BTS | Barcelona Technology School

Classical Data Analysis

Master in Big Data Solutions 2020-2021

Victor Pajuelo Madrigal

victor.pajuelo@bts.tech

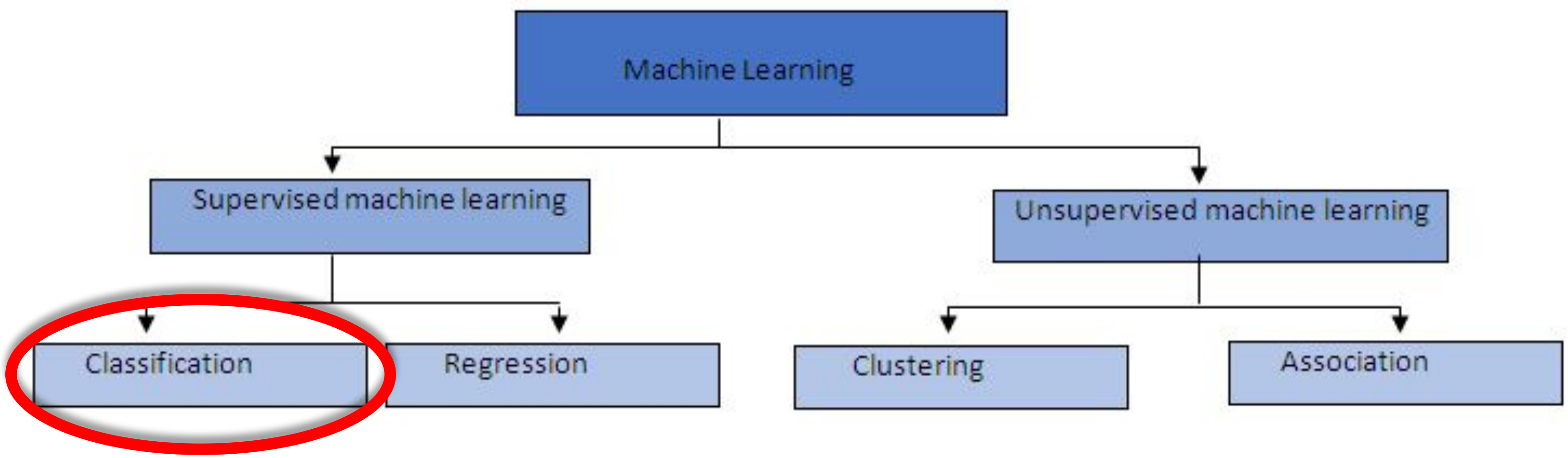
Today's Objective

- Review concepts from previous sessions
- Regression analysis (Polynomial, Ridge, Lasso, Elastic Net)
  - Theory
  - Regularized linear regression with Python

Contents

Previous Session: Logistic Regression

Supervised learning: Classification



The Logistic Regression algorithm solve **classification** problems. Don’t get confused by the name “Regression”!

Logistic Regression

- The main goal of logistic regression is to perform binary classification

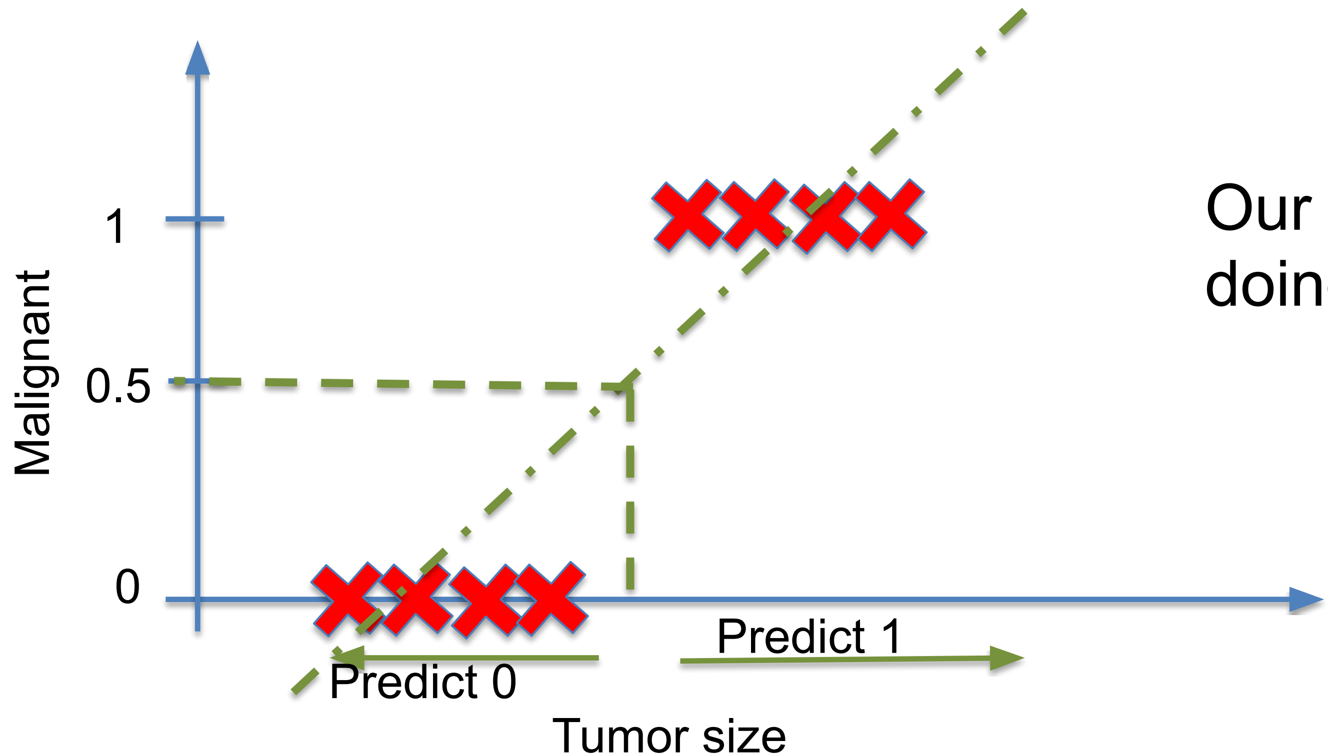
- Examples of applications:
  - In medical research, in the field of **predictive food microbiology**, to describe bacterial growth/no growth interface, in the 0–1 format.
  - Predict the **risk of developing a given disease** based on observed characteristics of the patient.
  - In credit risk modeling in identifying **potential loan defaulters**.

All the above problems are **binary** classification problems.
Possible results:
  - y = 1
  - y = 0

- Feature Engineering plays an important role in regards to the performance of Logistic

Logistic Regression

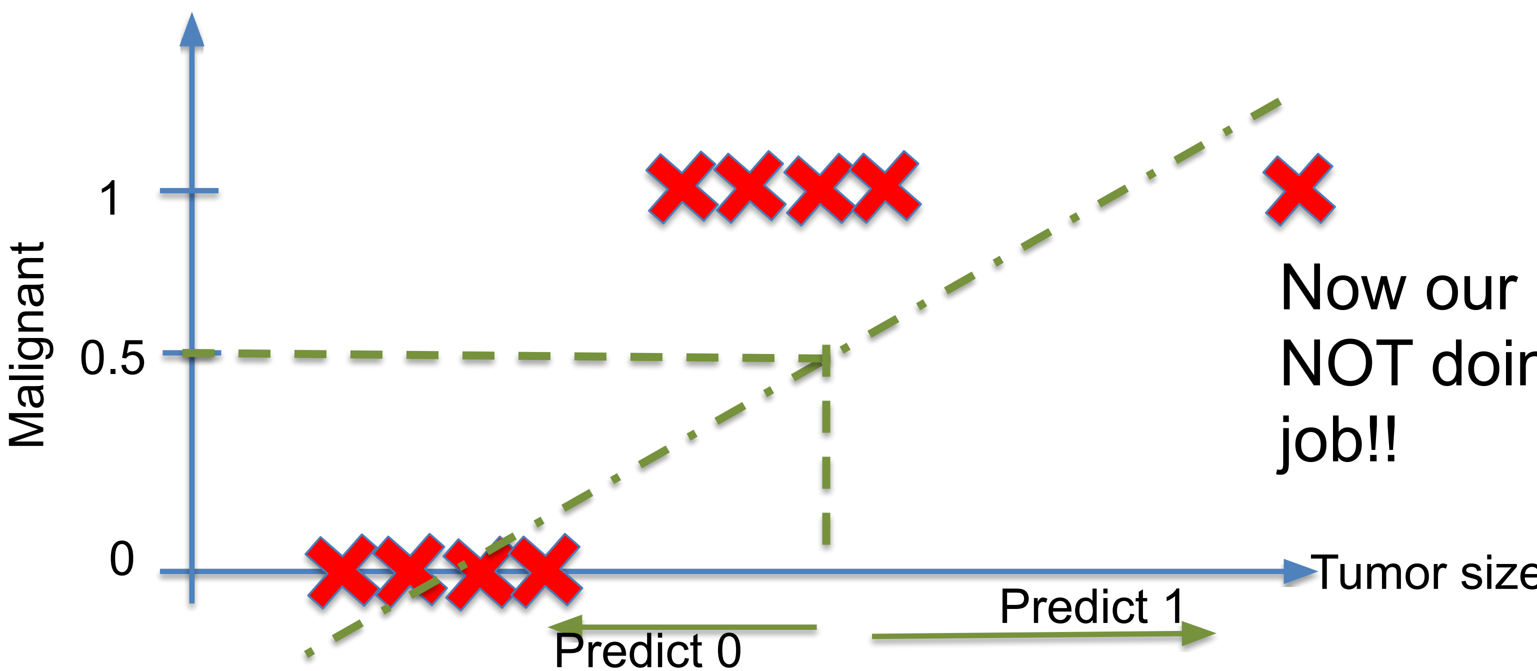
- How to develop a classification algorithm?
  - Let's try applying a linear regression model to classify a tumor as malignant or not!



Our model is doing a great job!!

Logistic Regression

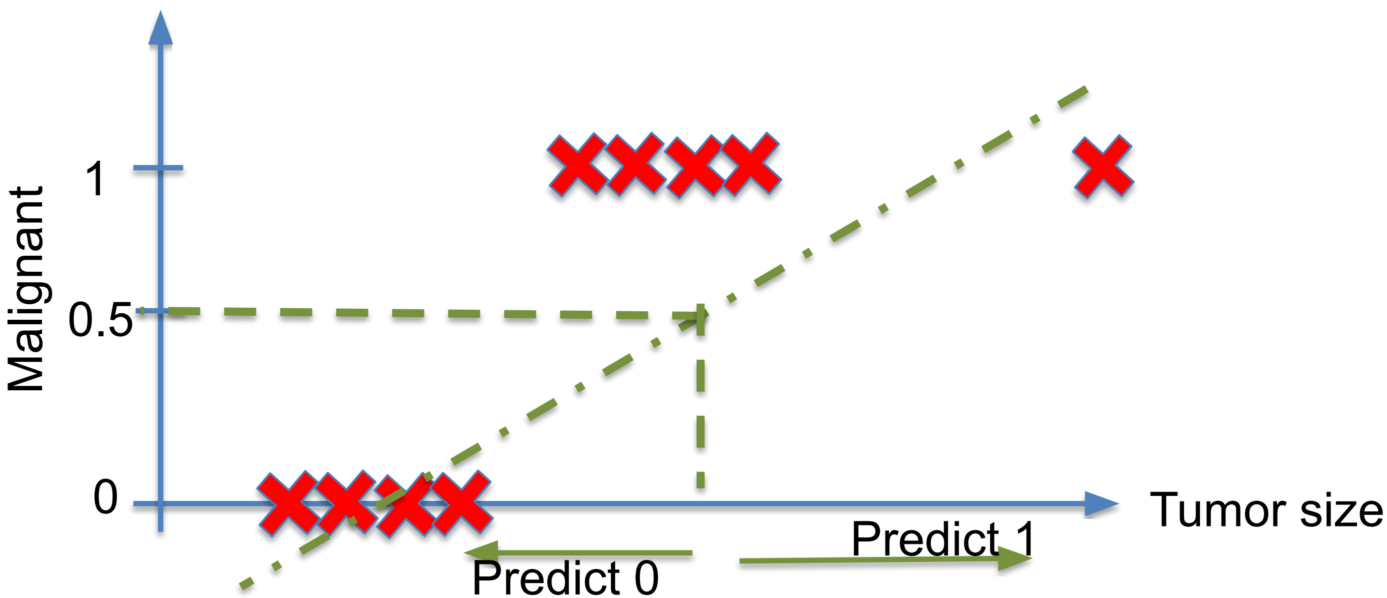
- How to develop a classification algorithm?
  - Let’s try applying a linear regression model to classify a tumor as malignant or not!
  - Let’a add an additional example



Now our model is NOT doing a great job!!

Many positive examples are now classified as negative.

Logistic Regression



- Many positive examples are now classified as negative.
- It also doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$.

Logistic Regression

Linear regression hypothesis function:

$$h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1 + \ ... + \theta_n x_n$$

In Logistic regression we want the output of the hypothesis function to be between 0 and 1. We apply the **sigmoid function (aka logistic function)** to the output of the linear regression, obtaining the hypothesis function for the logistic regression:

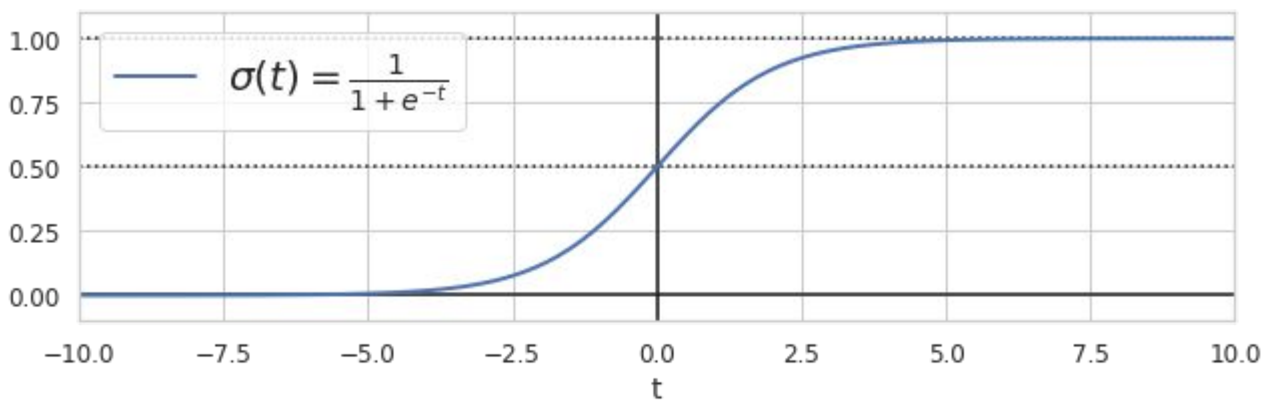
$$h_\theta(x) = g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \ ... + \theta_n x_n)$$

Logistic Regression is **just like a linear regression model**, it computes a weighted sum of the input features (plus a bias), but instead of outputting the linear result directly, it outputs the logistic of that result

Logistic Regression

- Sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}} \quad \longrightarrow \quad \frac{1}{1 + e^{-(\Theta^T X)}}$$



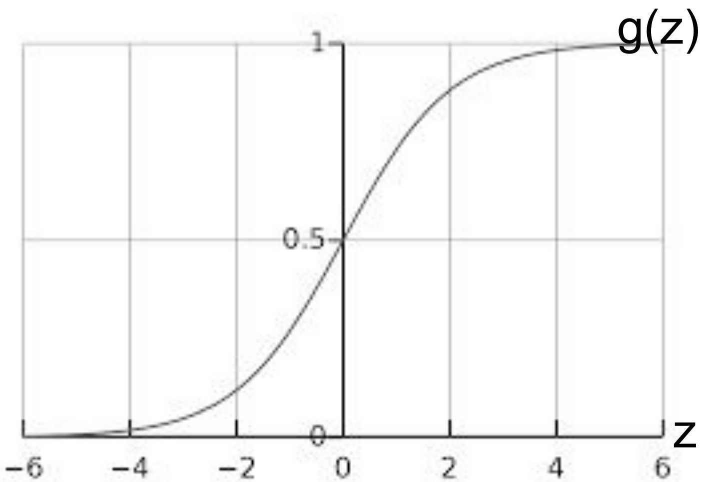
The logistic is a sigmoid function that outputs a number between 0 and 1

Go to @S2C1

Logistic Regression

Logistic Regression hypothesis function:

$$g(z) = \frac{1}{1 + e - ^{z}}$$



When our hypothesis $h_\theta(x)$ outputs a number, we treat that value as the estimated probability that y=1 on input x

Notice that g(z) tends towards 1 as $z \rightarrow \infty$, and g(z) tends towards 0 as $z \rightarrow -\infty$. Moreover, g(z), and hence also h(x), is always bounded between 0 and 1.

Logistic Regression aka *Logit Regression*

- It is commonly used to estimate the **probability that an instance belongs to a particular class**
  - What is the probability that an email is spam?
  - What is the probability that I will pass this class?
- If the estimated probability is above 50%, the model is predicting that the instance belongs to that class (positive class, labelled 1)
- If the estimated probability is below 50%, the model predicts that the instance belongs to the other class (negative class, labelled 0)

Making predictions is easy…

$$\hat{y} = \begin{cases} 0 \rightarrow \hat{p} < 0.5 \\ 1 \rightarrow \hat{p} \geq 0.5 \end{cases}$$

Logistic Regression

Cost function

- Alright, we know now that Logistic Regression estimates probabilities and makes predictions out of it but…
  - How is it trained?
- The **objective** of the training is to set the parameter vector hθ() so that…
  - Estimates high probabilities for positive instances (y=1)
  - Estimates low probabilities for negative instances (y=0)

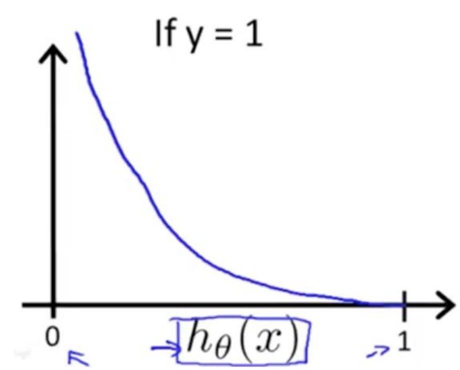
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Logistic Regression

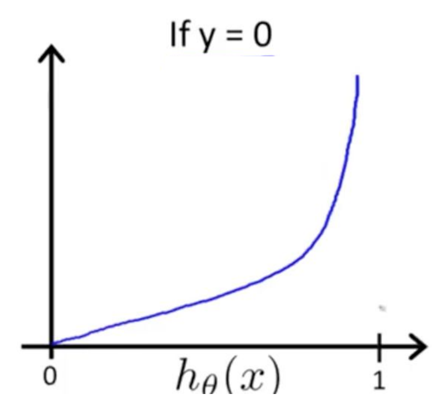
Cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

- In this example for a single instance (x), -log(hθ(x)), grows very large when hθ(x) approaches 0
  - So the cost will be very large if the model predicts a probability close to 0 for a positive instance



- -log(1- hθ(x)), grows very large when hθ(x) approaches 1
  - So the cost will be very large if the model predicts a probability close to 1 for a negative instance

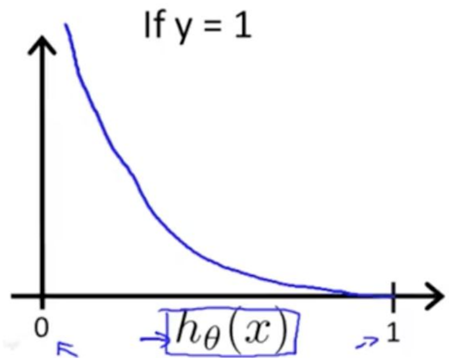


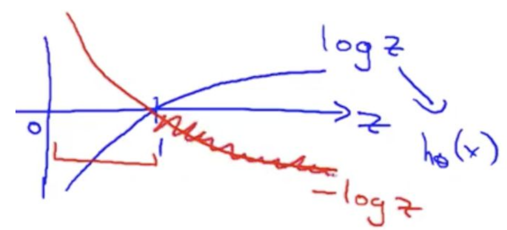
Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





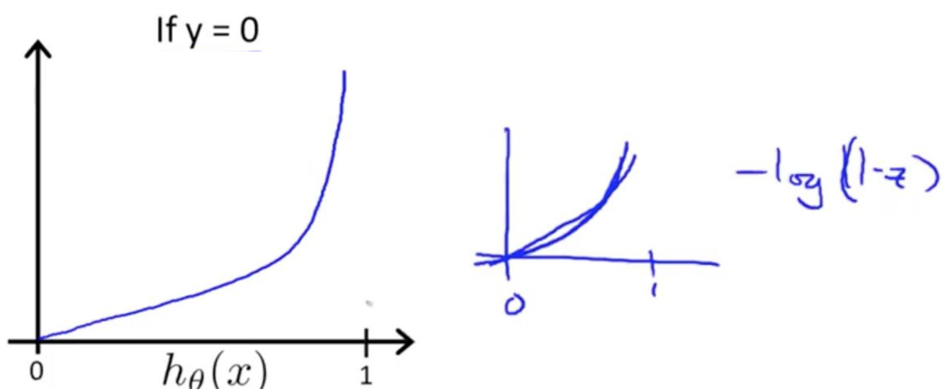
- if the hypothesis predicts that y = 1 and y = 1 cost = 0.
- if the hypothesis predicts that y = 0 and y = 1 cost -> inf.

Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



- if the hypothesis predicts that y = 0 and y = 0 cost = 0.
- if the hypothesis predicts that y = 1 and y = 0 cost -> inf.

Logistic Regression

Cost function

- Given the logistic regression model, how do we fit the θ parameters?

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

- This cost function can be written as follow

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

- This cost function can be derived from statistics using the principle of maximum likelihood estimation

- We can apply the **gradient descent** algorithm to find the parameter that minimize the cost function.

Logistic Regression

Guaranteed success! (almost)

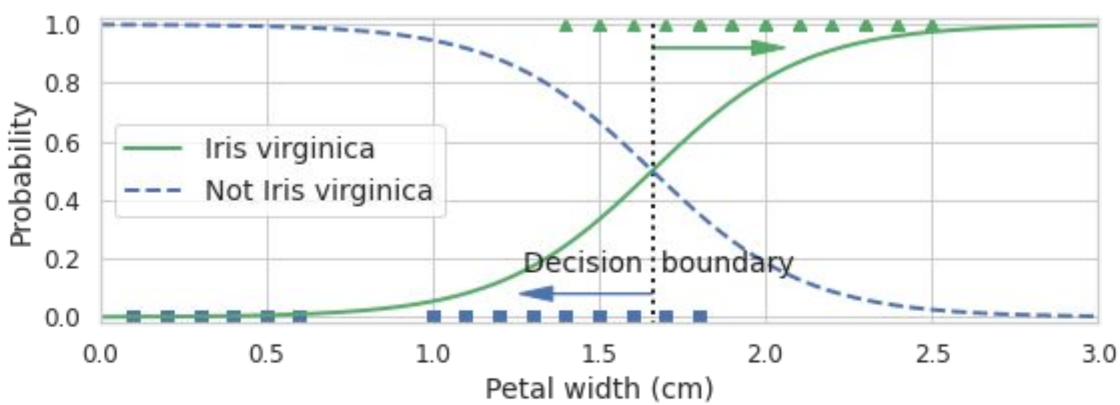
$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

- The cost function for the logistic regression is **convex**, so that *Gradient Descent* or any other optimization algorithm, will guarantee to **find the global minimum**
    - That is… if the learning rate is not too large and one has enough time to wait
    - This can be controlled with the **tolerance criteria** and the **number of iterations** defined in Sklearn

Logistic Regression

A note on Decision Boundaries

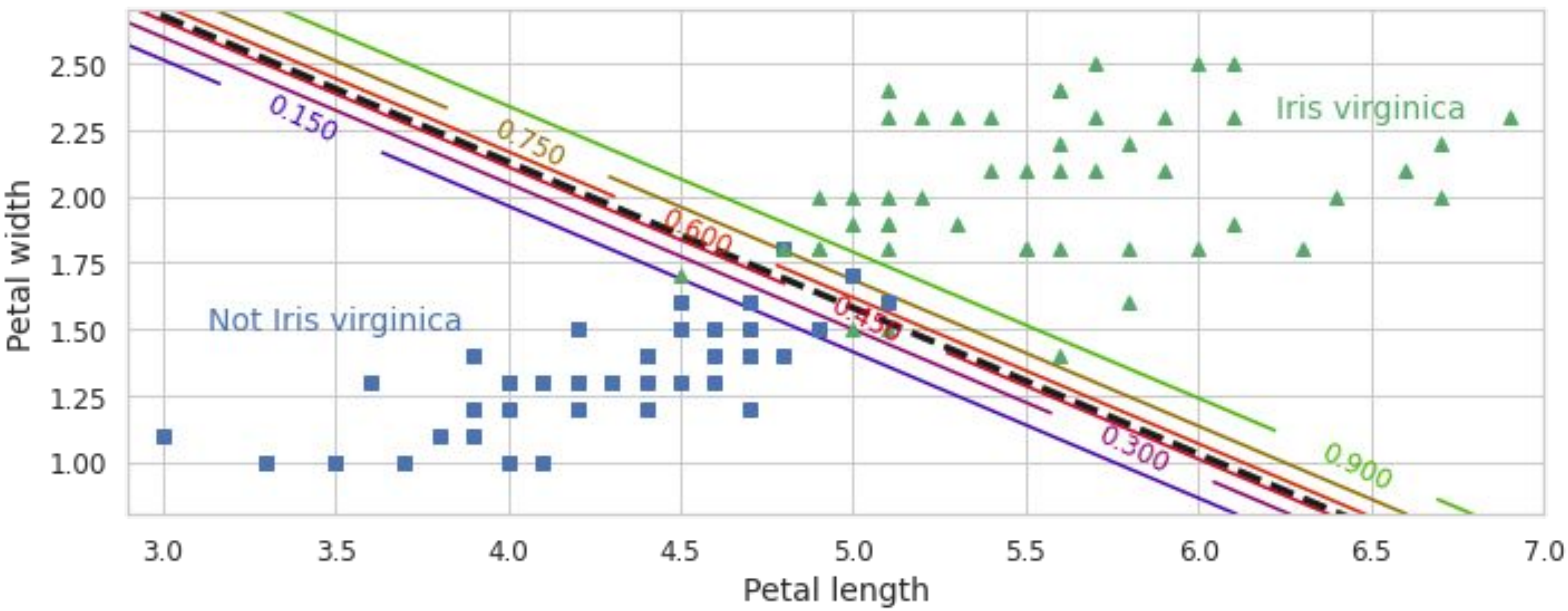
- Decision boundaries in LogReg are usually set when negative and positive class probabilities cross at y=0.5
- If you want to get probabilistic results rather than classes you can use sklearns `predict_proba()` instead of `predict()`
- That can be used for further refinement of the decision boundary (in case that one wants to classify something as positive if it goes below 0.5)



Go to @S2C2

Logistic Regression
A note on Decision Boundaries

- `predict_proba()` returns the probabilities of the class to be the class that represents in its vector space



Go to @S2C2

Logistic Regression

A note on Decision Boundaries

```
>> np.set_printoptions(suppress= True,
formatter={'float_kind':'{:0.9f}'.format})
>> y_proba = log_reg.predict_proba(X_new)
```

```
[0.999999998, 0.000000002]
```

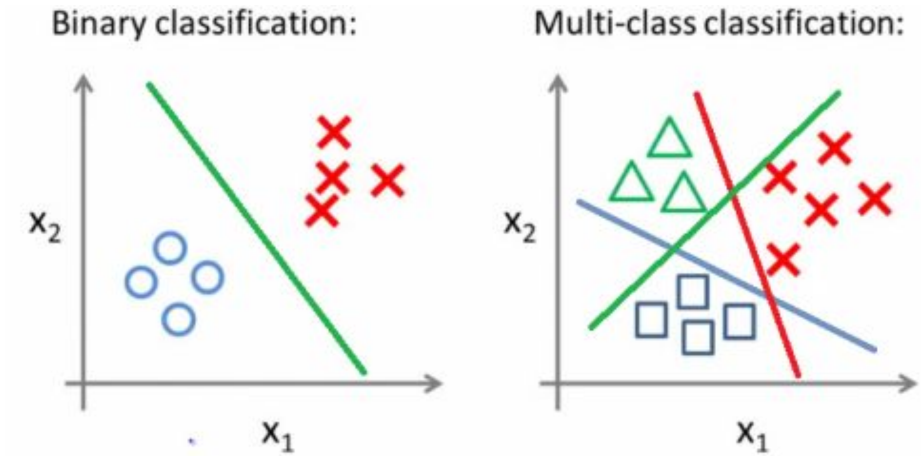
- Probabilities in results always sum up to 1

Go to @S2C2

Logistic Regression

Multi-class Classification

- **One vs. all classification**
  - classification problem with N distinct classes
  - Train N distinct binary classifiers, each designed for recognizing a particular class

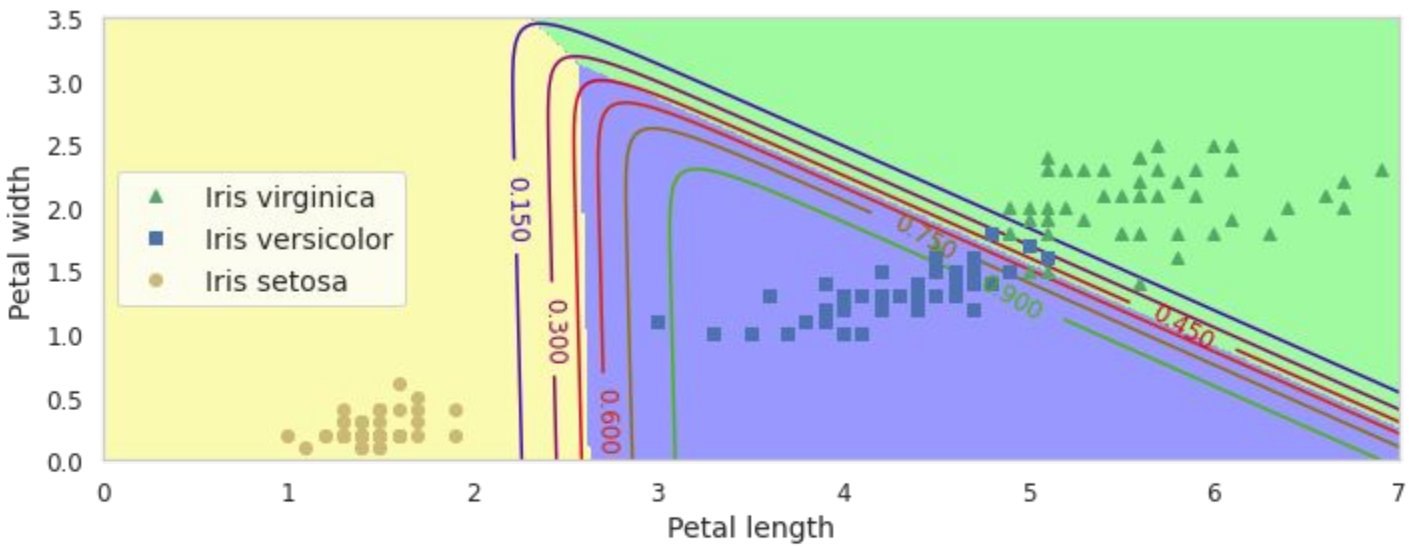


MBDS 2020-2021

Logistic Regression

Multi-class Classification

We can use multi class classification in Sklearn LogReg by passing the `multiclass` parameter



Go to @S2C2

Logistic Regression

Dealing with overfit and underfit

A model generalization error can be expressed as:

- BIAS: this part of the generalization error is due to wrong assumptions, such as assuming that the data is linear when it is actually quadratic. High-bias model is highly likely to underfit the training data

- VARIANCE: this error is due to having a model that is too sensitive to small variations in training data. A model with a lot of degrees of freedom, will have high variance and it will overfit our training data

Logistic Regression

How do we regularize those errors?

- Lasso, L1 penalty
  - *L1 penalizes sum of absolute value of weights.*
- Ridge, L2 penalty (added by default by Sklearn)
  - *L2 regularization penalizes sum of square weights.*

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda\sum_{j=0}^{M}|W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda\sum_{j=0}^{M}W_j^2}_{\text{Regularization Term}}$$

In Sklearn, this value is represented by C, and it is the inverse of lambda. So, the higher the value of C, the less the model will be regularized

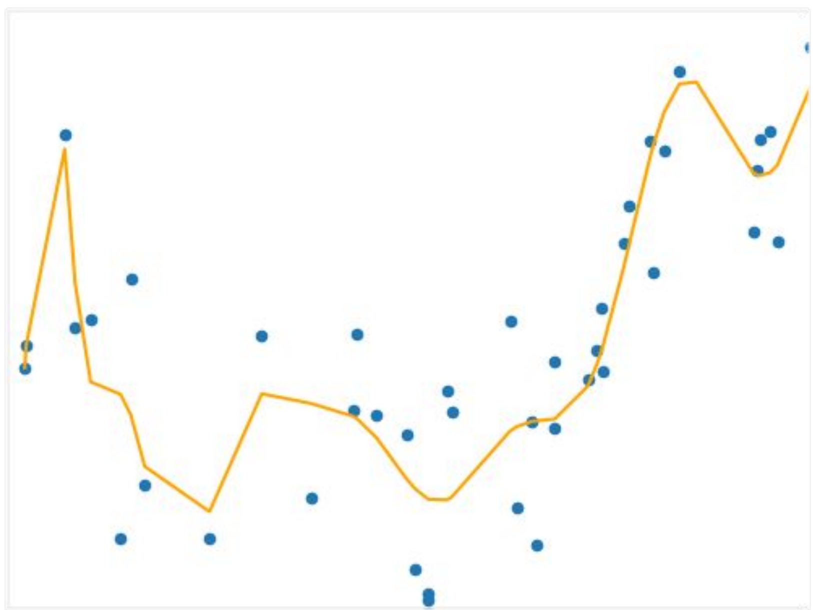
Huge part of next session will be spend on understanding this fully

Polynomial Regression

Polynomial regression

Limitation of linear Regression

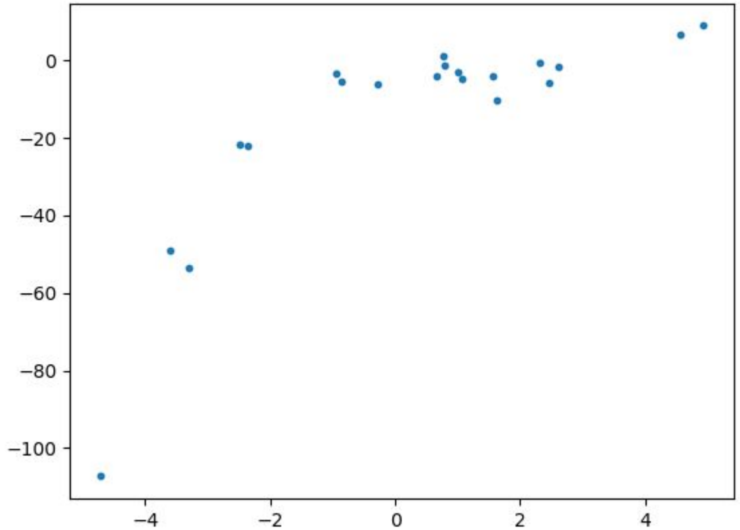
- **Linear regression** requires the **relation** between the dependent variable and the independent variable to be **linear**.



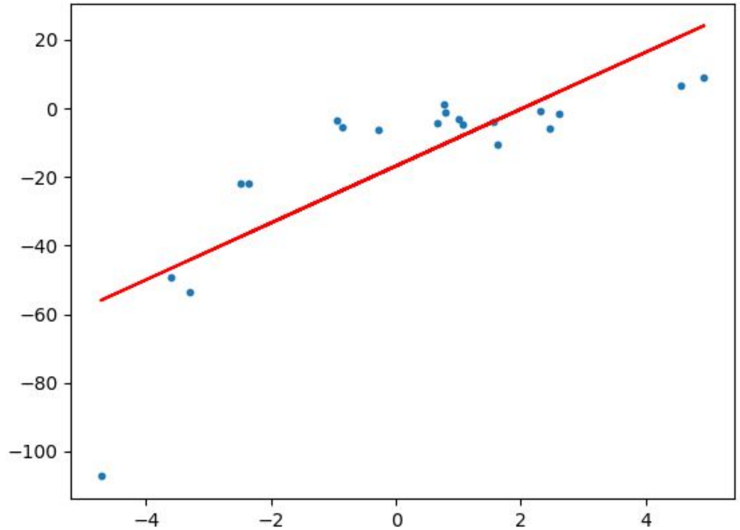
Can linear models be used to fit non-linear data?

Polynomial regression

Why Polynomial Regression?



Linear regression model

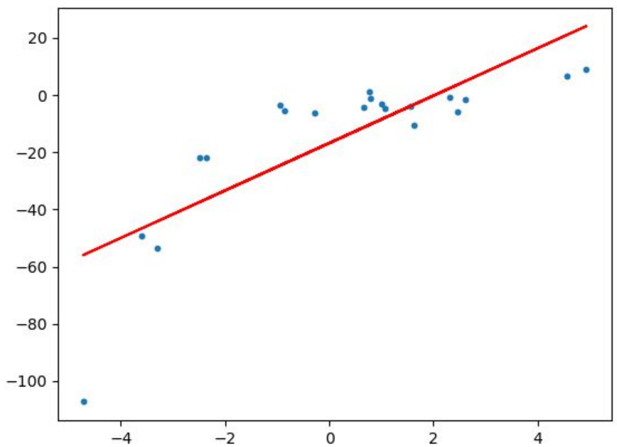


The model is unable to capture the patterns in the data. This is an example of **under-fitting**.

Polynomial regression

Why Polynomial Regression?

Linear regression model



How to overcome **under-fitting**?

Increase the complexity of the model.
We need to add more features to the data

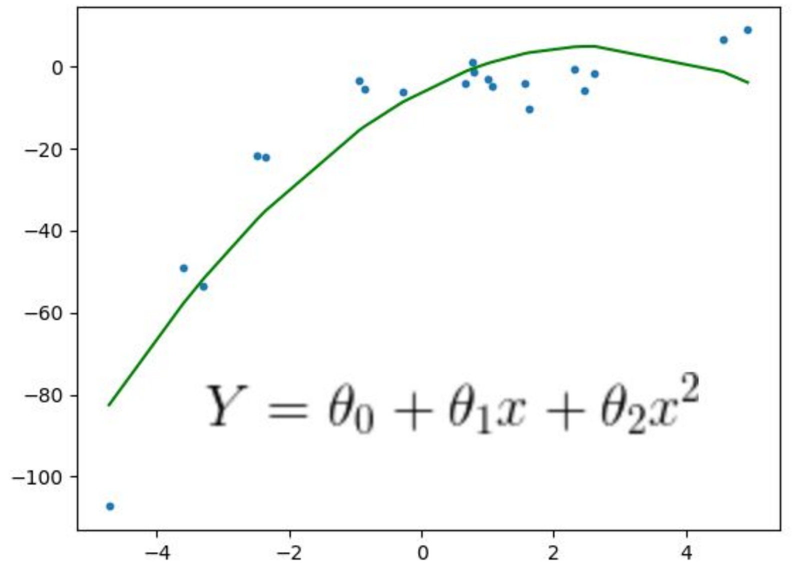
$$Y = \theta_0 + \theta_1 x \quad \Rightarrow \quad Y = \theta_0 + \theta_1 x + \theta_2 x^2$$

This is still considered to be a **linear model** as the coefficients/weights associated with the features are still linear. x^2 is only a new feature.

Polynomial regression

Why Polynomial Regression?

Linear regression model 2nd degree polynomial

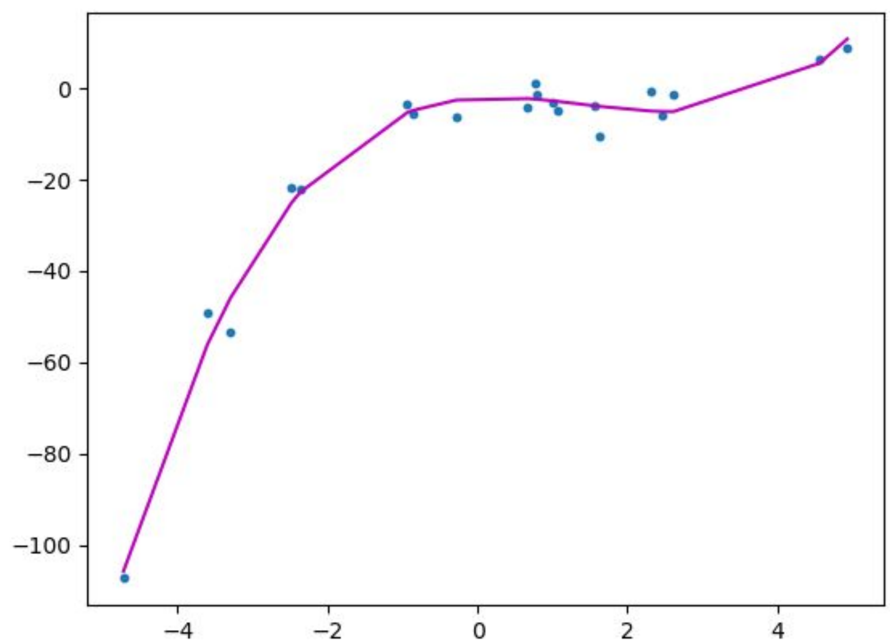


It is clear from the plot that the quadratic curve (2nd degree polynomial) is able to fit the data better than the linear line.

Polynomial regression

Why Polynomial Regression?

Linear regression model polynomial of degree 3

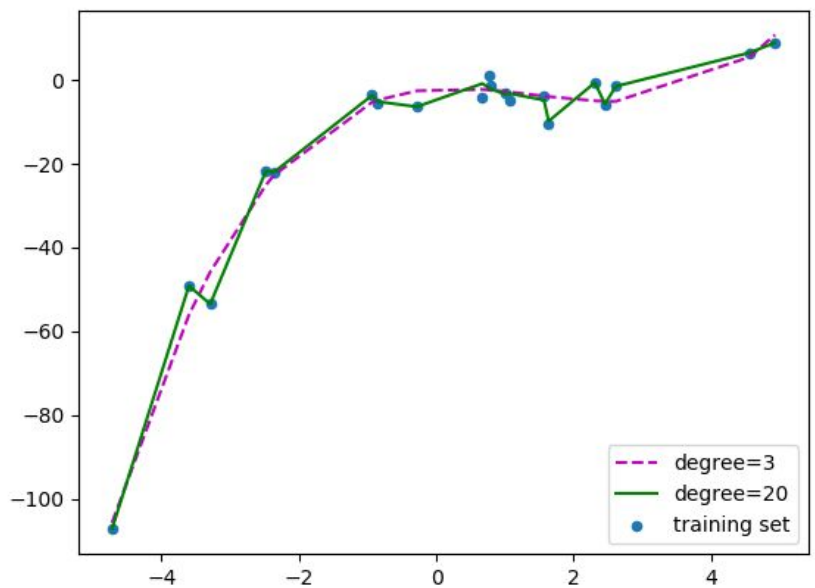


It passes through more data points than the quadratic and the linear plots.

Polynomial regression

Why Polynomial Regression?

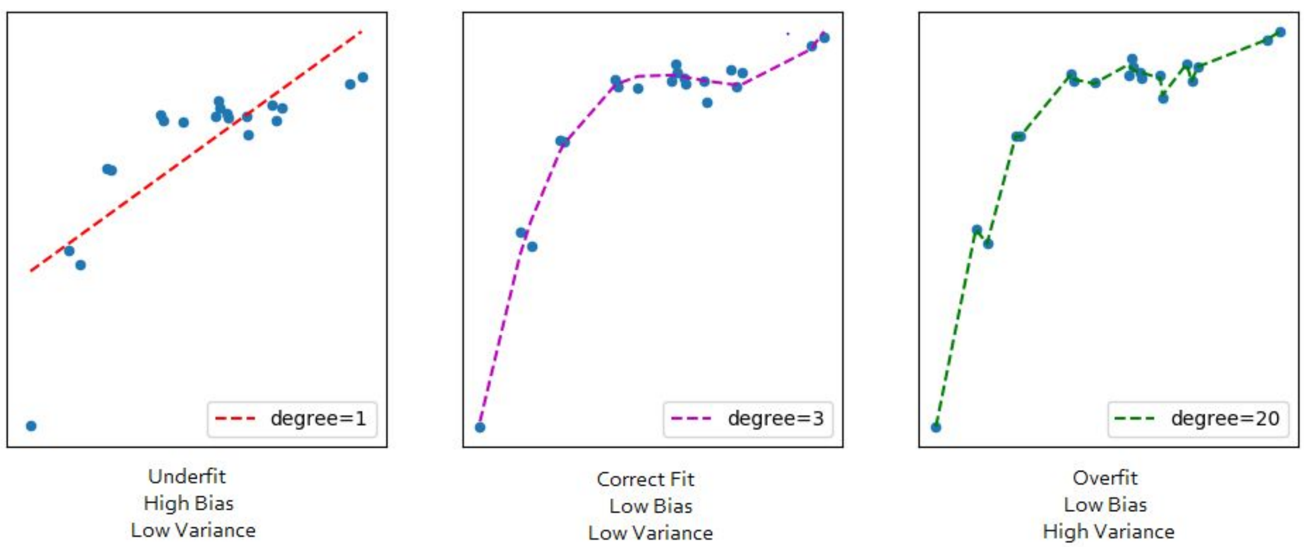
Linear regression model polynomial of degree 20



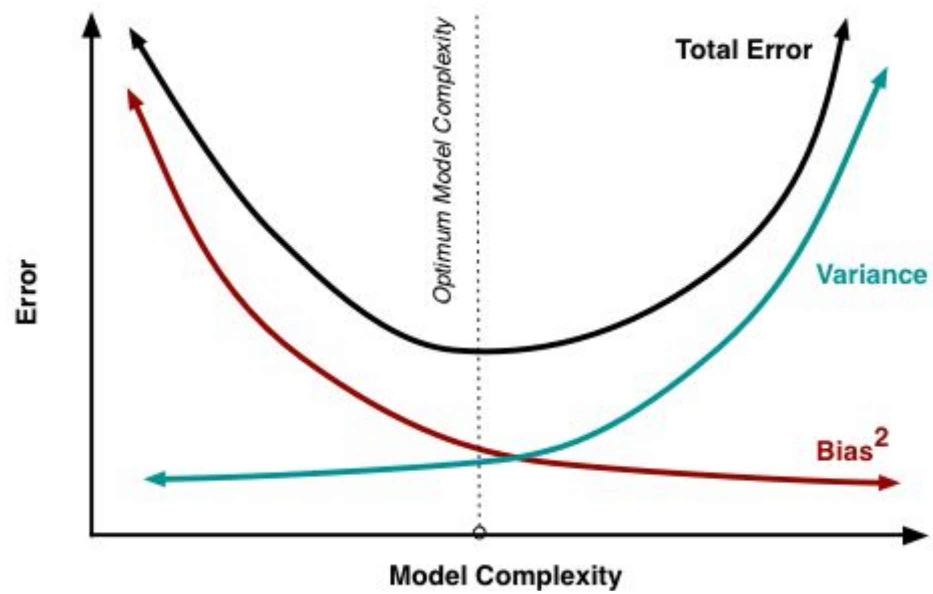
It passes through more data points but here we have a problem: **Overfitting**, i.e., the model is also capturing the noise in the data and even if it works well on the training set it will fail to generalize (the performance in the test set will be poor).

Bias vs Variance trade-off

- **Bias**: error due to the model's simplistic assumptions in fitting the data. A **high bias** means that the model is unable to capture the patterns in the data and this results in **underfitting**.
- **Variance:** error due to the complex model trying to fit the data. **High variance** results in **over-fitting** the data. Variance refers to how much the model is dependent on the training data.



Bias vs Variance trade-off



Bias is related with a model failing to fit the training set and variance is related with a model failing to fit the testing set.

Regularization: Ridge, Lasse and Elastic Net

Regularization

How to address overfitting

- In classical linear regression if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This might lead to **overfitting** in small datasets.

- Potential solutions:
  - Reduce Number of feature (Manually or Model selection algorithm)
  - **Regularization**
  - Collect More data
  - **Cross Validation**
  - Ensembling (Combine prediction from multiple separate models)

Regularization

Cross Validation

Regularization

How to address overfitting

- In classical linear regression if the model feels like one particular feature is particularly important, the model may place a large weight to the feature. This might lead to **overfitting** in small datasets.

- **Regularization techniques consist of modifying the cost function, adding a penalty term, to restrict the values of our coefficients.**

- Regularization techniques:
  - **Lasso**, also called L1-Norm
  - **Ridge**, also called L2-Norm
  - **Elastic Net**: is a combination of Lasso and Ridge

Regularization

LASSO

- Adds an additional term to the cost function, adding the sum of the coefficient values (the L-1 norm) multiplied by a constant lambda.

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

- Lambda set the coefficients of the bad predictors mentioned above 0 (**feature selection**).
- If lambda=0, we effectively have no regularization.
- Large lambda leads coefficients to 0.

Regularization

RIDGE

- Sums the **squares of coefficient** values (the L-2 norm) and multiplies it by some constant lambda.

$$\hat{\beta}^{ridge} = \underset{\beta \in \mathbb{R}}{argmin} \|y - XB\|_2^2 + \lambda \|B\|_2^2$$

- will decrease the values of coefficients, but is unable to force a coefficient to exactly 0 (**No feature selection**).
- If lambda=0, we effectively have no regularization
- It will also select groups of colinear features (grouping effect)

[Analysis of both Lasso and Ridge regression has shown that neither technique is consistently better than the other; try both methods to determine which to use!]

Regularization

Elastic Net

- Includes both L-1 and L-2 norm regularization terms.

$$\hat{\beta} \equiv \underset{\beta}{\operatorname{argmin}}(\|y - X\beta\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1).$$

- It seems to have the predictive power better than Lasso, while still performing feature selection. We therefore get the best of both worlds, performing **feature selection** of Lasso with the **feature-group selection** of Ridge.

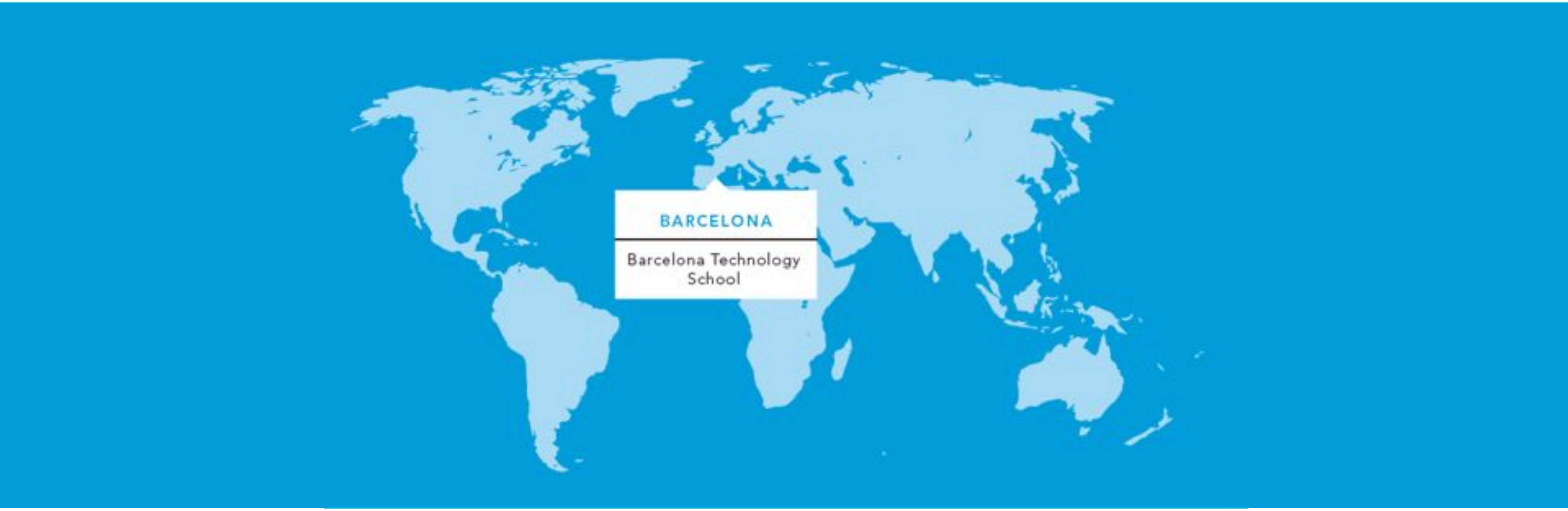
Regularization

- So what do we use?
  - Ridge is faster to converge, Lasso tends to bounce around during Gradient Descent
  - It is always preferable to have at least a little bit of regularization, so try to avoid regression without regularization
  - Ridge is good default to start with, and to check convergence potential (whether we train or not)
  - If you suspect that only few features are useful (and you don't know which ones) you should go for Lasso or ElasticNet because they tend to reduce the useless feature's weights down to zero

Resources

- Tan, Pang-Ning. *Introduction to data mining*. Pearson Education India, 2006.

- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. New York: Springer series in statistics, 2001.


BARCELONA
Barcelona Technology School