Introduction to Statistical Machine Learning CSC/DSCC 265/465

Lecture 6: Supervised Learning – Part III

Cantay Caliskan

Notes and updates

Notes and updates

- The deadline for the 2nd Problem Set is **Friday, February 4, 11:59 PM!**
- Anything you want to share?

Today is special!

- Why?
- **2/2/22 2:22:22:22 PM** will happen in this lecture.

Plan for today

- ***Example with Var-Cov matrix***
- ***Types of Gradient Descent***
- ***Maximum Likelihood Estimation***
- ***Classification***
- ***Model: Logistic Regression***

Plan for today

- ***Example with Var-Cov matrix***
- *Types of Gradient Descent*
- *Maximum Likelihood Estimation*
- *Classification*
- *Model: Logistic Regression*

Example of Var-Cov Matrix

- Let's go through a simple example:
- **Reminder:** $cov(\boldsymbol{X}) = \mathrm{E}[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T]$

Student	Math	English	Art
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

Raw data

$$\mathbf{X} = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

$\mathbf{X}$ matrix

$$\mathrm{E}[\mathbf{X}] = \begin{bmatrix} 66 & 60 & 60 \\ 66 & 60 & 60 \\ 66 & 60 & 60 \\ 66 & 60 & 60 \\ 66 & 60 & 60 \end{bmatrix}$$

$\mathrm{E}[\mathbf{X}]$

$$\mathbf{X} - \mathrm{E}[\mathbf{X}] = \begin{bmatrix} 24 & 0 & 30 \\ 24 & 30 & -30 \\ -6 & 0 & 0 \\ -6 & 0 & 30 \\ -36 & -30 & -30 \end{bmatrix}$$

$\mathbf{X} - \mathrm{E}[\mathbf{X}]$

Example of Var-Cov Matrix

- Let's go through a simple example:
- **Reminder:** $cov(\boldsymbol{X}) = \mathrm{E}[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T]$

$$\begin{bmatrix} 24 & 24 & -6 & -6 & -36 \\ 0 & 30 & 0 & 0 & -30 \\ 30 & -30 & 0 & 30 & -30 \end{bmatrix} \begin{bmatrix} 24 & 0 & 30 \\ 24 & 30 & -30 \\ -6 & 0 & 0 \\ -6 & 0 & 30 \\ -36 & -30 & -30 \end{bmatrix}$$

$$\begin{bmatrix} 2520 & 1800 & 900 \\ 1800 & 1800 & 0 \\ 900 & 0 & 3600 \end{bmatrix}$$

$$\begin{bmatrix} 2520/5 & 1800/5 & 900/5 \\ 1800/5 & 1800/5 & 0/5 \\ 900/5 & 0/5 & 3600/5 \end{bmatrix}$$

$(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T$ $(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T$ $\mathrm{E}[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T]$

Question: What is 5?

Answer: Number of rows in ***X***

Example of Var-Cov Matrix

- Let's go through a simple example:
- **Reminder:** $cov(\boldsymbol{X}) = \mathrm{E}[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^T]$

$$\begin{bmatrix} 2520/5 & 1800/5 & 900/5 \\ 1800/5 & 1800/5 & 0/5 \\ 900/5 & 0/5 & 3600/5 \end{bmatrix} = \begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix}$$

Question: Is this population or sample variance-covariance matrix?

Question: How is the sample variance-covariance matrix different?

Plan for today

- *Example with Var-Cov matrix*
- ***Types of Gradient Descent***
- *Maximum Likelihood Estimation*
- *Classification*
- *Model: Logistic Regression*

Types of Gradient Descent

Types of Gradient Descents

1. Stochastic Gradient Descent

- Iterates over each training example in the dataset while updating the model

2. Batch Gradient Descent

- Also called vanilla gradient descent
- Calculates the error for whole training dataset at once
- The model is updated only after all examples have been evaluated

3. Mini Batch Gradient Descent

- The method of having a combination of concepts of both SGD and Batch Gradient Descent
- It splits the dataset into small batches and then performs the update on each of these batches balancing

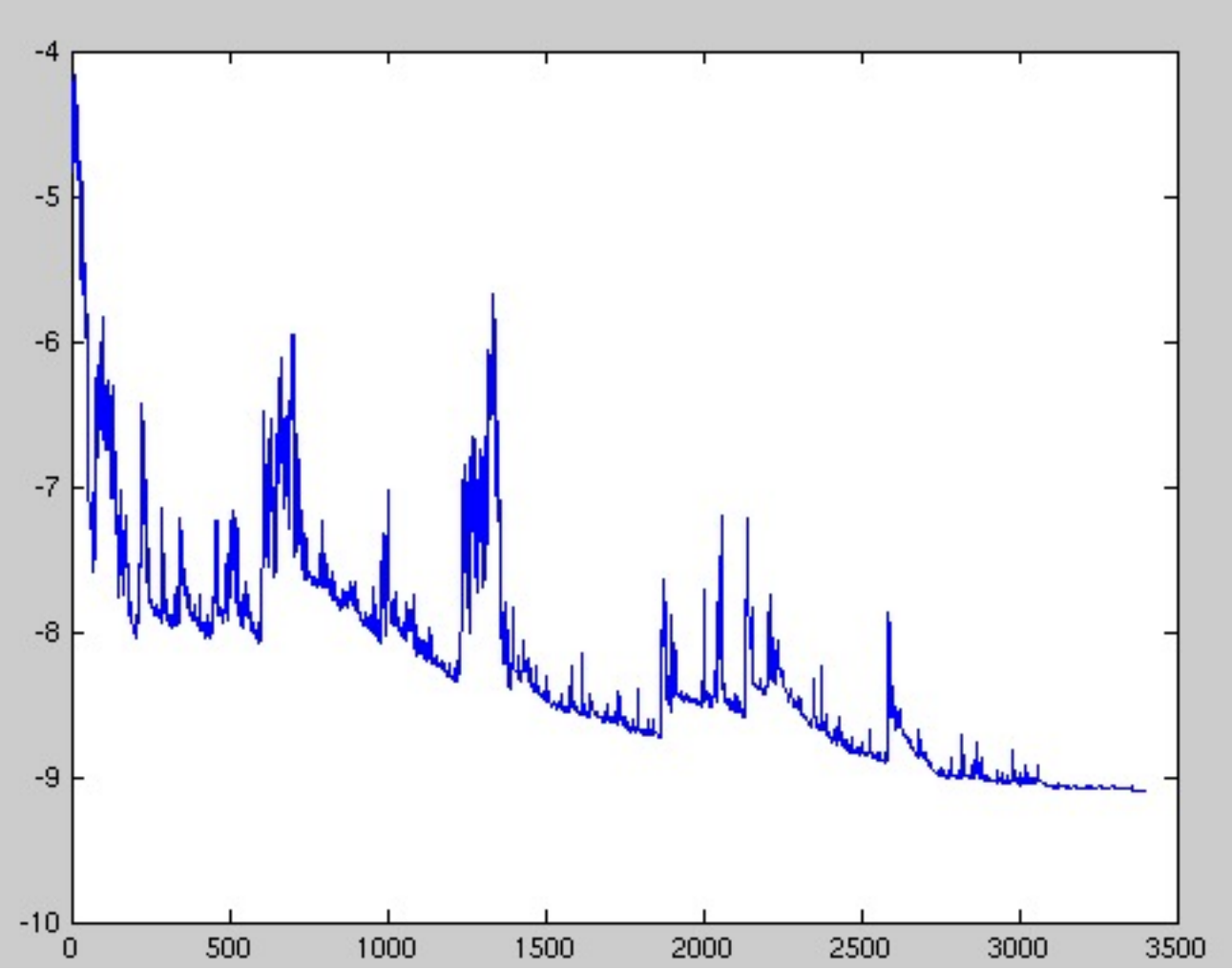
Stochastic gradient descent (SGD)

- In this method, one training sample (example) is passed through the model at a time and the parameters of your model are updated with the computed gradient.
  - Analogy from cross-validation: ***LOOCV***

- Example: If the training set contains 100 samples, then the parameters are updated 100 * Number of iterations many times

- Question: How do we select the next training sample?
- Answer: Randomly!

Stochastic gradient descent (SGD)



With SGD, you can have fluctuations in the cost function output

Advantages of SGD

Advantages:

- Easier to fit into memory due to a single training sample being processed in the model
- Computationally fast as only one sample is processed at a time
- For larger datasets, it can converge faster as it causes updates to the parameters more frequently
- Due to frequent updates, the steps taken towards the minima of the loss function have oscillations which help getting out of local minima

Disadvantages of SGD

Disadvantages:

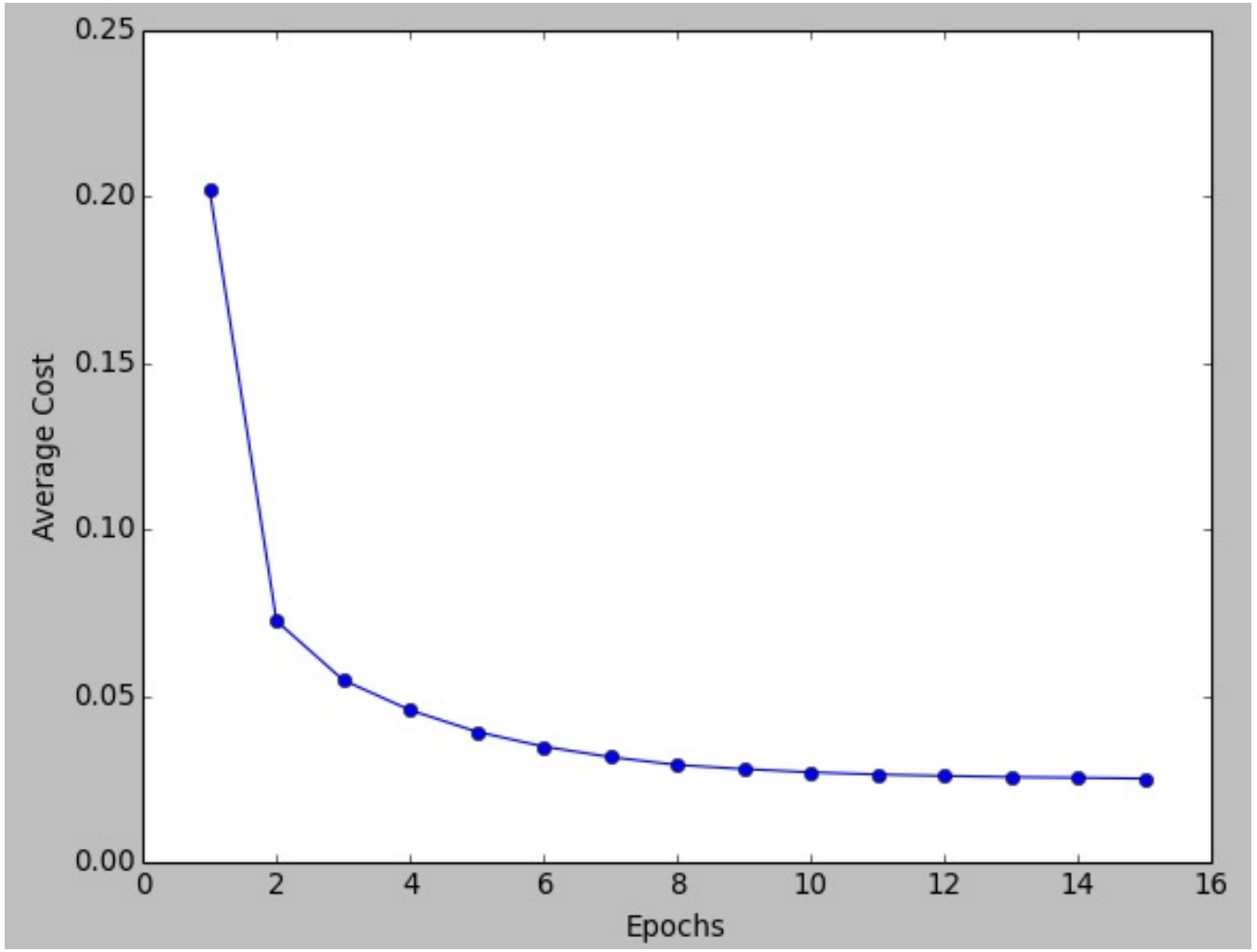
- Due to frequent updates the steps taken towards the minima are very noisy. This can often lead the gradient descent into other directions
- Also, due to noisy steps it may take longer to achieve convergence to the minima of the loss function
- Frequent updates are computationally expensive due to using all resources for processing one training sample at a time
- It loses the advantage of vectorized operations as it deals with only a single example at a time

Batch Gradient Descent

- Instead of updating the parameters of the model after calculating the loss function for every training sample:
  - The parameters are updated after **all the training examples** have been passed through the network
  - Analogy from cross-validation: ***Train/Test Split***

- Example: If the training dataset contains 100 training examples, then the parameters are updated when the gradient descent equation is iterated only once.
  - And the algorithm is completed when max. number of iterations is achieved.

Batch gradient descent



Here, we move more smoothly towards the solution

Advantages of Batch Gradient Descent

Advantages:

- Less oscillations and noisy steps taken towards the global minima of the loss function due to updating the parameters by computing the average of all the training samples rather than the value of a single sample

- It can benefit from the vectorization which increases the speed of processing all training samples together

- It produces a more stable gradient descent convergence and stable error gradient than stochastic gradient descent

- It is computationally more efficient as all computer resources are not being used to process a single sample

Disadvantages of Batch Gradient Descent

Disadvantages:

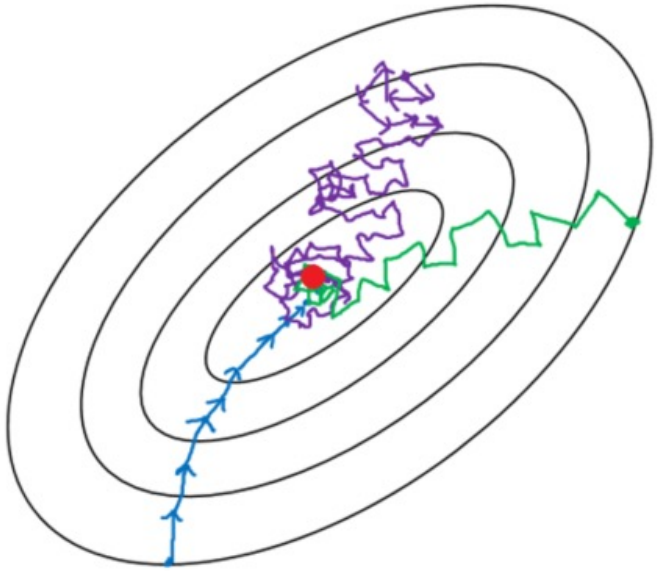
- Sometimes a stable error gradient can lead to a local minimum and unlike stochastic gradient descent no noisy steps are there to help get out of local minimum
- The entire training set can be too large to process in the memory due to which additional memory might be needed
- Depending on computer resources it can take too long for processing all the training samples as a batch

Mini Batch Gradient Descent

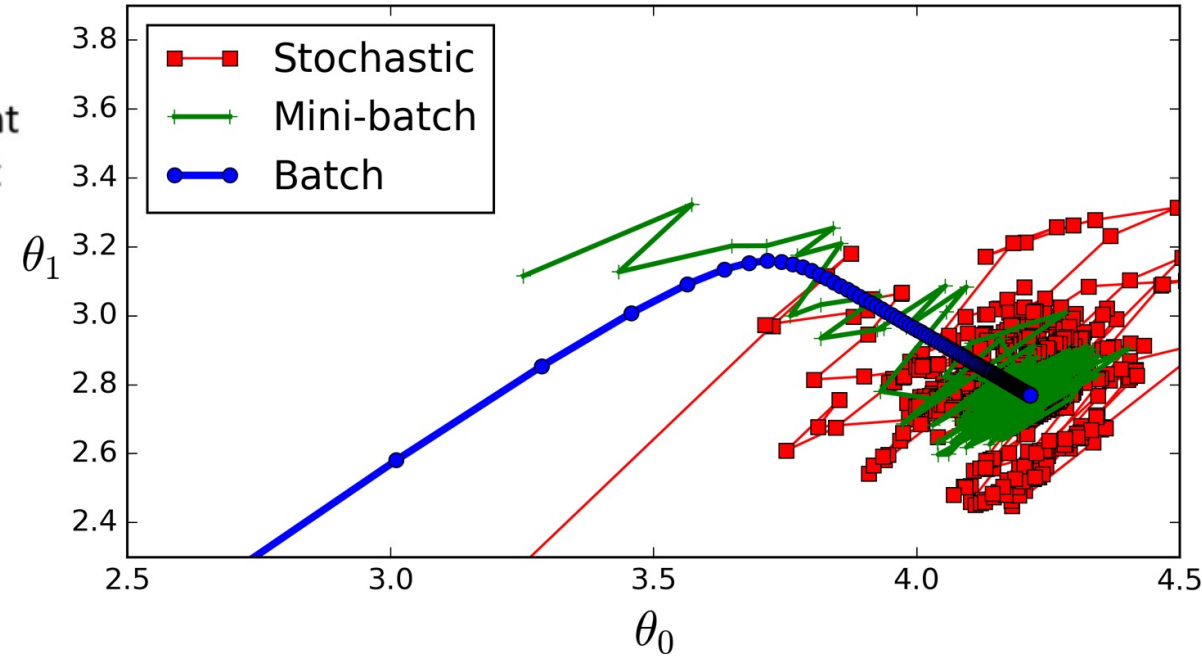
- A mixture of stochastic and batch gradient descent
- The training set is divided into multiple group called “mini batches”
- Each batch has some training samples in it
- A single batch is passed through the model which computes the loss of every sample in the batch and uses their average to update the parameters of the model
- Analogy from cross-validation: ***K-Fold***

- Example: The training set has 100 training examples which is divided into 5 batches with each batch containing 20 training examples.

Mini Batch Gradient Descent



Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent



We have a little bit of SGD and Batch GD here!

Advantages of Mini Batch Gradient Descent

Advantages:

- Easily fits in the memory
- Computationally efficient
- Benefit from vectorization
- If stuck in local minima, some noisy steps can lead the way out of them
- Average of the training samples produces stable error gradients and convergence

Table 4-1. Comparison of algorithms for Linear Regression

Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	n/a
SVD	Fast	No	Slow	0	No	`LinearRegression`
Batch GD	Slow	No	Fast	2	Yes	`SGDRegressor`
Stochastic GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	`SGDRegressor`

We didn't cover this one !

Source: O'Reilly

Plan for today

- *Example with Var-Cov matrix*
- *Types of Gradient Descent*
- ***Maximum Likelihood Estimation***
- *Classification*
- *Model: Logistic Regression*

Maximum Likelihood for Linear Regression

Maximum Likelihood

- So far, what we have done:
  - ***Direct minimization:*** Cost function -> Distance to output
  - ***Gradient descent:*** Cost function -> Distance to output

- An alternate view:
  - Data (x, y) is generated by an **unknown process**
  - However: We can only observe a **noisy version**
  - How can we model this uncertainty?

- Question: Do we need an alternative cost function?
  - Or something else?

Parametric vs. non-parametric

- We typically use parametric models!
  - What is a **parametric model**?
  1) Make an assumption about the distribution of the dependent variable
  2) Use the assumption to estimate a set of ***parameters*** – typically a set of parameters that relate ***y*** (an outcome variable) to a set of ***X*** variables.

Example: In the classical linear model, we assume:

$$y(\boldsymbol{x}, \boldsymbol{w}) = \sum_{j=1}^{M-1} \boldsymbol{w}^T \phi_j(\boldsymbol{x}) + \epsilon$$

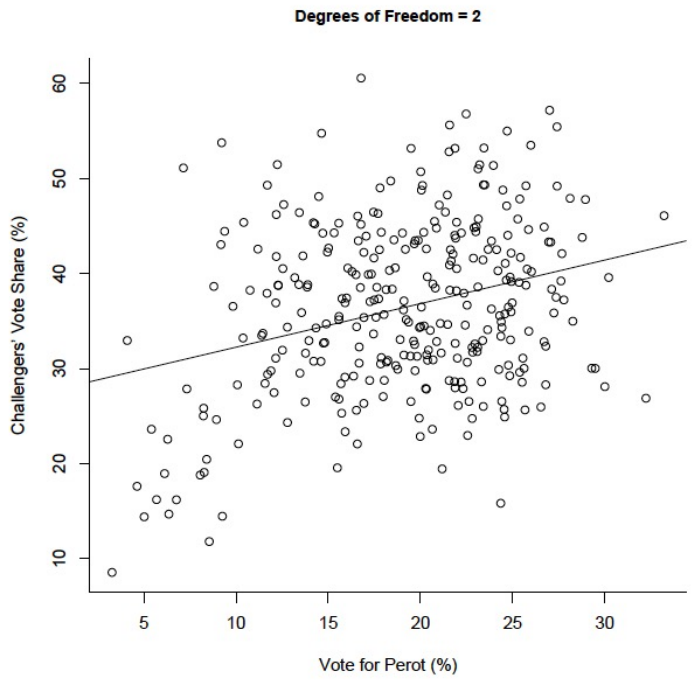
Note: ϵ is sometimes not written ☹

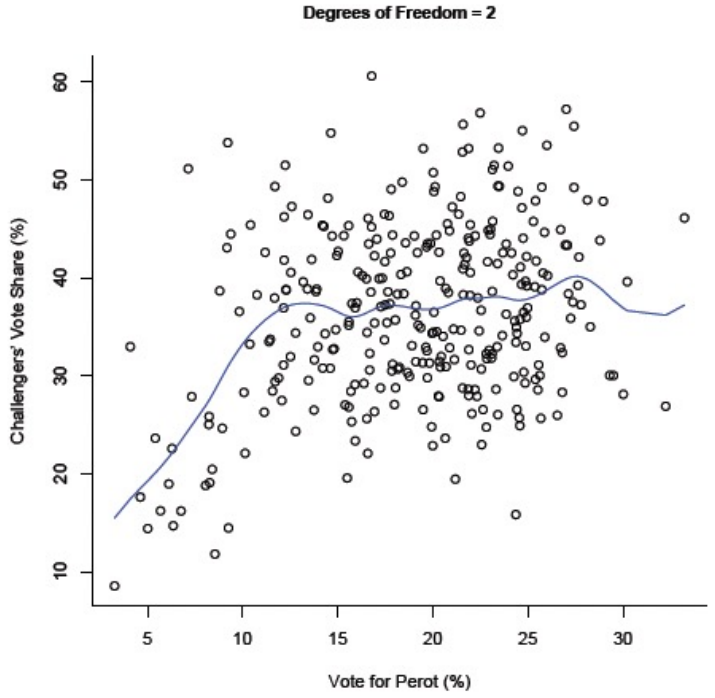
$$\epsilon \sim N(0, \sigma^2)$$

Question: What is the disadvantage of making this assumption?

Example

- In a 1994 AJPS article Gary Jacobson argues that support for Perot in the 1992 election was an indicator of support for Congressional challengers
- Regressed the challengers' vote share on the percentage of votes that Perot received in the Congressional district





Question: Which curve describes the relationship better?

MLE in a nutshell

- Idea: A method for estimating the parameters of an assumed probability distribution, given some observed data

- Goal: ***Maximize*** a likelihood function so that, under the assumed statistical model, the observed data is most probable

- The point in parameter space that maximizes the likelihood function is called the ***maximum likelihood estimate***

- Question: What is the 'most probable' point in *Normal distribution*?
  - Answer: Mean (μ)

MLE (more formally)

- A probabilistic framework for estimating the parameters of a model
- Goal: Maximizing the conditional probability of observing the data (***X***) given a specific probability distribution and its parameters (***θ***):

1) Probability: $P(X;\theta)$

2) Probability (sum rule): $P(x_1, x_2, \dots, x_n;\ \theta) = \sum_{i=1}^{n} \log\ P\ (x_i, \theta)$

3) Likelihood function: $L(X;\theta)$

4) Optimization: Find the max. for $\sum_{i=1}^{n} \log\ P\ (x_i, \theta)$

Question: What could the name for this one be?

Answer: Log-likelihood

Maximum Likelihood: Example

- Question: A coin is flipped **100 times**. Given that there were 55 heads, find the maximum likelihood estimate for the probability p of heads on a single toss.
  - Before we start: What is the probability if we flip the coin ∞ times?
  - **Frequentist**: $P(55\ heads) = \binom{100}{55} * p^{55} * (1-p)^{45}$
  - **Bayesian**: $P(55\ heads|p) = \binom{100}{55} * p^{55} * (1-p)^{45}$

Each flip is a Bernoulli random variable

- Goal: Find the p that maximizes $P(55\ heads|p)$
- Take the first derivative set it to zero:

$$\frac{d}{dp}P(55\ heads|p) = \binom{100}{55} * 55p^{54} * (1-p)^{45} * (-1) * 45p^{55} * (1-p)^{44}) = \mathbf{0}$$

Calculus!

Maximum Likelihood: Example

$$\frac{d}{dp}P(55\ heads|p) = \binom{100}{55} * 55p^{54} * (1-p)^{45} * (-1) * 45p^{55} * (1-p)^{44}) = \mathbf{0}$$

$$55p^{54} * (1-p)^{45} = 45p^{55} * (1-p)^{44}$$

$$55 * (1-p) = 45p$$

$$55 = 100p$$

$$(Best\ MLE\ Estimate\ for)\ p = 0.55$$

Note: MLE is Bayesian stats with 'flat priors'

- More formally: You should check that the critical point is indeed a maximum. You can do this with the second derivative test.

MLE in Linear Regression

Goal in MLE: **max.** $\sum_{i=1}^{n} log\ P(x_i, \theta)$

Goal in supervised learning framed probabilistically: **max.** $P(y|X)$

Combining both: **max.** $\sum_{i=1}^{n} log\ P(y_i|x_i; h)$ ⟵ **You can replace h with linear regression here**

Adding linear regression: **max.** $\sum_{i=1}^{n} log \frac{1}{\sqrt{2*\pi*\sigma^2}} - \frac{1}{2*\sigma^2} * (y_i - h(x_i, \beta)^2)$

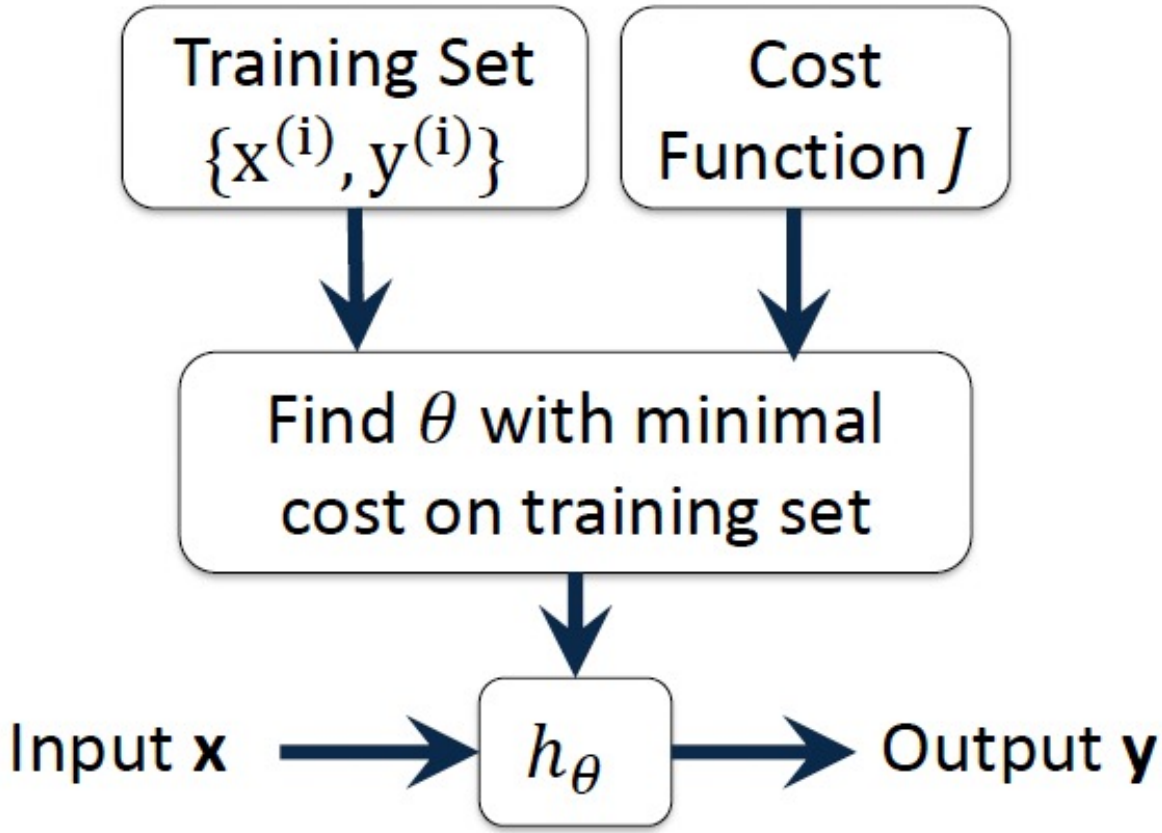
... After a few steps

Question: What is this similar to?

↓

Minimizing cost in least squares: min. $\sum_{i=1}^{n} (y_i - h(x_i, \beta))^2$

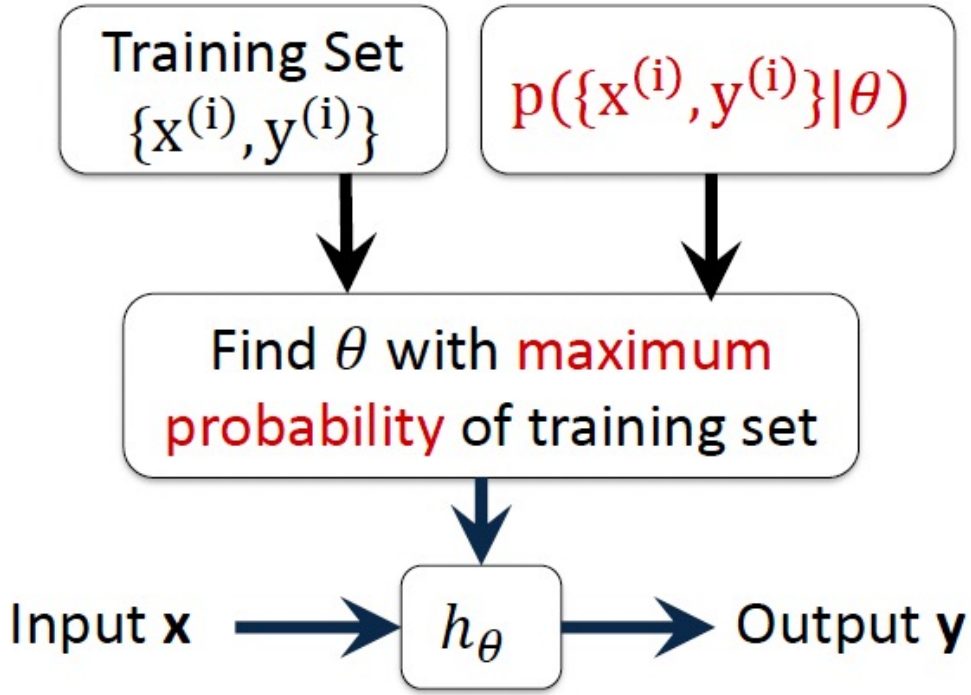
Cost Function in traditional ML



Cost Function in MLE

Alternative View:

"Maximum Likelihood"



Plan for today

- *Example with Var-Cov matrix*
- *Types of Gradient Descent*
- *Maximum Likelihood Estimation*
- ***Classification***
- ***Model: Logistic Regression***

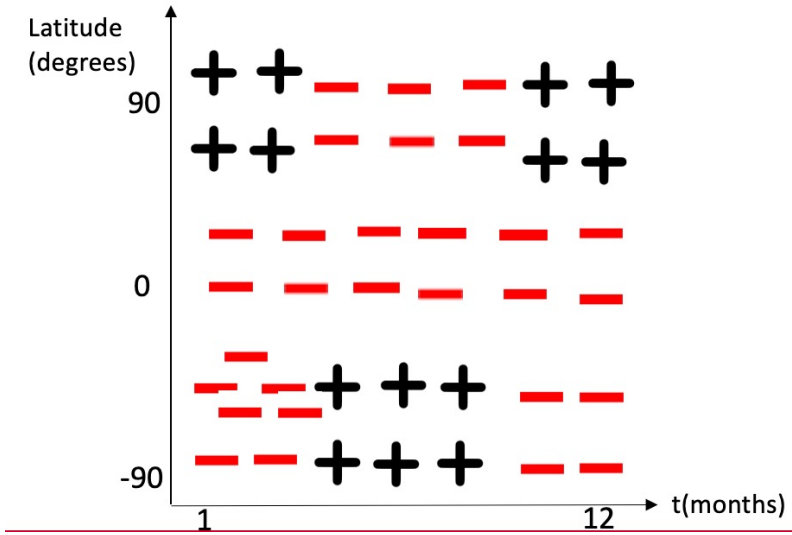
Intro: Classification

Classification

- Goal: Assigning a **class label** to points in vector space
  - ***Two-class***: $y \in \{0,1\}$
  - ***Multi-class***: $y \in \{0,1,2,\dots,k\}$

0 usually denotes 'the lack of something' (=negative class)

- A range of puzzles:
  - Email: *Spam / Not Spam?*
  - Social media user: *Bot / Not Bot?*
  - Video: *Viral / Not Viral?*
  - Covid: Mask / No Mask?
  - Winter: Skiing / No skiing
  - etc.



Generative and Discriminative Classifiers

- ***Generative*** model:
  - Associates each feature of an input with a probability
  - Captures ***P(X|Y)*** and ***P(Y)***
  - Uses Bayes Rule to figure out class association
  - They model the distribution of individual classes
  - Computes a probabilistic output based on features
  - Examples: Naive Bayes, Bayesian networks, Hidden Markov Models
- ***Discriminative*** model:
  - Only learns the boundary / differences between two classes
  - Captures ***P(Y)***
  - The boundary might be ***hard*** or ***soft***
  - Can be probabilistic, but also non-probabilistic
  - Examples: Logistic regression, SVM

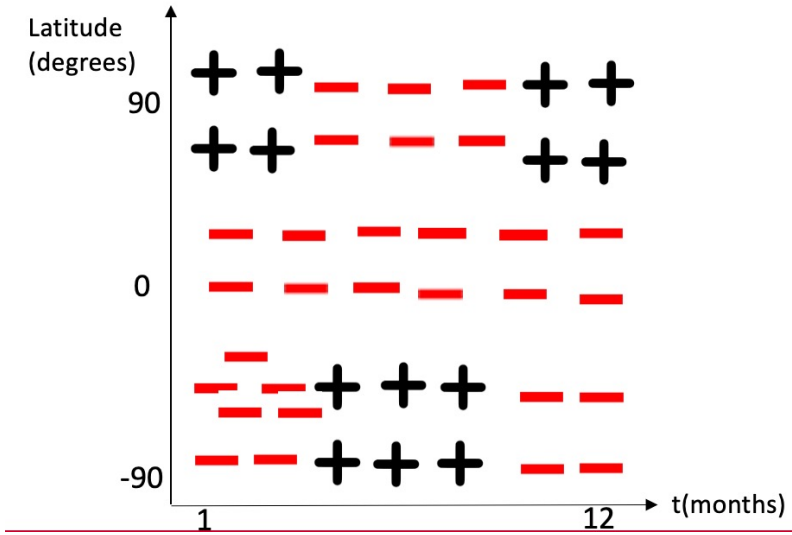
Classification

- Question: Can we just ***minimize*** the following?

$$J(\theta_0, \theta_1, \ldots, \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

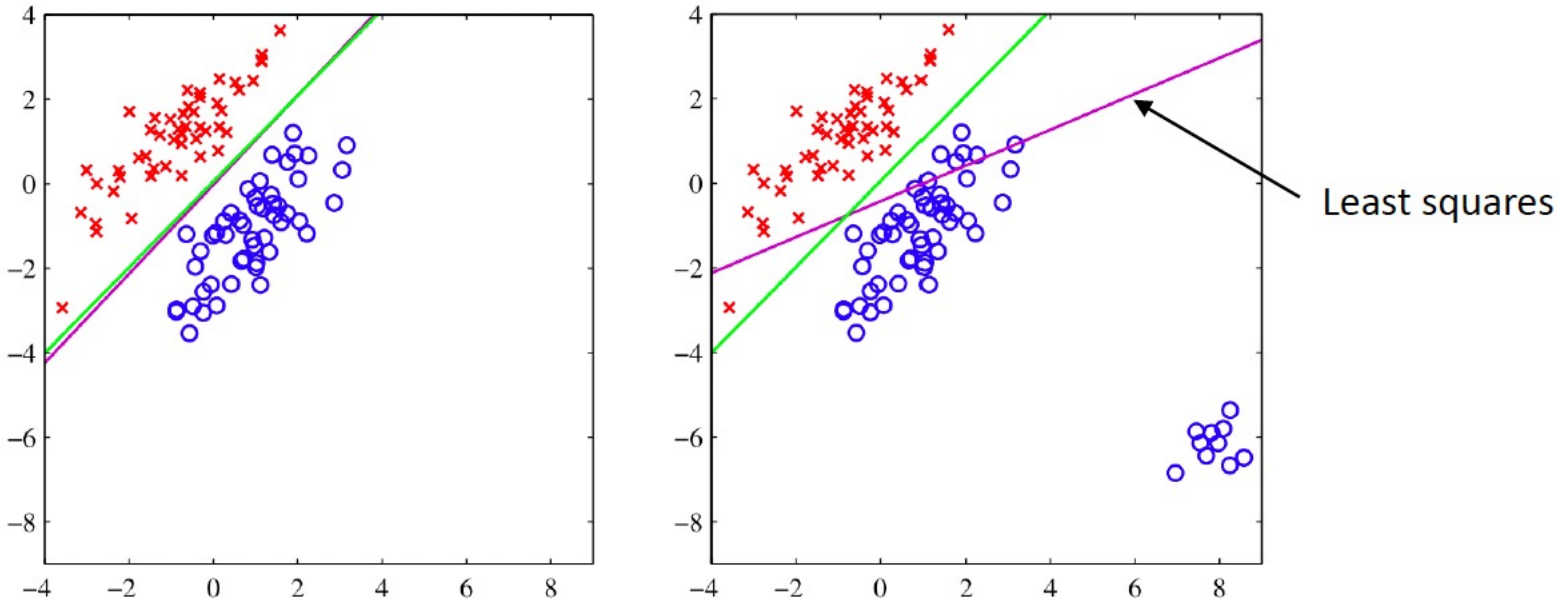
- **Yes, we can!**

- Problem: Outliers may be 'too influential'
- Solution: Sigmoidal cost functions
  - Example: Logistic regression



Least Squares vs. Logistic Regression for Classification

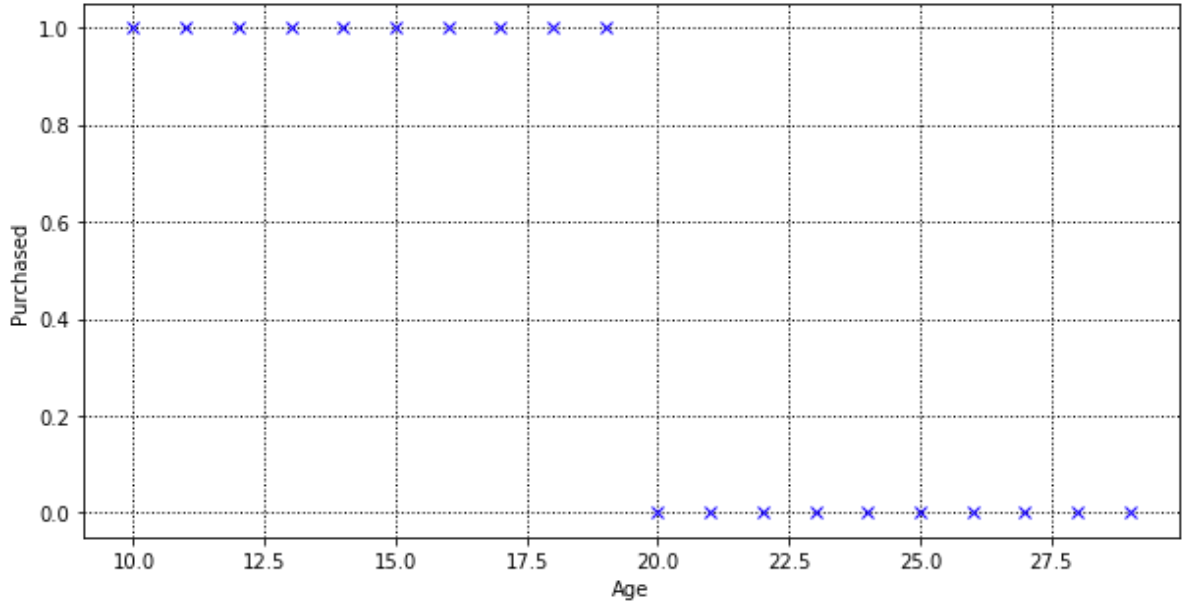
Source: Bishop



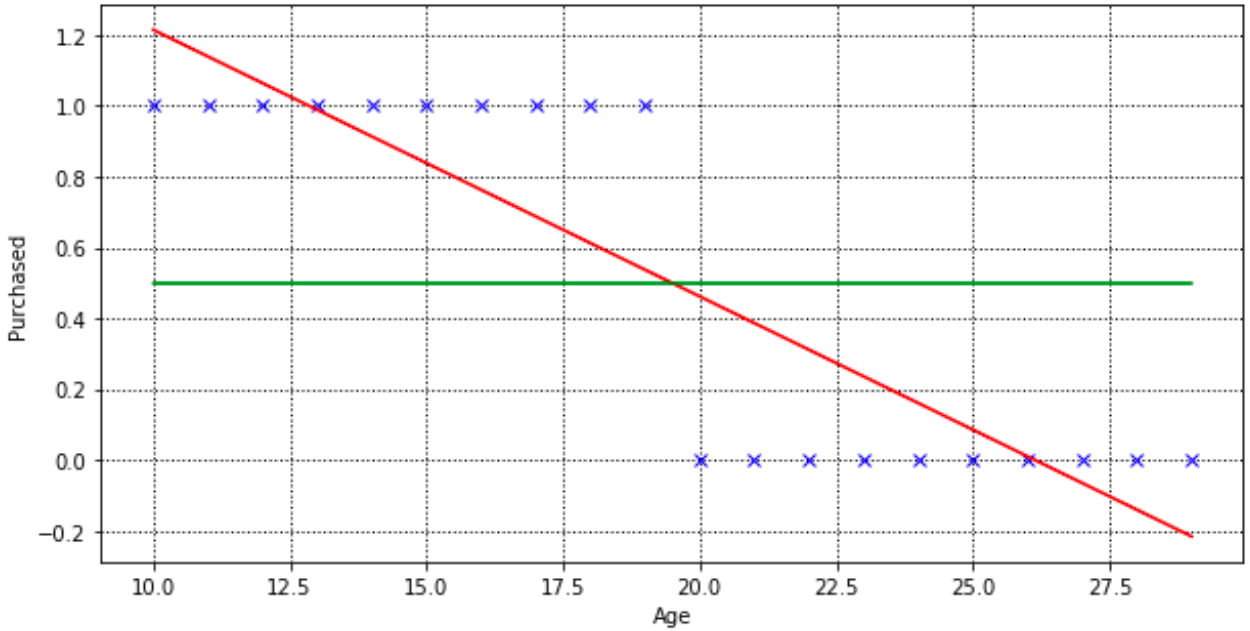
- There are two classes: **red** and **blue**
- Magenta curve denotes the decision boundary found by ***least squares***
- Green curve denotes the decision boundary identified by the ***logistic regression***
  - Difference: ***Logistic regression*** is less sensitive to outliers

Using Continuous vs. Binary Explanatory Variables

- Our data

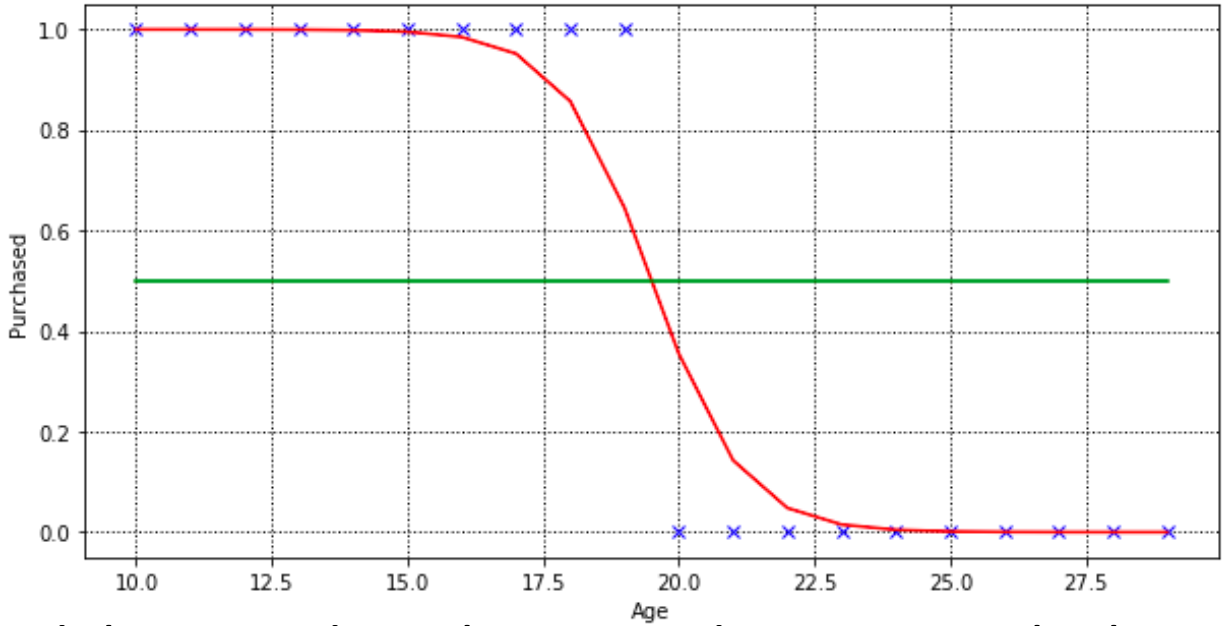


- Set a linear regression line



- **X**: 10, **Predicted Y-Value**: 1.21428571
- **X**: 15, **Predicted Y-Value**: 0.83834586
- **X**: 20, **Predicted Y-Value**: 0.46240602
- **X**: 25, **Predicted Y-Value**: 0.08646617
- **X**:30, **Predicted Y-Value**: -0.28947368

Logistic Function



Now we have 2 models trained on the same dataset, one by linear regression, and another by logistic regression. We can compare both models performance by using **RMSE** and **R^2**:

	R^2	RMSE
Linear regression	0.7518796992481203	0.06203007518796935
Logistic regression	0.9404089597242656	0.014897760068933596