Regression II

Fundamentals of Artificial Intelligence

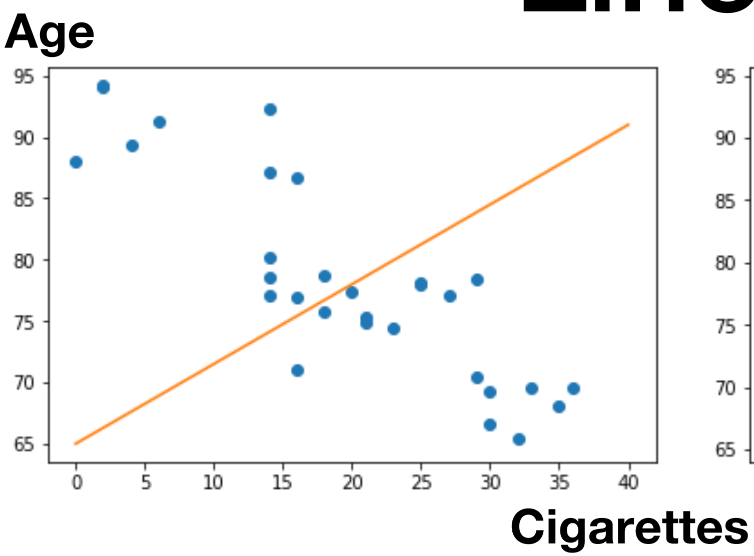
Instructor: Chenhui Chu

Email: chu@i.kyoto-u.ac.jp

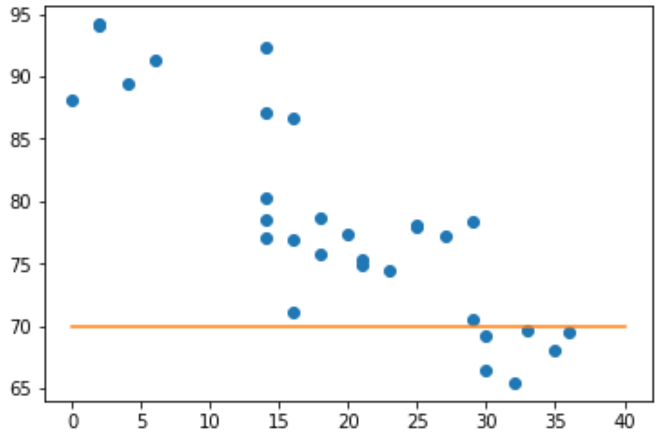
Teaching Assistant: Youyuan Lin

E-mail: youyuan@nlp.ist.i.kyoto-u.ac.jp

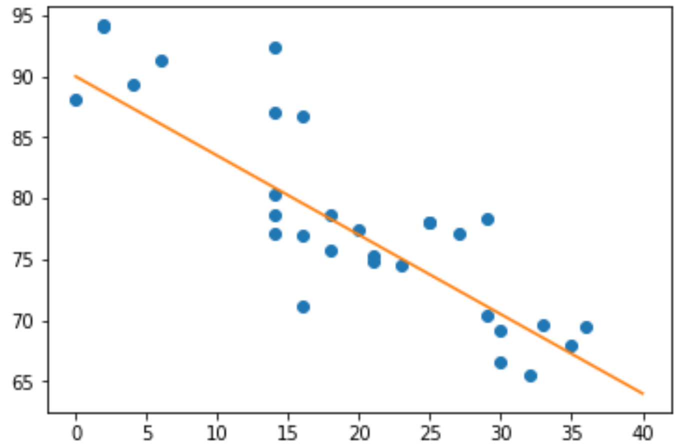
Linear Regression



$\theta_0 \approx 65 \quad \theta_1 \approx 0.7$



$\theta_0 \approx 70 \quad \theta_1 \approx 0$



$\theta_0 \approx 90 \, \theta_1 \approx -0.7$

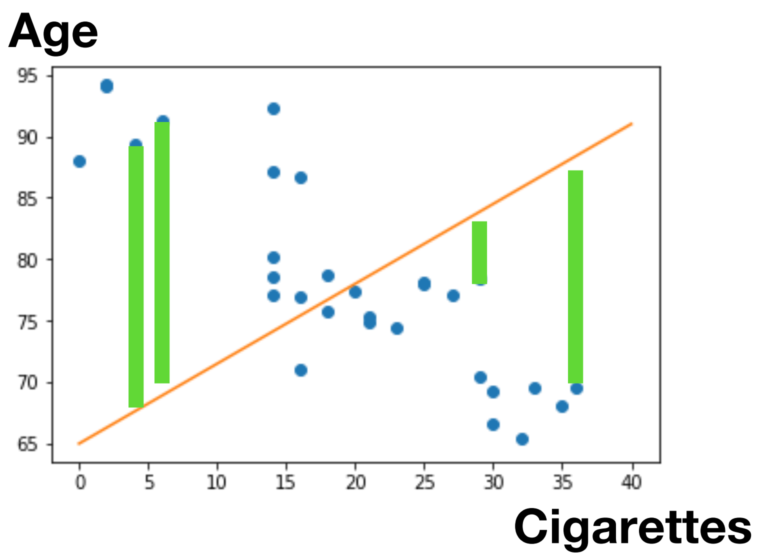
So, which one is best?

$$age = \theta_0 + \theta_1 \times cig$$

How to know which values of the parameters θ_0, θ_1 we should use?

Mean Squared Error

- Finally, we take **average** of the squared error for **all examples**



$\theta_0 \approx 65 \ \theta_1 \approx 0.7$

$$MeanSquaredError = \frac{1}{N} \cdot \sum_i (error_i)^2$$

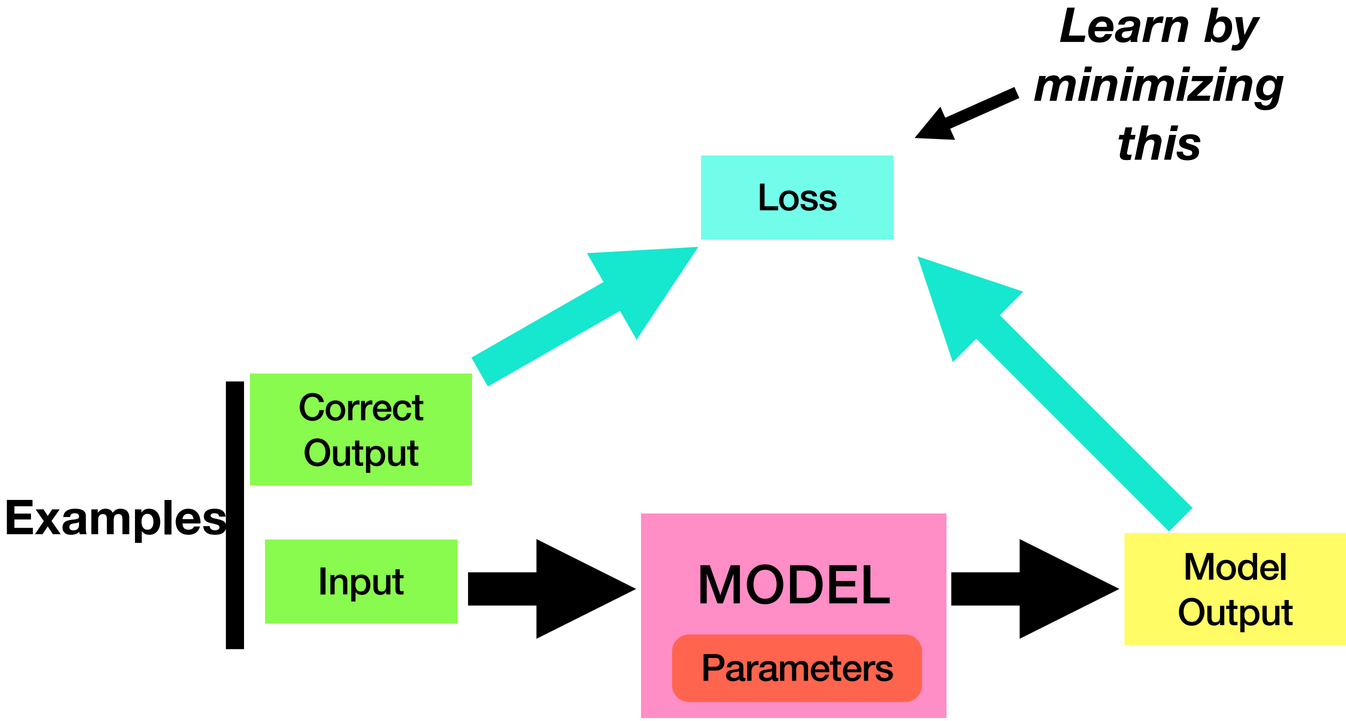
$$MeanSquaredError = \frac{1}{N} \cdot \sum_i (f(x_i) - y_i)^2$$

N: total number of examples
x_i: number of cigarettes smoked by person i
y_i: age person i died
$f(x_i)$: prediction of our model

In our case: $f(x_i)$ ~ $age_{\theta_0,\theta_1}(cig)$

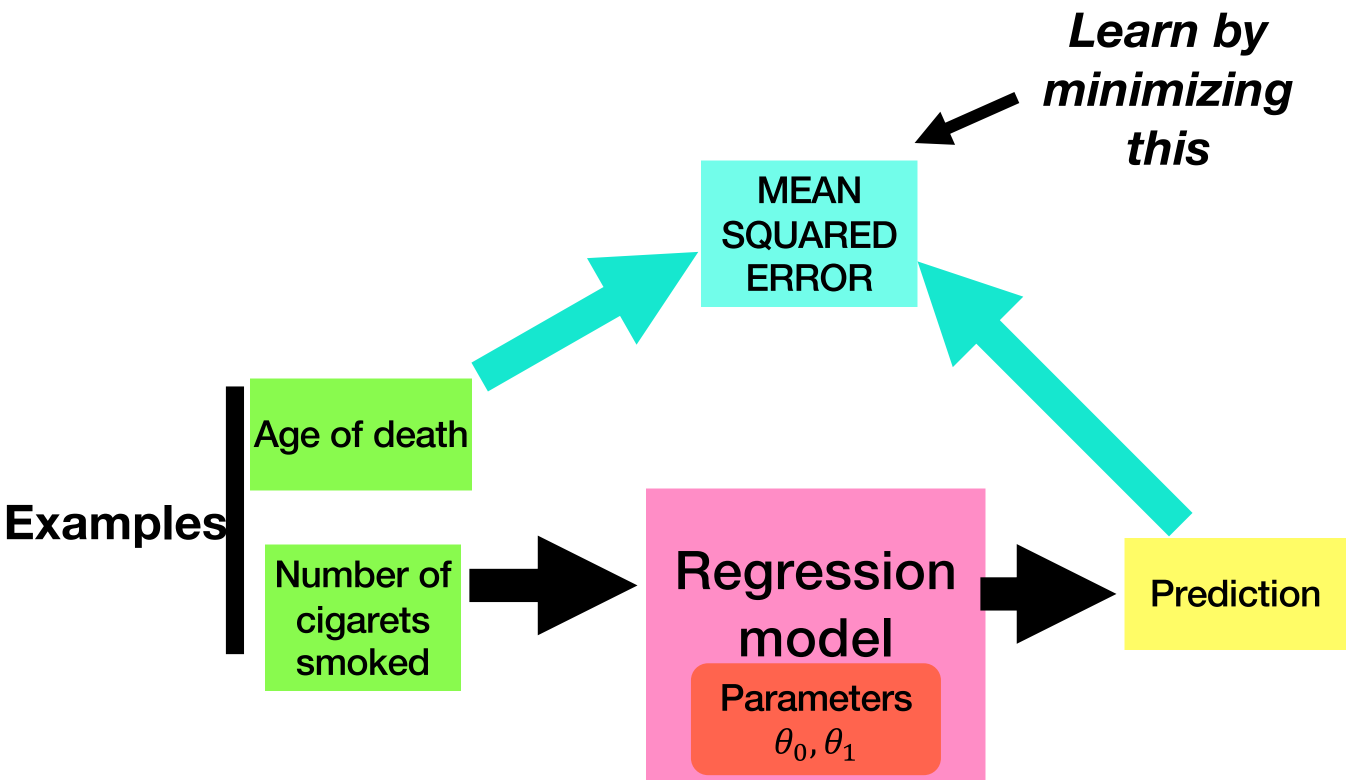
Supervised Learning (1/2)

- In supervised learning, we usually have:
  - A **MODEL**: a "parameterized" function that takes input and produce output
  - A ***Loss***: A function that computes how different the model output is from the correct output
  - ***Examples*** of input and correct output



Supervised Learning (2/2)

- In supervised learning, we usually have:
  - A **MODEL**: a "parameterized" function that takes input and produce output
  - A ***Loss***: A function that computes how different the model output is from the correct output
  - ***Examples*** of input and correct output



$$age_{\theta_0,\theta_1}(cig) = \theta_0 + \theta_1 \times cig$$

Stochastic Gradient Descent

- Stochastic gradient descent says that we can replace the average of the gradient over all examples by the gradient given by a randomly chosen example
- Slower Convergence
- But if we have one million examples: one million times faster to compute!

$$\theta_0 := \theta_0 - lr \times \frac{2}{N} \cdot \sum_i (\theta_0 + \theta_1 \times x_i - y_i)$$

Choose example i randomly

$$\theta_0 := \theta_0 - lr \times 2 \cdot (\theta_0 + \theta_1 \times x_i - y_i)$$

In practice, we often average over a few examples (instead of just one). This is called mini-batch gradient descent

Schedule

- 1. Overview of AI and this Course (4/14)
- 2. Introduction to Python (4/21)
- 3, 4. Mathematics Concepts I, II (4/28, 5/12)
- 5, 6. Regression I, II (5/19, 5/26)
- 7. Classification (6/2)
- 8. Introduction to Neural Networks (6/9)
- 9. Neural Networks Architecture and Backpropagation (6/16)
- 10. Fully Connected Layers (6/23)
- 11, 12, 13. Computer Vision I, II, III (6/30, 7/7, 7/14)
- 14. Natural Language Processing (7/17)

Overview of This Course

11, 12, 13. Computer Vision I, II, III

14. Natural language processing

Deep Learning Applications

8. Neural network Introduction

9. Architecture and Backpropagation

10. Feedforward neural networks

Deep Learning

5. Regression I

6. Regression II

7. Classification

Basic Supervised Machine Learning

2. Python

3, 4. Mathematics Concepts I, II

Fundamental of Machine Learning

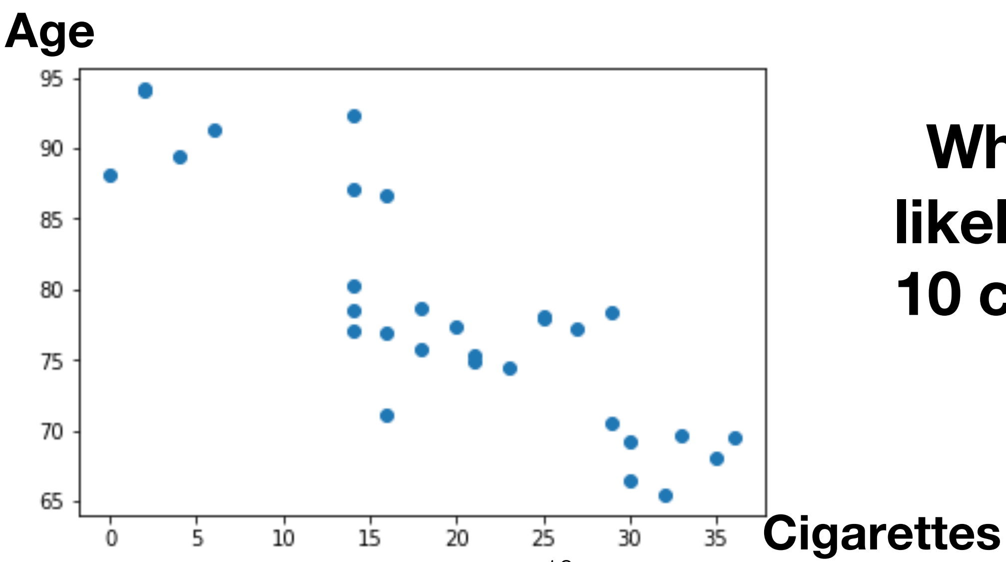
Today

- We expand on this by considering:
  - More than one input feature
  - More complex functions
- We will have a look at the important concept of overfitting

Other Factors?

- The number of cigarettes smoked is **not the only** important **factor** for **predicting** the age of death
  - Physical fitness, biological Sex, Wealth….

daily cigarettes	age of death
32.0	73
7.0	88
30.0	82
17.0	85
27.0	76
15.0	84
20.0	72
28.0	77
….	…..



What age am I most likely to die if I smoke 10 cigarettes per day?

Adding More Information

- Let us suppose now that, for the same 30 persons, we **also** have their BMI index and know their sex

- Vocabulary Note: In Machine Learning, we often call a "*feature*" or "*feature function*" each of this piece of information about an example

	daily cigarettes	bmi	is male	age of death
0	5.0	18.5	1.0	79.8
1	9.0	45.1	0.0	56.8
2	38.0	14.2	0.0	61.4
3	12.0	48.5	1.0	37.5
4	34.0	19.2	0.0	68.4
5	5.0	38.6	0.0	69.3
6	31.0	33.8	1.0	54.8
7	25.0	33.6	1.0	63.0
8	24.0	45.2	1.0	39.3
9 …	….	….	….	

$$\boxed{bmi = \frac{weight}{height^2}}$$

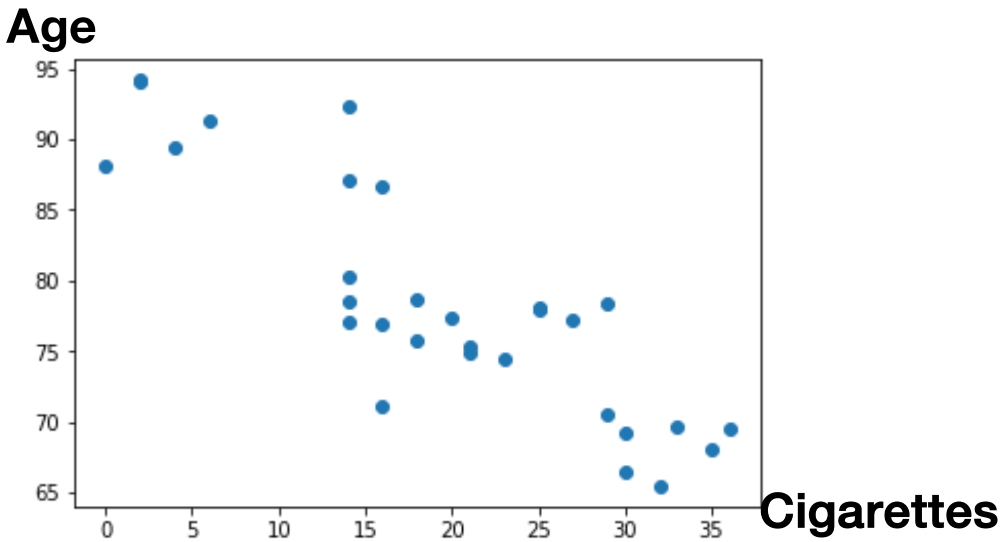
Is male: 1 if the person is a male, 0 if female

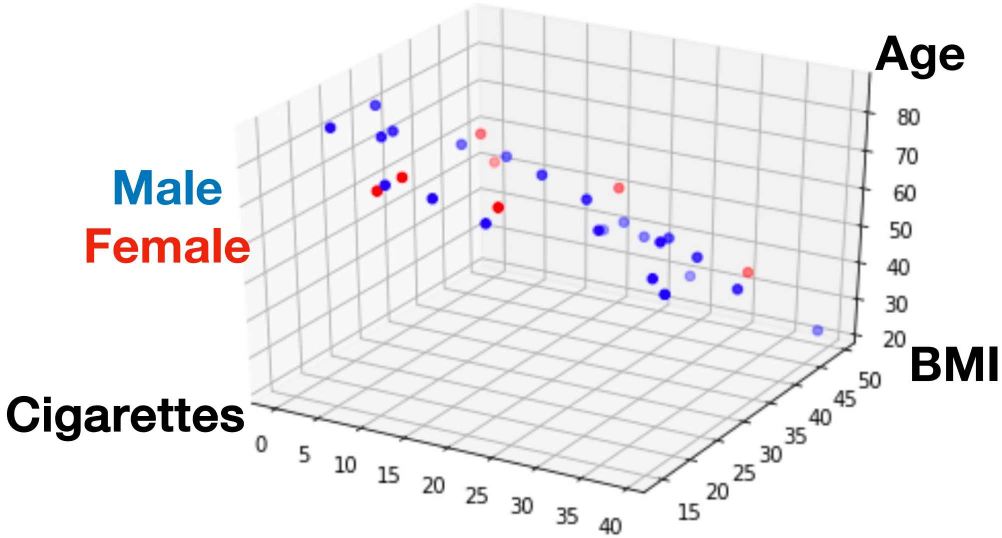
Note: most of the time, this is how we represent "categorical data" in Machine Learning: a feature equal to 1 if the example belongs to the category in question, and equal to zero otherwise

Visual Representation of the Examples

	daily cigarettes	age of death
0	5.0	79.8
1	9.0	56.8
2	38.0	61.4
3	12.0	37.5
7	25.0	63.0
8	24.0	39.3
9	…	….

	daily cigarettes	bmi	is male	age of death
0	5.0	18.5	1.0	79.8
1	9.0	45.1	0.0	56.8
2	38.0	14.2	0.0	61.4
3	12.0	48.5	1.0	37.5
7	25.0	33.6	1.0	63.0
8	24.0	45.2	1.0	39.3
9	…	….	….	….





Linear Regression with More Than One Feature

- We can still suppose that the *age of death* is a **linear function** of *the features*

$$age = \theta_0 + \theta_1 \times cig \quad \Rightarrow \quad age = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

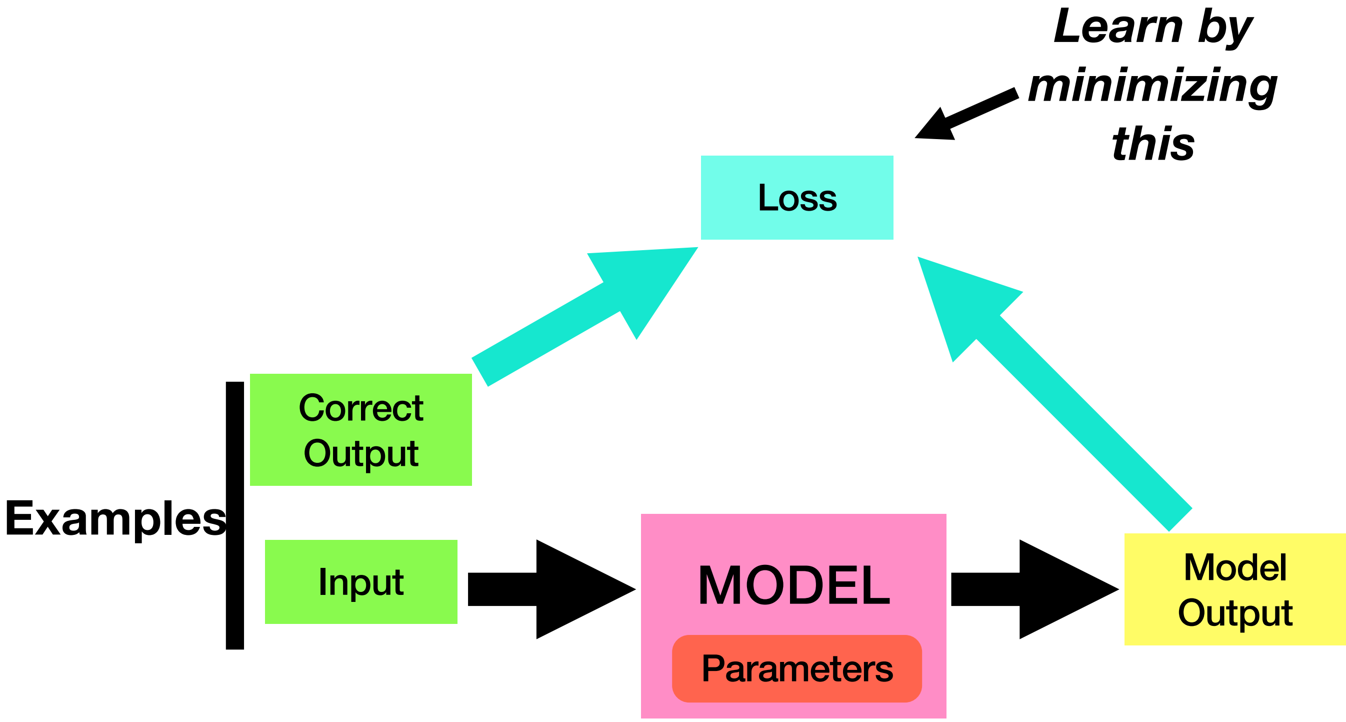
- We now have 2 more parameters (because we have 2 more features)

- For a total of 4 parameters

- But finding the parameters will be done in exactly the same way as in the case with one feature

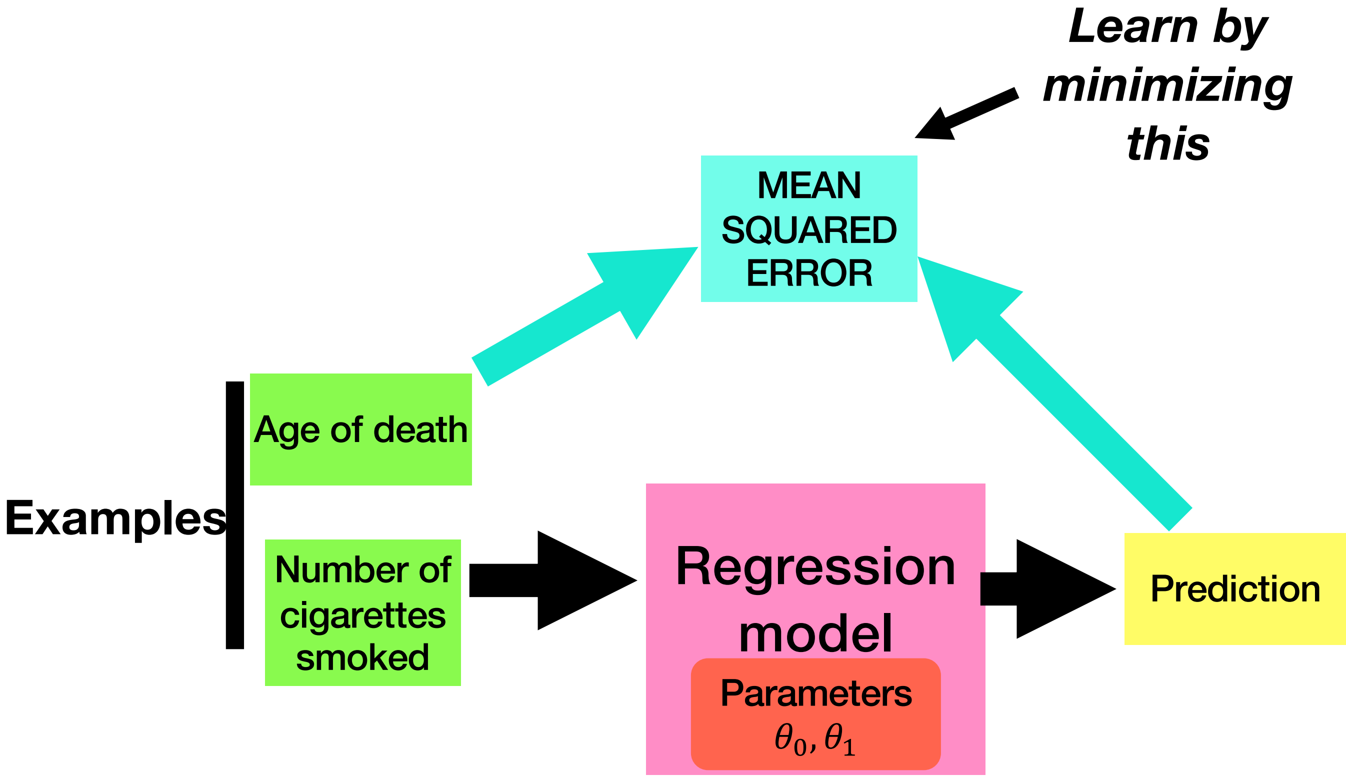
Supervised Learning (1/3)

- In supervised learning, we usually have:
  - A **MODEL**: a "parameterized" function that takes input and produce output
  - A ***Loss***: A function that computes how different the model output is from the correct output
  - ***Examples*** of input and correct output



Supervised Learning (2/3)

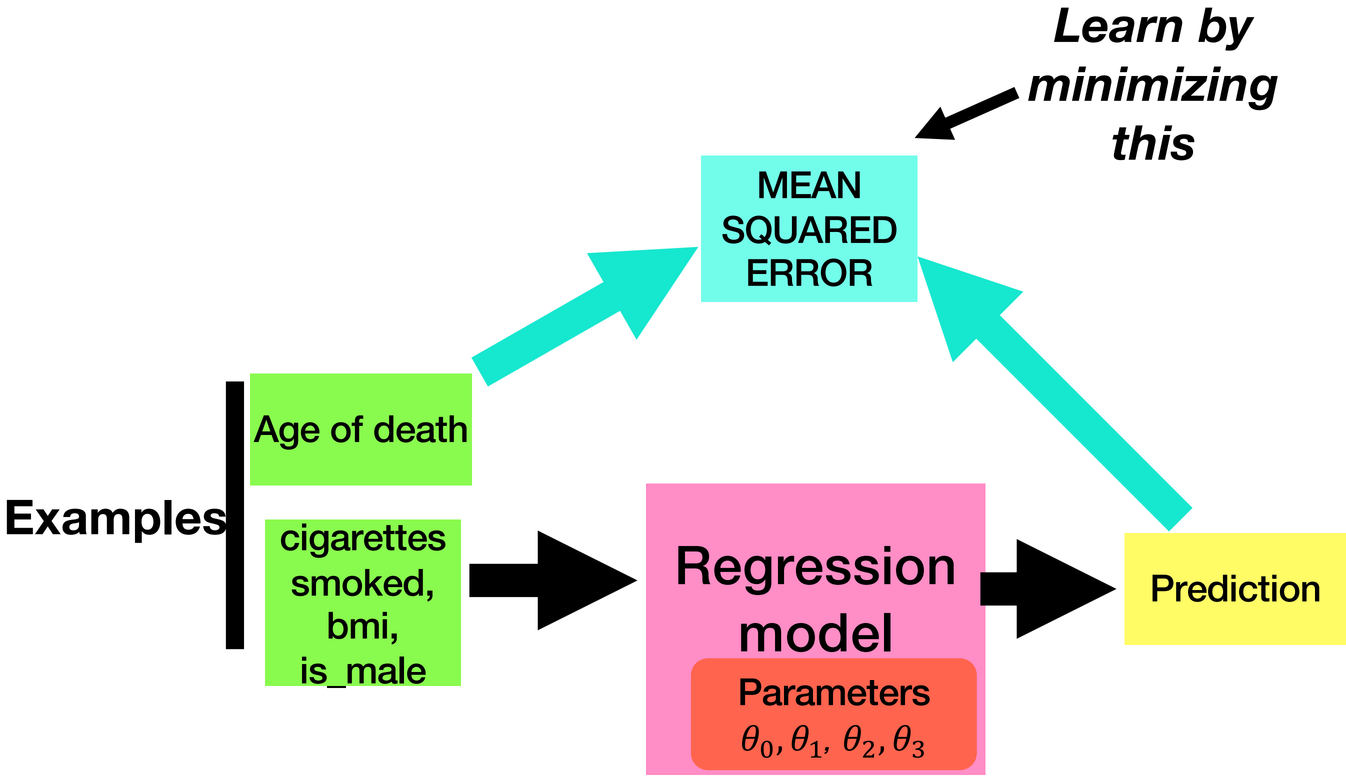
- In supervised learning, we usually have:
  - A **MODEL**: a "parameterized" function that takes input and produce output
  - A ***Loss***: A function that computes how different the model output is from the correct output
  - ***Examples*** of input and correct output (cigarettes smoked, age of death)



$$age_{\theta_0,\theta_1}(cig) = \theta_0 + \theta_1 \times cig$$

Supervised Learning (3/3)

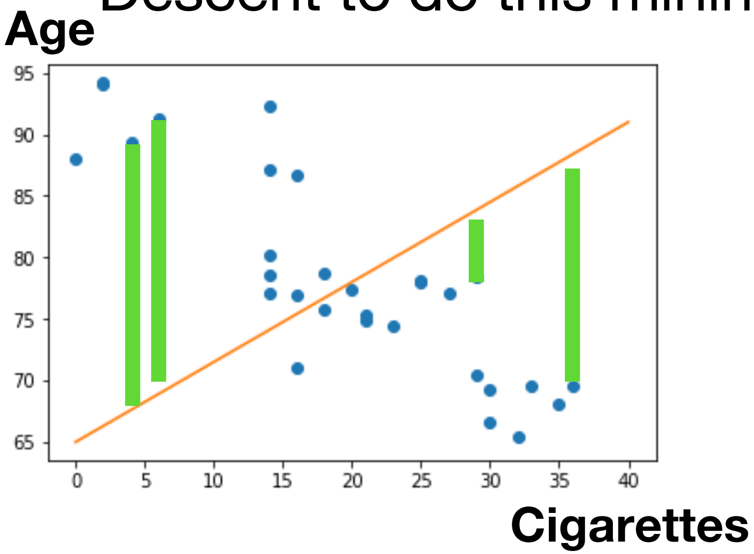
- In supervised learning, we usually have:
  - A **MODEL**: a “parameterized” function that takes input and produce output
  - A ***Loss***: A function that computes how different the model output is from the correct output
  - ***Examples*** of input and correct output (cigarettes smoked, bmi, is_male, age of death)



$$age_{\vec{\theta}}(cig) = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

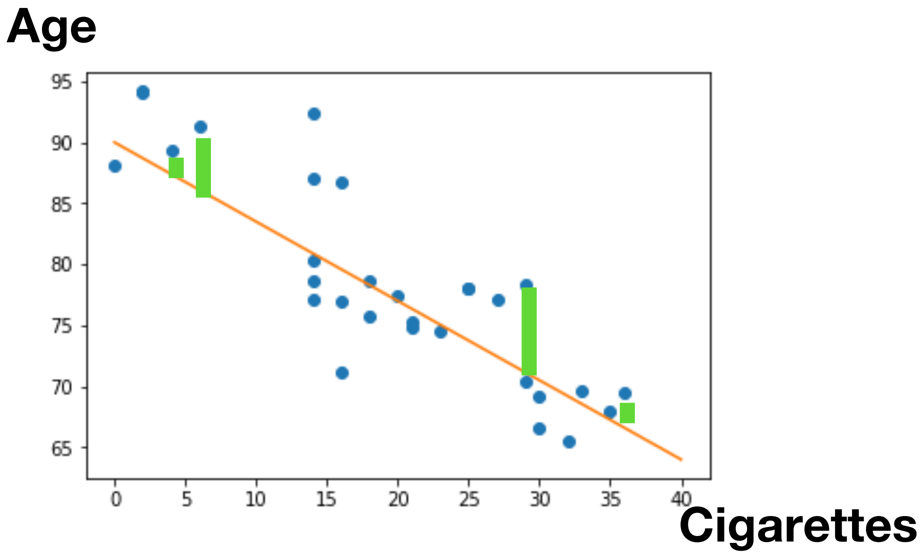
Reminder: Loss with One Feature

- We saw that we could find the parameters of this linear relation by minimizing the Mean Squared Distance between the prediction of the model and the actual value; We saw we could use Gradient Descent to do this minimization



$\theta_0 \approx 65 \quad \theta_1 \approx 0.7$

$MeanSquaredError = 294.7$



$\theta_0 \approx 90 \quad \theta_1 \approx -0.7$

$MeanSquaredError = 18.3$

Linear Regression with More Than One Feature

- The **loss** is defined as before: the average of the squared distance between model prediction and real example values

$$MeanSquaredError = \frac{1}{N} \cdot \sum_i (\underbrace{model(cig_i, bmi_i, ismale_i)}_{\text{Model prediction for example i}} - \underbrace{age_i}_{\text{Real value for example i}})^2$$

	daily cigarettes	bmi	is male	age of death
0	5.0	18.5	1.0	79.8
1	9.0	45.1	0.0	56.8
2	38.0	14.2	0.0	61.4
3	12.0	48.5	1.0	37.5
7	25.0	33.6	1.0	63.0
8	24.0	45.2	1.0	39.3
9	…	….	….	….

$$age_{model} = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

Small Exercise: Compute the Loss

- The loss is defined as before: the average of the squared distance between model prediction and real example values

$$MeanSquaredError = \frac{1}{N} \cdot \sum_i (\underbrace{model(cig_i, bmi_i, ismale_i)}_{\text{Model prediction for example i}} - \underbrace{age_i}_{\text{Real value for example i}})^2$$

	daily cigarettes	bmi	is male	age of death
0	5.0	18	0.0	80
1	9.0	45	1.0	57
2	38.0	16	0.0	61

$$age_{model} = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

$$\theta_0 = 90 \quad \theta_1 = -1 \quad \theta_2 = -0.1 \quad \theta_3 = -6$$

Compute the model prediction for each of the examples.
Then, compute the loss.

$$MeanSquaredError = \frac{1}{N} \cdot \sum_i (\underbrace{model(cig_i, bmi_i, ismale_i)}_{\text{Model prediction for example i}} - \underbrace{age_i}_{\text{Real value for example i}})^2$$

	daily cigarettes	bmi	is male	age of death
0	5.0	18	0.0	80
1	9.0	45	1.0	57
2	38.0	16	0.0	61

$$age_{model} = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

$$\theta_0 = 90 \quad \theta_1 = -1 \quad \theta_2 = -0.1 \quad \theta_3 = -6$$

What is the Gradient (1/2)?

- To do it a bit differently than last time: let us note the error on example i as:

$$error_i = model(cig_i, bmi_i, ismale_i) - age_i$$

Then our loss is equal to:

$$MeanSquaredError = \frac{1}{N} \cdot \sum_i (error_i)^2$$

(The mean squared distance is actually also often called the mean squared error)

Then our gradient is equal to:

Using linearity and the fact that

$$\frac{d}{dx}[f(x)]^2 = 2 \times f(x) \times \frac{d}{dx} f(x)$$

$$\frac{\partial}{\partial \theta_k} MeanSquaredDistance = \frac{1}{N} \cdot \sum_i 2 \times error_i \times \frac{\partial}{\partial \theta_k} error_i$$

What is the Gradient (2/2)?

$$\frac{\partial}{\partial \theta_k} MeanSquaredError = \frac{1}{N} \cdot \sum_i 2 \times error_i \times \frac{\partial}{\partial \theta_k} error_i$$

$$error_i = model(cig_i, bmi_i, ismale_i) - age_i$$

$$age_{model} = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

$$\frac{\partial}{\partial \theta_0} error_i = 1$$

$$\frac{\partial}{\partial \theta_1} error_i = cig_i$$

$$\frac{\partial}{\partial \theta_2} error_i = bmi_i$$

$$\frac{\partial}{\partial \theta_3} error_i = ismale_i$$

(we know have 4 parameters, so gradient is a 4-dimesnional vector)

$$gradient = \left| \begin{array}{l} \frac{1}{N} \cdot \sum_i 2 \times error_i \times 1 \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times cig_i \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times bmi_i \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times ismale_i \end{array} \right|$$

Small Exercise: Compute the Gradient

- Let us compute the gradient for this examples and θk:

$$error_i = model(cig_i, bmi_i, ismale_i) - age_i$$

$$age_{model} = \theta_0 + \theta_1 \times cig + \theta_2 \times bmi + \theta_3 \times ismale$$

$$\theta_0 = 90 \quad \theta_1 = -1 \quad \theta_2 = -0.1 \quad \theta_3 = -6$$

	daily cigarettes	bmi	is male	age of death
0	5.0	18	0.0	80
1	9.0	45	1.0	57
2	38.0	16	0.0	61

Note you already computed the errors when you computed the loss in previous exercise

$$gradient = \begin{vmatrix} \frac{1}{N} \cdot \sum_i 2 \times error_i \times 1 \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times cig_i \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times bmi_i \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times ismale_i \end{vmatrix}$$

- Let us compute the gradient for this examples and θ_k:

$$\theta_0 = 90 \quad \theta_1 = -1 \quad \theta_2 = -0.1 \quad \theta_3 = -6$$

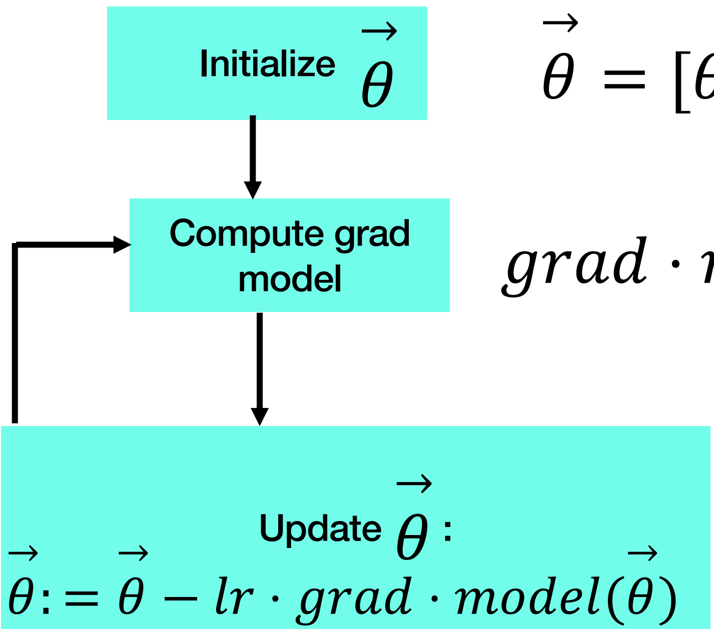
	daily cigarettes	bmi	is male	age of death
0	5.0	18	0.0	80
1	9.0	45	1.0	57
2	38.0	16	0.0	61

$$gradient = \begin{vmatrix} \frac{1}{N} \cdot \sum_i 2 \times error_i \times 1 \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times cig_i \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times bmi_i \\ \frac{1}{N} \cdot \sum_i 2 \times error_i \times ismale_i \end{vmatrix}$$

Note you already computed the errors when you computed the loss in previous exercise

Gradient Descent

- Now that we know how to compute the gradient, we can find the optimal parameters with gradient descent:



$$\vec{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3]$$

$$grad \cdot model(\vec{\theta}) = [\frac{\partial f}{\partial \theta_0}, \ldots \frac{\partial f}{\partial \theta_3}]$$

Feature Scaling (1/2)

- We can see that our features have very different **ranges and means:** “daily cigarettes” is from 0 to 50, “bmi” is from 15 to 50, “is_male” is from 0 to 1
- Usually, features having different ranges make learning/gradient descent more difficult

	daily cigarettes	bmi	is male	age of death
0	5.0	18.5	1.0	79.8
1	9.0	45.1	0.0	56.8
2	38.0	14.2	0.0	61.4
3	12.0	48.5	1.0	37.5
7	25.0	33.6	1.0	63.0
8	24.0	45.2	1.0	39.3
9	…	….	….	….

The solution is to scale all the features to the same range

$$new = \frac{old - mean(old)}{max(old) - min(old)}$$

This way, all features have ranges between -1 and 1, and mean equal to zero

Feature Scaling (2/2)

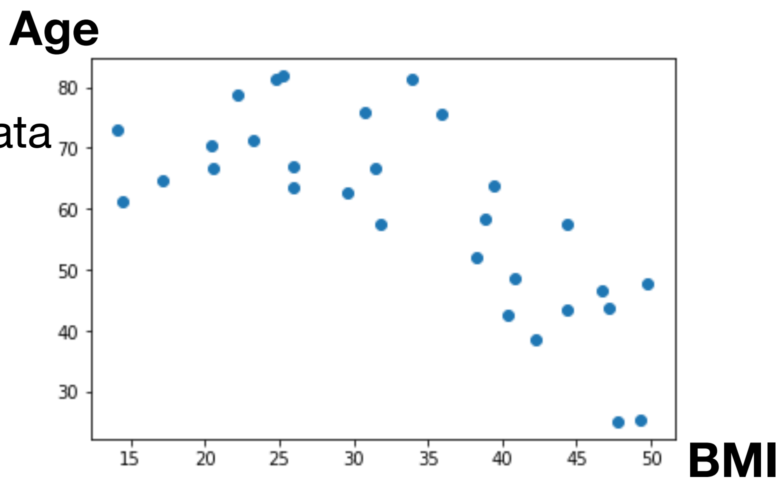
- After the features have been scaled, we apply Gradient Descent as usual:

$$new = \frac{old - mean(old)}{max(old) - min(old)}$$

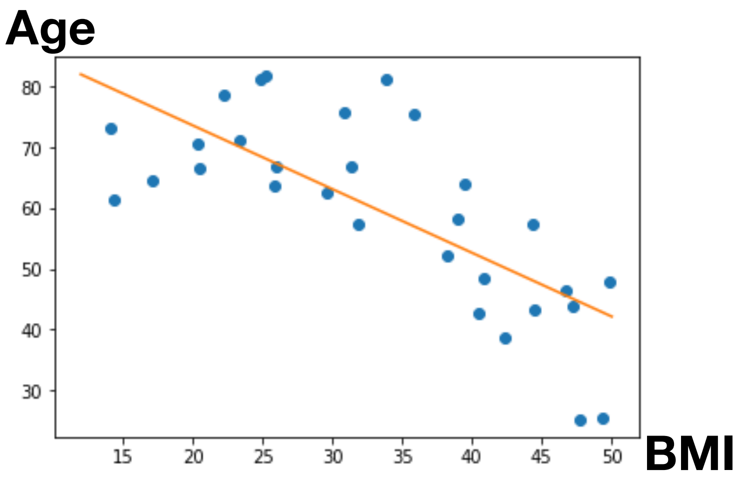
	daily cigarettes	bmi	is male	age of death
0	24.0	44.4	1.0	43.4
1	37.0	47.7	1.0	25.2
2	11.0	14.1	0.0	73.1
3	33.0	49.3	1.0	25.5
4	27.0	20.4	0.0	70.5
5	27.0	38.2	1.0	52.1
6	6.0	22.2	1.0	78.6
7	31.0	17.1	0.0	64.6
8	28.0	23.3	0.0	71.3
9	15.0	31.4	1.0	66.8

	daily cigarettes	bmi	is male	age of death
0	0.064957	0.313072	0.433333	43.4
1	0.398291	0.405509	0.433333	25.2
2	-0.268376	-0.535668	-0.566667	73.1
3	0.295726	0.450327	0.433333	25.5
4	0.141880	-0.359197	-0.566667	70.5
5	0.141880	0.139402	0.433333	52.1
6	-0.396581	-0.308777	0.433333	78.6
7	0.244444	-0.451634	-0.566667	64.6
8	0.167521	-0.277965	-0.566667	71.3
9	-0.165812	-0.051074	0.433333	66.8

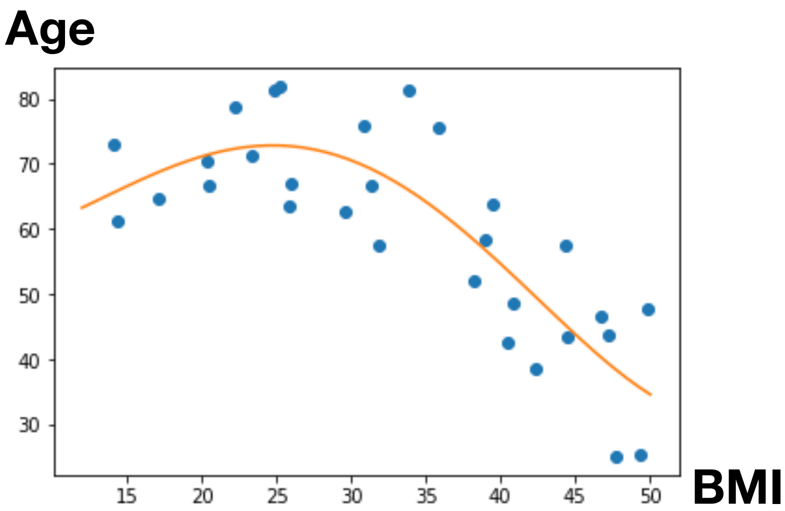
Learning More Complicated Functions



- So far, we have only tried to learn linear functions of the data
- We might want to learn more complicated functions
- For example, having a high BMI reduce life expectancy
- But having a very low BMI also reduces life expectancy



We need a more complex model!



Learning More Complex Functions: "Expanding the Feature Space"

- One neat trick to learn more complex functions: Create additional features from existing features; Then apply linear regression
- Example: From the feature bmi, we add bmi^2, bmi^3 and bmi^4:

	bmi	age of death
0	44.4	43.4
1	47.7	25.2
2	14.1	73.1
3	49.3	25.5
4	20.4	70.5
5	38.2	52.1
6	…	…

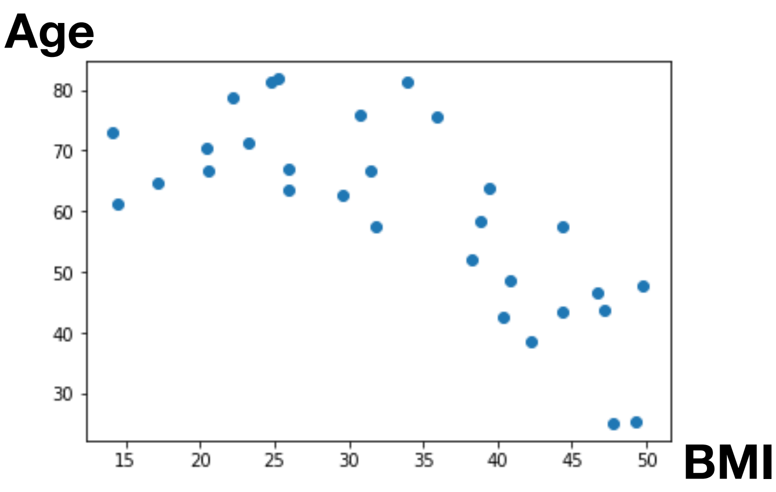
$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^3 + \theta_4 \times bmi^4$$

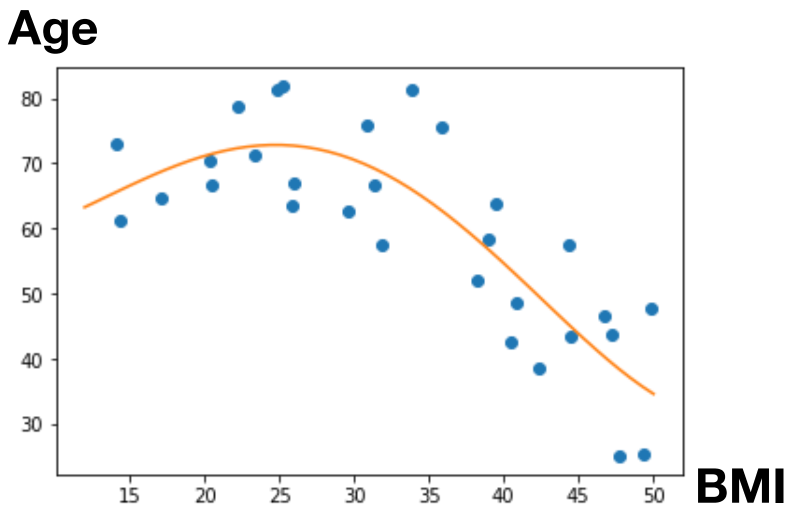
	bmi	bmi^2	bmi^3	bmi^4	age of death
0	44.4	1971.36	87528.384	3.88E+06	43.4
1	47.7	2275.29	108531.333	5.17E+06	25.2
2	14.1	198.81	2803.221	3.95E+04	73.1
3	49.3	2430.49	119823.157	5.90E+06	25.5
4	20.4	416.16	8489.664	1.73E+05	70.5
5	38.2	1459.24	55742.968	2.12E+06	52.1
6	…	…	…	…	…

Learning More Complicated Functions

- This way, we can learn much more complex functions
- After finding the optimal parameters by gradient descent on the Mean Squared Error:

$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^3 + \theta_4 \times bmi^4$$





Learning Functions That are Too Complicated (1/3)

- Now, we might think:

Cool. I am going to add as many variations of the features as I can. Then my loss (Mean Squared Error) will drop to zero!

	bmi	bmi^2	bmi^3	bmi^4	bmi^5	sin(bmi)	log(bmi)	…	age of death
0	44.4	1971.36	87528.384	3.886260E+06	1.725500E+08	0.405662	3.793239	…	43.4
1	47.7	2275.29	108531.333	5.176945E+06	2.469403E+08	-0.544766	3.864931	…	25.2
2	14.1	198.81	2803.221	3.952542E+04	5.573084E+05	0.999309	2.646175	…	73.1
3	49.3	2430.49	119823.157	5.907282E+06	2.912290E+08	-0.822324	3.897924	…	25.5
4	20.4	416.16	8489.664	1.731891E+05	3.533059E+06	0.999793	3.015535	…	70.5
5	38.2	1459.24	55742.968	2.129381E+06	8.134237E+07	0.480205	3.642836	…	52.1
6	22.2	492.84	10941.048	2.428913E+05	5.392186E+06	-0.207336	3.100092	…	78.6
7	17.1	292.41	5000.211	8.550361E+04	1.462112E+06	-0.984065	2.839078	…	64.6
8	…	…	…	…	…	….	…	…	…

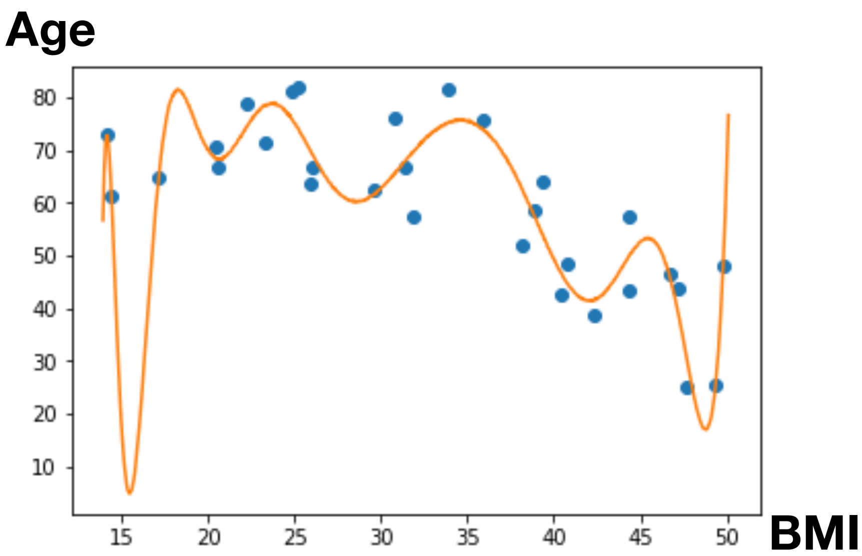
$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^3 + \theta_4 \times bmi^4$$
$$+ \theta_5 \times bmi^5 + \theta_6 \times sin(bmi) + \theta_7 \times log(bmi) + \ldots$$

Learning Functions That are Too Complicated (2/3)

- The problem: this is what you are going to get:

Cool. My loss (Mean Squared Error) is lower than before. My model is better now!

Or is it really better???



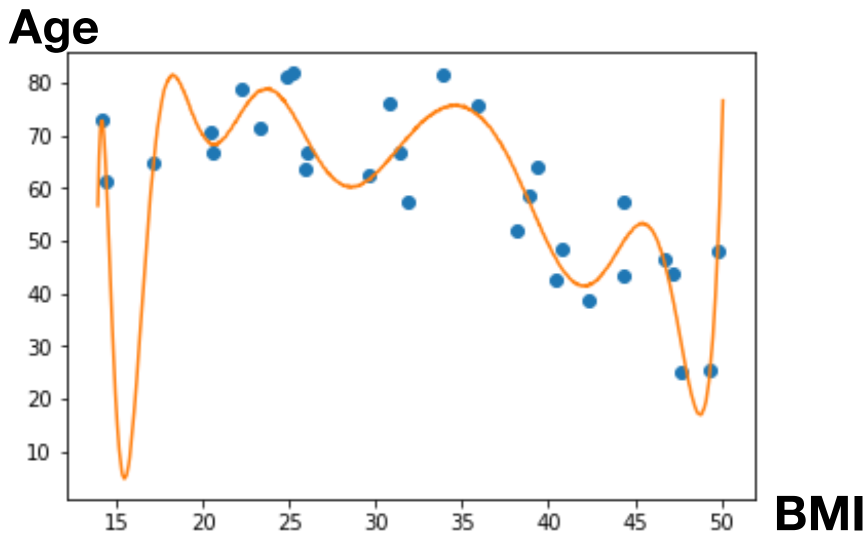
$$MeanSquaredError = 1.3$$

Learning Functions That are Too Complicated (3/3)

- The problem: this is what you are going to get:

Cool. My loss (Mean Squared Error) is lower than before. My model is better now!

Or is it really better???



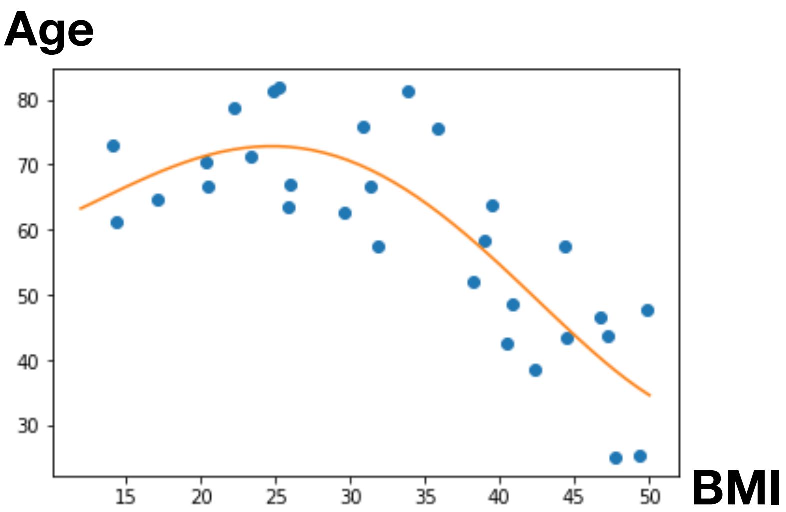
$$MeanSquaredError = 1.3$$

The model predict that somebody with BMI 48 will die at 18

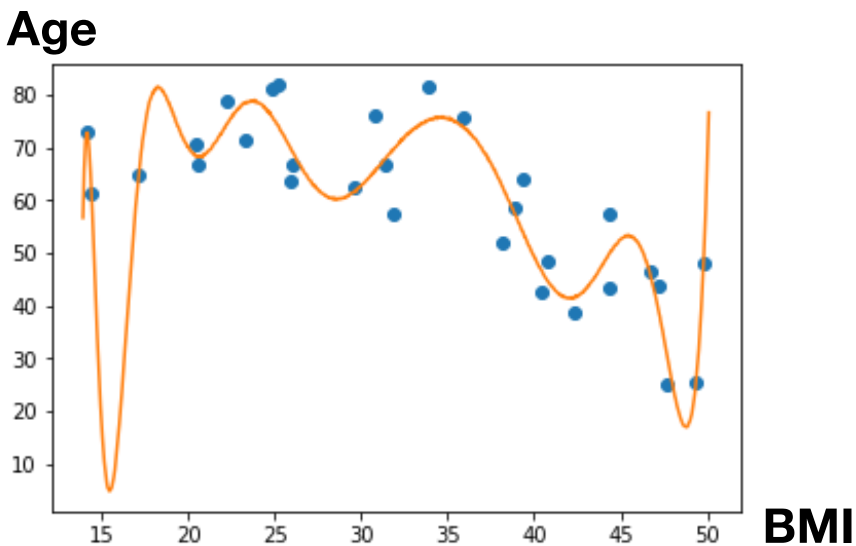
But somebody with BMI 51 is predicted to die at 80

Overfitting (1/2)

- Minimizing the loss does not always give the best model….
- This phenomenon is called overfitting



$$MeanSquaredError = 11.2$$



$$MeanSquaredError = 1.3$$

Overfitting (2/2)

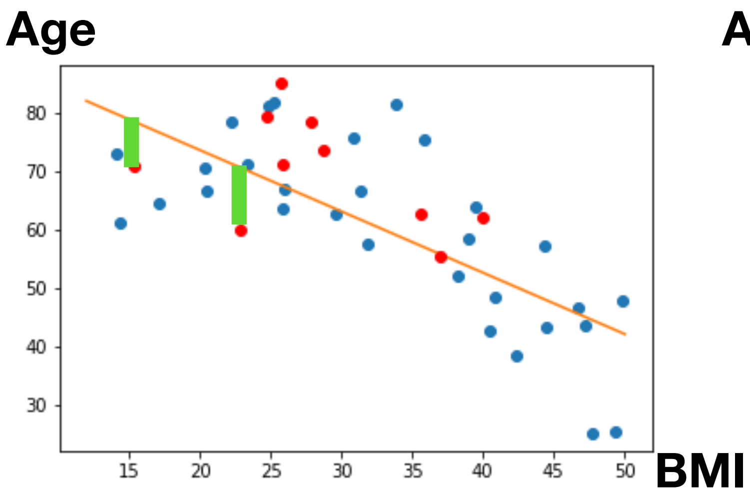
- **Overfitting** is to Machine Learning what **Rote Learning** is to Human Learning
- Instead of understanding the data, the model just memorized all of the examples
- If we ask it to predict the age of death of an example it has seen, it will give very good prediction
- But it will give very bad prediction as soon as we ask him to make a prediction for someone that was not in the example data.
- Very similar to a student that memorize without understanding the answer to a set of exercises in a class. He will do very well in the exam if the exam contains the exercises he studied, but very bad if the exercises are a bit different.

How to Detect Overfitting (1/3)?

- Pretty much like we do with humans: We evaluate them with different exercises than the ones they were trained for.
- In practice, it means we separate our example data in 2:
  - Training Data
  - Test Data
- We use the training data to do the Learning (we train the model)
- We use the test data to evaluate the quality of the learning (we test the model)

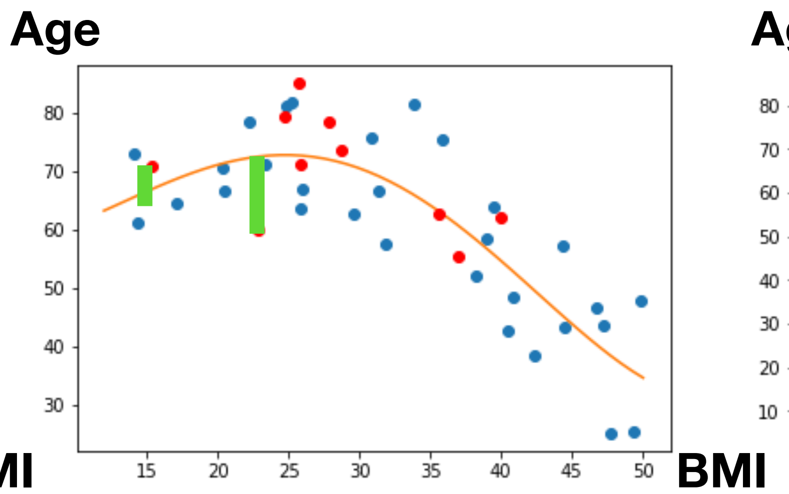
How to Detect Overfitting (2/3)?

- In blue: training examples
- In red: test examples



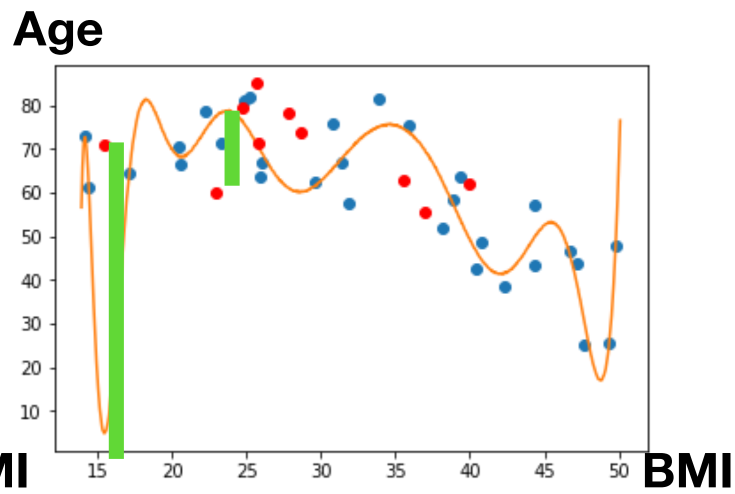
$MeanSquaredError = 18.6$

$TestMeanSquaredError = 17.3$



$MeanSquaredError = 11.2$

$TestMeanSquaredError = 11.8$



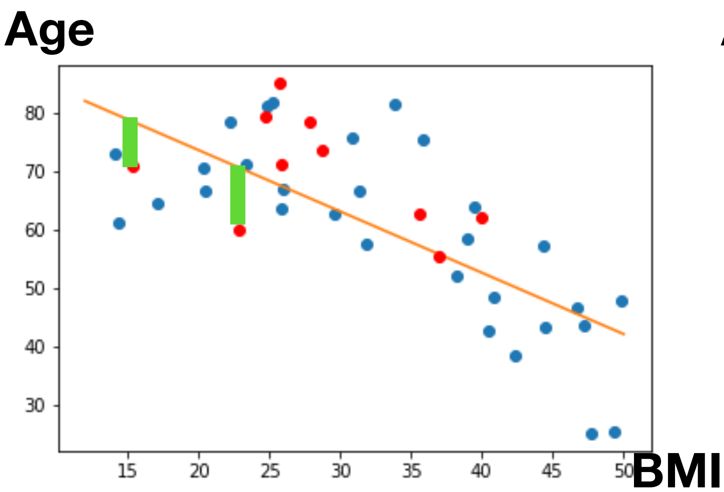
$MeanSquaredError = 1.3$

$TestMeanSquaredError = 24.5$

How to Detect Overfitting (3/3)?

- If we see a model with very low training loss and high test loss: it is overfitting!

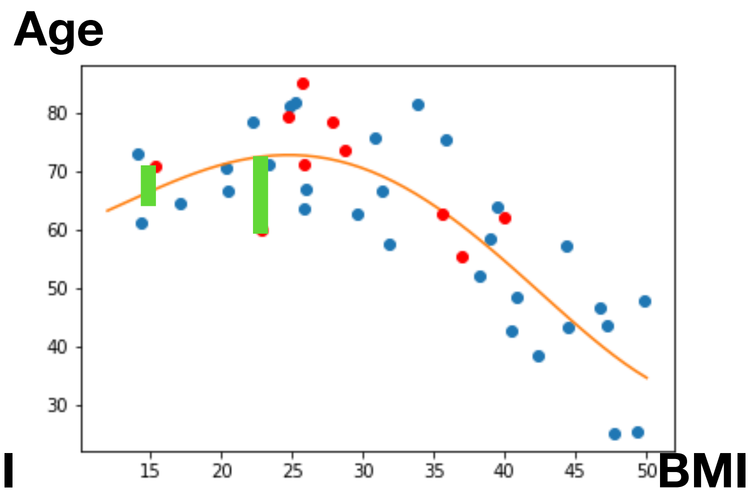
Linear Model



$MeanSquaredError = 18.6$

$TestMeanSquaredError = 17.3$

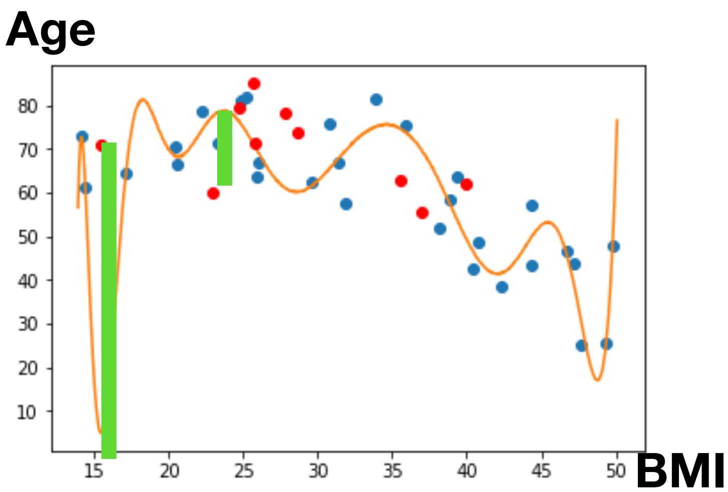
More Complex Model



$MeanSquaredError = 11.2$

$TestMeanSquaredError = 11.8$

“Too Complex” Model



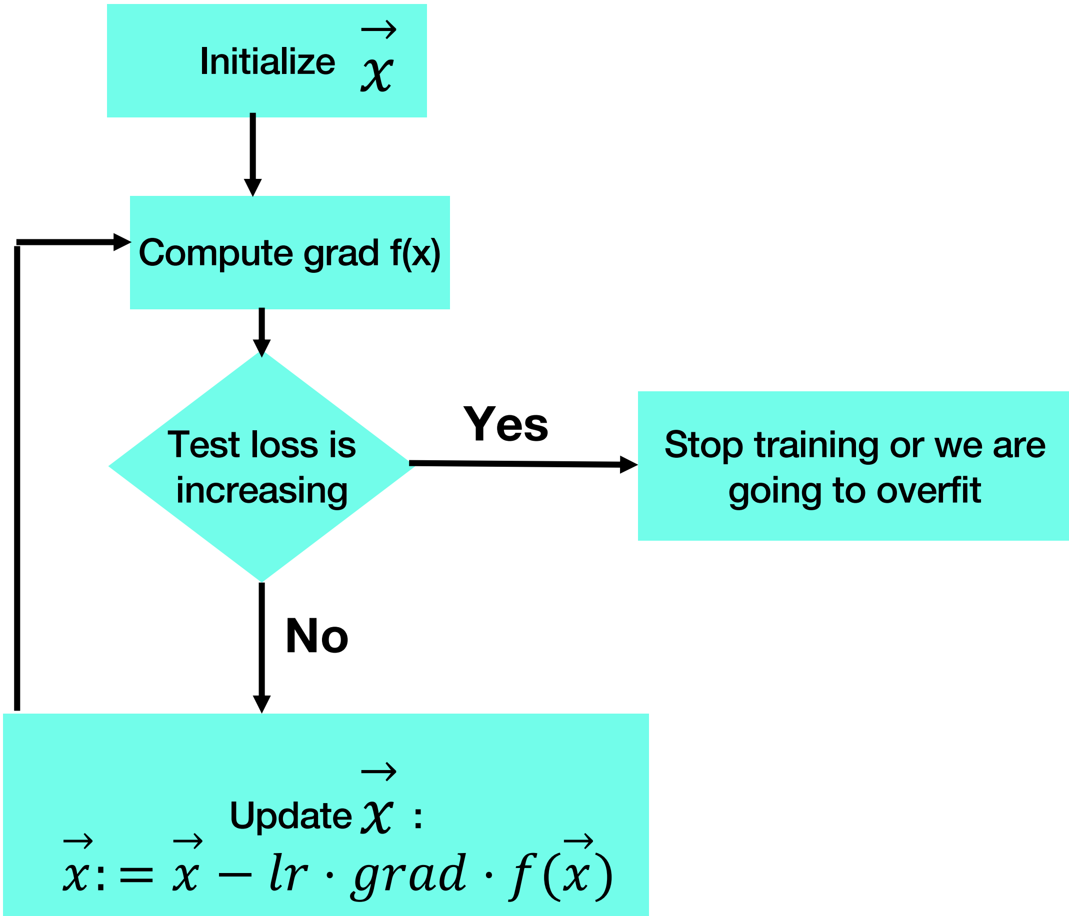
$MeanSquaredError = 1.3$

$TestMeanSquaredError = 24.5$

How to Prevent Overfitting: Early Stopping (1/2)

- A very **simple** and very **efficient** method for preventing overfitting is called “***early stopping***”
- During the gradient descent, we check the test loss at each iteration. When the test loss starts to increase (or stay unchanged) for a few iteration, we stop the training

How to Prevent Overfitting: Early Stopping (2/2)



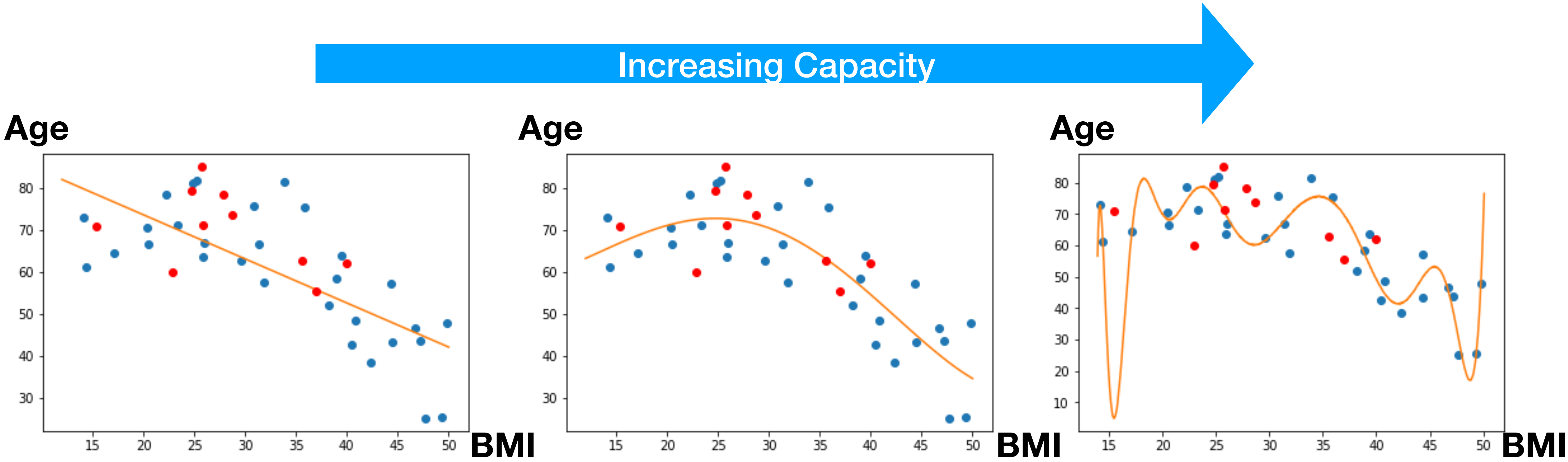
How to Prevent Overfitting: Capacity (1/5)

- Another method is to reduce the ***capacity*** of the model
- Roughly speaking, the capacity of a model is its ability to adapt to a large number of examples
- The capacity of a model will increase with the number of parameters

How to Prevent Overfitting: Capacity (2/5)

- Models with high capacity can learn more complicated relations
- But they overfit more easily
- A model with very high capacity has the ability to memorize all the training examples (the rote learning problem)
- If we reduce the capacity of a model, we can make it less prone to overfitting

How to Prevent Overfitting: Capacity (3/5)



$$age = \theta_0 + \theta_1 \times bmi$$

$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^3 + \theta_4 \times bmi^4$$

$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^3 + \theta_4 \times bmi^4 + \theta_5 \times bmi^5 + \theta_6 \times sin(bmi) + \theta_7 \times log(bmi) + \ldots$$

How to Prevent Overfitting: Capacity (4/5)

- The simplest way to reduce capacity is to remove some parameters in the model:

$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^3 + \theta_4 \times bmi^4$$
$$+\theta_5 \times bmi^5 + \theta_6 \times sin(bmi) + \theta_7 \times log(bmi) + \ldots$$

$$age_{model} = \theta_0 + \theta_1 \times bmi + \theta_2 \times bmi^2 + \theta_3 \times bmi^4 + \theta_4 \times bmi^5$$

How to Prevent Overfitting: Capacity (5/5)

- Another way to reduce the capacity is “force” the model to use small values for the parameters
  - By default, the parameters θ_k can take any value: -100 000, 0.1, 1 000 000
  - If we forbid the model to use very high values for θ_k, we reduce its capacity: it cannot adapt to data as well as before
- Most practical way to forbid high values: “L2 Regularization”

L2 Regularization

- We note $|\vec{\theta}|^2$ the sum of the square of all parameters θ_k (this is called the "L2 Norm")

- Then we add this quantity to the loss we want to minimize:

$$Loss = MeanSquaredError = \frac{1}{N} \cdot \sum_i (model(cig_i, bmi_i, ismale_i) - age_i)^2$$

$$\Downarrow$$

$$Loss = \frac{1}{N} \cdot \sum_i (model(cig_i, bmi_i, ismale_i) - age_i)^2 + \lambda |\vec{\theta}|^2$$

Then we apply Gradient Descent to this new loss

Next

- Next time, we will consider <u>Classification Problems:</u>
  - Predicting if some symptoms are the sign of a disease or not…
  - Predicting if an image represents a cat or a dog
  - Predicting if a text is in French, German or Japanese

Report

- Submit a report “summarizing what overfitting is and ways to detect and prevent overfitting” **in pdf** via PandA
    - Submission due: **next lecture**
    - Name the pdf file as **student id_name**.

Google Colab Notebook

https://shorturl.at/Iy52R