MACHINE LEARNING

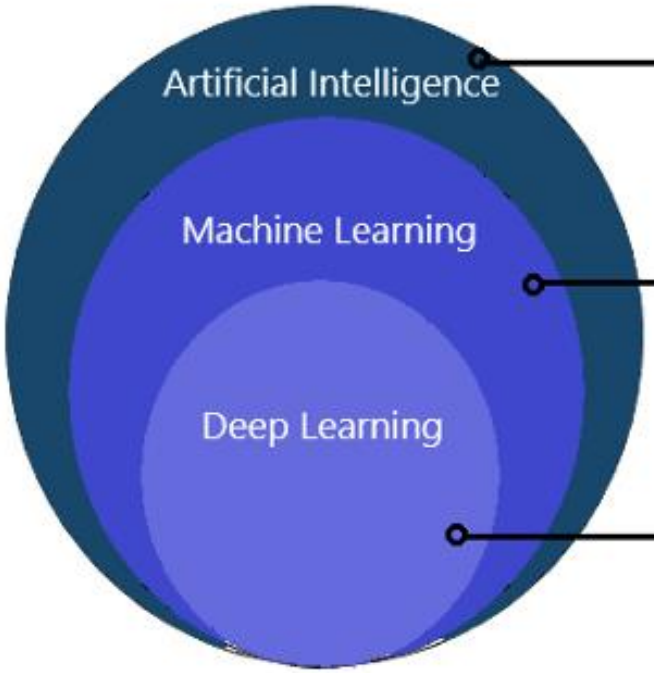
Electrical Summer Workshops (ESW) 2022

Electrical Engineering Department - Sharif University of Technology

Instructors: *Alireza Gargoori Motlagh – Amir Mirrashid – Ali Nourian*

INTRODUCTION

What is ML?

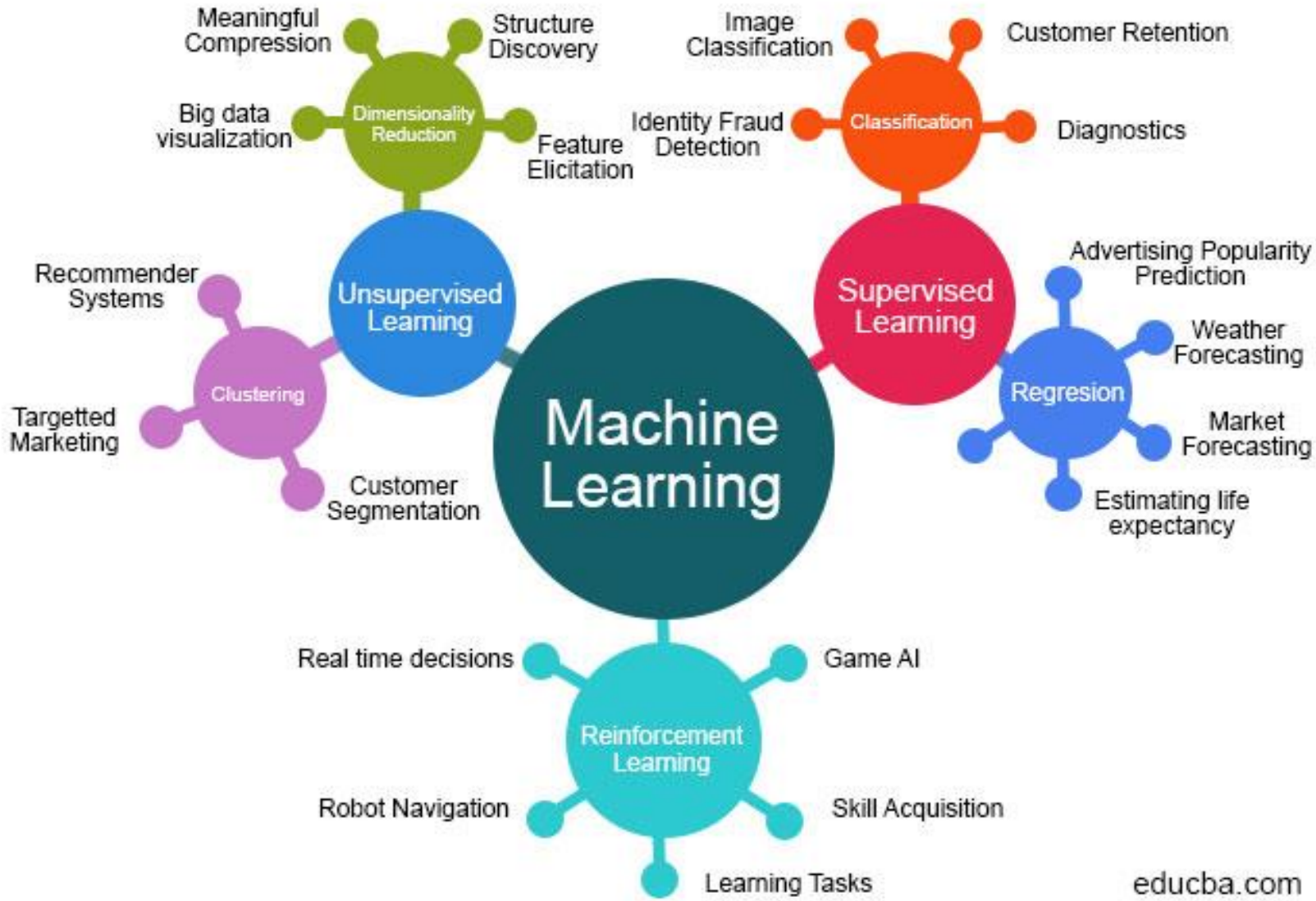


Artificial Intelligence
AI enables machines to think without any human intervention.

Machine Learning
Subset of AI that uses statistical learning algorithms that learn pattern in data over time

Deep Learning
Subset of ML that filters the data through multiple layers

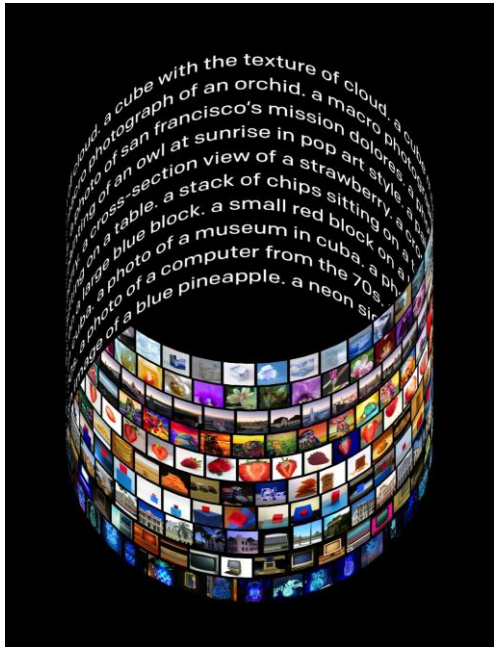
Machine Learning Algorithms



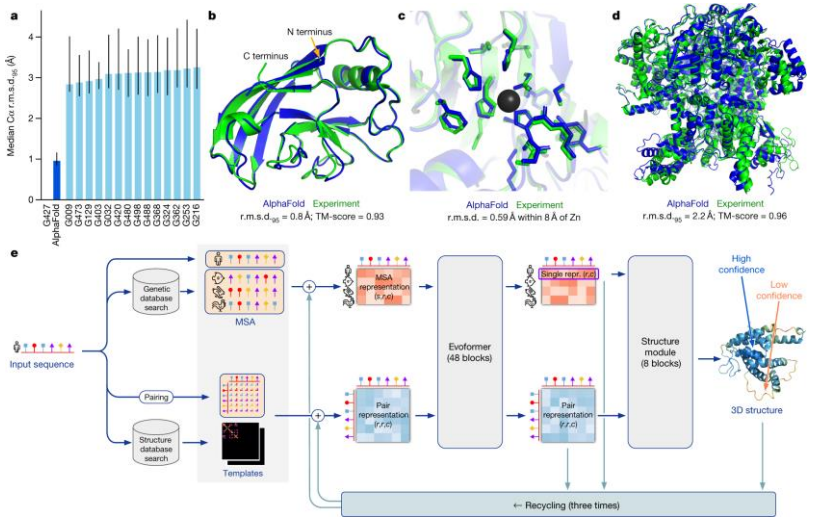
educba.com

Why ML?


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATURE.COM/NATURE

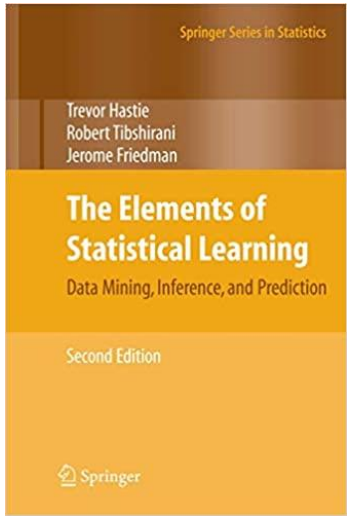

a cube with the texture of cloud. a
photograph of an orchid. a macro
photo of san francisco's mission dolores
of an owl at sunrise in pop art style.
on a table. a cross-section view of a strawberry.
blue block. a stack of chips sitting on a
a photo of a museum in cuba. a
a photo of a computer from the 70s.
of a blue pineapple. a neon

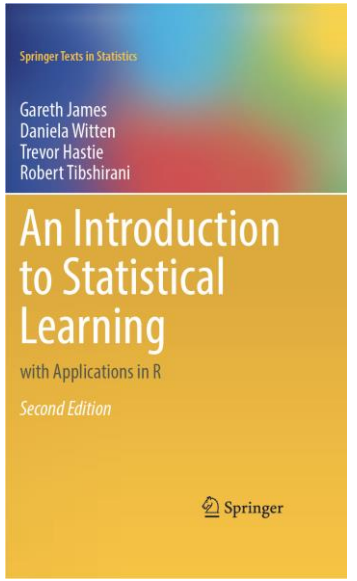

GPT - 3


a
b
c
d
N terminus
C terminus
AlphaFold Experiment
r.m.s.d. = 0.59 Å within 8 Å of Zn
r.m.s.d95 = 2.2 Å; TM-score = 0.96
e
Input sequence
Genetic database search
MSA
Pairing
Structure database search
Templates
Evoformer (48 blocks)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)

Course Overview

- Week 1
  - Introduction to ML
  - Linear Regression
  - Regularization
  - Logistic Regression
- Week 2
  - Support Vector Machines (SVM)
  - Decision Tress & Ensemble Methods (bagging, boosting, random forests, XGboost)
- Week 3
  - Dimensionality Reduction (PCA, Kernel PCA, t-SNE)
  - Clustering (K-means, Expectation Maximization, …)
- Week 4
  - Neural Networks
    - Multi Layer Perceptrons (MLP)
    - Convolutional Neural Networks (CNN)
    - Recurrent Neural Networks (RNN)





Prerequisites

- Linear Algebra
  - Basic Matrix Operations
  - Matrix Decompositions (SVD, Eigen-decomposition)
  - Matrix Calculus
- Probability and Statistics
  - Expectation and Variance
  - Covariance Matrix
- Programming in Python

LINEAR REGRESSION

Notation Conventions

- Y refers to true target values.
- $\hat{Y}$ refers to value of predictions of our model.

- Subscript j refers to j'th variable (feature).
- Superscript i refers to i'th sample.
  - So $x_j^{(i)}$ means j'th feature of i'th sample.

- Non-bold letters denote scalers.
- Capital bold letters denote matrices.
- Small bold letters denote vectors.

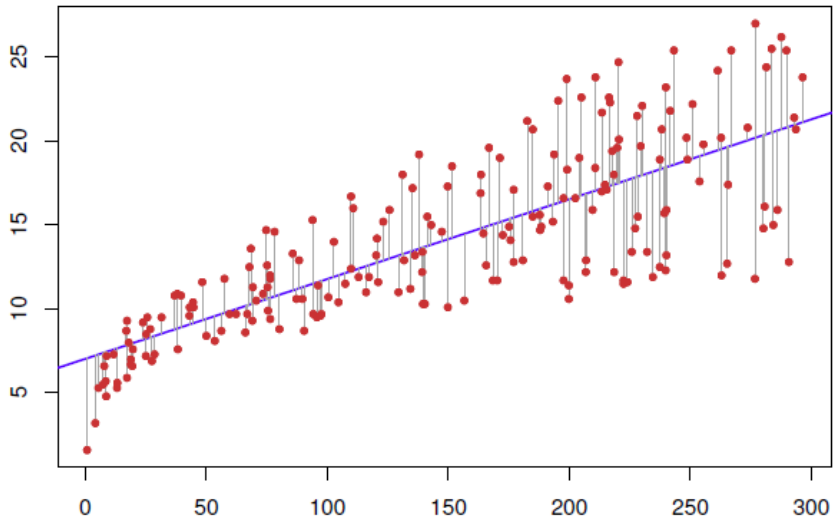
Linear Regression

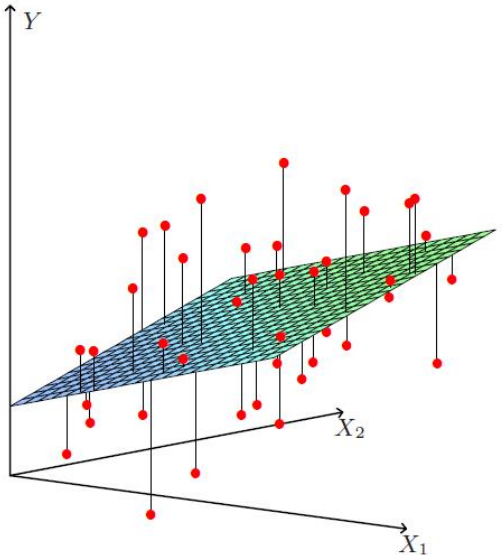
Relationship between the input variables and the output is modeled through a linear function

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$$

The linear model either assumes that the regression function $E(Y|X)$ is linear, or that the linear model is a reasonable approximation.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon = \beta_0 + \sum_{j=1}^{p} \beta_j X_j + \epsilon$$





Coefficients and Features

Here the β_j's are unknown parameters or coefficients, and the variables X_j can come from different sources:

- quantitative inputs;
- transformations of quantitative inputs, such as log, square-root or square;
- basis expansions, such as $X_2 = X_1^2$, $X_3 = X_1^3$, leading to a polynomial representation;
- numeric or "dummy" coding of the levels of qualitative inputs. For example, if G is a five-level factor input, we might create X_j, j = 1, . . . , 5, such that $X_j = I(G = j)$. Together this group of X_j represents the effect of G by a set of level-dependent constants, since in $\sum_{j=1}^{5} \beta_j X_j$,one of the X_j's is one, and the others are zero;
- interactions between variables, for example, $X_3 = X_1.X_2$;

Estimating the Parameters

Estimation of the output(target value) for training samples would be:

$$\hat{y}^{(i)} = \hat{\beta}_0 + \hat{\beta}_1 x_1^{(i)} + \hat{\beta}_2 x_2^{(i)} + \cdots + \hat{\beta}_p x_p^{(i)}$$

We want to find the optimal values of $\hat{\beta}_j$ in order to minimize the cost function defined as:
(Residual Sum of squares)

$$RSS = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

Or equivalently:
(Mean Squared Error)

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2$$

- Convex function of the regression parameters.
- Unbiased estimation of the population regression line. (if some requirements are met)

Cost Function in Matrix Form

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix} \qquad \hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix}$$

$$RSS = \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 = (\boldsymbol{y} - \hat{\boldsymbol{y}})^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) = \left|\left|\boldsymbol{y} - \hat{\boldsymbol{y}}\right|\right|_{L2}^2$$

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \vdots \\ \hat{y}^{(n)} \end{bmatrix} = \begin{bmatrix} \hat{\beta}_0 + \hat{\beta}_1 x_1^{(1)} + \hat{\beta}_2 x_2^{(1)} + \cdots + \hat{\beta}_p x_p^{(1)} \\ \vdots \\ \hat{\beta}_0 + \hat{\beta}_1 x_1^{(n)} + \hat{\beta}_2 x_2^{(n)} + \cdots + \hat{\beta}_p x_p^{(n)} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & x_1^{(1)} & \dots & x_p^{(1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & x_1^{(n)} & \dots & x_p^{(n)} \end{bmatrix} \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_p \end{bmatrix} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$$

Least-Squares Problem

$$RSS = (\boldsymbol{y} - \widehat{\boldsymbol{y}})^T(\boldsymbol{y} - \widehat{\boldsymbol{y}}) = \left|\left|\boldsymbol{y} - \widehat{\boldsymbol{y}}\right|\right|_{L2}^2, \qquad \widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$$

So, the cost would be:

$$RSS = (\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})$$

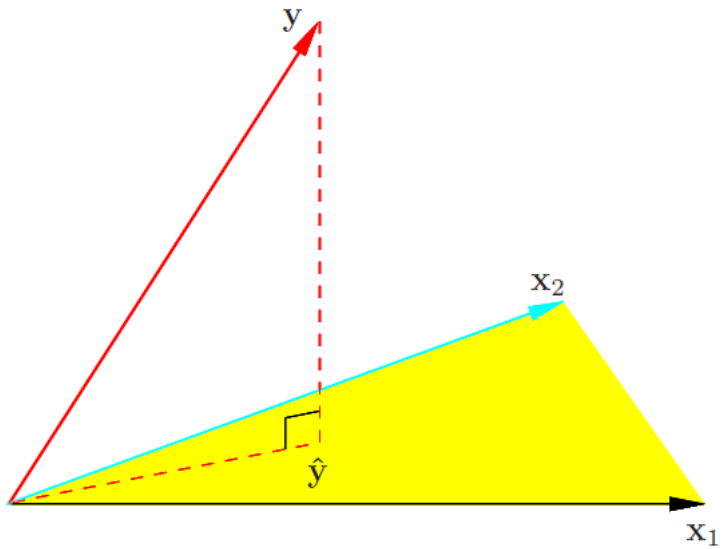
Since this is a convex function in regression parameters, we can set its derivative to zero to find the optimal values of $\widehat{\beta_j}$ for the minimum value of RSS:

$$\frac{\partial\, RSS}{\partial\, \boldsymbol{\beta}} = -2\boldsymbol{X^T}(\boldsymbol{y} - \boldsymbol{X\beta}) = 0 \rightarrow \boldsymbol{X^T X}\,\widehat{\boldsymbol{\beta}} = \boldsymbol{X^T y} \quad (Normal\ eq.)$$

Hence, considering that X is full column-rank (i.e. columns of X, features, are linearly independent), the *unique* solution would be:

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X^T X}\right)^{\boldsymbol{-1}}\boldsymbol{X^T y}$$

Geometric Interpretation



The prediction of least-squares problem($\hat{\boldsymbol{y}}$), would be the orthogonal projection of outcome vector ($\boldsymbol{y}$) onto the hyperplane spanned by the features of X.

So the residual error, $\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}}$ is always orthogonal to this subspace and it's the part of the target that the model cannot predict.

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\boldsymbol{1}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{y} = \boldsymbol{H}\boldsymbol{y}$$

$\boldsymbol{H} = \boldsymbol{X}\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\right)^{-\boldsymbol{1}}\boldsymbol{X}^{\boldsymbol{T}}$ is referred as *Hat* matrix or *Projection* matrix.

*Correlated Variables

- What if there is a perfect collinearity between features? (e.g. $X_2 = 4X_1$)
- In this case $\boldsymbol{X}$ is not full column rank and hence $\boldsymbol{X^T X}$ is singular and not invertible.
- The solution still exists, but it is not unique. (The solution are still the projection of y onto the column space of X but there are more than one way to express that projection.)

One simple solution is to drop the redundant (dependent) columns from X so that we can still use the previous results. (Most regression software package detect these redundancies and automatically implement some strategies for removing them.)

- Also, there are cases when 2 or more features are highly correlated.
- P-values of regression coefficients are not significant. (by changing the train set, the coefficients of these features can vary too much.)

There are some methods to detect and solve this problem:

- Backward Selection
- Forward Selection
- Shrinkage Methods

*Unbiasedness of Linear Regression

$$\hat{\beta} = (X^T X)^{-1} X^T y = (X^T X)^{-1} X^T (X\beta + \epsilon) \rightarrow$$
$$\hat{\beta} = \beta + (X^T X)^{-1} X^T \epsilon$$

Assumptions:

- iid samples (indepndent and identically distributed)
- Full column-rank (No perfect collinearity among features)
- $\epsilon | \boldsymbol{X} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$
  - Zero conditional mean ($E(\epsilon|X) = 0$)
  - Constant variance of error term and iid.
  - Normally distributed. (Not really required, but for some other inferences such as distributions of β_j's and their p-values, estimation of variance of noise, etc.)

Under the above assumptions, we can write:

$$E(\widehat{\boldsymbol{\beta}}|\boldsymbol{X}) = E\left(\boldsymbol{\beta} + (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T \epsilon} \middle| \boldsymbol{X}\right) = E(\boldsymbol{\beta}|\boldsymbol{X}) + E\left((\boldsymbol{X^T X})^{-1} \boldsymbol{X^T \epsilon} \middle| \boldsymbol{X}\right) \rightarrow$$
$$E(\widehat{\boldsymbol{\beta}}|\boldsymbol{X}) = \boldsymbol{\beta} + (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T}\, E(\boldsymbol{\epsilon}|\boldsymbol{X}) = \boldsymbol{\beta}$$

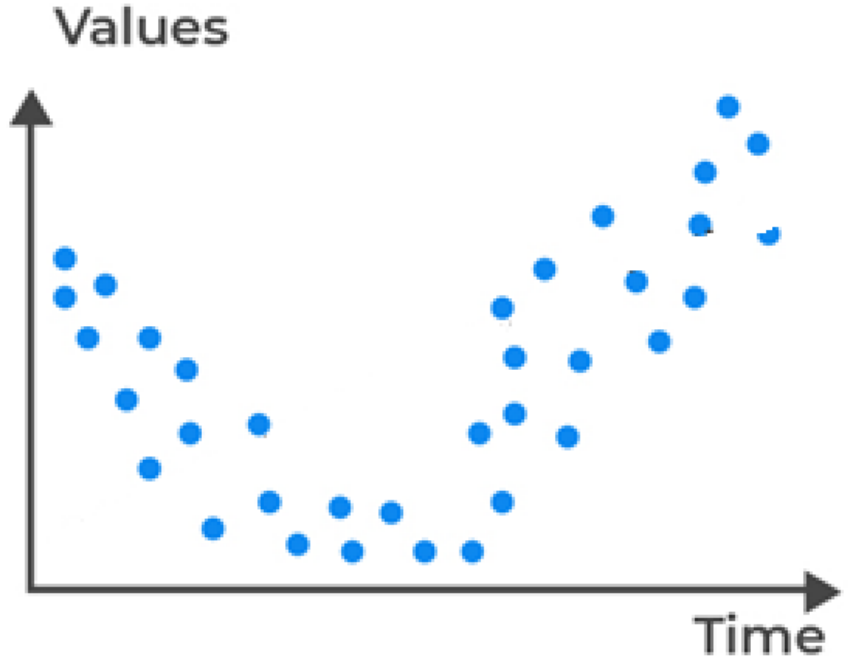
Gauss-Markov theorem: ordinary least squares (OLS) regression produces unbiased estimates that have the smallest variance of all possible linear estimators.

A Case Study: Polynomial Fitting

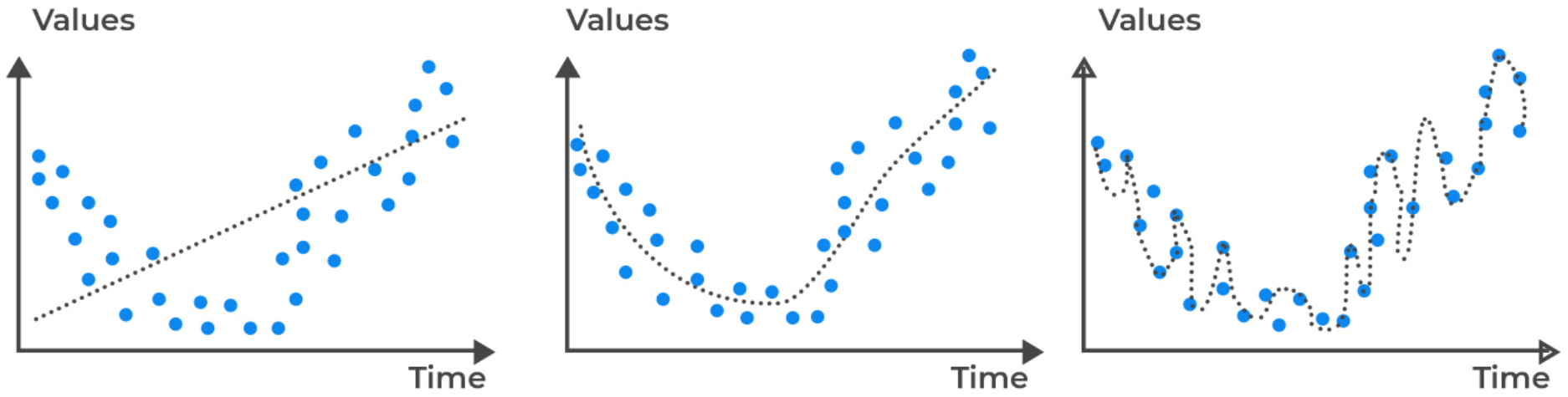
$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \cdots + \beta_p X^p + \epsilon = \sum_{j=0}^{p} \beta_j X^j + \epsilon$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x^{(1)} & x^{2(1)} & \dots & x^{p(1)} \\ 1 & x^{(2)} & x^{2(2)} & \dots & x^{p(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x^{(n)} & x^{2(n)} & \dots & x^{p(n)} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$



Overfitting and Underfitting



Underfitted

(High bias error)

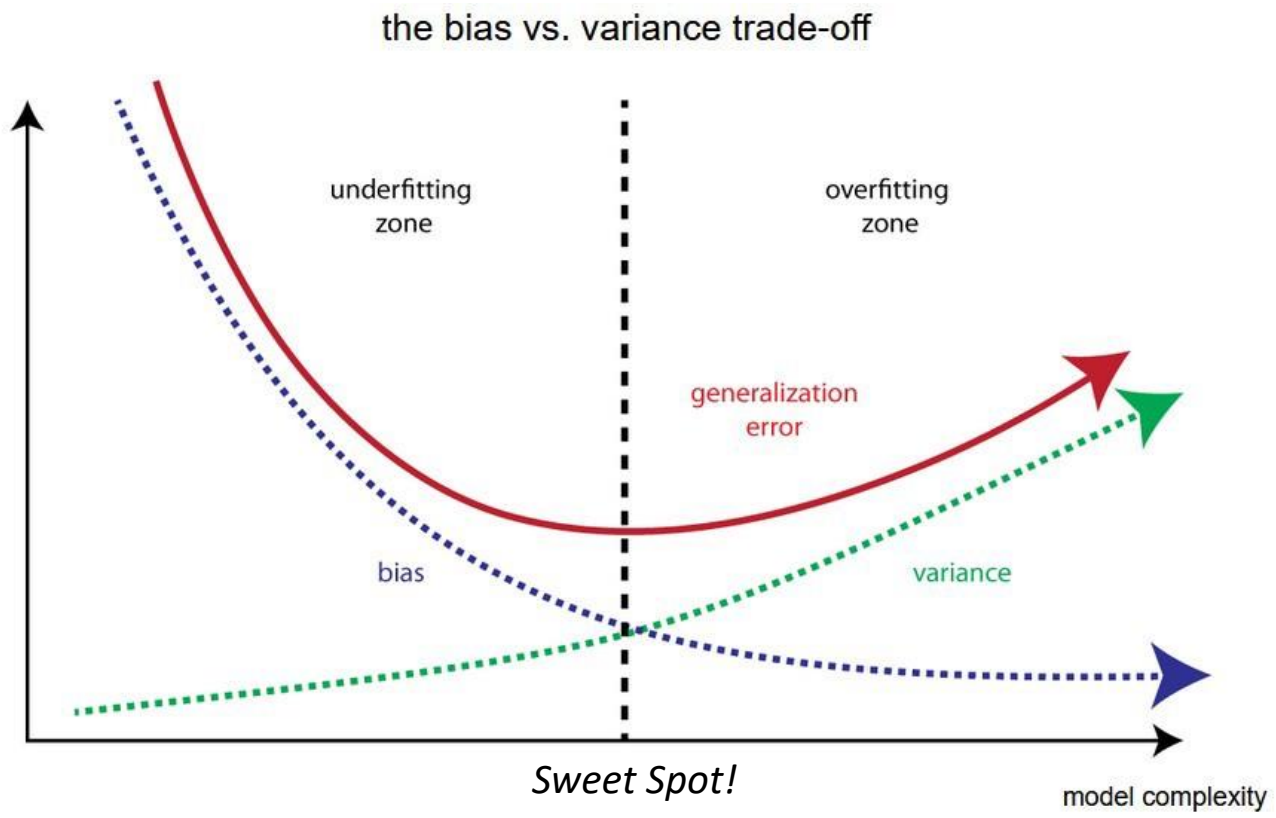
Good Fit/Robust

(Balance between bias and variance)

Overfitted

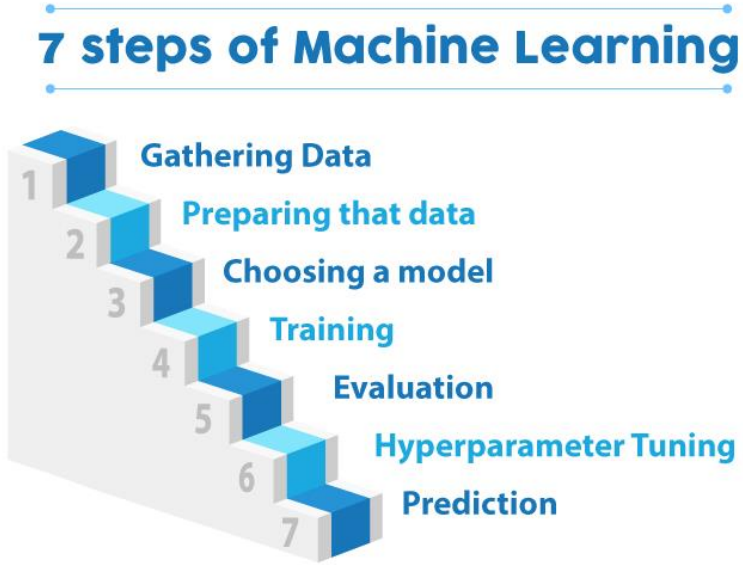
(High variance error)

Bias-Variance Tradeoff



Approach to a ML Problem

1. Collecting your data.
2. Prepare and preprocess the dataset and Splitting the data randomly to a training set, a validation set and a test set.
3. Choose an appropriate algorithm.
4. Training the model on the train set.
5. Evaluating the model on test.
6. Hyperparameter tuning on the validation set.
7. Making predictions.



REGULARIZATION AND SHRINKAGE METHODS

Ridge Regularization

Ridge regularization is a shrinkage method which its goal is to reduce the variance of the model through reducing the absolute value of the regression parameters (coefficients).

The RSS cost function is modified by the L_2 norm of the coefficients:

$$cost = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$\rightarrow cost = \sum_{i=1}^{n} \left(y^{(i)} - \beta_0 - \beta_1 x_1^{(i)} - \beta_2 x_2^{(i)} - \cdots - \beta_p x_p^{(i)}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

So a regularization term is added to the residual term in order to limit the radius of the parameters.

λ is called the hyperparameter of the model and it controls the reduction of the coefficients' values.

Ridge Regression Solution

First, by removing the intercept (β_0) from the cost function, we can rewrite the cost as follows:

$$cost = \left|\left|y - X\beta\right|\right|_{L2}^2 + \lambda\left|\left|\beta\right|\right|_{L2}^2 = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta$$

Again, this is a convex function of regression parameters and we can set its derivative w.r.t. $\boldsymbol{\beta}$ to zero to find the optimal solution:

$$\frac{\partial\, RSS}{\partial\, \boldsymbol{\beta}} = -2\boldsymbol{X^T}(\boldsymbol{y} - \boldsymbol{X\beta}) + 2\lambda\boldsymbol{\beta} = 0 \rightarrow (\boldsymbol{X^TX} + \lambda\boldsymbol{I})\,\boldsymbol{\beta} = \boldsymbol{X^Ty}$$

Hence the solution would be:

$$\widehat{\boldsymbol{\beta}} = \left(\boldsymbol{X^TX} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X^Ty}$$

(which is *always unique even if the features are correlated and* X is not full column-rank)

Lasso Regularization

Lasso regularization is a shrinkage method which its goal is to reduce the variance of the model through reducing the absolute value of the regression parameters (coefficients).

The RSS cost function is modified by the L_1 norm of the coefficients:

$$cost = \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

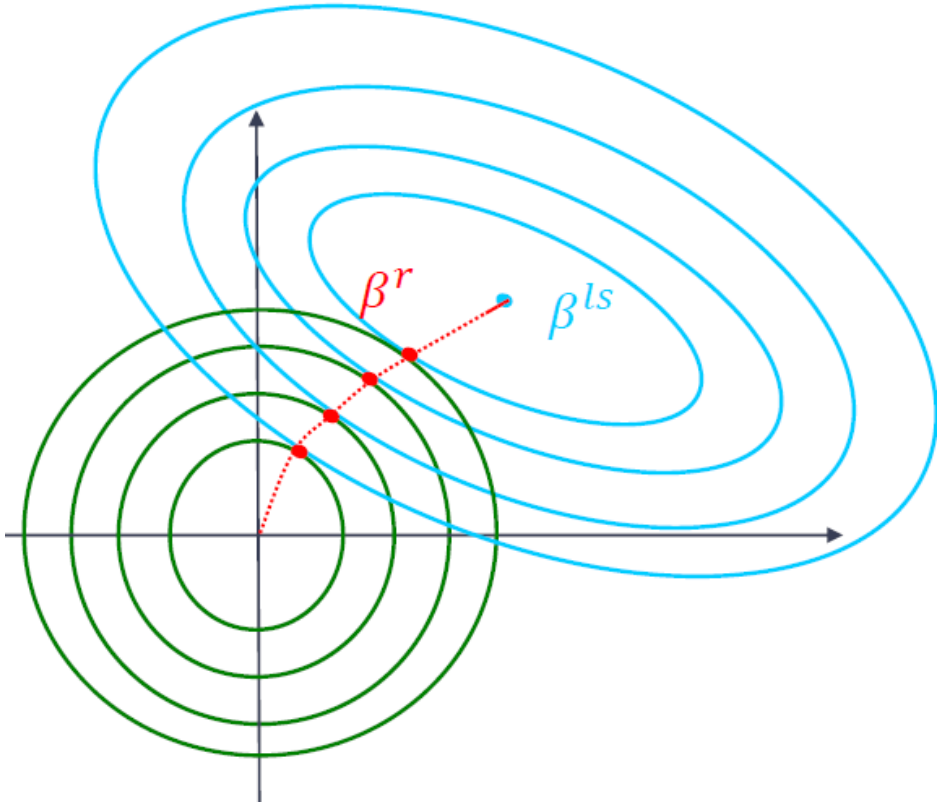
$$\rightarrow cost = \sum_{i=1}^{n} \left(y^{(i)} - \beta_0 - \beta_1 x_1^{(i)} - \beta_2 x_2^{(i)} - \cdots - \beta_p x_p^{(i)}\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

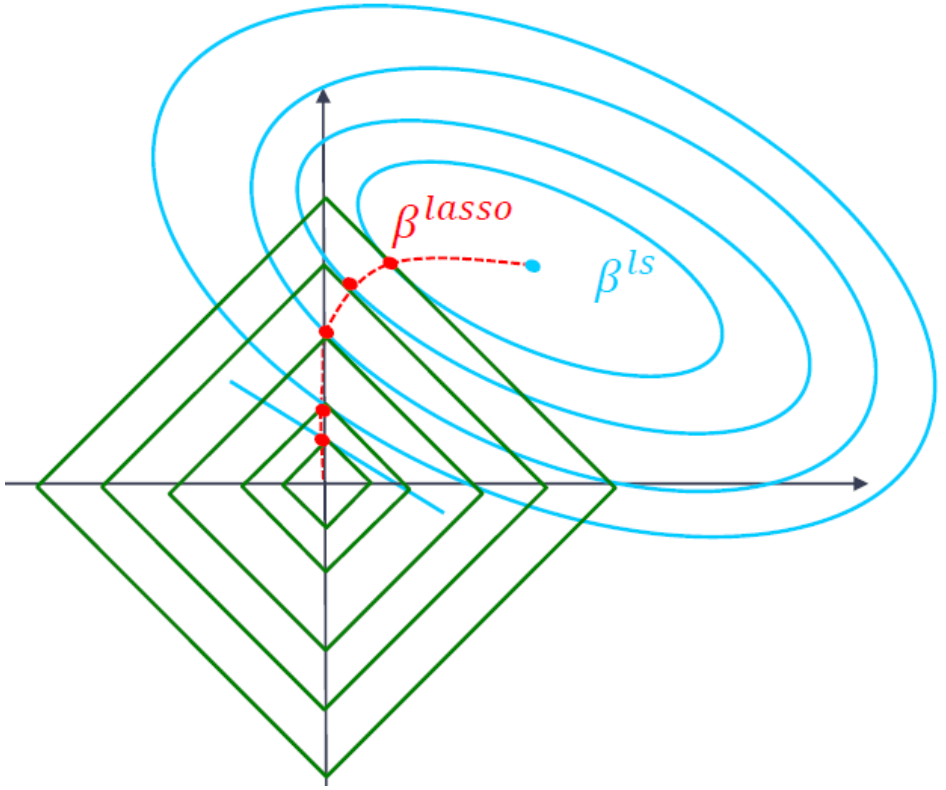
So a regularization term is added to the residual term in order to limit the radius of the parameters.

λ is called the hyperparameter of the model and it controls the reduction of the coefficients' values.

This is still a convex function of the regression parameters but with no analytical solution.

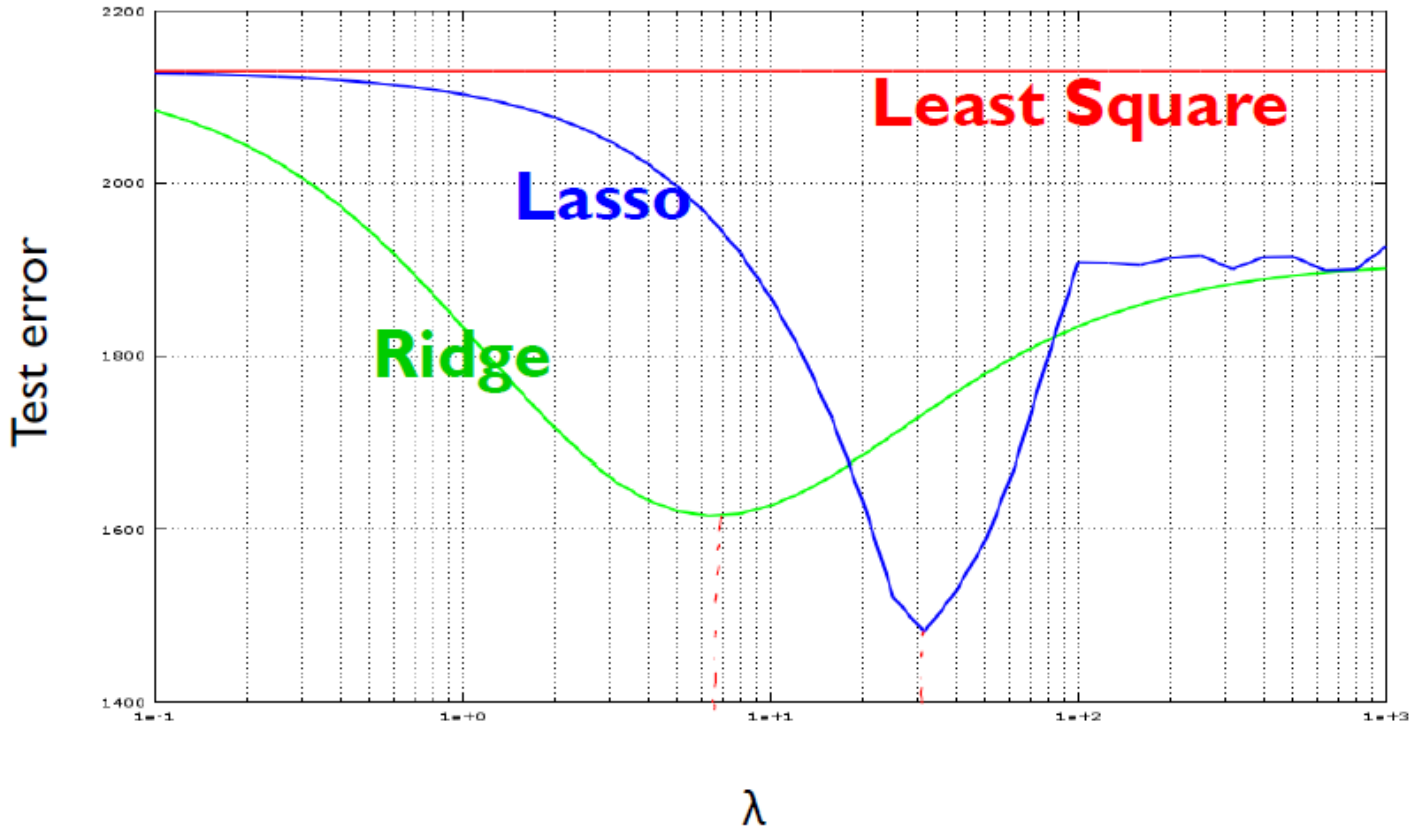
Ridge and Lasso Coefficients Path





Linear Regression vs Ridge vs Lasso

The effect of λ hyperparameter also results in U-shape curve for test error.



LOGISTIC REGRESSION

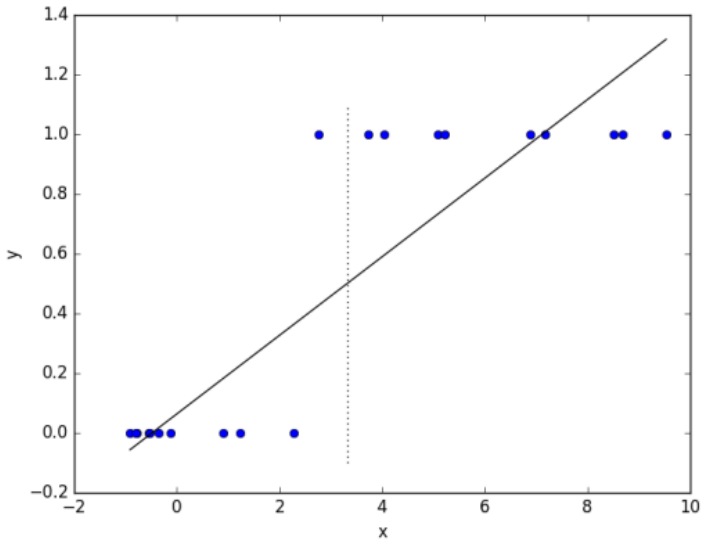
Linear Regression as Classifier

A basic approach to solve classification problems could use linear regression as follows: $\left(y^{(i)} \in \{0,1\}\right)$

$$RSS = \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\beta}^T \boldsymbol{x}^{(i)}\right)^2$$

$$\rightarrow y^{(i)} = \begin{cases} 1 & \boldsymbol{\beta}^T \boldsymbol{x}^{(i)} > 0.5 \\ 0 & otherwise \end{cases}$$

- Arbitrary threshold
- Optimal coefficients try to make data points hug the line as closely as possible, no matter the effect on classification performance



- Our loss function needs to be modified.

Logit Transform

Predicting probabilities : $\mathbb{R}^n \to (0,1)$

- Dot product to $\mathbb{R}_+^n$ through exponentiation
- Divide the result to $1 + itself$

$\to sigmoid$ function:

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

where: $z = \boldsymbol{\beta}^T \boldsymbol{x}$ and $\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{bmatrix}$

$$\mathbb{P}(Y=1|\boldsymbol{X}) = \frac{1}{1+e^{-\boldsymbol{\beta^T X}}}$$

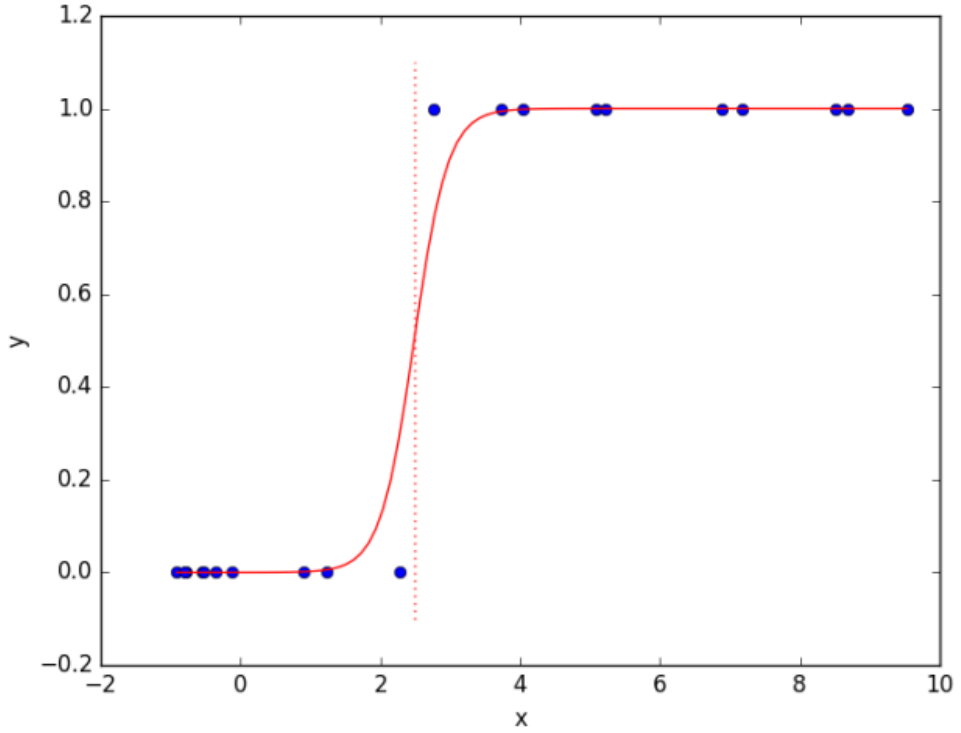
$$\mathbb{P}(Y=0|\boldsymbol{X}) = \frac{1}{1+e^{\boldsymbol{\beta^T X}}}$$

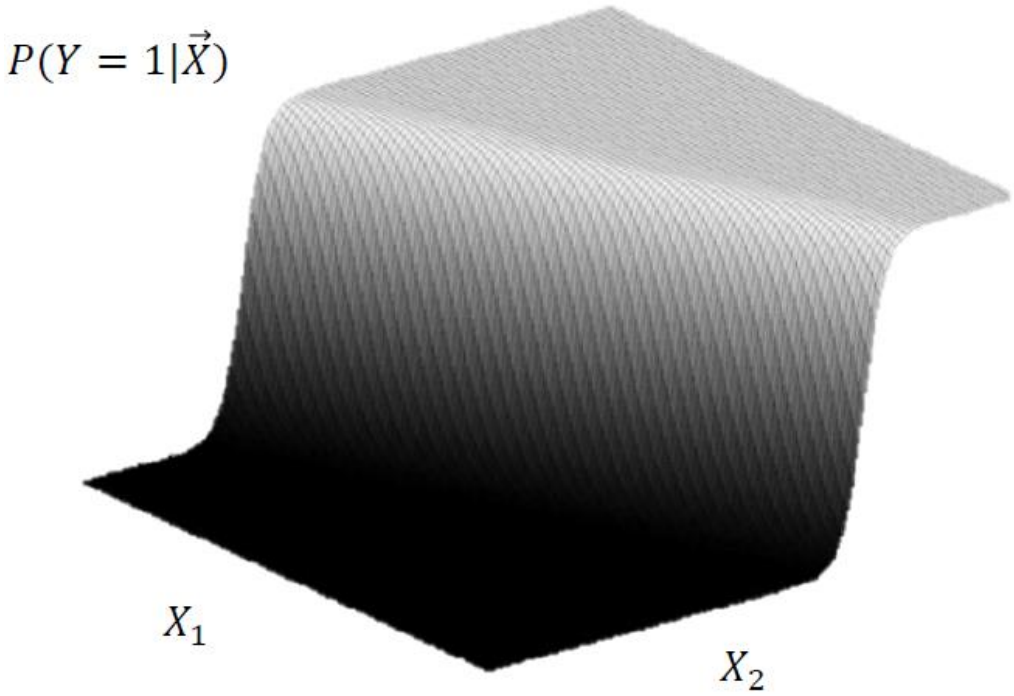
$$\xrightarrow{log-odds} \log\left(\frac{\mathbb{P}(Y=1|\boldsymbol{X})}{\mathbb{P}(Y=0|\boldsymbol{X})}\right) = \boldsymbol{\beta}^T \boldsymbol{X} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Logistic Regression Decision Boundary

$$y^{(i)} = \begin{cases} 1 & \sigma(\boldsymbol{\beta}^T \boldsymbol{x}^{(i)}) > 0.5 \\ 0 & otherwise \end{cases}$$

- Other thresholds might be needed in some cases!





Maximum Likelihood Estimation (MLE)

The problem of finding two probabilities reduces to estimation of regression parameters:

- *Maximum likelihood estimation (MLE)*: Finding the parameters that maximize the joint probability of occurring all observed samples

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \mathbb{P}\left(Y = y^{(i)} \middle| \boldsymbol{X} = \boldsymbol{x^{(i)}}\right) = \prod_{i=1}^{n} \mathbb{P}\left(Y = 1 \middle| \boldsymbol{X} = \boldsymbol{x^{(i)}}\right)^{y^{(i)}} \mathbb{P}\left(Y = 0 \middle| \boldsymbol{X} = \boldsymbol{x^{(i)}}\right)^{1-y^{(i)}}$$

$$\rightarrow L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \sigma\left(\boldsymbol{\beta^T x^{(i)}}\right)^{y^{(i)}} \left(1 - \sigma\left(\boldsymbol{\beta^T x^{(i)}}\right)\right)^{1-y^{(i)}}$$

$$l(\boldsymbol{\beta}) = \log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} y^{(i)} \log\left(\sigma\left(\boldsymbol{\beta^T x^{(i)}}\right)\right) + \left(1 - y^{(i)}\right)\left(\log\left(1 - \sigma\left(\boldsymbol{\beta^T x^{(i)}}\right)\right)\right) =$$

$$= \sum_{i=1}^{n} \left(y^{(i)}\left(\boldsymbol{\beta^T x^{(i)}}\right) - \log\left(1 + e^{\boldsymbol{\beta^T x^{(i)}}}\right)\right)$$

Cost Function

Since sigmoid outputs probability, we use negative log likelihood to represent the error:

$$J(\boldsymbol{\beta}) = -\frac{1}{n} l(\boldsymbol{\beta}) = -\frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} \left(\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(\boldsymbol{i})} \right) - \log \left(1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}} \boldsymbol{x}^{(i)}} \right) \right)$$

We can update the parameters by minimizing the cost function (or equivalently, maximizing likelihood function)

Estimation of the Parameters

$$J(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\right) - \log\left(1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}\right)\right)$$

$$\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}\frac{e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}{1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}\right) = -\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\left(y^{(i)} - \frac{e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}{1 + e^{\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}}}\right) =$$

$$\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\left(\sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\right) - y^{(i)}\right)$$

$$\rightarrow \frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}\left(\sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{i})}\right) - y^{(i)}\right) = \frac{1}{n}\boldsymbol{X}^{T}(\boldsymbol{\sigma}(\boldsymbol{X\beta}) - \boldsymbol{y})$$

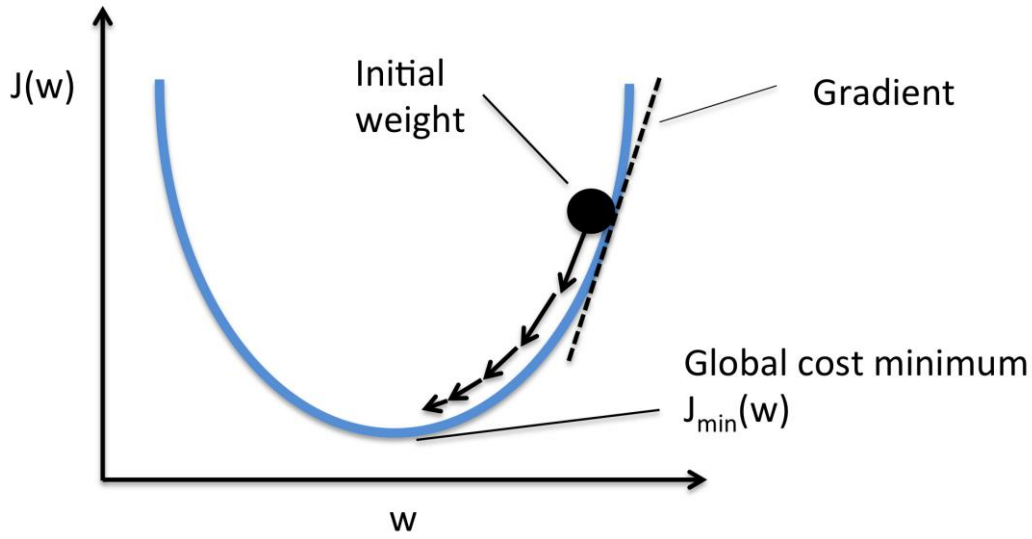
where $\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots x_p^{(1)} \\ \vdots & \vdots & \dots \vdots \\ 1 & x_1^{(n)} & \dots x_p^{(n)} \end{bmatrix}$, $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$, $\boldsymbol{\sigma}(\boldsymbol{X\beta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{1})}\right) \\ \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{2})}\right) \\ \vdots \\ \sigma\left(\boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{n})}\right) \end{bmatrix}$

Updating the Parameters using Gradient Descent

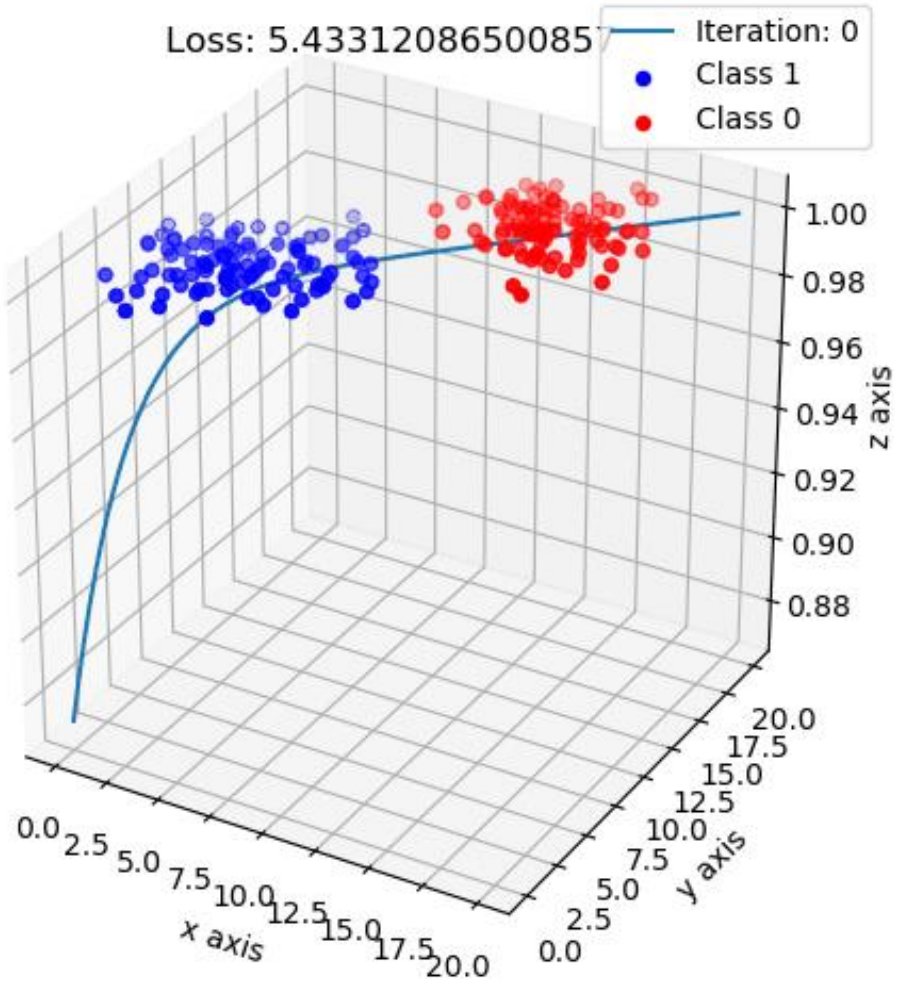
Now that we have the derivatives we can update the parameters using *Gradient Descent:*

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} - \alpha \frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta}^{old} - \alpha \frac{1}{n} \boldsymbol{X}^T (\boldsymbol{\sigma}(\boldsymbol{X}\boldsymbol{\beta}) - \boldsymbol{y})$$

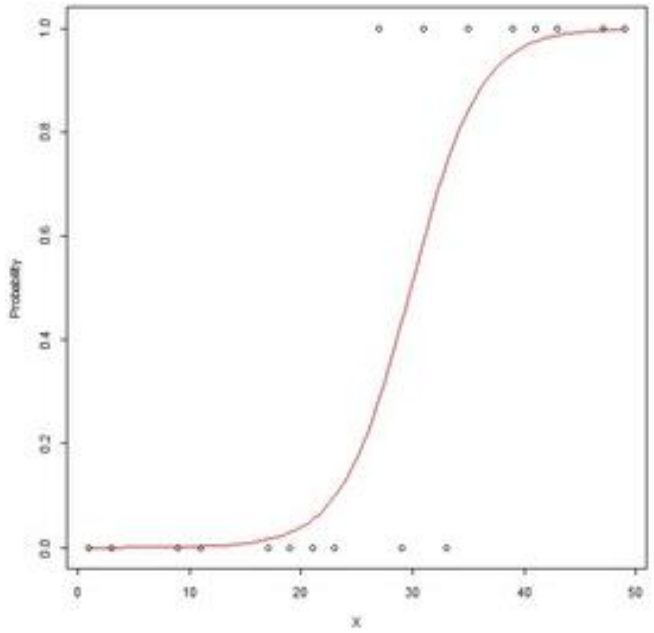
- α $(learning\ rate)$: controls the rate of convergence (should be moderate).



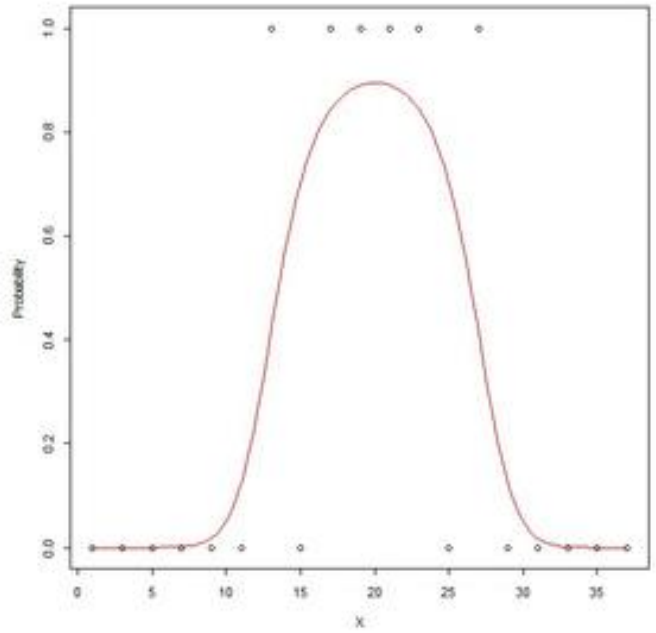
Example


Loss: 5.4331208650085
Iteration: 0
Class 1
Class 0
1.00
0.98
0.96
0.94
0.92
0.90
0.88
z axis
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
x axis
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
y axis

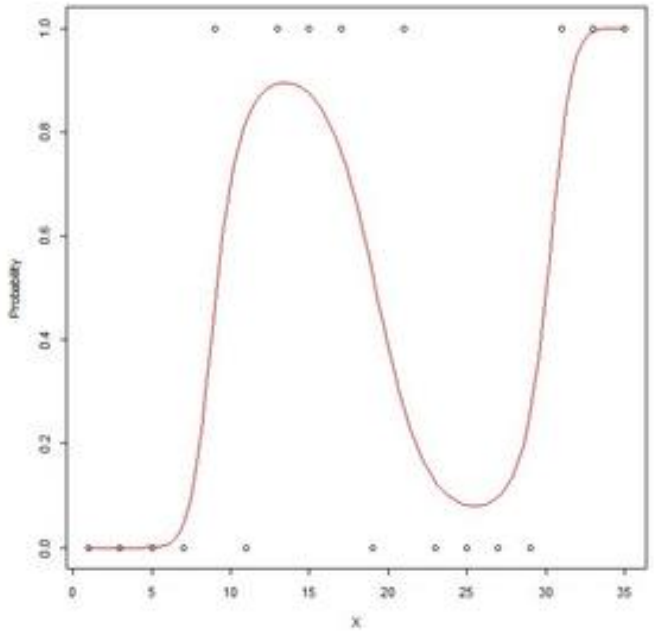
Non-Linear Logistic Regression



Linear



Quadratic



Cubic

*Multi-Class Logistic Regression

We can use logistic regression for multi-class classification as well. This is called *Multinomial Logistic Regression* or *Softmax Regression*.

- Each class has its own probability function with its own parameter vector, s.t. for any given x, they all sum to one.

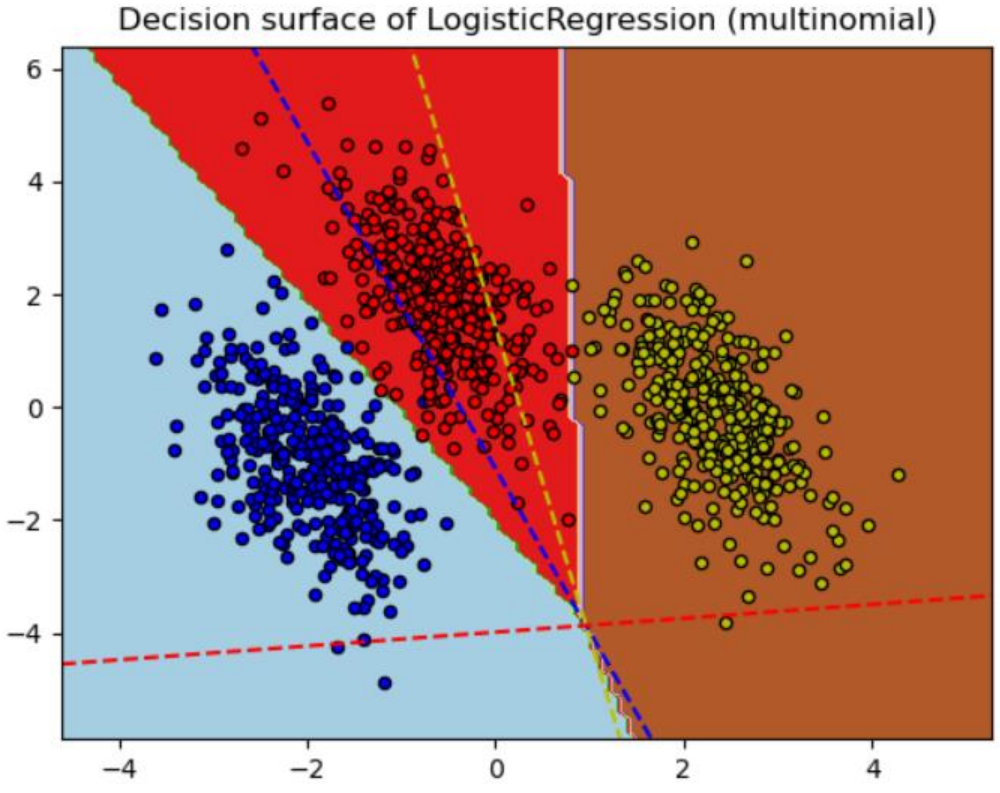
$$\mathbb{P}_j(\boldsymbol{x}, \boldsymbol{\beta}) = \mathbb{P}(Y = j|\boldsymbol{x}, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}_j^T \boldsymbol{x}}}{\sum_{j=1}^{K} e^{\boldsymbol{\beta}_j^T \boldsymbol{x}}}, \qquad j = 1,2, \dots, K$$

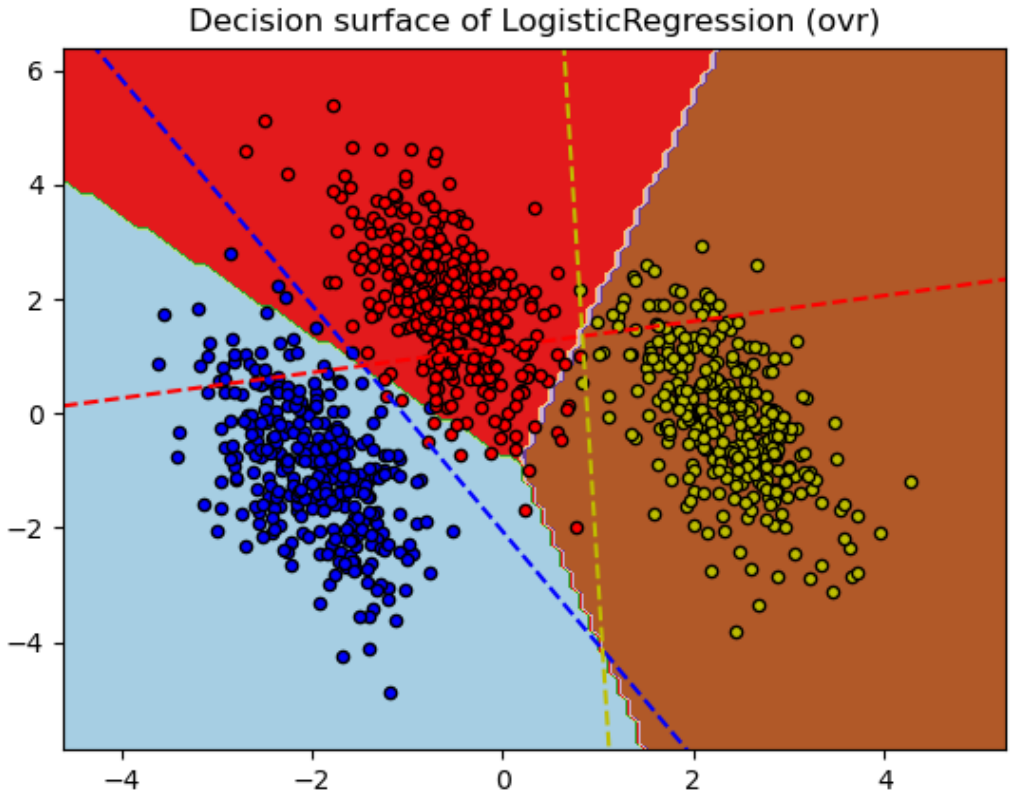
Other approaches:

- One-vs-All
- One-vs-One

- In softmax model we should use penalty term to keep the parameters small. (L_2 or L_1 regularizers)

*Multi-Class Logistic Regression Decision Surface





*Regularized Logistic Regression

Like linear regression, in order to reduce the variance of the model, we can use regularization for the logistic regression as well:

- L_2 norm regularizer:

$$cost = -l(\boldsymbol{\beta}) + \lambda \left|\left|\boldsymbol{\beta}\right|\right|_{L2}^{2} = -l(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- L_1 norm regularizer:

$$cost = -l(\boldsymbol{\beta}) + \lambda \left|\left|\boldsymbol{\beta}\right|\right|_{L1} = -l(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \left|\beta_j\right|$$

- Scikit-learn uses L_2 norm regularization by default.