Lecture 2: Linear Predictors

Gonzalo De La Torre Parra, Ph.D.

Fall 2021

Quick Recap

- **What is supervised ML?**

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.
- **What is unsupervised ML?**

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.

- **What is unsupervised ML?**

  It is a ML problem in which we are given n data points,

  $$x_1, x_2, \dots, x_n,$$

  and are asked to find or learn a pattern in x, for example, divide them into clusters.

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.
- **What is unsupervised ML?**

  It is a ML problem in which we are given n data points,

  $$x_1, x_2, \ldots, x_n,$$

  and are asked to find or learn a pattern in x, for example, divide them into clusters.
- **What is regression?**

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.

- **What is unsupervised ML?**

  It is a ML problem in which we are given n data points,

  $$x_1, x_2, \ldots, x_n,$$

  and are asked to find or learn a pattern in x, for example, divide them into clusters.

- **What is regression?**

  It is a supervised ML problem in which the variable to be predicted y can take any real value.

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.
- **What is unsupervised ML?**

  It is a ML problem in which we are given n data points,

  $$x_1, x_2, \ldots, x_n,$$

  and are asked to find or learn a pattern in x, for example, divide them into clusters.
- **What is regression?**

  It is a supervised ML problem in which the variable to be predicted y can take any real value.
- **What is classification?**

Quick Recap

- **What is supervised ML?**

  It is a ML problem in which we are given n pairs of data points,

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

  and are asked to find or learn a relationship between x and y.

- **What is unsupervised ML?**

  It is a ML problem in which we are given n data points,

  $$x_1, x_2, \ldots, x_n,$$

  and are asked to find or learn a pattern in x, for example, divide them into clusters.

- **What is regression?**

  It is a supervised ML problem in which the variable to be predicted y can take any real value.

- **What is classification?**

  It is a supervised ML problem in which the variable to be predicted y can take only a finite set of possible values or labels.

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

  A predictor is a function that maps d-dimensional vectors to real numbers.

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

  A predictor is a function that maps d-dimensional vectors to real numbers.

- **What is sample error?**

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

  A predictor is a function that maps d-dimensional vectors to real numbers.

- **What is sample error?**

  Sample error for a predictor h is given by

  $$L_S(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i).$$

  It is the average loss incurred by the predictor h on the given training data.

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

  A predictor is a function that maps d-dimensional vectors to real numbers.

- **What is sample error?**

  Sample error for a predictor h is given by

  $$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i).$$

  It is the average loss incurred by the predictor h on the given training data.

- **Write the expression for sample error for regression for the squared error loss.**

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

  A predictor is a function that maps d-dimensional vectors to real numbers.

- **What is sample error?**

  Sample error for a predictor h is given by

$$L_S(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i).$$

  It is the average loss incurred by the predictor h on the given training data.

- **Write the expression for sample error for regression for the squared error loss.**

$$L_S(h) = \frac{1}{n}\sum_{i=1}^{n} (h(x_i) - y_i)^2.$$

- **Write the expression for sample error for classification.**

- **What is a predictor? Assume that $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$.**

  A predictor is a function that maps d-dimensional vectors to real numbers.

- **What is sample error?**

  Sample error for a predictor h is given by

  $$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i).$$

  It is the average loss incurred by the predictor h on the given training data.

- **Write the expression for sample error for regression for the squared error loss.**

  $$L_S(h) = \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2.$$

- **Write the expression for sample error for classification.**

  $$L_S(h) = \frac{\#\{i : h(x_i) \neq y_i\}}{n}.$$

- **What is training?**

- **What is training?**

  Training is the process of minimizing the sample error $L_S(h)$ over the predictors h, i.e., training is solving the optimization problem:

$$\min_h L_S(h) = \min_h \ \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

- **What is training?**

  Training is the process of minimizing the sample error $L_S(h)$ over the predictors h, i.e., training is solving the optimization problem:

$$\min_h L_S(h) = \min_h \ \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

- **What is generalization error?**

- **What is training?**

  Training is the process of minimizing the sample error $L_S(h)$ over the predictors h, i.e., training is solving the optimization problem:

$$\min_h L_S(h) = \min_h \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

- **What is generalization error?**

  If h_s is the output of training or the solution to the optimization problem $\min_h L_S(h)$, then the generalization error of h_s is the performance of h_s over new and unseen data of size m (for m large) $(x_{n+1}, y_{n+1}), (x_{n+2}, y_{n+2}), \ldots, (x_{n+m}, y_{n+m})$:

$$L_G(h_s) = \frac{1}{m} \sum_{j=1}^{m} \ell(h_s(x_{n+j}), y_{n+j}).$$

- **Find the sample error of the predictor $h(x_1, x_2) = \frac{x_1+x_2}{10}$ for the following regression problem using squared error loss.**

y	x_1	x_2
0.1	1	1
0.2	2	2
0.3	3	3
0.4	4	4

- **Find the sample error for the following classification problem for the predictor that always chooses A, no matter what the sample point.**

y	x_1	x_2
A	10	21
A	2	22
B	3	1
B	100	4

When is the Training Process called Successful?

The training process is called successful if the generalization error of the predictor h_s is small.

How to interpret the generalization error? Recall that the generalization of the predictor h_s is

$$L_G(h_s) = \frac{1}{m} \sum_{j=1}^{m} \ell(h_s(x_{n+j}), y_{n+j}).$$

The generalization error is the performance of the learned predictor h_s on dataset that it has not seen. Intuitively, this is the true test of a predictor.

The most important fact about ML is that if we are not careful in the training process, the generalization error can be really poor.

Factors Affecting Successful Training

Successful Training = low generalization error of the learned predictor on potentially infinitely large unseen data.

Factors affecting successful training:

- **Independent and Identically Distributed (i.i.d.) Data**: The $x - y$ relationship that you have learned should be the same in the new data.
- **Sample size**: Too few training data will not work.
- **Overfitting**: We need to avoid building a predictor that 'fits' too closely the training data.

To appreciate these points, let us look at an example.

Predicting Financial Market

There are two companies: company A and company B. Let

$$\begin{aligned} x = & \quad \text{the stock value of company A,} \\ y = & \quad \text{the stock value of company B .} \end{aligned}$$

You are given stock values of last n days

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

and asked to learn to predict the stock value of y from x. You are being told that the learned predictor will be used to predict future stock values of company B using stock values of company A.

Let us understand what it takes to successfully learn a predictor using this example.

- **Role of data size** n: Intuitively, can you learn the relationship between the companies using one or two days of data? You would say, "no, I need possibly months of data, even data from last year!"
- **Constant** $x - y$ **relationship**: Now suppose you have learned a predictor. Let us call it h_s. You sell h_s in the market. But, due to some unforeseen circumstances (e.g., changes in govt. policies), the relationship between stock values of company A and B evolves or changes. Since you don't see the future, your trained a predictor without taking into account this new phenomenon. Your predictor may not work as well you have thought it would. **So, for a predictor to work in the future, the** $x - y$ **relationship you have learned should stay the same**.

  The mathematical way to say that the $x - y$ **relationship should stay the same is to say that the data should be i.i.d.. We will come back to these concepts when we discuss statistical inference and ML theory later in the course.**

- **Avoid Overfitting**: What is wrong with this predictor?

$$h_s(x) = \begin{cases} y_i, & \text{if } x = x_i \\ 0, & \text{otherwise.} \end{cases}$$

  - In the future, if we see the stock value of company A at $x = x_i$ (which itself is very unlikely because stock values evolve), this predictor will predict the value company B had the last time we saw $x = x_i$.
  - This will predict the stock value of company B to be 0 no matter what the value of stock of company A is ($x \neq x_i$ for any i.)

You might say, "obviously, this is a terrible predictor and why would I choose this one?"

Recall that training is solving the following optimization problem:

$$\min_h L_S(h) = \min_h \; \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i).$$

And the predictor given above has the minimum possible error of ZERO (because it makes perfect prediction on the training data).

This phenomenon is called overfitting and we need to avoid this.

Tips for Successful Training

So, what are the tips for successful training?

- **Need I.I.D. data or same $x - y$ relationship**.
- **Need large sample size or training data size**.
- **Need to avoid the phenomenon of overfitting**.

Note that our arguments here are based purely on intuitions. But, these ideas can be established using rigourous mathematical statements. We will come back to these in Part II of the course.

Tips for Successful Training: How do we make sure these conditions are satisfied?

- **Large sample size**: Intuitively, we can just ensure this condition is satisfied by just collecting or using a large amount of data. But, how much is considered large? In general, the answer depends on the complexity of the problem and you never know how many samples you need. The theoretical number needed is infinite amount of data. There are some nonasymptotic analysis available. But, they are only useful for developing insights and not meant as actual practical tools.
- **Same $x - y$ relationship**: Intuitively, we can just look at the test data and make a guess on whether or not same $x - y$ relationship will hold. However, in general, the correct way to check same $x - y$ relationship is through statistical tests. This is never done in practice!
- **Avoid overfitting**: This question cannot be answered based on intuitions. This brings us to the notion of hypothesis class.

Hypothesis Class

Modified Approach to Training: Search for predictors over a class of functions $\mathcal{H}$ (called a Hypothesis class).

$$\min_{h \in \mathcal{H}} L_S(h) = \min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)$$

Examples:

- $\mathcal{H} = \{$Class of linear predictors$\}$.
- $\mathcal{H} = \{$Class of polynomial predictors up to a given order$\}$.
- $\mathcal{H} = \{$Class of neural networks of a certain type$\}$.

- It is called a Hypothesis class because it represents your hypothesis (based on prior knowledge) that the best predictor either belongs to this class or can be well-approximated by it. **In practice, you never know which $\mathcal{H}$ to choose because you may not have prior knowledge**.
- If $\mathcal{H}$ is too large (e.g., all possible functions), then it will lead to overfitting and learning will fail.
- If $\mathcal{H}$ is too small, then the best predictor will not lie in it and the generalization error might be quite large.
- So, $\mathcal{H}$ should be moderately sized. **Since we never know the truth, we never know if a given hypothesis class is of moderate size!!**

Recipe for Successful ML

The recipe for successful machine learning (acceptable' generalization error) is:

1. Large sample size.
2. Independent and Identically Distributed (i.i.d.) Data.
3. Avoid overfitting by solving the following over moderately sized hypothesis class $\mathcal{H}$:

$$\min_{h\in\mathcal{H}} L_S(h) = \min_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i)$$

In practice:

- People may have prior knowledge based on published results or past experience with similar data.
- People use hypothesis class $\mathcal{H}$ such that $\mathcal{H}$ is "simple or less complex" and for which the training $\min_{h\in\mathcal{H}} L_S(h)$ can be done efficiently. The motivation for doing this is being practical Occam's Razor

Occam's Razor

Theorem (Occam's Razor)

All things being equal, the simplest explanation tends to be the right one.

Movie CONTACT:
`https://en.wikipedia.org/wiki/Contact_(1997_American_film)`.

What is relationship to ML? It turns out that while searching for the best predictor, the search should be restricted to simple predictors.

Restricting the search to simpler predictors will affect your ability to predict perfectly. But, you will be able to quantify your performance using ML theory.

Linear Predictors: Notations

Before we begin our study of linear predictors, let us set some notation that we will try to use the rest of the course.

- We collect the x values of the n training samples in a matrix

$$\mathbb{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}_{n \times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}_{n \times p} .$$

Thus, we assume that each x value is a p-dimensional vector:

$$\begin{aligned} x_1 &= (x_{11}, x_{12}, \ldots, x_{1p}) \\ x_2 &= (x_{21}, x_{21}, \ldots, x_{2p}) \\ &\vdots \\ x_n &= (x_{n1}, x_{12}, \ldots, x_{np}). \end{aligned}$$

Notations: Continued

- We collect the y values of the n training samples in a vector

$$\mathbb{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Thus, we assume that each y value is a real number.

- Given n training data points for supervised ML

$$(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n),$$

where $x \in \mathbb{R}^p$ and $y \in \mathbb{R}$, we collect them in the matrix $\mathbb{X}$ and the vector $\mathbb{Y}$.

The **diabetes data** contains data from 442 patients. It has 10 variables. We need to use those 10 variables (I have shown only 6 of them below) to predict a quantitative measure of disease (diabetes) progression one year after baseline.

```
Y                AGE         SEX        BMI          BP          S1          S2
151.0       [ 0.03807591  0.05068012  0.06169621  0.02187235 -0.0442235  -0.03482076]
75.0       [-0.00188202 -0.04464164 -0.05147406 -0.02632783 -0.00844872 -0.01916334]
141.0       [ 0.08529891  0.05068012  0.04445121 -0.00567061 -0.04559945 -0.03419447]
206.0       [-0.08906294 -0.04464164 -0.01159501 -0.03665645  0.01219057  0.02499059]
135.0       [ 0.00538306 -0.04464164 -0.03638469  0.02187235  0.00393485  0.01559614]
97.0       [-0.09269548 -0.04464164 -0.04069594 -0.01944209 -0.06899065 -0.07928784]
138.0       [-0.04547248  0.05068012 -0.04716281 -0.01599922 -0.04009564 -0.02480001]
63.0       [ 0.06350368  0.05068012 -0.00189471  0.06662967  0.09061988  0.10891438]
110.0       [ 0.04170844  0.05068012  0.06169621 -0.04009932 -0.01395254  0.00620169]
310.0       [-0.07090025 -0.04464164  0.03906215 -0.03321358 -0.01257658 -0.03450761]
```

For this example, $n = 442$ and $p = 10$. As a result, the matrix $\mathbb{X}$ will be a 442×10 matrix and the vector $\mathbb{Y}$ will be a 442×1 vector or matrix.

Linear Predictors

Definition

A linear predictor is a linear function of its variables: Let u and β be p-dimensional vectors $u = (u_1, u_2, \ldots, u_p)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$. Then a linear predictor as a function of u is defined as

$$\begin{aligned} h(u_1, u_2, \ldots, u_p) &= \beta_1 u_1 + \beta_2 u_2 + \cdots + \beta_p u_p \\ h(u) &= \beta^T u. \end{aligned}$$

Examples:

- Given a data point $x_1 = (x_{11}, x_{12}, \ldots, x_{1p})$, the linear predictor maps x_1 to

$$h(x_1) = h(x_{11}, x_{12}, \ldots, x_{1p}) = \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_p x_{1p} = \beta^T x_1.$$

- Given a data point $x_2 = (x_{21}, x_{22}, \ldots, x_{2p})$, the linear predictor maps x_2 to

$$h(x_2) = h(x_{21}, x_{22}, \ldots, x_{2p}) = \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_p x_{2p} = \beta^T x_2.$$

- Training using linear predictors is trying to predict the vector $\mathbb{Y}$ using the matrix $\mathbb{X}$:

$$\mathbb{Y} \longleftarrow \mathbb{X}\beta.$$

- In detail, this equation is equal to

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \longleftarrow \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}.$$

- Element-wise, this looks like

$$\begin{aligned} y_1 &\longleftarrow \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_p x_{1p} \\ &\vdots \\ y_n &\longleftarrow \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_p x_{np} \end{aligned}$$

- **Remember that you are using the same β vector on every training data. This is because you fix a predictor and see its performance on the entire training data. If you are not happy with the performance you change your predictor. In case of linear predictors, you change the vector β.**

Hypothesis Class of Linear Predictors

- Choosing a linear predictor amounts to choosing the coefficients of the linear function $\beta = (\beta_1, \ldots, \beta_p)$.

- Note that applying a predictor means applying the **same** function h to all the training data. So, when we move from one training point to another, the x vector changes, but the βs remain the same.

- The class of linear predictors is the simplest class of predictors you can choose. Motivated by this, let us define

$$\mathcal{H} = \{\text{Class of linear predictors}\}.$$

- Training process then becomes (for regression and squared error loss)

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 = \min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2.$$

Linear Methods in Regression (Overview)

- **Data**: We are given n training point $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$. Now, $x \in \mathbb{R}^p$ and $y \in \mathbb{R}$.

- **Linear regression or ordinary least square**: The training process for the search for the best linear predictor is given by the optimization problem

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2.$$

  Here, $\beta = (\beta_1, \dots, \beta_p)$.

- **Ridge regression**:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 + \lambda \sum_{i=1}^{p} \beta_i^2.$$

- **LASSO (least absolute shrinkage and selection operator)**:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 + \lambda \sum_{i=1}^{p} |\beta_i|.$$

- **Elastic Net**

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 + \lambda_1 \sum_{i=1}^{p} \beta_i^2 + \lambda_2 \sum_{i=1}^{p} |\beta_i|.$$

Linear Regression: Special Case of $p = 1$

- **Data**: We are given n training points

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

  Here x and y variables are real numbers. We have n training points n. We are again asked to learn a linear predictor using the data.
- **Training**: The training for a linear predictor looks the same:

  $$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 = \min_{\beta_1} \frac{1}{n} \sum_{i=1}^{n} (\beta_1 x_i - y_i)^2$$

  Here, $\beta = (\beta_1, \ldots, \beta_p) = \beta_1$. For $p = 1$, this is fitting a line to a set of points.
- **Solving the optimization problem**: How do you solve the optimization problem to learn the best β_1?
  - This is a *convex* function with a unique minimum (more on this in the coming lectures).
  - It is also differentiable.
  - The optimal point can be obtained by taking the derivative with respect to β_1 and setting the derivative to zero.

- **Convex Functions**: A convex function looks like as shown in the figure below (Taken from Boyd's book).

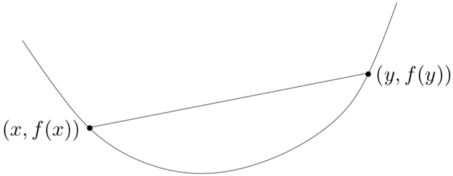


Figure 3.1 Graph of a convex function. The chord (*i.e.*, line segment) between any two points on the graph lies above the graph.

- **Optimization of a convex function**: Suppose $f(\beta)$ is a convex function and we want to solve

$$\min_{\beta} f(\beta).$$

If a minimum exists, it is unique and can be obtained by setting the derivative of $f(\beta)$ with respect to β to zero.

- **Solving for β_1**: We solve the optimization problem

$$\min_{\beta_1} \frac{1}{n} \sum_{i=1}^{n} (\beta_1 x_i - y_i)^2$$

by again taking the derivative with respect to β_1 and setting it to zero.

- **The derivative**: The derivative is

$$\frac{d}{d\beta_1} \frac{1}{n} \sum_{i=1}^{n} (\beta_1 x_i - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} 2(\beta_1 x_i - y_i)(x_i).$$

Simplification gives

$$2\left(-\frac{1}{n} \sum_{i=1}^{n} x_i y_i + \beta_1 \, \frac{1}{n} \sum_{i=1}^{n} x_i^2 \right).$$

- **The solution**: Setting this to zero gives us the solution to the ordinary lease square or training problem for general n:

$$\beta_{ols} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}.$$

- **The learned predictor**: Note that your learned predictor is

$$h_s(u) = \beta_{ols} u = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2} u.$$

  So, it is obviously an explicit function of your training data.

- **Application to New Data**: How do you use the learned predictor on new data? Given a new data point x^* (remember, no corresponding y^* will be given), your predicted value at this new x^* will be

$$\hat{y^*} = h_s(x^*) = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2} x^*.$$

- **Generalization Error**: If I now give you a new (possibly infinite) set of $x - y$ labels to test your predictor (in this case you will be given the y values) $(x_{n+1}, y_{n+1}), (x_{n+2}, y_{n+2}), \ldots, (x_{n+m}, y_{n+m})$, the generalization error would be

$$\begin{aligned} \frac{1}{m} \sum_{j=1}^{m} \ell(h_s(x_{n+j}), y_{n+j}) &= \frac{1}{m} \sum_{j=1}^{m} \left(y_{n+j} - \beta_{ols} x_{n+j}\right)^2 . \\ &= \frac{1}{m} \sum_{j=1}^{m} \left(y_{n+j} - \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2} x_{n+j}\right)^2 . \end{aligned}$$

Linear Regression: Special Case with $p = 2$

- **Data**: We are given n training points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

Now, each x_i is a vector of dimension $p = 2$:

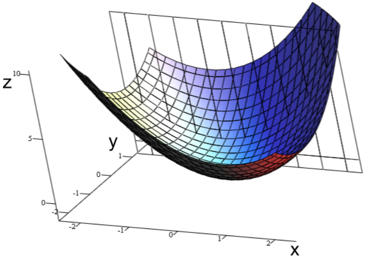
$$\begin{aligned} x_1 &= (x_{11}, x_{12}) \\ x_2 &= (x_{21}, x_{22}) \\ &\vdots \\ x_n &= (x_{n1}, x_{n2}). \end{aligned}$$

- **Training**: The training process for a linear predictor is:

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 = \min_{\beta_1, \beta_2} \frac{1}{n} \sum_{i=1}^{n} (\beta_1 x_{i1} + \beta_2 x_{i2} - y_i)^2$$

Here, $\beta = (\beta_1, \ldots, \beta_p) = (\beta_1, \beta_2)$. **Depending on the problem, this can be fitting a line (with intercept) or fitting a plane**.

- **Convex Function in Two Variables**: A convex function in two variables $f(\beta_1, \beta_2)$ looks like as shown in the figure below (taken from Wikipedia):



- **Optimization of a convex function**: We want to solve

$$\min_{\beta_1, \beta_2} f(\beta_1, \beta_2).$$

If a minimum exists, it is unique and can be obtained by setting the gradient

$$\nabla_\beta f(\beta_1, \beta_2) := \begin{bmatrix} \frac{\partial}{\partial \beta_1} f(\beta_1, \beta_2) \\ \frac{\partial}{\partial \beta_2} f(\beta_1, \beta_2) \end{bmatrix}$$

to zero.

- **The gradient**: The gradient is computed as follows.

$$\frac{\partial}{\partial \beta_1}\frac{1}{n}\sum_{i=1}^{n}(\beta_1 x_{i1} + \beta_2 x_{i2} - y_i)^2 \quad = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} 2(\beta_1 x_{i1} + \beta_2 x_{i2} - y_i)(x_{i1})$$

$$\frac{\partial}{\partial \beta_2}\frac{1}{n}\sum_{i=1}^{n}(\beta_1 x_{i1} + \beta_2 x_{i2} - y_i)^2 \quad = \tfrac{1}{n}\textstyle\sum_{i=1}^{n} 2(\beta_1 x_{i1} + \beta_2 x_{i2} - y_i)(x_{i2}).$$

Simplification gives

$$\nabla_\beta f(\beta_1, \beta_2) = \begin{bmatrix} \frac{\partial}{\partial \beta_1} f(\beta_1, \beta_2) \\ \frac{\partial}{\partial \beta_2} f(\beta_1, \beta_2) \end{bmatrix}$$

$$= \begin{bmatrix} 2\left(-\frac{1}{n}\sum_{i=1}^{n} x_{i1} y_i \quad + \quad \beta_1\ \frac{1}{n}\sum_{i=1}^{n} x_{i1}^2 \quad + \quad \beta_2\ \frac{1}{n}\sum_{i=1}^{n} x_{i1} x_{i2}\right) \\ 2\left(-\frac{1}{n}\sum_{i=1}^{n} x_{i2} y_i \quad + \quad \beta_1\ \frac{1}{n}\sum_{i=1}^{n} x_{i1} x_{i2} \quad + \quad \beta_2 \frac{1}{n}\sum_{i=1}^{n} x_{i2}^2\right) \end{bmatrix}$$

- **Setting the gradient to zero**: In order to find the optimal point, we need to set the gradient $\nabla_\beta f(\beta_1, \beta_2)$ to zero. This gives us

$$\begin{bmatrix} 2\left(-\frac{1}{n}\sum_{i=1}^n x_{i1}y_i + \quad \beta_1 \ \frac{1}{n}\sum_{i=1}^n x_{i1}^2 + \quad \beta_2 \ \frac{1}{n}\sum_{i=1}^n x_{i1}x_{i2}\right) \\ 2\left(-\frac{1}{n}\sum_{i=1}^n x_{i2}y_i + \quad \beta_1 \ \frac{1}{n}\sum_{i=1}^n x_{i1}x_{i2} + \quad \beta_2 \frac{1}{n}\sum_{i=1}^n x_{i2}^2\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The above equation is equivalent to (remove scaling by 2 and $\frac{1}{n}$)

$$\begin{bmatrix} \beta_1 \ \sum_{i=1}^n x_{i1}^2 + \quad \beta_2 \ \sum_{i=1}^n x_{i1}x_{i2} \\ \beta_1 \ \sum_{i=1}^n x_{i1}x_{i2} + \quad \beta_2 \sum_{i=1}^n x_{i2}^2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_{i1}y_i \\ \sum_{i=1}^n x_{i2}y_i \end{bmatrix}.$$

This in turn can be written in the matrix format:

$$\begin{bmatrix} \sum_{i=1}^n x_{i1}^2 & \sum_{i=1}^n x_{i1}x_{i2} \\ \sum_{i=1}^n x_{i1}x_{i2} & \sum_{i=1}^n x_{i2}^2 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_{i1}y_i \\ \sum_{i=1}^n x_{i2}y_i \end{bmatrix}.$$

- **The solution**: The OLS solution for $p = 2$ is given by (assuming invertibility of the matrix on the left above)

$$\beta_{ols} = \begin{bmatrix} \beta_{1,ols} \\ \beta_{2,ols} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_{i1}^2 & \sum_{i=1}^n x_{i1}x_{i2} \\ \sum_{i=1}^n x_{i1}x_{i2} & \sum_{i=1}^n x_{i2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^n x_{i1}y_i \\ \sum_{i=1}^n x_{i2}y_i \end{bmatrix}.$$

- **The learned predictor**: Note that your learned predictor is (with $u = (u_1, u_2)$)

$$
\begin{aligned}
h_s(u) &= \beta_{ols}^T u = \beta_{1,ols} u_1 + \beta_{2,ols} u_2 \\
&= \left(\begin{bmatrix} \sum_{i=1}^n x_{i1}^2 & \sum_{i=1}^n x_{i1}x_{i2} \\ \sum_{i=1}^n x_{i1}x_{i2} & \sum_{i=1}^n x_{i2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^n x_{i1}y_i \\ \sum_{i=1}^n x_{i2}y_i \end{bmatrix} \right)^T \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}
\end{aligned}
$$

- **Application to New Data**: Prediction on new $x^* = (x_1^*, x_2^*)$:

$$
\begin{aligned}
\hat{y^*} &= h_s(x^*) = \beta_{ols}^T x^* = \beta_{1,ols} x_1^* + \beta_{2,ols} x_2^* \\
&= \left(\begin{bmatrix} \sum_{i=1}^n x_{i1}^2 & \sum_{i=1}^n x_{i1}x_{i2} \\ \sum_{i=1}^n x_{i1}x_{i2} & \sum_{i=1}^n x_{i2}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^n x_{i1}y_i \\ \sum_{i=1}^n x_{i2}y_i \end{bmatrix} \right)^T \begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix}.
\end{aligned}
$$

- **Generalization Error**: the generalization error for new data $(x_{n+1}, y_{n+1}), (x_{n+2}, y_{n+2}), \ldots, (x_{n+m}, y_{n+m})$ would be

$$
\frac{1}{m} \sum_{j=1}^{m} \left(y_{n+j} - \beta_{ols}^T x_{n+j} \right)^2 .
$$

Affine Functions

- **Affine Function of one variable**: An affine function of one variable is defined as

$$h(u) = b + au$$

  for real numbers a and b. This is also a line with an intercept b. It passes through the origin only if $b = 0$.

- **Affine functions of two variables**: An affine function of two variables is defined as

$$h(u_1, u_2) = a_1 u_1 + a_2 u_2 + b$$

  for some constants a_1, a_2, b. It is a plane with an intercept.

- **Affine function of many variables**: More generally, we define anaffine function of many variables as

$$h(u_1, u_2, \ldots, u_p) = a_1 u_1 + a_2 u_2 + \cdots + a_p u_p + b.$$

Training With Affine Predictors

- **Training an affine precdictor**: This looks like

$$\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} (h(x_i) - y_i)^2 = \min_{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i + \beta_0 - y_i)^2.$$

  Here,

$$\beta = (\beta_1, \ldots, \beta_p).$$

- **Not a new problem**: This is exactly like a OLS problem but in dimension $p+1$: for each x data point x_i, add a 1 to the first component

$$(1, x_{i1}, \cdots, x_{ip}).$$

  Then, train a linear predictor in dimension $p+1$, i.e., search for the optimal $\beta \in \mathbb{R}^{p+1}$ where now the β is modified to include β_0:

$$\beta = (\beta_0, \beta_1, \cdots, \beta_p).$$

Linear Regression for any p, n.

- **Data**: We are given n training point

  $$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

  Now, $x \in \mathbb{R}^p$ and $y \in \mathbb{R}$.

- **Training**: The training process for the search for the best linear predictor is given by the optimization problem

  $$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2.$$

  Here, $\beta = (\beta_1, \ldots, \beta_p)$.

  We now want to understand how to solve this more general optimization problem over the p-dimensional space $\mathbb{R}^p$. The approach is again to take the gradient with respect to β and set the p-dimensional gradient vector to zero. In the next few slides, we look at a cleaner way to do this using matrix algebra.

- Recall that

$$\mathbb{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \qquad \mathbb{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}_{n \times p}$$

where each x_i is collected in the rows of the matrix $\mathbb{X}$:

$$\begin{aligned} x_1 &= (x_{11}, x_{12}, \ldots, x_{1p}) \\ &\vdots \\ x_n &= (x_{n1}, x_{12}, \ldots, x_{np}). \end{aligned}$$

- Now note that the vector obtained by the product $\mathbb{X}\beta$ can be written as

$$\mathbb{X}\beta = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} = \begin{bmatrix} x_1^T \beta \\ \vdots \\ x_n^T \beta \end{bmatrix}$$

- The difference of two vectors $\mathbb{Y} - \mathbb{X}\beta$ can be written as

$$\mathbb{Y} - \mathbb{X}\beta = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1^T\beta \\ \vdots \\ x_n^T\beta \end{bmatrix} = \begin{bmatrix} y_1 - x_1^T\beta \\ \vdots \\ y_n - x_n^T\beta \end{bmatrix}$$

- **Important observation**: Now note that the training or sample error is nothing but a scaled version of the dot product of the vector $\mathbb{Y} - \mathbb{X}\beta$ with itself:

$$\frac{1}{n}\sum_{i=1}^{n}(\beta^T x_i - y_i)^2 = \frac{1}{n}\langle \mathbb{Y} - \mathbb{X}\beta, \mathbb{Y} - \mathbb{X}\beta \rangle = \frac{1}{n}\left(\mathbb{Y} - \mathbb{X}\beta\right)^T\left(\mathbb{Y} - \mathbb{X}\beta\right).$$

To understand the above equation, recall that the inner product or the dot product between two n-dimensional vectors:

$$\langle a, b \rangle = a^T b = \sum_{i=1}^{n} a_i b_i \qquad \langle a, a \rangle = a^T a = \sum_{i=1}^{n} a_i^2.$$

- **The optimization problem**: The OLS optimization problem is then given by

$$\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} (\beta^T x_i - y_i)^2 = \min_{\beta \in \mathbb{R}^p} \frac{1}{n} \left(\mathbb{Y} - \mathbb{X}\beta\right)^T \left(\mathbb{Y} - \mathbb{X}\beta\right).$$

- In order to solve the problem, we expand the expression for the objective function and obtain

$$\frac{1}{n} \left(\mathbb{Y} - \mathbb{X}\beta\right)^T \left(\mathbb{Y} - \mathbb{X}\beta\right) = \frac{1}{n} \left(\mathbb{Y}^T\mathbb{Y} + \beta^T\mathbb{X}^T\mathbb{X}\beta - 2\beta^T\mathbb{X}^T\mathbb{Y}\right).$$

  As discussed earlier, we can show that this is a convex function and the optimal solution can be obtained by setting the gradient with respect to β to zero. **But, how do we find gradient of this expression?**

Gradient Calculations for Quadratic Forms

- The gradient ∇_β of a real valued function $f(\beta)$ of β is the column vector of partials:

$$\nabla_\beta f(\beta) = \begin{bmatrix} \frac{\partial f(\beta)}{\partial \beta_1} \\ \frac{\partial f(\beta)}{\partial \beta_2} \\ \vdots \\ \frac{\partial f(\beta)}{\partial \beta_p} \end{bmatrix}$$

- For any square matrix A,

$$\nabla_\beta \; (\beta^T A \, \beta) = (A + A^T)\beta.$$

- To verify this, let us try a 2×2 example:

$$\begin{aligned} \beta^T A \, \beta &= \begin{bmatrix} \beta_1 & \beta_2 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \beta_1 & \beta_2 \end{bmatrix} \begin{bmatrix} a\beta_1 + b\beta_2 \\ c\beta_1 + d\beta_2 \end{bmatrix} \\ &= \beta_1(a\beta_1 + b\beta_2) + \beta_2(c\beta_1 + d\beta_2). \end{aligned}$$

- Thus, we have

$$\beta^T A\,\beta = \beta_1(a\beta_1 + b\beta_2) + \beta_2(c\beta_1 + d\beta_2).$$

- The gradient is given by

$$\begin{aligned}
\nabla_\beta\,(\beta^T A\,\beta) &= \begin{bmatrix} \frac{\partial(\beta^T A\,\beta)}{\partial\beta_1} \\ \frac{\partial(\beta^T A\,\beta)}{\partial\beta_2} \end{bmatrix} = \begin{bmatrix} 2a\beta_1 + b\beta_2 + c\beta_2 \\ b\beta_1 + c\beta_1 + 2d\beta_2 \end{bmatrix} \\
&= \begin{bmatrix} 2a & b+c \\ b+c & 2d \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \\
&= \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \\
&= A\beta + A^T\beta.
\end{aligned}$$

- What happens when A is symmetric, i.e, $A = A^T$? We get

$$\nabla_\beta\,(\beta^T A\,\beta) = (A + A^T)\beta = 2A\beta.$$

- What happens when $A = I$, the identity matrix? We get

$$\nabla_\beta\,(\beta^T A\,\beta) = 2A\beta = 2\beta.$$

- For any column vector $a = [a_1, \cdots, a_p]^T$,

$$a^T\beta = \beta^T a = \sum_{i=1}^{p} a_i\beta_i$$

- What is the gradient of $a^T\beta$? To obtain this, just take partial derivatives of $a^T\beta = \sum_{i=1}^{p} a_i\beta_i$ with respect to components of β and arrange them in a column vector:

$$\nabla_\beta\left(a^T\beta\right) = \begin{bmatrix} \frac{\partial(a^T\beta)}{\partial\beta_1} \\ \frac{\partial(a^T\beta)}{\partial\beta_2} \\ \vdots \\ \frac{\partial(a^T\beta)}{\partial\beta_p} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix}.$$

- Thus,

$$\nabla_\beta\left(a^T\beta\right) = a.$$

- So, do we know how to compute the following gradient?

$$\nabla_\beta \left(\mathbb{Y}^T \mathbb{Y} + \beta^T \mathbb{X}^T \mathbb{X} \beta - 2\beta^T \mathbb{X}^T \mathbb{Y} \right)$$

  Yes, because gradient (partial derivative or total derivatives) of a sum of functions is equal to the sum of partial derivatives of individual functions.

- This gives us

$$\begin{aligned} &\nabla_\beta \left(\mathbb{Y}^T \mathbb{Y} + \beta^T \mathbb{X}^T \mathbb{X} \beta - 2\beta^T \mathbb{X}^T \mathbb{Y} \right) \\ &= \nabla_\beta \left(\mathbb{Y}^T \mathbb{Y} \right) + \nabla_\beta \left(\beta^T \mathbb{X}^T \mathbb{X} \beta \right) - 2\nabla_\beta \left(\beta^T \mathbb{X}^T \mathbb{Y} \right) \\ &= 2\mathbb{X}^T \mathbb{X} \beta - 2\mathbb{X}^T \mathbb{Y}. \end{aligned}$$

- Setting $\nabla_\beta = 0$ will give us the solution. Setting the gradient equal to the all zero vector gives us

$$\beta_{ols} = \left(\mathbb{X}^T \mathbb{X} \right)^{-1} \mathbb{X}^T \mathbb{Y}.$$

Understanding the Linear Regression Solution

We should ask the following questions:

1. What does the matrix $\mathbb{X}^T\mathbb{X}$ looks like?
2. When is it invertible?
3. What if it is not invertible?
4. We used convex optimization techniques without verifying if the results are really applicable. So, what is convex optimization and what is the mathematics behind it?
5. We found the solution of the linear regression or ordinary least square problem using calculus or convex optimization techniques. But, do we really understand the solution? How to interpret the results?
6. Are there other linear predictors apart from the one obtained using the linear regression or ordinary least square techniques?

Understanding the Linear Regression Solution

We should ask the following questions (**now with answers**):

1. What does the matrix $\mathbb{X}^T\mathbb{X}$ looks like? **Look at the $p = 2$ case.**
2. When is it invertible? **When columns of $\mathbb{X}$ are linearly independent. For example, use Ridge Regression.**
3. What if it is not invertible? **Use other linear predictors or ML techniques.**
4. We used convex optimization techniques without verifying if the results are really applicable. So, what is convex optimization and what is the mathematics behind it? **We have seen the figures and intuition. We just have to express them in math.**
5. We found the solution of linear regression using convex optimization techniques. But, do we really understand the solution? How to interpret the results? **Keyword is projection: OLS solution is the projection of $\mathbb{Y}$ on the column space of $\mathbb{X}$.**
6. Are there other linear predictors apart from the one obtained using the linear regression or ordinary least square techniques? **Of course, we will discuss RIDGE, LASSO, and ELASTIC NET**