

Let's do Polynomial Feature Transform with order j.



$\theta \in \mathbb{R}^{D\times 1}$

Append θ to accommodate more features

θ_0	1	1	1	1	...	x_0^j
θ_1	1	3	9	27		x_1^j
	1	4	16	64		x_2^j
	1	6	36	216		x_3^j
	1	7	49	343		x_4^j
	1	8	64	512		x_5^j

$X \in \mathbb{R}^{N\times D}$

Append *X* in the *D dimension*
With feature transformed X

43 of 91

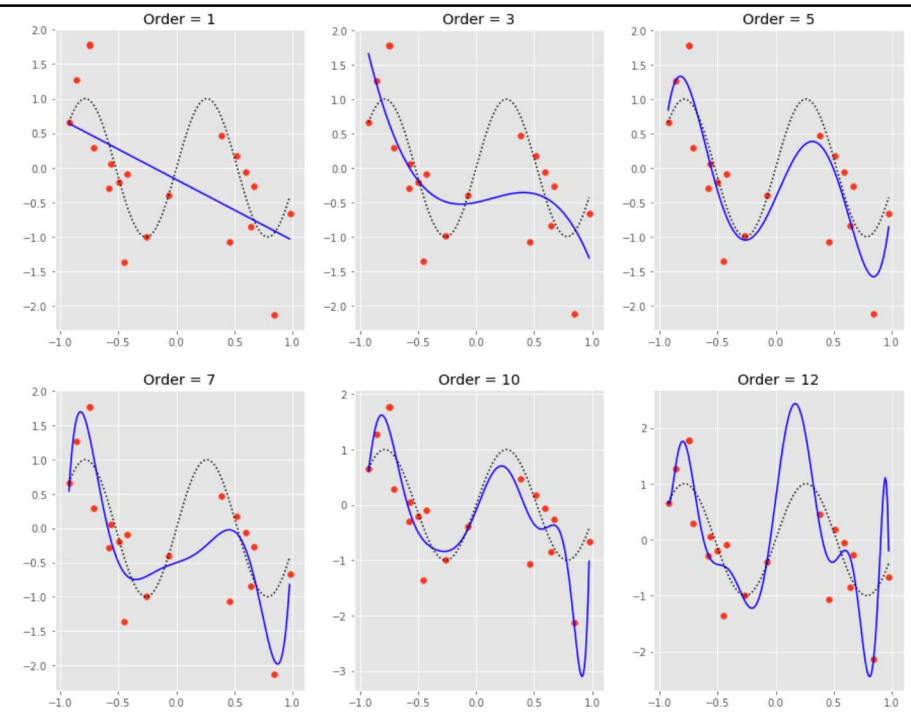
How do we generate non-linear regression curves / decision boundaries?

Before:

After:

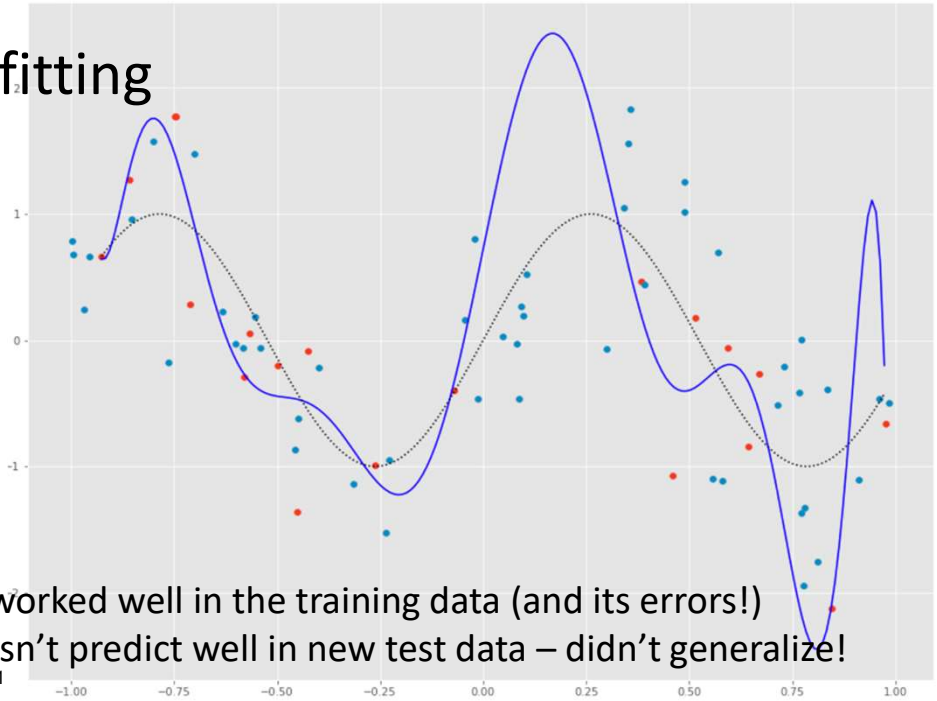
Everything else in the procedure is the same except we replace the original input x with its transformed version $\phi(x)$.

44 of 91



45 of 91

Overfitting



Model worked well in the training data (and its errors!)
but doesn't predict well in new test data – didn't generalize!

46 of 91

Anatomy of the error of an estimator

Anatomy of the error of an estimator

- $y = f(x) + \epsilon$ and $\epsilon \sim N(0, \sigma^2)$

For linear regression, we assume $h(x) = \theta^T x$

- $\underset{\theta}{\operatorname{argmin}}\ L(\theta) = \sum_{i=1}^{N} (h(x_i; \theta) - y)^2$

Possible Sources Errors:

- (bias) Systematic prediction error
- Variance
- Irreducible error (noise)

Anatomy of the error of an estimator

- Assuming that the data points are drawn independently and identically distributed (i.i.d.) from a unique underlying probability distribution.
- Suppose we have a model trained on training data X. Given a new test data point x, what is the expected prediction error?

$$E\left[\left(y - h(x)\right)^2 \mid X\right]$$

- We will decompose this expectation into three components that characterizes our errors.

49 of 91

Recall: Statistics

- Let X be a random variable with possible values $x_i, i = 1 \ldots N$ and with probability distribution $P(X)$.
- The **expected value** or **mean** of X is

$$E[X] = \sum_{i=1}^{N} x_i P(x_i)$$

The **variance** of X is:

$$\begin{aligned} Var[X] &= E[(X - E[X])^2] \\ &= E[X^2 - 2XE[X] + E[X]^2] \\ &= E[X^2] - 2E[X]E[X] + E[X]^2 \\ &= E[X^2] - 2E[X]^2 + E[X]^2 \\ &= E[X^2] - (E[X])^2 \end{aligned}$$

50 of 91

Bias-Variance Decomposition

$$E\left[\left(y-h(x)\right)^2\middle|X\right] = E\left[\left(h(x)\right)^2 - 2yh(x) + y^2\middle|X\right]$$
$$= E[(h(x)^2|X] + E[y^2|X] - 2E[y|X]E[h(x)|X]$$

$E[h(x)\|X] = \overline{h(x)}$	denotes the mean prediction of the hypothesis at x.
$E[y\|X] = E[f(x)+\epsilon\|X]$ $= E[f(x)] + E[\epsilon\|X]$ $= f(x)$	Linearity of expectation $\epsilon \sim N(0,\sigma^2)$
$E[h(x)^2\|X] = E\left[\left(h(x)-\overline{h(x)}\right)^2\|X\right] + \overline{h(x)}^2$	Based on $E[X^2] = Var[X] + (E[X])^2$
$E[y^2\|X] = E\left[\left(y-f(x)\right)^2\middle\|X\right] + \left(f(x)\right)^2$	Based on $E[X^2] = Var[X] + (E[X])^2$

51 of 91

Substituting back everything

$$E\left[\left(y-h(x)\right)^2\middle|X\right] = E\left[\left(h(x)-\overline{h(x)}\right)^2|X\right] + \overline{h(x)}^2 - 2f(x)\overline{h(x)} + \left(f(x)\right)^2 + E\left[\left(y-f(x)\right)^2\middle|X\right]$$
$$= E\left[\left(h(x)-\overline{h(x)}\right)^2|X\right] + (\overline{h(x)} - f(x))^2 + E\left[\left(y-f(x)\right)^2\middle|X\right]$$

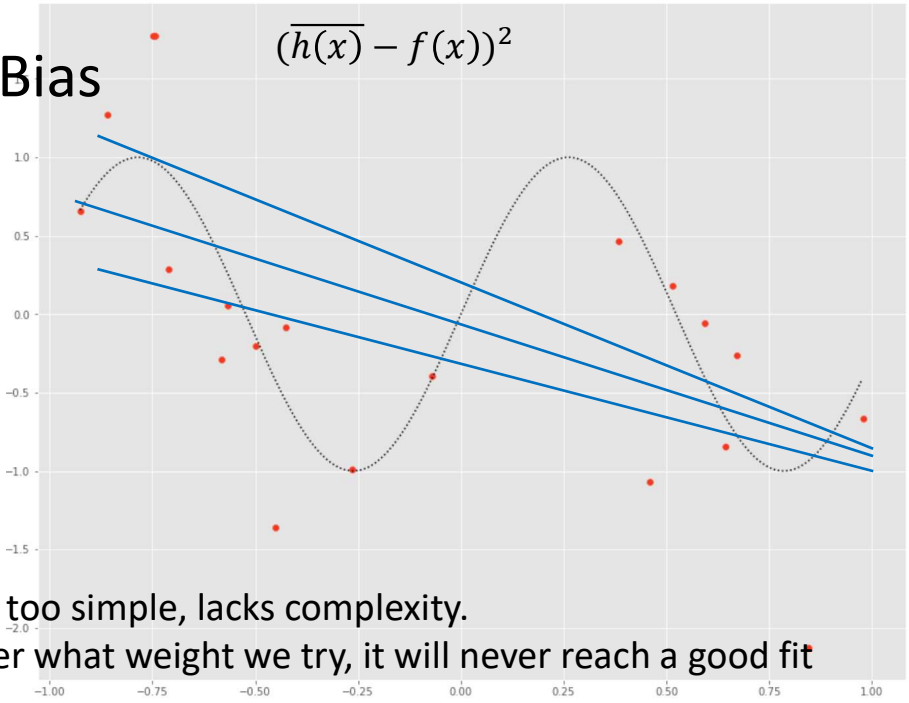
$E\left[\left(h(x)-\overline{h(x)}\right)^2\|X\right]$	Variance of the hypothesis
$(\overline{h(x)} - f(x))^2$	Squared bias (or systematic error) associated with the class of hypothesis we are considering
$E\left[\left(y-f(x)\right)^2\middle\|X\right]$ $y = f(x)+\epsilon$	Irreducible noise / error (cannot be avoided)

These are the three components that characterizes our errors.

52 of 91

High Bias

$$(\overline{h(x)} - f(x))^2$$

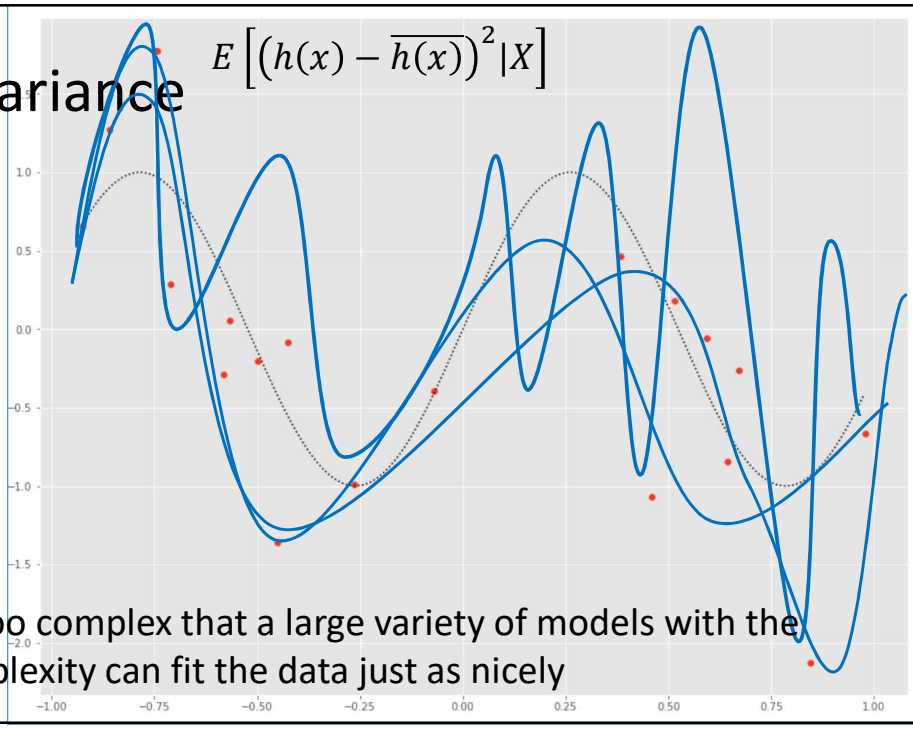


Model is too simple, lacks complexity.
No matter what weight we try, it will never reach a good fit

53 of 91

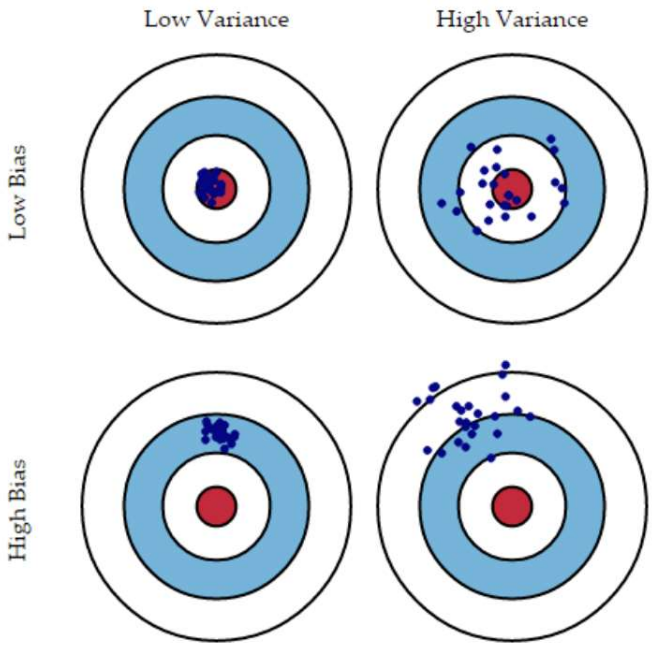
High Variance

$$E\left[\left(h(x) - \overline{h(x)}\right)^2 | X\right]$$



Model is too complex that a large variety of models with the same complexity can fit the data just as nicely

54 of 91



55 of 91

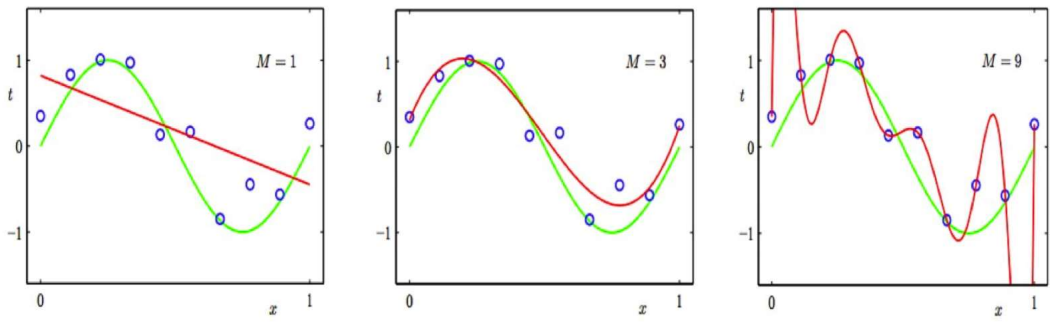
	Training error	Test error
Underfit model	HIGH	HIGH
Overfit model	LOW	HIGH

56 of 91

Problem with complex functions.. Overfitting!

- More complex functions have more expressive power and can fit your data better. However, they fit the noise better as well. **Overfitting** is when we are fitting to the noise rather than the general trends of the data.
- Typical **overfitting** means that error on the training data is very low, but error on new instance is high.
- Typical **underfitting** means that error on the training data is very high.

57 of 91

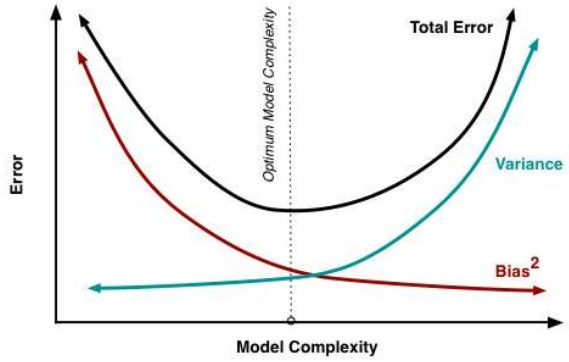


predictor too inflexible: cannot capture pattern

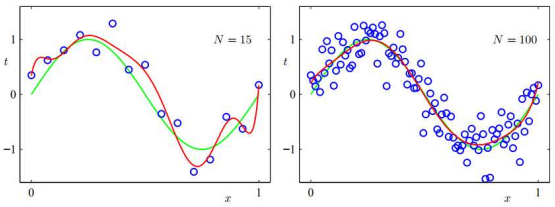
predictor too flexible: fits noise in the data

58 of 91

Bias-Variance Trade-off

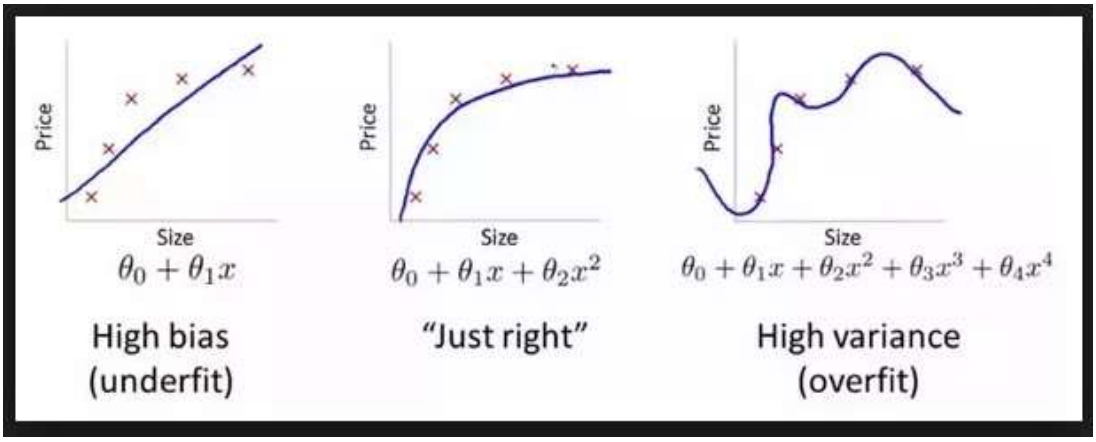


Hypothesis complexity (decreasing left-to-right)



59 of 91

Bias-Variance Trade-off



60 of 91

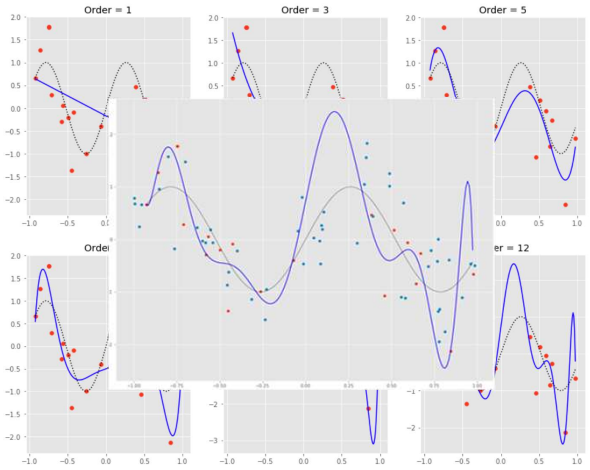
What can we do?

Definitely not underfit!

Stop before it gets too weird!
(Stop at order = 5)

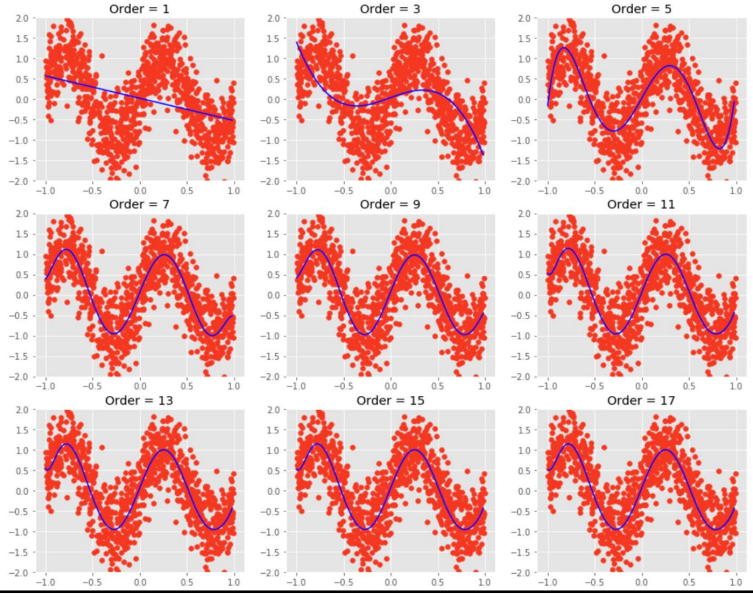
Get more data!
(This way we just overfit on all of them)

Make the algo more robust to overfitting.
(But … how?)



61 of 91

If we had more data..



62 of 91

Can we reduce variance (i.e. overfitting) without collecting more data or stopping the optimization early?

Yes! Add Regularization!

63 of 91

What happens when we increase the degree

poly degree	weight_1	weight_2	weight_3	weight_4	weight_5
1	-0.87	-0.17			
3	-2.15	0.89	0.34	-0.50	
5	13.40	-1.31	-17.28	1.36	3.79
7	31.30	-5.53	-37.29	4.48	6.87
10	179.74	94.89	-378.87	-185.43	259.32
12	-540.45	-1015.67	1487.13	2623.60	-1490.51
30	-14156072.98	16329720.76	21300299.24	-14967132.99	13066492.56

Big weights mean feature is important, but these are too large!

64 of 91

$$\underset{\theta}{\operatorname{argmin}} L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2 + R(\theta)$$

+ a high number if weights are far from 0

$$\underset{\theta}{\operatorname{argmin}} L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2 + \lambda\sum_{j=1}^{D}\theta^2$$

or

$$\underset{\theta}{\operatorname{argmin}} L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2 + \lambda\sum_{j=1}^{D}|\theta|$$

65 of 91

$$\underset{\theta}{\operatorname{argmin}} L(\theta) = \underbrace{\frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2}_{\text{Least Squares error}} + \underbrace{\lambda R(\theta)}_{\text{Regularization error}}$$

Regularization error
θ near 0 is better

Contradicting each other. They must find a balance!

Can you think of how λ will affect this?

66 of 91

What does lambda have to do with it?

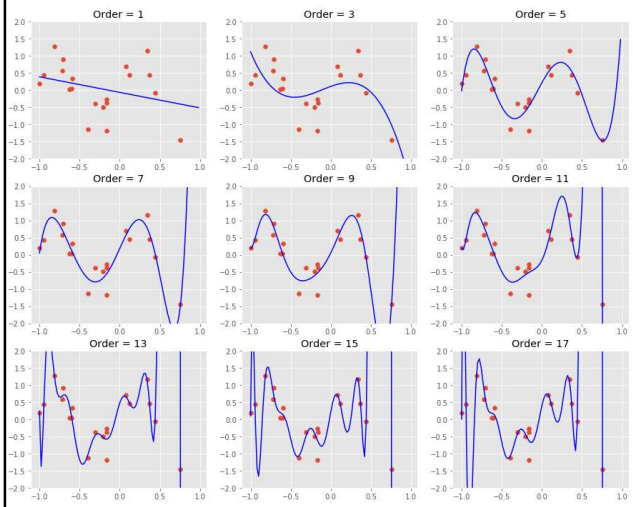
$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y)^2 + \frac{1}{2}\lambda||\theta||_2^2$$

If λ is large (e.g. 1000)	Least Squares error has little impact on loss
If λ is 0	No effect, it's like removing regularization
If λ is negative	The higher the weights, the lower the loss!

Need to find λ that is not too restrictive (large) and not too flexible (very small)

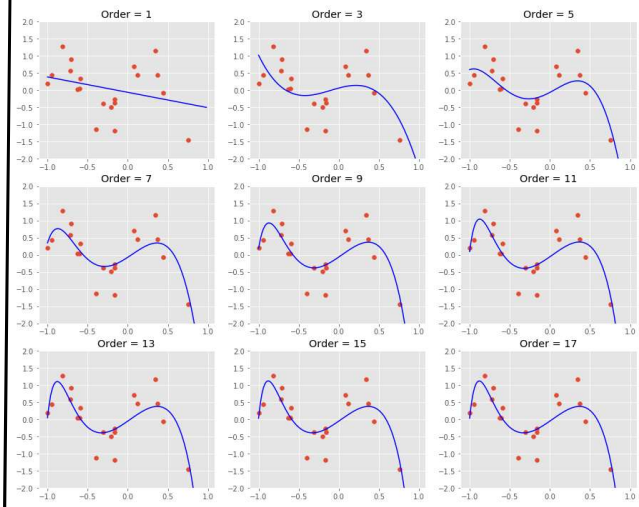
67 of 91

Without Regularization



68 of 91

With Regularization



Two common regularization methods

Ridge Regression

L2 Regularization

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y)^2 + \frac{1}{2}\lambda||\theta||_2^2$$

Lasso Regression

L1 Regularization

$$L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y)^2 + \lambda||\theta||_1$$

69 of 91

Remember this?

poly degree	weight_1	weight_2	weight_3	weight_4	weight_5
1	-0.87	-0.17			
3	-2.15	0.89	0.34	-0.50	
5	13.40	-1.31	-17.28	1.36	3.79
7	31.30	-5.53	-37.29	4.48	6.87
10	179.74	94.89	-378.87	-185.43	259.32
12	-540.45	-1015.67	1487.13	2623.60	-1490.51
30	-14156072.98	16329720.76	21300299.24	-14967132.99	13066492.56

70 of 91

This is what happens with Ridge (L_2)

poly degree	weight_1	weight_2	weight_3	weight_4	weight_5	weight_6	weight_7	weight_8	weight_9	weight_10
1	-0.44									
3	0.17	0.22	-0.27							
5	1.90	0.34	-2.16	-0.08	0.42					
7	0.30	0.03	0.84	0.28	-1.16	-0.09	0.24			
10	1.78	-0.46	-3.06	-0.59	2.11	1.35	-0.82	-0.61	0.12	0.08
12	1.95	0.06	-3.43	-1.35	2.22	1.31	-0.70	-0.15	0.05	-0.12
20	2.78	-1.04	-6.02	1.21	4.23	-0.73	-0.68	0.15	-0.27	0.06

Small weights!

71 of 91

This is what happens with Lasso (L_1)

poly degree	weight_1	weight_2	weight_3	weight_4	weight_5	weight_6	weight_7	weight_8	weight_9	weight_10
1	-0.44									
3	0.17	0.22	-0.27							
5	1.90	0.34	-2.16	-0.08	0.42	0.00				
7	0.29	0.03	0.85	0.28	-1.16	-0.09	0.24			
10	2.44	0.57	-4.84	-3.22	3.63	3.49	-1.32	-1.30	0.17	0.16
12	4.99	-1.21	-15.19	0.00	18.06	0.60	-10.16	0.00	2.61	-0.13
20	3.97	-1.58	-9.05	-0.03	5.82	2.95	0.00	-2.38	-0.77	0.26

Also small weights! And some of them are zero! What do zero weights imply?

72 of 91

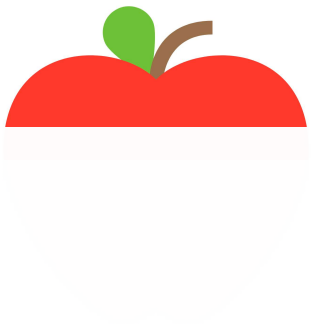
What's with this behavior?

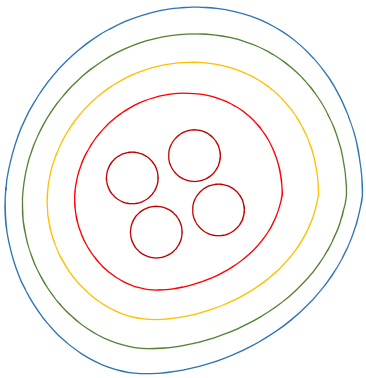
Why does Lasso zero out weights, while Ridge does not?

73 of 91

Watch this video for contour plots

https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/visualizing-scalar-valued-functions/v/contour-plots

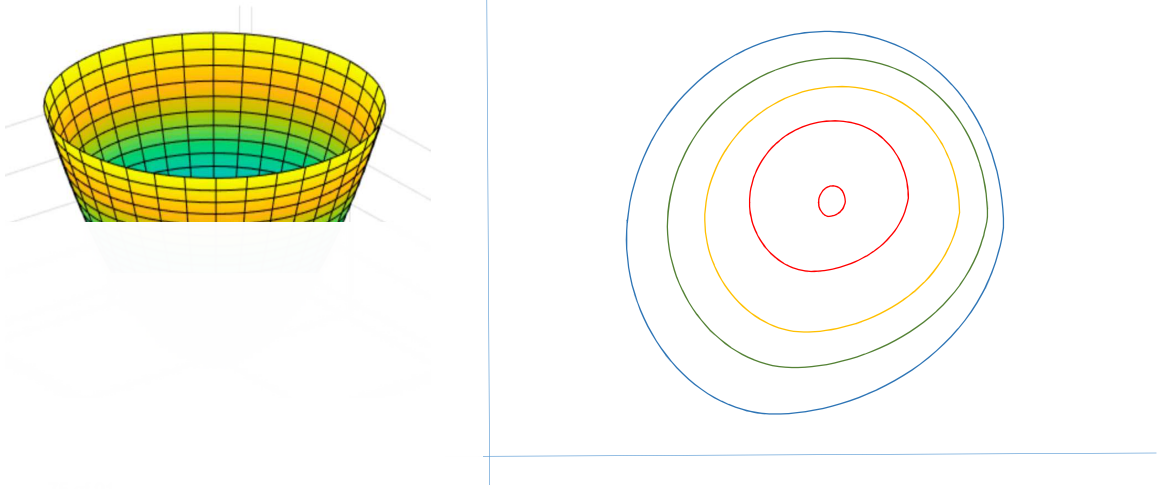




74 of 91

Watch this video for contour plots

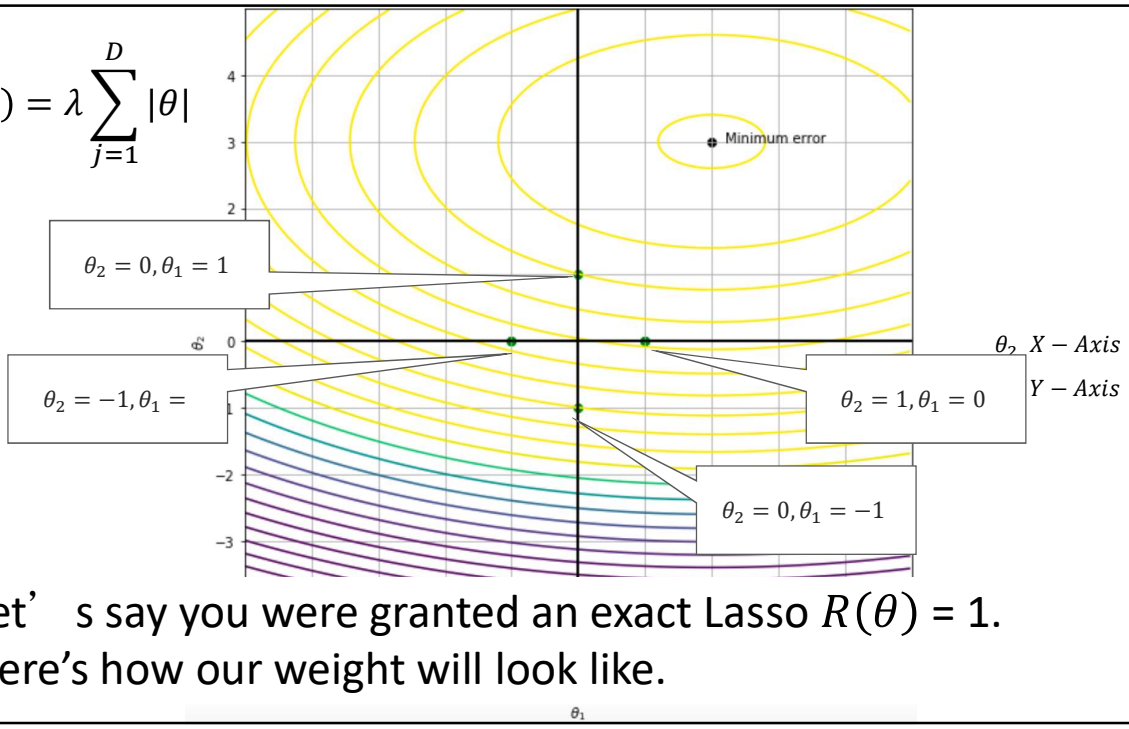
https://www.khanacademy.org/math/multivariable-calculus/thinking-about-multivariable-function/visualizing-scalar-valued-functions/v/contour-plots



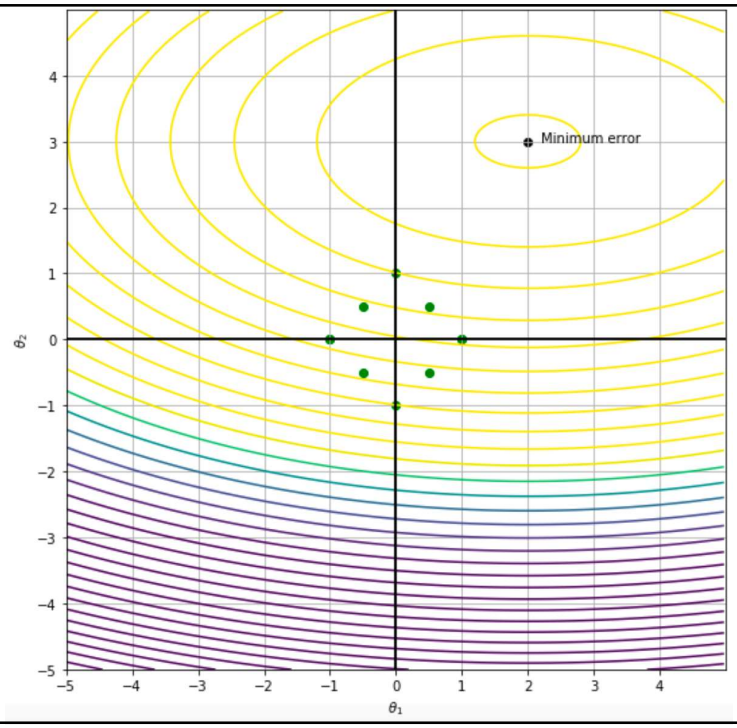
$$\underset{\theta}{\operatorname{argmin}} L(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\theta^T x_i - y_i)^2 + \lambda R(\theta)$$

76 of 91

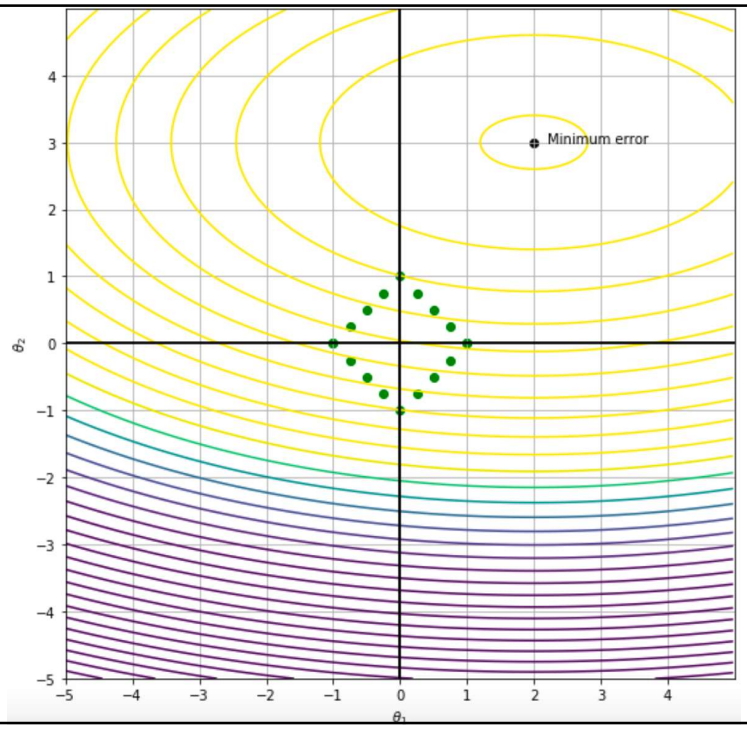
$$R(\theta) = \lambda \sum_{j=1}^{D} |\theta|$$



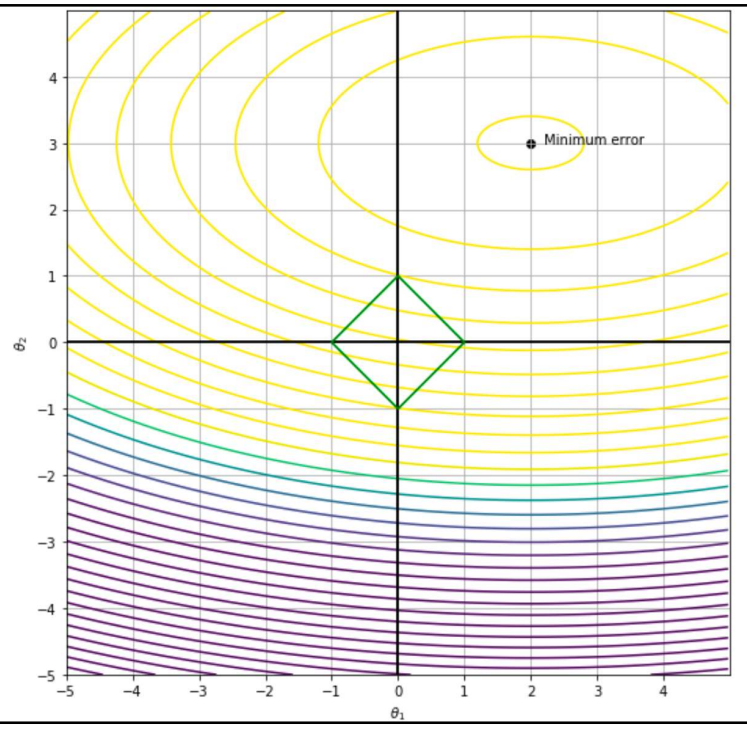
Let's say you were granted an exact Lasso $R(\theta)$ = 1. Here's how our weight will look like.



78 of 91

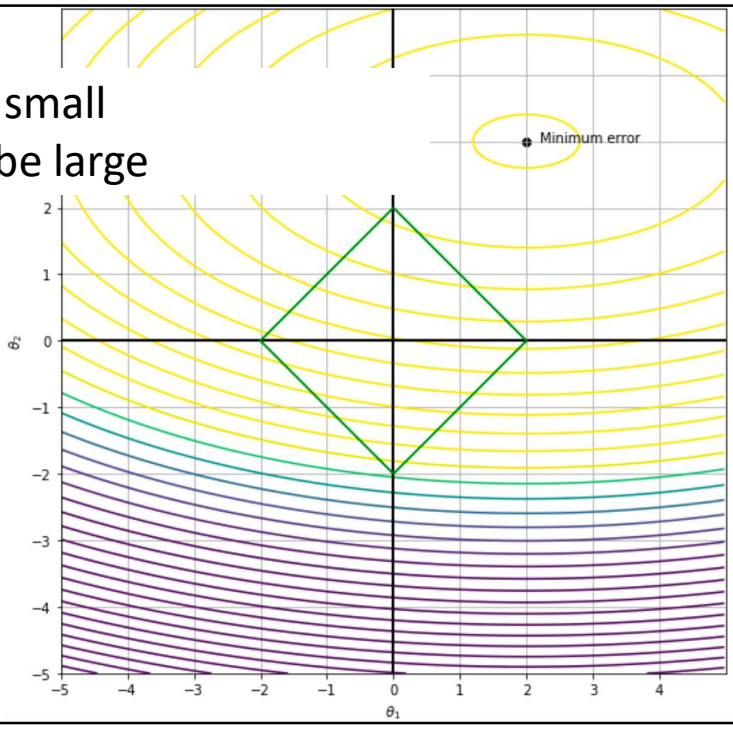


79 of 91



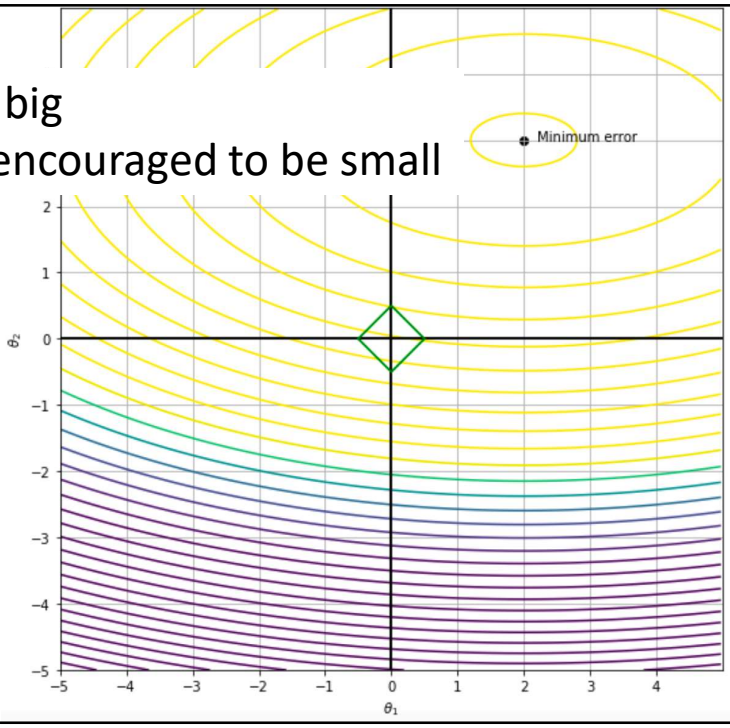
80 of 91

Here is if λ is small
Weights can be large

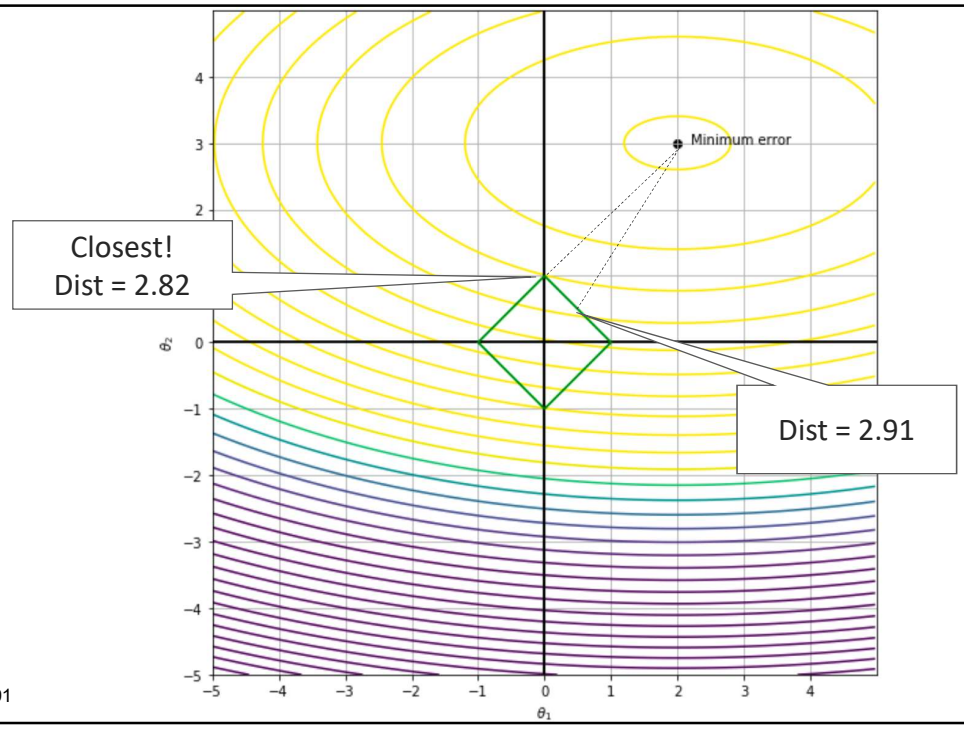


81 of 91

Here is if λ is big
Weights are encouraged to be small

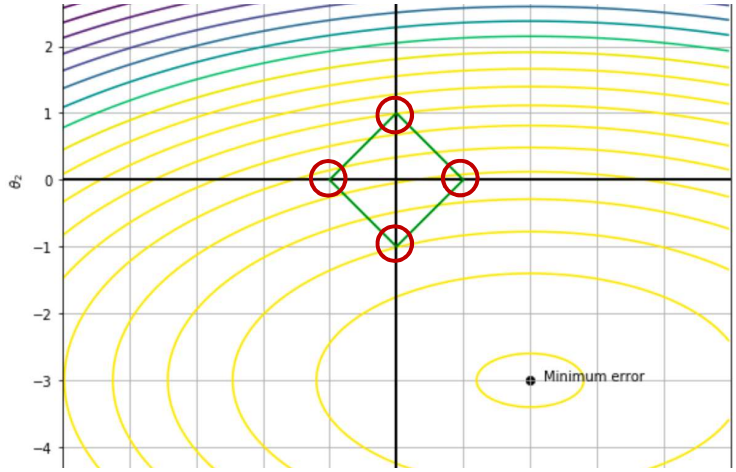


82 of 91



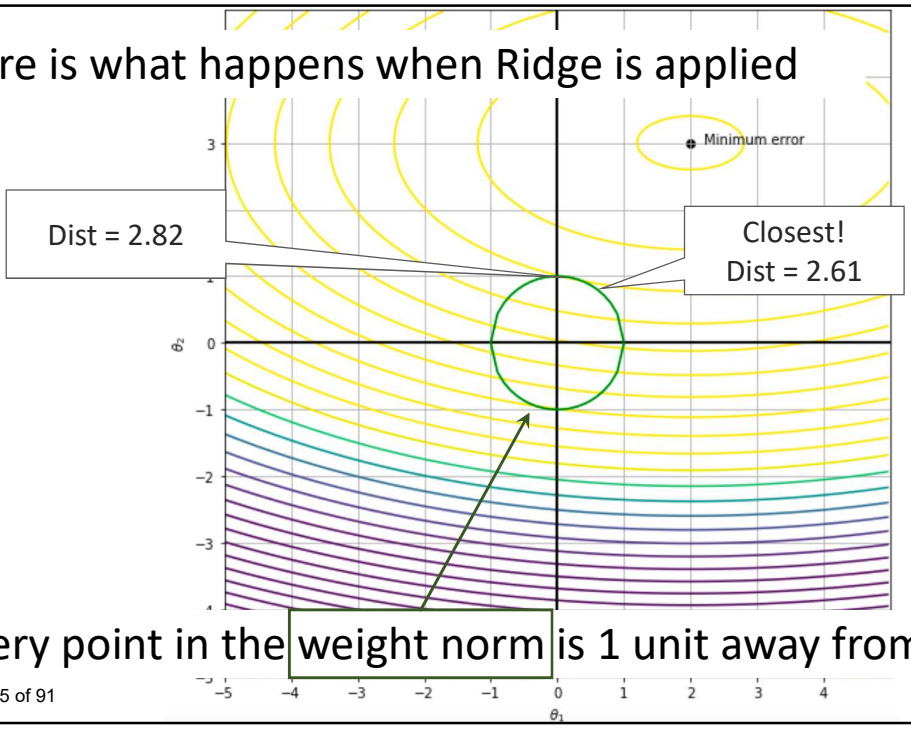
83 of 91

Weights most likely reach the min error when θ is along the axis
These are where its reach is farthest when using Lasso



These are also where some weights are 0

Here is what happens when Ridge is applied



Every point in the weight norm is 1 unit away from 0

85 of 91

Two common regularization methods

Ridge Regression

L2 Regularization

Weights approach 0, but almost always never reaches 0

Lasso Regression

L1 Regularization
Least Absolute Shrinkage and Selection Operator (LASSO)

Weights approach 0
This makes the feature useless

Good for removing features from dataset with a large number of features

86 of 91

Summary

- Transform features to generate more complex functions
- Simple functions are prone to under-fitting (high bias, low variance)
- Complex functions are prone to over-fitting (low bias, high variance)
- Overfitting depends on the amount of data relative to the complexity of the hypothesis.
- A good way to prevent overfitting is to use regularization
  - L_2 regularization (Ridge regression) – likely to make weights of irrelevant features small
  - L_1 regularization (LASSO) – likely to make weights of irrelevant features 0

87 of 91

Next meeting

Moving from regression to classification

88 of 91

fin

References / Slide Credits

- Andrew Ng – CS229
- Eric Xing – CMU 10-701

Homework -2

- For the next homework, we will implement our own linear regression
- We will be working on a house price prediction case study
- Will be posted tonight
- Deadline – Next Week Friday (March 20)