Machine Learning & its Application

Linear Models

02/10/2023
Done by: Issa Hammoud

Outline

- Introduction
- Linear Regression
- Linear Classification
- Unified Framework
- Conclusion

Introduction

- The easiest problem to begin with are linear problems.
- In the linear case, we suppose that the data is linear.
- We can have 2 use cases:
  - Regression: we suppose there is a line that fits the data.
  - Classification: we suppose there is a line that separates the data.
- We will present and solve each problem apart.

Outline

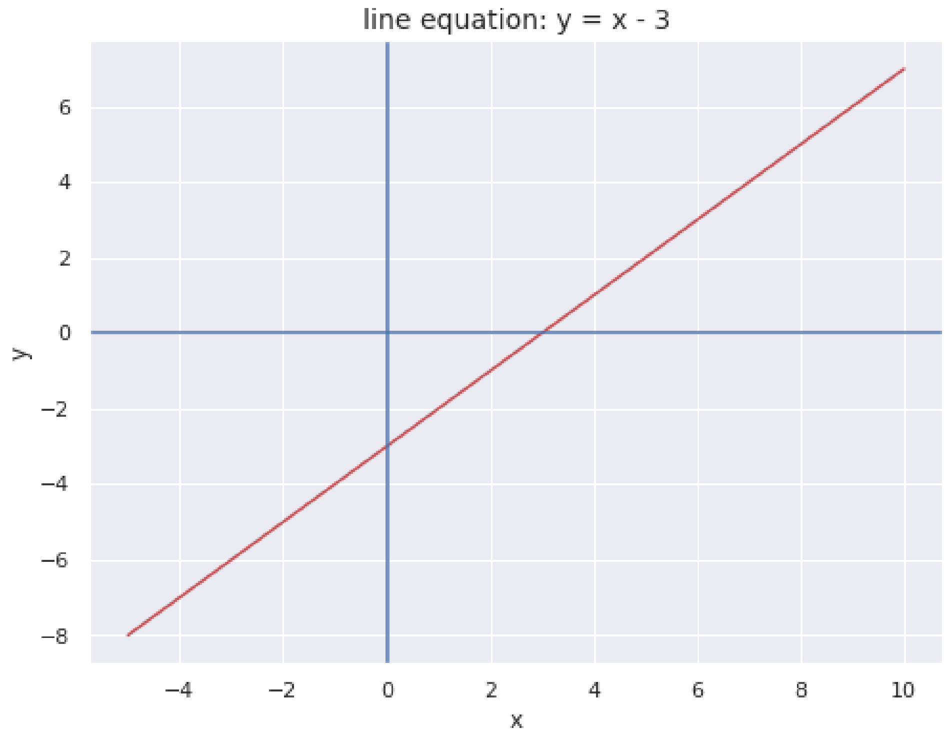
- Introduction
- Linear Regression
- Linear Classification
- Unified Framework
- Conclusion

Linear Regression

- Introduction
- Line of best fit
- High dimensional space
- The normal equation
- Exercise

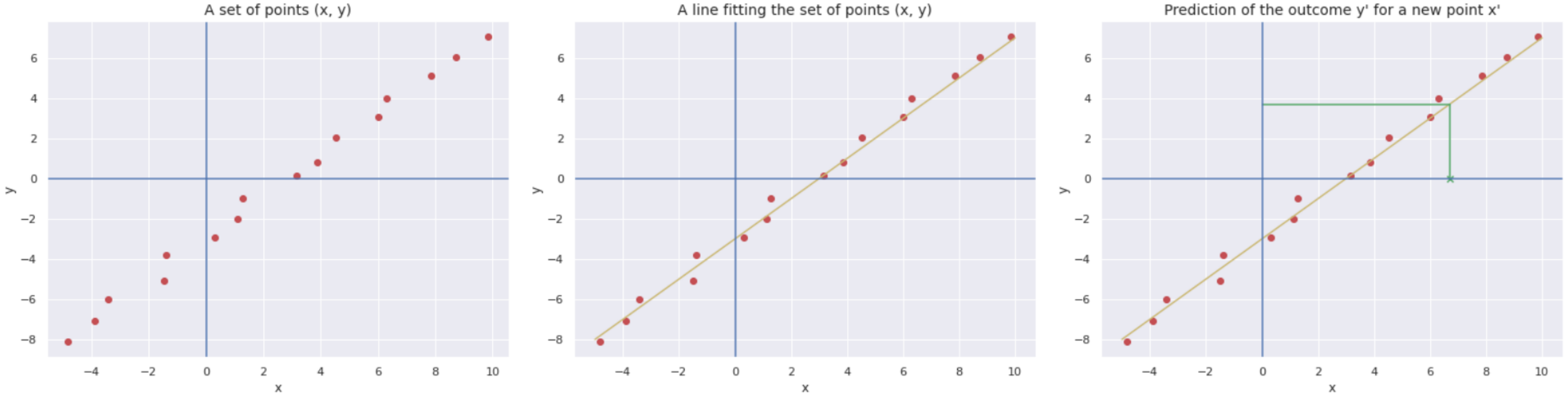
Introduction

- We will begin in 2D for simplicity.
- A line equation has the form: $y=ax+b$
- We can draw a line using 2 points only.
- But what can we do with a line?



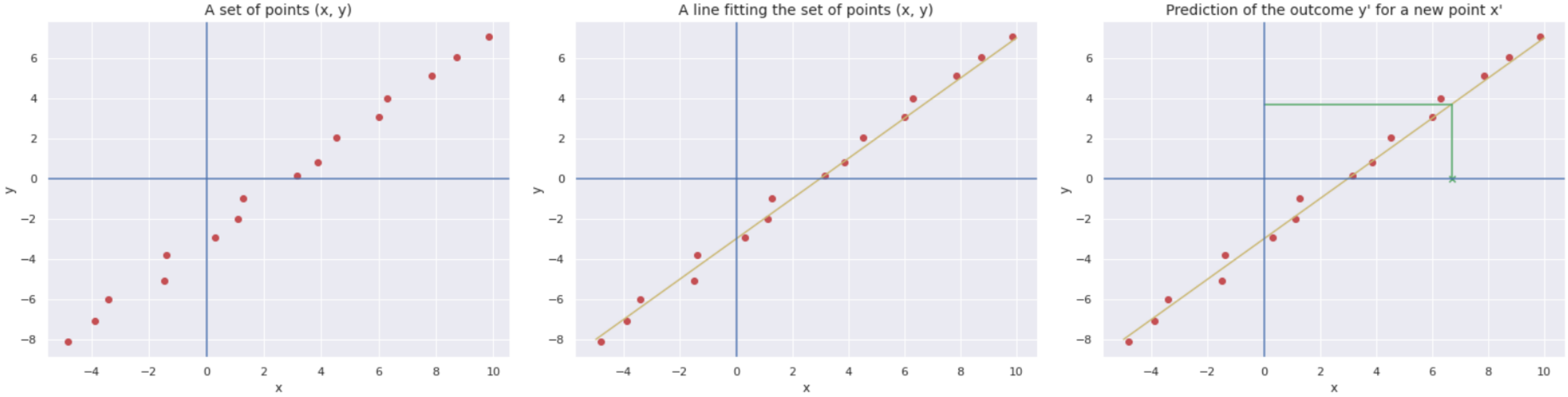
Introduction

- Imagine we have a set of points (x, y): x is the area of a house and y its price.
- We want to find the line that fits those points (x, y).
- Thus, if we get a new point x' (house area), we can predict the outcome y' (its price).



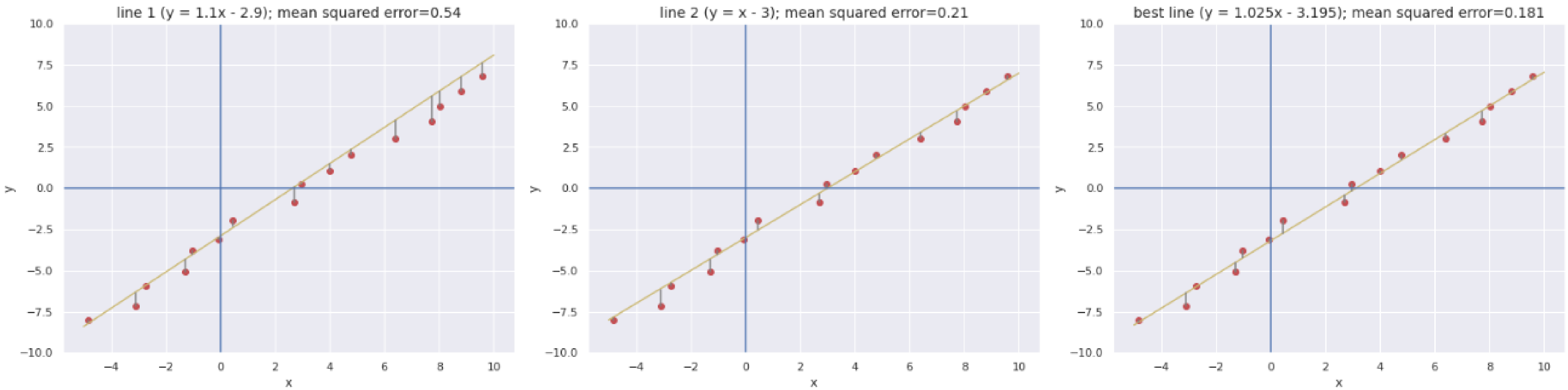
Introduction

- The data is not perfectly aligned, it has some noise.
- Thus, we cannot use any 2 points to find the line.
- We need to find the line that fits best the data, a.k.a the line of best fit.



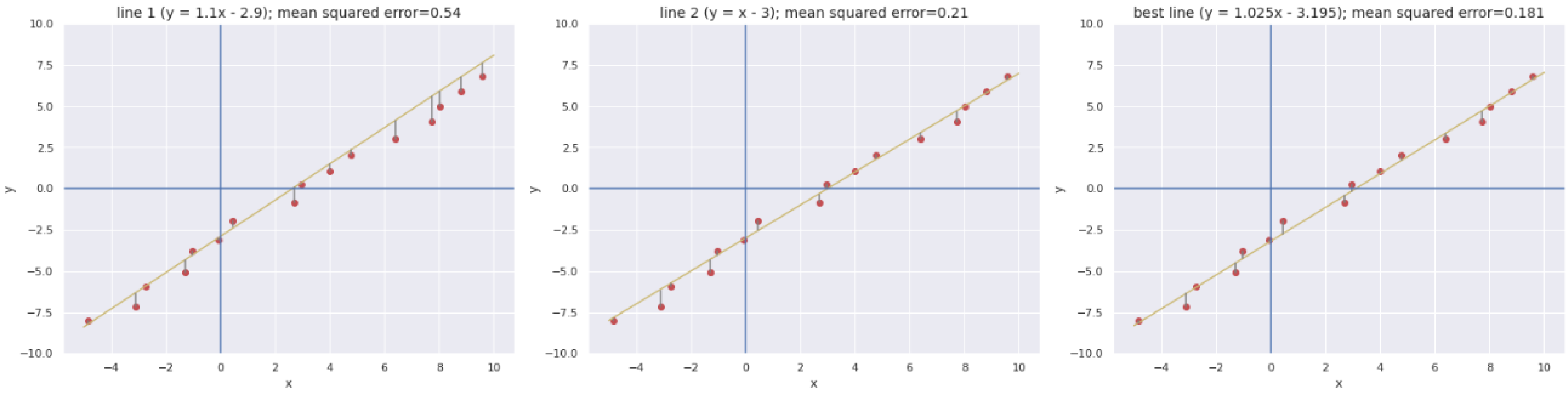
Line of Best Fit

- The word "best" should be defined with respect to a metric.
- Let's take 3 lines (for different values of *a* and *b*).
- We need to compute an error metric in order to find which line is better.



Line of Best Fit

- The error should represent how much a point is far from the line on average
- It should a positive value, so the values don't cancel each other.
- One choice is to use mean squared error. And thus, line 3 is better than lines 1 and 2.



Line of Best Fit

- We found the best line over 3 possible values of *a* and *b*.
- However, we have an infinite number of possibilities.
- We can't compare all possible values, we need a way to find the smallest mean square error.
- The problem is to find the minimum of a function, which we know how to solve:

Compute the derivative and set it to zero

Line of Best Fit

- Let's formalize the problem and define the optimization to solve:

$$\text{given } n \text{ data points } X = \{x_1,\ x_2,\ \ldots, x_n\} \text{ and } Y = \{y_1,\ y_2,\ \ldots,\ y_n\}\ ;\ x_i,\ y_i\ \in R$$

$$\min_{a,\,b}\ (Y' - Y)^2 = \min_{a,\,b}\ \frac{1}{n}\sum_{i}^{n}(ax_i + b\ -\ y_i)^2$$

- We want to find (a, b) that gives minimum average distance between the line and all points (xi, yi).

$$\frac{\partial \frac{1}{n}\sum_i^n (ax_i + b\ -\ y_i)^2}{\partial a} = \frac{2}{n}\sum_{i}^{n}(ax_i + b - y_i)^2 x_i\ = 0$$

$$\frac{\partial \frac{1}{n}\sum_i^n (ax_i + b\ -\ y_i)^2}{\partial b} = \frac{2}{n}\sum_{i}^{n}(ax_i + b - y_i)^2\ = 0$$

Line of Best Fit

- We have 2 equations with 2 unknowns.
- By rearranging the terms, we can find the line of best fit.

$$a = \frac{\sum_i^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_i^n (x_i - \overline{x})^2} \ ; \ b = \overline{y} - a\overline{x}$$

$$\text{where } \overline{x} = \frac{1}{n}\sum_i^n x_i; \ \overline{y} = \frac{1}{n}\sum_i^n y_i$$

High Dimensional Space

- We solved the problem in a simple setting when **x** and ***y*** are both scalars.
- For the example of house pricing, many features contribute to estimating the price.
- We call each feature a "dimension", and thus **x** is a vector no more a scalar.
- We also may want to predict multiple quantity not only the price.
- We will take the general case when ***X*** and ***Y*** are both vectors, of different dimensions.

High Dimensional Space

- We can still define the relationship between a data point Xi and Yi as: ***Yi = AXi + b***.

- To be able to relate ***Xi*** and ***Yi***, ***A*** should be a matrix and ***b*** a vector, as follows:

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \\ \cdot \\ \cdot \\ \cdot \\ Y_{iq} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1p} \\ A_{21} & A_{22} & \dots & A_{2p} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A_{q1} & A_{q2} & \dots & A_{qp} \end{pmatrix} \begin{pmatrix} X_{i1} \\ X_{i2} \\ \cdot \\ \cdot \\ \cdot \\ X_{ip} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_q \end{pmatrix}$$

- ***Yi*** a vector of dimension *q*, ***Xi*** of dimension *p*, ***b*** of dimension *q*, ***A*** a matrix of size *(q, p)*.

High Dimensional Space

- The relationship between all data points can also be expressed as : ***Y = AX + b.***
- ***A*** and ***b*** are the same because they are independent of the number of examples.
- ***X*** and ***Y*** are now matrices containing all the data points ,as follows:

$$\begin{pmatrix} Y_{11} & Y_{21} & \dots & Y_{n1} \\ Y_{12} & Y_{22} & \dots & Y_{n2} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ Y_{1q} & Y_{2q} & \dots & Y_{nq} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1p} \\ A_{21} & A_{22} & \dots & A_{2p} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ A_{q1} & A_{q2} & \dots & A_{qp} \end{pmatrix} \begin{pmatrix} X_{11} & X_{21} & \dots & X_{n1} \\ X_{12} & X_{22} & \dots & X_{n2} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X_{1p} & X_{2p} & \dots & X_{np} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_q \end{pmatrix}$$

- ***X*** is of shape *(p, n)* and ***Y*** of shape *(q, n)*.

High Dimensional Space

- A final trick to make the equation easier is to set A and b into a single matrix.
- We can easily prove that the following formulation is equivalent:

$$\begin{pmatrix} Y_{11} & Y_{21} & \dots & Y_{n1} \\ Y_{12} & Y_{22} & \dots & Y_{n2} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ Y_{1q} & Y_{2q} & \dots & Y_{nq} \end{pmatrix} = \begin{pmatrix} b_1 & A_{11} & A_{12} & \dots & A_{1p} \\ b_2 & A_{21} & A_{22} & \dots & A_{2p} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ b_q & A_{q1} & A_{q2} & \dots & A_{qp} \end{pmatrix} \begin{pmatrix} 1 & 1 & \dots & 1 \\ X_{11} & X_{21} & \dots & X_{n1} \\ X_{12} & X_{22} & \dots & X_{n2} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ X_{1p} & X_{2p} & \dots & X_{np} \end{pmatrix}$$

- We can write the equation as ***Y = AX***, where ***A*** contains ***b*** as well.

High Dimensional Space

- Now it is time to find A and b the fits well the data as we did for the scalar case.
- We will define an optimization problem and set the derivative to zero as usual.
- The problem can be expressed as follows:

$$\min_A || Y' - Y ||_2^2 = \min_A || AX - Y ||_2^2 = \min_A \sum_i^q \sum_j^n ((AX)_{ij} - Y_{ij})^2$$

- However, computing the matrices derivative is not easy, we will define some properties.

The Normal Equation

For a given matrix ***M*** of shape (***n, n***), and ***Q*** of shape (***n, p***)

1. We define the trace (*tr*) of **M** as the sum of its diagonal elements. The trace of a matrix is a scalar value:

$$\mathrm{tr}(M) = \sum_i M_{ii}$$

2. The transpose of a matrix is the swapping of its rows and columns. Then, the shape of transpose **Q** is (***p, n***):

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & \dots & Q_{1p} \\ Q_{21} & Q_{22} & \dots & Q_{2p} \\ . & . & & . \\ . & . & . & . \\ . & . & . & . \\ Q_{n1} & Q_{n2} & \dots & Q_{np} \end{pmatrix}; \; Q^T = \begin{pmatrix} Q_{11} & Q_{21} & \dots & Q_{n1} \\ Q_{12} & Q_{22} & \dots & Q_{n2} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ Q_{1p} & Q_{2p} & \dots & Q_{np} \end{pmatrix}; \; (MQ)^T = Q^T M^T$$

3. The L2 norm square of a matrix equals the trace of the matrix times its transpose:

$$||Q||_2^2 = \mathrm{tr}\left(Q.Q^T\right) = \mathrm{tr}\left(Q^T.Q\right)$$

4. The derivative of the trace of a matrix with respect to itself is the identity matrix:

$$\frac{\partial\,\mathrm{tr}(M)}{\partial M} = I$$

5. The derivative of scalar ***b*** with respect to a matrix **M** is a matrix of the same dimension as **M** (point 4 is an example).

6. Some useful rules:

$$\frac{\partial MQ}{\partial M} = Q^T; \; \frac{\partial MQ^T}{\partial Q} = M; \; \frac{\partial MM^T}{\partial M} = 2M; \; \frac{\partial MZM^T}{\partial M} = MZ + MZ^T; \; \left(QQ^T\right)^T = QQ^T$$

The Normal Equation

- We want to compute the derivative with respect to ***A***.
- We will use the aforementioned properties to solve it step by step.

$$\min_A \| AX - Y \|_2^2 = \min_A \operatorname{tr}\big((AX - Y)(AX - Y)^T\big) \text{ (property 3)}$$

$$\frac{\partial \operatorname{tr}\big((AX - Y)(AX - Y)^T\big)}{\partial A} = \frac{\partial (AX - Y)(AX - Y)^T}{\partial A} \cdot \frac{\partial \operatorname{tr}\big((AX - Y)(AX - Y)^T\big)}{\partial (AX - Y)(AX - Y)^T} \text{ (chain rule)}$$

$$\frac{\partial \operatorname{tr}\big((AX - Y)(AX - Y)^T\big)}{\partial (AX - Y)(AX - Y)^T} = I \text{ (property 4)}$$

$$\frac{\partial \operatorname{tr}\big((AX - Y)(AX - Y)^T\big)}{\partial A} = \frac{\partial (AX - Y)(AX - Y)^T}{\partial A}$$

The Normal Equation

- We simplified the expression in order to compute the derivatives.
- This equation is known as the ***normal equation***.

$$(AX - Y)(AX - Y)^T = (AX - Y)\left(X^TA^T - Y^T\right) = AXX^TA^T - AXY^T - YX^TA^T + YY^T$$

$$\frac{\partial\left(AXX^TA^T - AXY^T - YX^TA^T + YY^T\right)}{\partial A} = AXX^T + AXX^T - YX^T - YX^T = 2AXX^T - 2YX^T = 0$$

$$AXX^T = YX^T$$

$$A = YX^T\left(XX^T\right)^{-1}$$

N.B.: You may find a slightly different formulation in most references. This is due to the way we define the matrices X and Y. In this derivation, we defined X and Y as matrices of shape (number_of_features, number_of_samples). In practice, we use (number_of_samples, number_of_features). Thus the difference.

Exercise

- We will do a small exercise on Google Colab.
- The goal is to implement a linear regression function in Python.
- We will code the scalar version and the matrix version for higher dimensional data.
- We will compare the result to sklearn LinearRegression class.

Outline

- Introduction
- Linear Regression
- Linear Classification
- Unified Framework
- Conclusion

Linear Classification

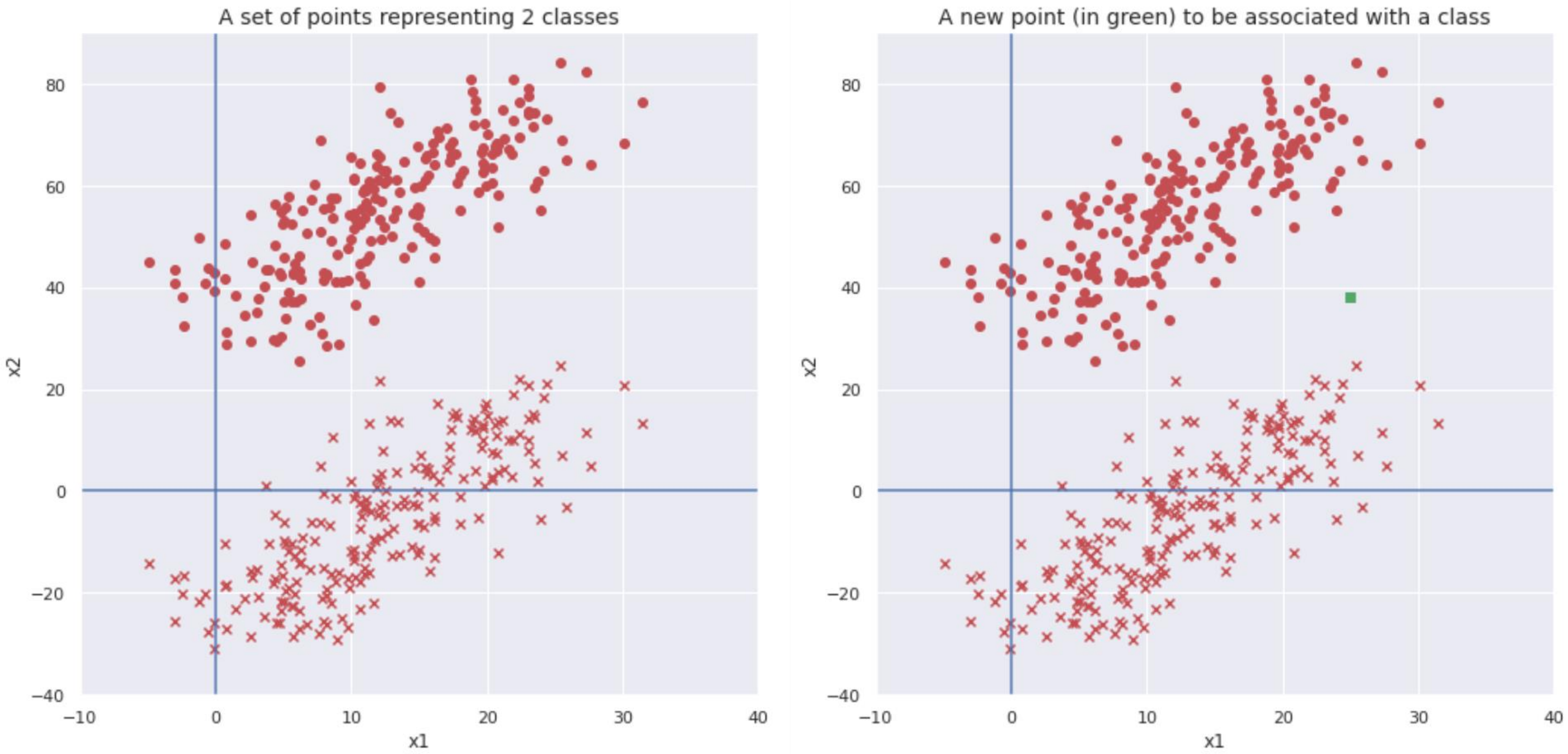
- Introduction
- Binary classification
  - Probabilistic modelling
  - Maximum likelihood estimation
  - Binary classification as Bernoulli distribution
- Multiclass classification

Introduction

- Classification is assigning a label ***yi*** to an input ***xi***, where ***yi*** is a discrete value.
- In case yi can take only 2 values (e.g. cat vs dog) we call it binary classification.
- If ***yi*** can take more than 2 values, we call it multiclass classification.
- In both cases, we suppose that an input ***xi*** can take only a single label ***yi***.
- We call this problem single label classification (against multi label classification).
- An example of multilabel classification is movie type prediction, where a movie can belong to multiple type at the same time. It is out of the scope of this course.

Binary Classification

- Let's take a simple binary classification problem as illustrated below.
- Given a set of linearly separable points (x, y), we want to assign for a new point x' a label y'.
- Here y is a circle or cross.



Binary Classification

- There are two main approaches to solve this problem:
  1. Find the line that maximize the margin between the 2 set of points.
  2. Learn a probabilistic model to predict a probability of belonging to one of the classes.
- One well-known classical algorithm that follows the first approach is ***SVM***.
- The problem with this approach is that it doesn't give a confidence measurement.
- Deep Learning algorithms follow the second approach.
- We will present a probabilistic linear classifier known as ***logistic regression***.

Probabilistic Modeling

- Probabilistic classifiers should output a probability of belonging to one class.
- Thus, the output of a binary classifier is not 0 or 1 but a value in the range [0-1].
- To be able to define a probability, we need to define random variables.
- Let $\boldsymbol{x}$ be a random variable that can take on different realizations $\boldsymbol{X_i}$.
- Let $\boldsymbol{y}$ be a function of $\boldsymbol{x}$ and some unknown parameters $\boldsymbol{A}$, $\boldsymbol{y=f(x; A)}$.
- Thus, $\boldsymbol{y}$ is also a random variable that can take different realization $\boldsymbol{Y_i}$.

Probabilistic Modeling

- We define the data-generating process, ***p(x, y)*** that can generate any ***(x, y)*** value.
- We suppose that the data is independent and identically distributed (*iid*).
- Our goal is to compute the probability of a label ***y***, given **x** and ***A***: ***p(y/x; A)***.
- For instance, for the green point x', based on the learned knowledge A:
  - Compute p(y=circle/x'; A) and p(y=cross/x'; A)
  - If p(y=circle/x'; A) > p(y=cross/x'; A) then x' is a circle with a given probability.
- To solve this problem we should have 2 things:
  1. The value of the parameter ***A***.
  2. The expression of ***p(y/x; A)***.

Probabilistic Modeling

- Finding the value of A is also called parameter estimation.
- We need to find an estimator of A. An estimator in general is a function of the data.
- For instance, an estimator of the expectation is the average value.
- But how can we find estimators? The answer is maximum likelihood principle.

Maximum Likelihood Estimation

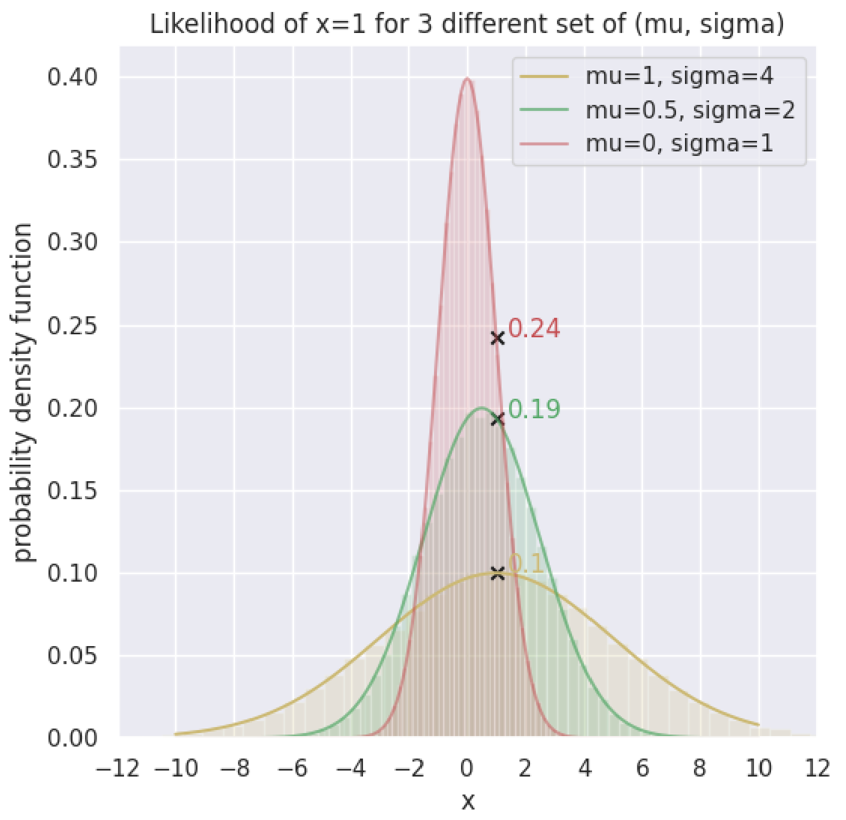
- Maximum likelihood is a way to find estimators. Let's take an example.
- Imagine we have a set of points generated from a Gaussian distribution of unknown mean and variance.
- The probability density function of a Gaussian distribution is as follows:

$$p(x;\ \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$$

- The question to answer is "what are the parameters that highly likely generated our data"?

Maximum Likelihood Estimation

- Let's take the first data point, let it be X=1 and plot 3 different Gaussians.
- For each we want to see which μ and σ are more likely to generate X=1.
- Among all curves, the red one has a bigger likelihood.
- We say that we will choose the parameters that maximize the likelihood.



Maximum Likelihood Estimation

- However, here we compared only 3 possible values.
- In addition, we should choose the maximum likelihood for all data points not only X=1.
- Thus, we can write the optimization problem as:

$$\max_A P(X;\ A) = \max_A P(X_1, X_2, \ldots, X_n;\ A) = \max_A \prod_i^n P(X_i;\ A)$$

- However, it is hard to optimize a product. We can transform it to sum with a log.
- Thus, an equivalent optimization problem is the log likelihood:

$$\max_A \prod_i^n P(X_i;\ A) \equiv \max_A \sum_i^n \log P(X_i;\ A)$$

N.B.: *P(X1, X2)= P(X1)xP(X2) because the data is independent.*

Maximum Likelihood Estimation

- Now let's apply maximum log likelihood to estimate μ and σ.

$$P(X_i;\ A = (\mu,\ \sigma))\ =\ \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{X_i\ -\ \mu}{\sigma}\right)^2\right)$$

$$\log P(X_i;\ A = (\mu,\ \sigma))\ =\ \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right)\ -\frac{1}{2}\left(\frac{X_i\ -\ \mu}{\sigma}\right)^2$$

$$\sum_i^n \log P(X_i;\ A = (\mu,\ \sigma))\ =\ -n\log\left(\sigma\sqrt{2\pi}\right)\ -\frac{1}{2\sigma^2}\sum_i^n (X_i\ -\ \mu)^2$$

$$\frac{\partial \sum_i^n \log P(X_i;\ A = (\mu,\ \sigma))}{\partial\mu}\ =\ \frac{1}{\sigma^2}\sum_i^n (X_i\ -\ \mu)\ =\ 0;\ \boxed{\mu\ =\frac{1}{n}\sum X_i = \overline{X}};\ \sigma \neq 0$$

$$\frac{\partial \sum_i^n \log P(X_i;\ A = (\mu,\ \sigma))}{\partial\sigma}\ =\ -n\frac{1}{\sigma}\ +\ \frac{1}{\sigma^3}\sum_i^n (X_i\ -\ \mu)^2\ =\ 0;\ \boxed{\sigma^2\ =\ \frac{1}{n}\sum_i^n (X_i\ -\ \overline{X})^2}$$

Binary Classification as Bernoulli

- Let's get back to binary classification.
- We mentioned that the goal is to compute ***p(y/x; A)***.
- We showed how maximum likelihood can be used to estimate the parameters A.
- What remain is to find the expression of the ***p(y/x; A)***.
- Binary classification is similar to a coin flipping task when the outcome is 1 or 0.
- Thus, it can be modeled with a Bernoulli distribution.

Binary Classification as Bernoulli

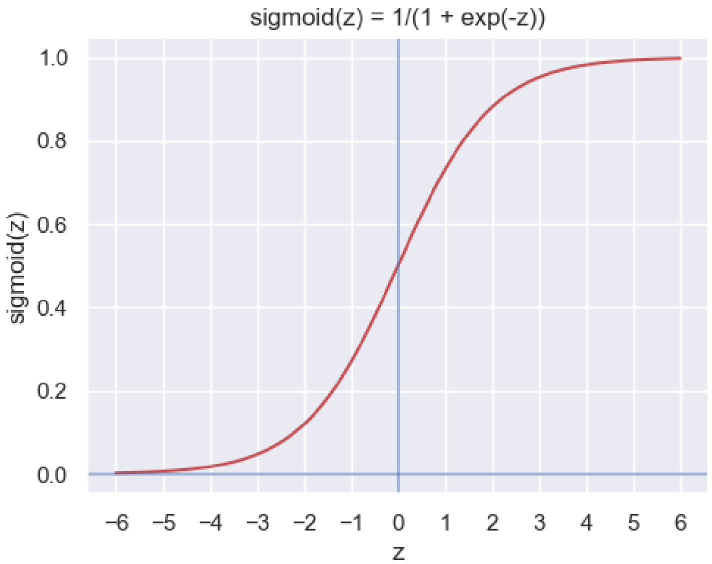
- Bernoulli distribution is defined as follows:

$$P(Y = y_i) = \begin{Bmatrix} \theta & \text{if } y_i = 1 \\ 1-\theta & \text{if } y_i = 0 \end{Bmatrix} = \theta^{y_i}(1-\theta)^{1-y_i}$$

- In coin flipping, ***θ*** is the probability of having a head.
- It is sufficient to compute ***P(Y=1)***, because ***P(Y=0) = 1 – P(Y=1)***.
- Thus, we can write Bernoulli as ***P(Y=1) = θ***.
- In our case, we don't have ***P(Y=1)***, instead we have ***P(Y=1/x; A)= θ***.
- But ***θ*** should be a function of **x** and ***A***, thus ***P(Y=1/x; A)= θ(x; A)***.

Binary Classification as Bernoulli

- We begin by talking about linear classifiers, so we should have a linear relationship.
- ***θ(x; A)*** is a function of x and A, and should be between 0 and 1.
- We can define a linear relation ***Z=AX***, then squash the values between 0 and 1.
- One function that do this is the sigmoid function, represented as follows:



Binary Classification as Bernoulli

- We finally get all the ingredients to solve linear classification problem.
- We have ***p(Y=1/X; A)=sigmoid(AX)***. This classifier is called ***logistic regression***.
- We can apply maximum likelihood to find A, or equivalently the Negative Log Likelihood (NLL) and set the derivative to 0.

$$P(Y = 1/X_i; A) = \sigma(AX_i);\ \text{where } \sigma(x) = \frac{1}{1+e^{-x}}$$

$$-\log P(Y = 1/X; A) = \log\left(1+e^{-AX}\right)$$

$$\min_A -\sum_i^n \log P(Y_i = 1/X_i; A) = \min_A \sum_i^n \log\left(1+e^{-AX_i}\right)$$

$$\frac{\partial \Sigma_i^n \log\left(1+e^{-AX_i}\right)}{\partial A} = \sum_i^n \frac{X_i e^{-AX_i}}{1+e^{-AX_i}} = 0$$

Binary Classification as Bernoulli

- There is no closed-form solution for this equation.
- We need an iterative algorithm, like gradient descent, to solve it (next lesson).
- Note that, applying the NLL to a Bernoulli distribution give:

$$P(Y = y_i) = \theta^{y_i}(1-\theta)^{1-y_i}$$

$$-\log P(Y = y_i) \; = \; -\log \theta^{y_i}(1-\theta)^{1-y_i} \; = \; -y_i \log \theta \; - (1-y_i)\log(1-\theta)$$

- This is the well-known binary cross entropy loss.
- Cross entropy maximizes the likelihood of a binary classifier parameters to have generated the training data.

Multiclass Classification

- Everything we did for binary classification applies here as well with few exceptions.
- First, we can't model a multiclass problem with Bernoulli distribution.
- In addition, we can't use the sigmoid function.
- Multiclass problem is similar to a dice roll, where each face has a probability to occur.
- We can model it with a Multinoulli distribution, that is defined as follows:

$$\text{Bernouilli}: \ P(Y = y_i) = \begin{cases} \theta & \text{if } y_i = 1 \\ 1 - \theta & \text{if } y_i = 0 \end{cases} = \theta^{y_i}(1-\theta)^{1-y_i} \ ; \ y_i \in \{0, 1\}; \ 0 \leqslant \theta \leqslant 1$$

$$\text{Multinouilli}: \ P(Y = y_i) = \prod_{j}^{k} \theta_j^{y_{ij}} \ ; \ y_i \in \{0,1\}^k \ ; \ 0 \leqslant \theta_j \leqslant 1 \ ; \ \sum_{j}^{k} \theta_j = 1$$

Multiclass Classification

- The output of a *k-class* classifier is a vector of probability of size *k*.
- Each value represents the probability of a class to occur.
- In addition, the *k* value should sum to 1.
- To obtain this from a linear projection Z=AX, we can apply the softmax function:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j^k \exp(z_j)} \; ; \; z = Ax; \; z \in R^k; \; z_i \in R$$

- Thus we can define our probabilistic classifier.

Multiclass Classification

- Once we have the expression of p(y/x; A) we can apply Maximum likelihood to find A.
- Applying it to a Multinoulli distribution gives the following:

$$P(Y = y_i) = \prod_{j}^{k} \theta_j^{y_{ij}}$$

$$-\log P(Y = y_i) = -\log \prod_{j}^{k} \theta_j^{y_{ij}} = -\sum_{j}^{k} \log \theta_j^{y_{ij}} = -\sum_{j}^{k} y_{ij} \log \theta_j$$

- This is the well-known cross entropy loss.
- Similarly to binary classification, we need an iterative algorithm to find the solution.

Outline

- Introduction
- Linear Regression
- Linear Classification
- Unified Framework
- Conclusion

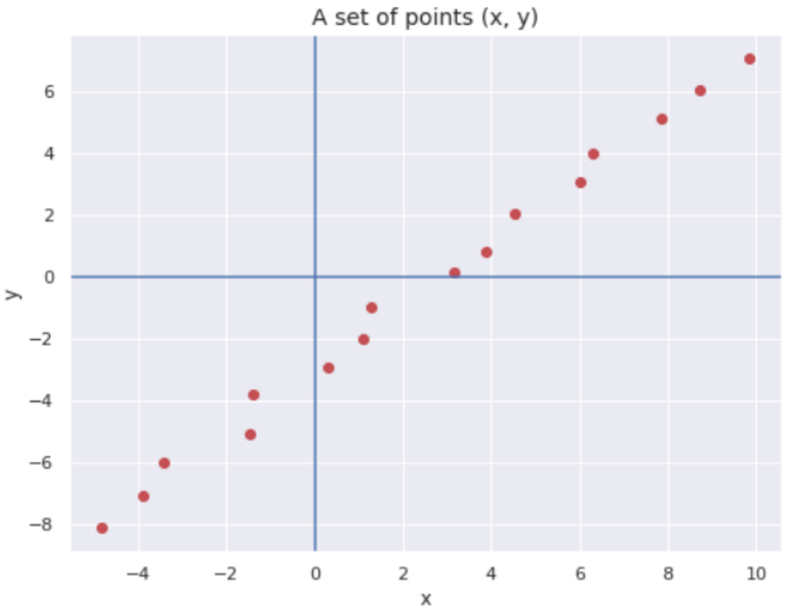
Unified Framework

- Let's summarize what we did in classification to solve the problem:
  - Define the distribution of p(y/x; A) (Bernoulli for binary, Multinoulli for multiclass).
  - Apply Maximum Likelihood principle to estimate the parameter A.
  - Compute the derivative to be used later with gradient descent (as there is no closed-form solution).
- We obtained the well-known cross entropy loss for classification.
- There are well-defined systemic steps to find classification solutions.
- It would be nice if we can have something similar to linear regression.

Unified Framework

- First, we need to define the probabilistic setting for linear regression.
- However, we should use a continuous distribution not a discrete one.
- Suppose there is a perfect line y=ax+b that should pass through all points.
- However, there is a noise term added to each point.
- We can express it as follows:

$$y = ax + b + \epsilon\ ;\ \epsilon \sim N(0,\ \sigma)$$



Unified Framework

- Thus, we can find p(y/x; A) and apply Negative Log likelihood as follows:

$$\epsilon = y - ax - b \sim N(0,\ \sigma) \rightarrow y \sim N(ax + b,\ \sigma)\ \rightarrow P(y/x;\ a, b) \sim N(ax + b,\ \sigma)$$

$$P(y/x;\ a, b) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(y - ax - b)^2}{\sigma^2}\right)$$

$$NLL:\ -\sum_i \log \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(y - ax - b)^2}{\sigma^2}\right) = -\sum_i \log \frac{1}{\sigma\sqrt{2\pi}}\ -\frac{1}{2}\frac{(y_i - ax_i - b)^2}{\sigma^2}\ \propto\ -\sum_i (y_i - (ax_i + b))^2$$

- As we can see, we get the mean squared error as before.
- It means that mean squared error maximize the likelihood of the data being generated by a Gaussian distribution.

Exercise

- Show the importance of using sigmoid in binary classification from an optimization point of view.
- Hints:
  - Take the case where the correct label y=1 for simplicity.
  - Plot the sigmoid function and the negative log likelihood applied to it.
  - See what is the error when the model predict correctly and what is the error in the other case.
  - The ideal behavior is to have very small error with correct prediction, and a proportional error otherwise.

Outline

- Introduction
- Linear Regression
- Linear Classification
- Unified Framework
- Conclusion

Conclusion

- We presented linear models for regression and classification.
- In the regression case, we derived and solved the normal equation in a closed-form.
- In classification, we had to define a probabilistic setting in order to output probability.
- The goal is to compute ***p(y/x; A):***
  - We need to model the problem following a distribution (e.g. Bernoulli for binary case).
  - Then, we can apply the negative log likelihood to derive loss functions to get ***A***.
- We applied the same framework for regression as well.
- We obtained the well-known loss functions for both classification and regression.