Contents

1 k Nearest Neighbors (kNN)

1.1 kNN for Classification Problem

Definition 1.1

In the **classification** task, the target space is finite:

$$\mathbb{Y} = \{1,\ 2,\ \ldots,\ C\} \quad \text{or} \quad \mathbb{Y} = \{0,\ 1,\ \ldots,\ C-1\}, \quad \text{or} \quad \mathbb{Y} = \{-1,\ 1\}. \tag{1.1}$$

In the **regression** task, the target can take any real number: $\mathbb{Y} = \mathbb{R}$.

Remark 1.1

There is a **multi-label classification,** when multiple labels can be assigned to one object $\mathbb{Y} = \{0, 1\}^C$, and **multiple regression,** when $\mathbb{Y} = \mathbb{R}^n$.

In Fig. 1.1, the data with two-dimensional feature space and a binary target is presented. There are three points: red, blue, and black. How would you classify them?

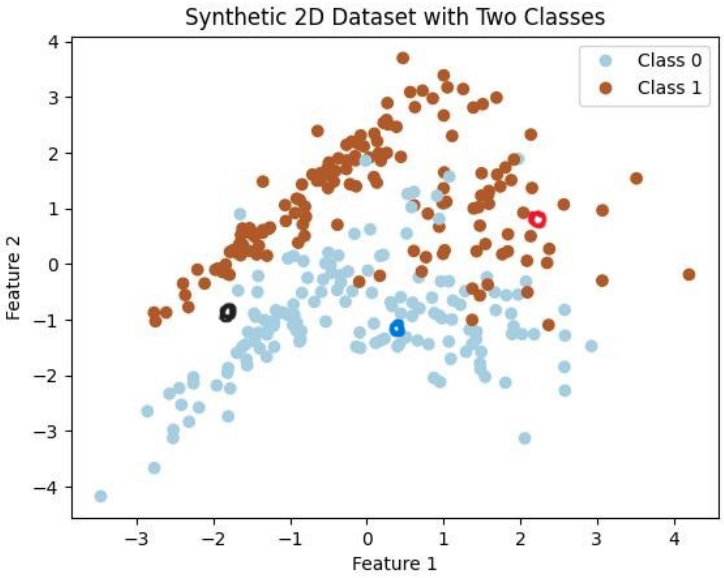


Figure 1.1: Data of two classes and three points to classify.

Most likely, you would classify the red point as Class 1, the blue point as Class 0, but you would have some difficulty to classify the black point.

This intuitive idea can be formalized in terms of neighboring point.

Definition 1.2

By a **neighbourhood** of a point x we will mean a ball of some radius r centered at the point x :

$$B_r(a) = \{x \in \mathbb{X} : d(a,\ x) < r\}. \tag{1.2}$$

The points inside the neighbourhood are called **neighbours.**

In the figure[1] below, the solid ball contains three neighbours and the dashed ball contains five neighbours.

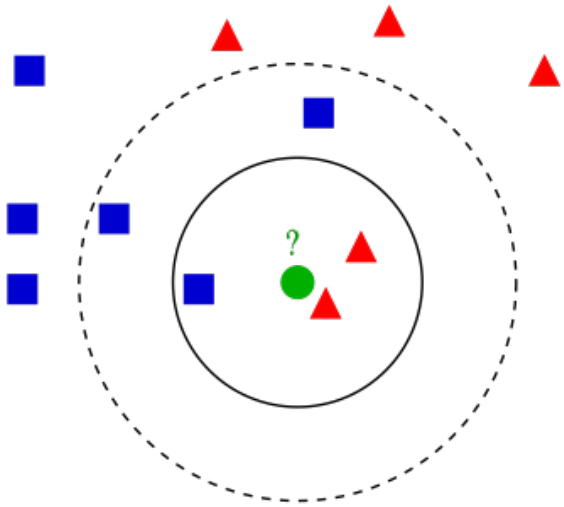


Figure 1.2

If we use three closest neighbours for the classification, then we would classify the green point as a triangle, because we have more triangles in the neighbourhood.

This example brings us to the k nearest neighbors method.

1.1.1 k Nearest Neighbors (kNN): training

- We are given $\left(x^{(i)},\ y^{(i)}\right)$, $i = 1,\ 2, \ldots,\ N$.
- $\mathbb{Y} = \{1,\ 2,\ \ldots,\ C\}$ (classification problem).
- training = memorizing of the given data.

[1] https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

1.1.2 k Nearest Neighbors (kNN): prediction

- We have a new feature x.
- Determine the distances between the new feature and $x^{(i)}$: $d\left(x,\ x^{(i)}\right)$.
- Rearrange the objects by the closeness to x :

$$d\left(x,\ x^{(i_1)}\right) \leq d\left(x,\ x^{(i_2)}\right) \leq \ldots \leq d\left(x,\ x^{(i_N)}\right). \tag{1.3}$$

- Look at the first k labels and assign the class with the highest number of representatives:

$$a(x) = \underset{c \in \mathbb{Y}}{\operatorname{argmax}} \sum_{s=1}^{k} \left[y^{(i_s)} = c\right]. \tag{1.4}$$

Let's illustrate this algorithm with $k = 3$ on the example in Fig. 1.2. Assume, the points were indexed somehow.

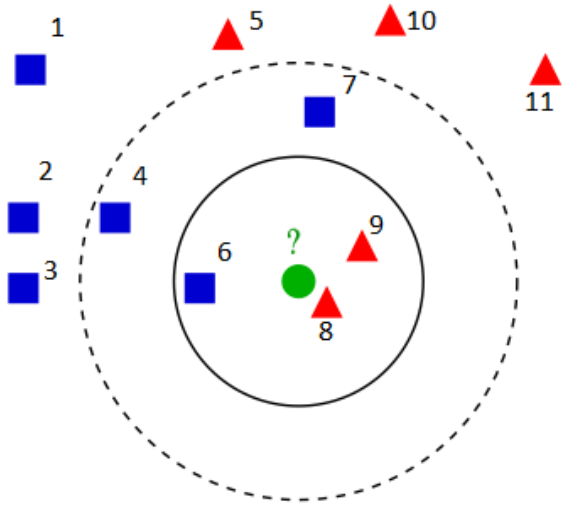


Figure 1.3

Reindex them in the increasing order of ℓ_2 distances from the green point:

$$i_1 = 8,\ i_2 = 9,\ i_3 = 6,\ i_4 = 7, \ldots$$

In this case, $\mathbb{Y} = \{\blacksquare,\ \blacktriangle\}$. Calculate two sums:

$$\sum_{s=1}^{3} \left[y^{(i_s)} = \blacksquare\right] = \left[y^{(8)} = \blacksquare\right] + \left[y^{(9)} = \blacksquare\right] + \left[y^{(6)} = \blacksquare\right] = 1 \quad \text{and} \quad \sum_{s=1}^{3} \left[y^{(i_s)} = \blacktriangle\right] = 2$$

Finally,

$$\max_{c \in \{\blacksquare,\ \blacktriangle\}} \sum_{s=1}^{3} \left[y^{(i_s)} = c\right] = 2 \quad \text{and} \quad a(\bullet) = \underset{c \in \{\blacksquare,\ \blacktriangle\}}{\operatorname{argmax}} \sum_{s=1}^{3} \left[y^{(i_s)} = c\right] = \blacktriangle.$$

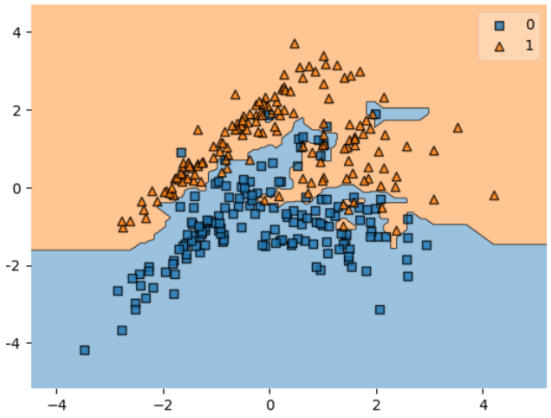


Figure 1.4: $k = 1,\ p = 1$

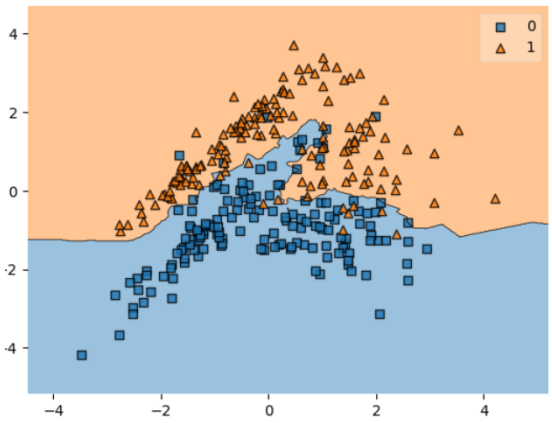


Figure 1.5: $k = 5,\ p = 2$

1.2 Model Comparison or How to Choose k?

Let's apply kNN model to the data in Fig. 1.1 with ℓ_1 distance and $k = 1$ (see Fig. 1.4) and with ℓ_2 distance and $k = 5$ (see Fig.1.5). The results of classification depend on the choice of the number of neighbors and the metric being used. These parameters are referred to as *hyper parameters.*

Definition 1.3

The parameters of the model that cannot be tuned during training are called **hyper parameters.**

We cannot tune k or the choice of distance during the training, these parameters correspond to the current model. But we can use different (hyper) parameters and train the model again, this will be a different model. Comparing somehow those two models, we can choose one that performs better, and this is how we can choose the hyper parameters.

The main method to evaluate the model's performance is to look at the quality metric on the new data. In practice, we don't want to collect new data, and we split the existing data into two folds: one we use for training and the other one is used for the final report.

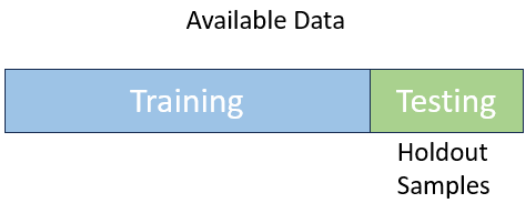


Figure 1.6: Train/Test Split

If we want to tune hyper parameters, we can use one of the following approaches[2]:

- Evaluate model on the validation set
- Use cross-validation

To tune the hyper parameters and choose one model, we can

1. Train different models with different hyper parameters.
2. Test them on Validation (or Developing) set and choose the best model.
3. Using the best model, report the final metric on the Testing set.

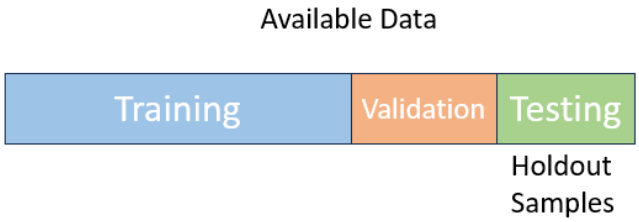


Figure 1.7: Train/Test Split

1.2.1 K-Fold Cross-Validation

The other approach is to split training set into K equal parts and use each part as the validation set.

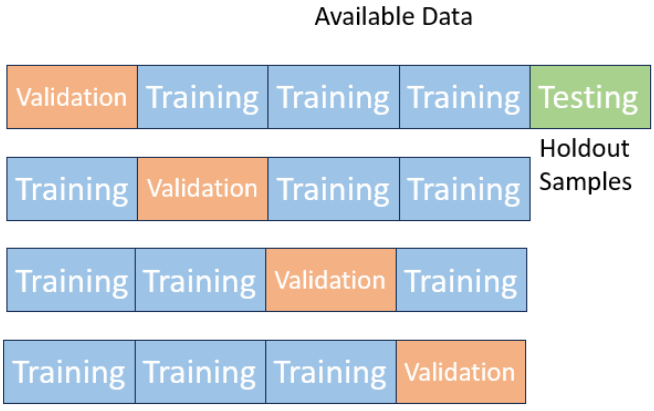


Figure 1.8: Train/Test Split

[2]There are other ways to compare the models, e.g., maximizing likelihood in probabilistic models. We will talk about them later.

1. For each model
   - Train the model on the Training data from split 1 and calculate the quality metric Q_1 on the first Validation set.
   - Train the model on the Training data from split 2 and calculate the quality metric Q_2 on the second Validation set.
   - ...
   - Train the model on the Training data from split k and calculate the quality metric Q_K on the K-th Validation set.
   - report the average quality metric over all splits.

$$Q = \frac{1}{K} \sum_{i=1}^{K} Q_i. \tag{1.5}$$

2. Choose the best model based on the quality metric (1.5).
3. Using the best model, report the final metric on the Testing set.

Cross-validation is used when the training set is not big enough to cut a Validation set from it. Additionally, it takes care of the uneven splits (when in Validation set similar data were sampled).

1.2.2 How to choose k in kNN?

Let's demonstrate, how to choose the best number of neighbors for a fixed distance function. With a slight abuse of the previous notation, we use the testing set as the validation set. We calculate accuracy on the Training set and on the Testing set (which

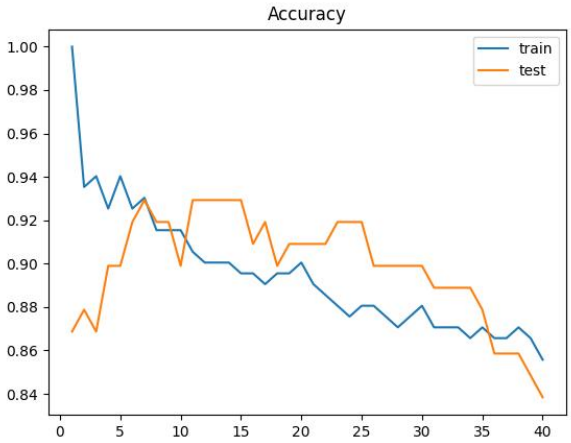


Figure 1.9: Accuracy for different value of k

should be the Validation set) for different k and plot them in one figure. The region, where these curves meet for the first time can be considered as an optimal one. In our case, the optimal $k \in [6, 9]$.

1.2.3 Overfitting and Generalization

We will try to explain, why we have chosen the number of neighbors at the point where training and validation curves met for the first time.

Let's start with an analogy. Imaging, you are preparing for an exam. You can choose two strategies: the first strategy is to memorize all the questions without understanding of the topics. It will take some time, but you will get a good grade. The other strategy is to understand the topics and instead of memorization derive the formulas from the simpler concepts. Which one would you choose?

In ML, your model can describe well the training data, but have a pour performance on the new data. This is called **memorisation** or **overfitting.** If your model performs well on the new data, then we say, it has a good **generalization** ability.

That is why we usually train the model until its performance on the unseen data begins worsen (or stops improving).

1.3 Weighted k Nearest Neighbors (kNN)

Let's look again at Fig. 1.3 with $k = 5$ neighbors. If we count the neighbors of each class

$$\sum_{s=1}^{5} \left[y^{(i_s)} = \blacktriangle\right] = \underset{1}{\left[y^{(8)} = \blacktriangle\right]} + \underset{1}{\left[y^{(9)} = \blacktriangle\right]} + \underset{0}{\left[y^{(6)} = \blacktriangle\right]} + \underset{0}{\left[y^{(7)} = \blacktriangle\right]} + \underset{0}{\left[y^{(4)} = \blacktriangle\right]} = 2,$$

$$\sum_{s=1}^{5} \left[y^{(i_s)} = \blacksquare\right] = \underset{0}{\left[y^{(8)} = \blacksquare\right]} + \underset{0}{\left[y^{(9)} = \blacksquare\right]} + \underset{1}{\left[y^{(6)} = \blacksquare\right]} + \underset{1}{\left[y^{(7)} = \blacksquare\right]} + \underset{1}{\left[y^{(4)} = \blacksquare\right]} = 3,$$

we should assign class ■ to the green point. But it doesn't look correct, because the green point is much closer to the red triangles, than to the blue squares.

To take the distance into account, we introduce the coefficients (weights) decreasing with the distance, for example,

$$\sum_{s=1}^{5} \left[y^{(i_s)} = \blacktriangle\right] = \frac{5}{5} \cdot 1 + \frac{4}{5} \cdot 1 + \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 + \frac{1}{5} \cdot 0 = \frac{9}{5},$$

$$\sum_{s=1}^{5} \left[y^{(i_s)} = \blacksquare\right] = \frac{5}{5} \cdot 0 + \frac{4}{5} \cdot 0 + \frac{3}{5} \cdot 1 + \frac{2}{5} \cdot 1 + \frac{1}{5} \cdot 1 = \frac{6}{5},$$

and choose the class with smaller sum. In this case, we would choose class ▲.

The general formula for prediction takes the following form:

$$a(x) = \underset{c \in \mathbb{Y}}{argmax} \sum_{s=1}^{k} w_s \left[y^{(i_s)} = c\right]. \tag{1.6}$$

1.3.1 Formulas for weights

The weights decay, we use above is given by

$$w_s = \frac{k+1-s}{k}, \qquad s = 1, 2, \ldots, k. \tag{1.7}$$

Another common choice of the weight is

$$w_s = q^s, \quad 0 < q < 1, \qquad s = 1, 2, \ldots, k. \tag{1.8}$$

A kernel

$$w_s = K\left(\frac{d(x,\ x^{(i_s)})}{h}\right), \tag{1.9}$$

where K is a non-increasing function defined on $[0,\ 1]$ or $[0,\ \infty)$.

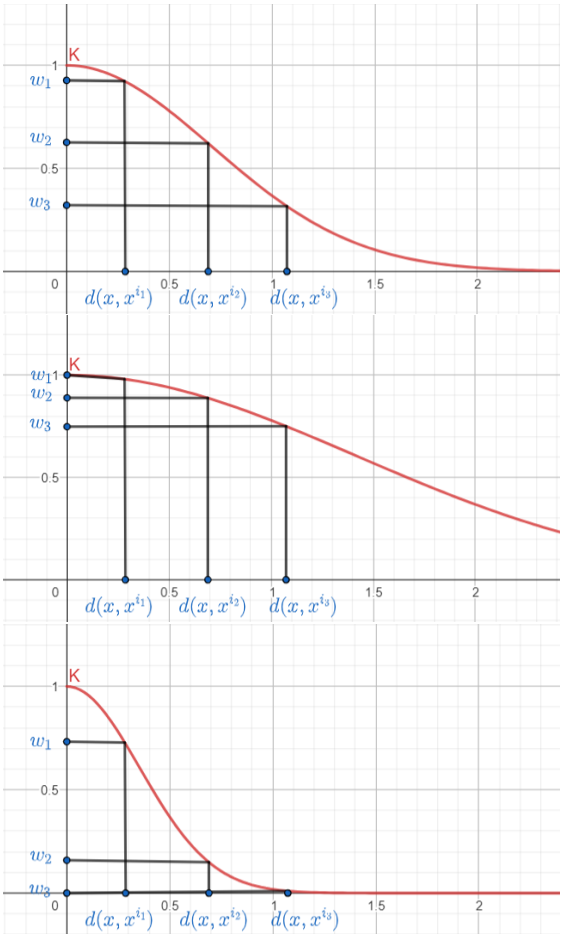


Figure 1.10: $K(d/h) = e^{-(d/h)^2}$ for $h = 1$, 2, and 0.5

With smaller parameter h we will assign smaller weights to larger distances.

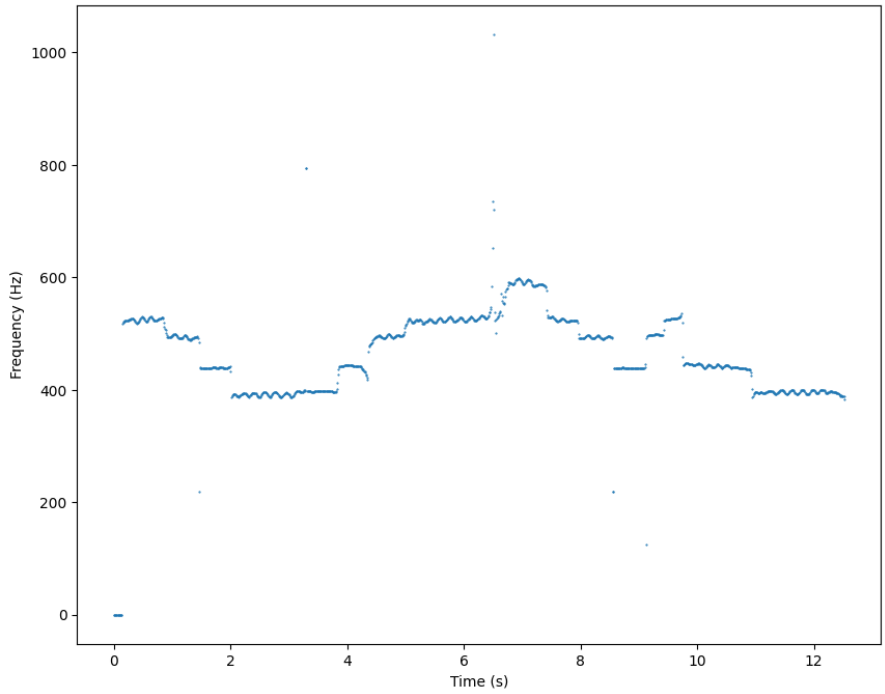


Figure 1.11: Pitch detection of a violin recording

1.4 kNN for Regression Problem

In previous section, we used neighboring data points to predict a class based on the most common labels. We want to use the information of the closer neighbors to predict continuous data. As an example, look at the output of the YIN algorithm for pitch detection[3] in Fig. 1.11. The data plotted $\left(x^{(i)}, y^{(i)}\right)$ represent time of the audio and corresponding pitch prediction. If we were interested in the pitch at time $x^{(i)} < x < x^{(i+1)}$, then most likely we would use the average

$$y \approx a(x) = \frac{y^{(i)} + y^{(i+1)}}{2}. \tag{1.10}$$

If we want to reduce the noise of the data, we can average over more points.[4] This brings us to the kNN algorithm for continuous data prediction.

[3] http://audition.ens.fr/adc/pdf/2002_JASA_YIN.pdf

[4] This idea is based on the Law of Large Numbers, which says that the sum of i.i.d. random variables $X^{(i)}$ tends to their expected value:

$$\frac{X^{(1)} + X^{(2)} + \ldots + X^{(N)}}{n} \approx E\left[X^{(1)}\right].$$

It is reasonable to assume that the noise appears independently and in average is close to zero.

Definition 1.4

By **regression task** in machine learning we mean a problem, which solution is an arbitrary number from some range, e.g., $\mathbb{Y} = \mathbb{R}$ in case of temperature prediction, or $\mathbb{Y} = [0, 1]$ in case we predict the probability.

Remark 1.2

Looking at formula (1.10) one may ask the question: isn't regression is the same as **interpolation?** The answer is no. In case of interpolation, we assign a value close to the near points. In case of the stock market prediction, the intermediate value can deviate a lot from the nearest values. In case of regression, we hope to find some pattern or functional dependency between features and the target.
This definition of the regression problem shouldn't be confused with the **statistical regression** problem, when we assume some probability distribution behind the data.
The other difference is that in the interpolation, the resulted curve goes through the data, in case of regression, it may not go through any of the data points.

1.4.1 k Nearest Neighbors (kNN) for Regression: training

- We are given $\left(x^{(i)},\ y^{(i)}\right),\ i = 1,\ 2, \ldots,\ N.$
- $\mathbb{Y} = \mathbb{R}$ (regression problem).
- training = memorizing of the given data.

1.4.2 k Nearest Neighbors (kNN) for Regression: prediction

- We have a new feature x.
- Define the distance between new feature and $x^{(i)}$: $d\left(x,\ x^{(i)}\right)$.
- Rearrange the objects by the closeness to x :

$$d\left(x,\ x^{(i_1)}\right) \leq d\left(x,\ x^{(i_2)}\right) \leq \ldots \leq d\left(x,\ x^{(i_N)}\right).$$

- Look at the first k labels and assign the average of those:

$$a(x) = \frac{1}{k} \sum_{s=1}^{k} y^{(i_s)}. \tag{1.11}$$

The result of kNN regression for the data in Fig. 1.11 with $k = 15$ represented in Fig. 1.12. We see some peaks caused by outliers, we will discuss it in the next chapter. This section we conclude with weighted version of kNN regression.

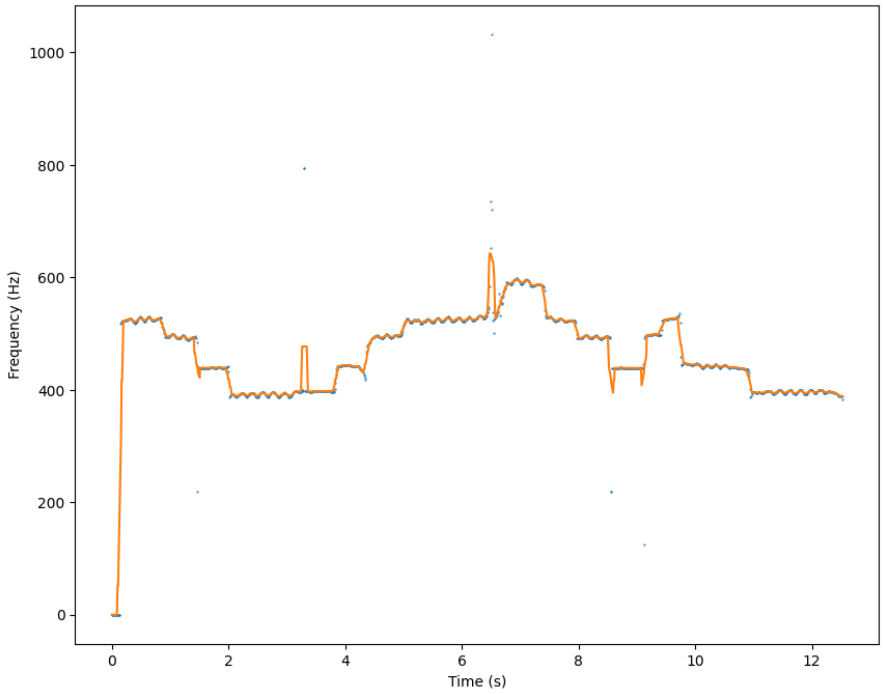


Figure 1.12: $k = 15$

1.4.3 k Nearest Neighbors (kNN) Weighted Regression

In case of weighted kNN regression, we normalize the weighted sum:

$$a(x) = \frac{\sum\limits_{s=1}^{k} w_s y^{(i_s)}}{\sum\limits_{s=1}^{k} w_s}. \tag{1.12}$$

If $w_s = 1$ for all $s = 1, \ldots, k$, then this formula coincides with average (1.11).

Remark 1.3

When the weights are defined by a kernel function: $w_s = K\left(\frac{d(x,\, x^{(i_s)})}{h}\right)$, this prediction coincides with the Nadaraya — Watson estimator in the statistical non-parametric regression.

1.4.4 Pros and Cons of kNN

Pros

- If there are many examples, the algorithm gives a good prediction.

- Simple training.
- Few hyperparameters.
- There exist problems, where kNN works better, e.g., texts classification with many classes.

Cons

- Usually, other models work better.
- Extensive usage of storage.
- Sorting for finding the neighbors.
- The model has limited tuning abilities.

1.5 Exercises

Problem 1. Apply the kNN method with 2 neighbors to the following data.

x	0.45	-0.1	2	0.3	-0.5	0.7	0
y	$+1$	-1	$+1$	$+1$	-1	-1	$+1$

Report the accuracy.

Hint: Order the data by x first. Consider the current point unlabeled. In cases of ambiguity choose the closest point, if the points (with different labels) are equidistant, return 0 as a refusal of classification.

2 Exploratory Data Analysis (EDA)

2.1 Box-Plot and Outliers

Fig. 1.11 represents frequencies detected by YIN algorithm, but some frequencies were detected incorrectly, like 1000 Hz. We call the data with a high deviation from the average them **outliers.**

Definition 2.1

Let

$$y^{(i_1)},\ y^{(i_2)},\ldots,\ y^{(i_N)}$$

be a variational series, i.e., the sorted sequence of the values $y^{(i)}$ $(i = 1, \ldots, N)$. For a given number $0 \leq \alpha \leq 1$ the value

$$y^{(i_s)}, \quad s = \lfloor \alpha N \rfloor \tag{2.1}$$

is called the $\boldsymbol{\alpha \cdot 100}$**-th percentile.**
25-th, 50-th, and 75-th percentiles are known as **1-st, 2-d, and 3-d quartiles.** The 2-d quartile coincides with the **median.**

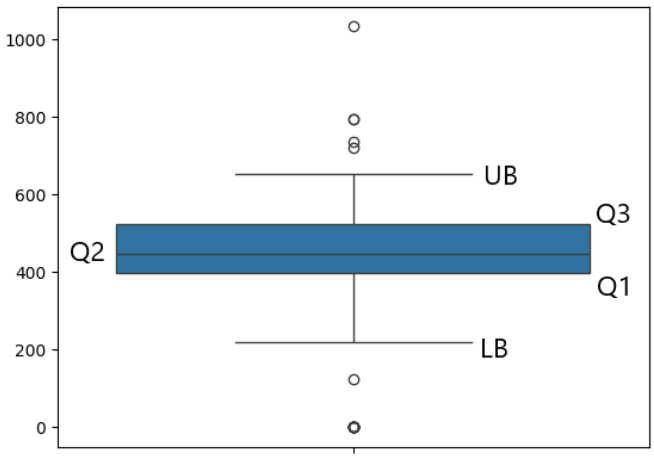


Figure 2.1: Box-plot for data in Fig. 1.11 (seaborn library)

The **box-plot diagram** represents the quartiles, there are also whiskies, we denote them by Low Bound (LB) and Upper Bound (UP). The points outside the whiskies

are considered as outliers. In mathplotlib and seaborn libraries, they are calculated as follows:

$$LB = Q1 - 1.5 \cdot (Q3 - Q1), \qquad UB = Q3 + 1.5 \cdot (Q3 - Q1). \tag{2.2}$$

The upper and lower bounds can be defined in different ways. In case of the pitch detection, we used the 3-d and 97-th percentiles as the threshold for outliers (see Fig.2.2).

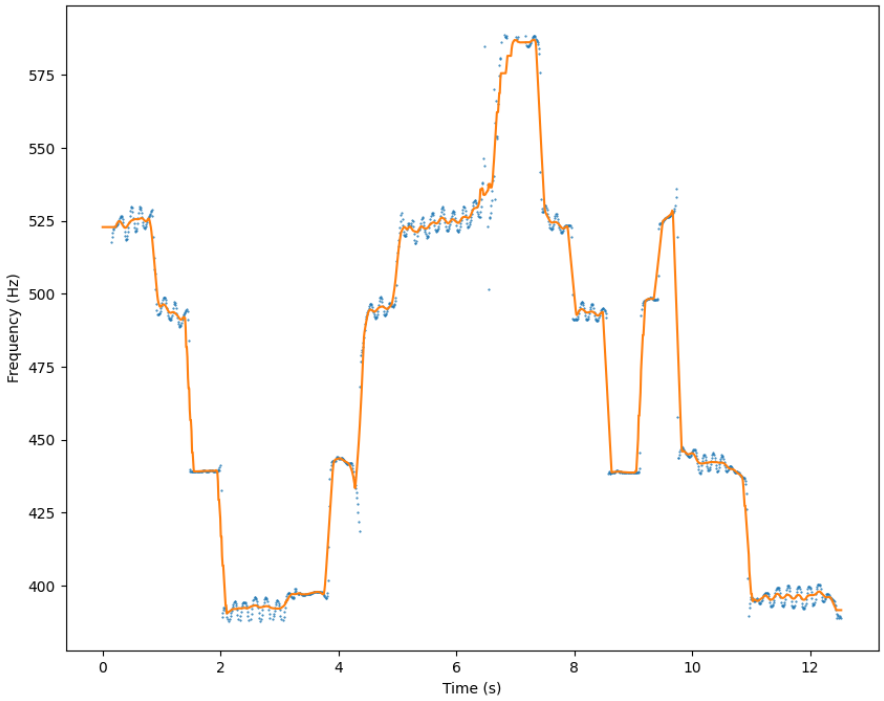


Figure 2.2: After removing points outside 3% and 97% percentiles

Remark 2.1

There is no strict definition of the outliers. Some times rare or very unusual events can be of high importance. For example, an airplane crash is a very rear event, but it can not be eliminated if we analyze the safety of the flights.

2.2 Binarization (or Binning)

In some cases, the small differences in data are not important. For example, age 60, or 61. In this case we can split the continuous feature into intervals (bins). E.g., in case of age

$$[0,\ 5),\ [5, 10),\ [10, 16),\ [16, 21),\ [21, 35),\ [35, 50),\ [50, \infty). \tag{2.3}$$

If we enumerate these intervals as $0,\ 1,\ \ldots, 6$, then we can create a discrete ordinal feature from the continuous one.

From Fig. 2.2 it is clear that only five notes were played. The result of binning of the detected frequencies is presented below.

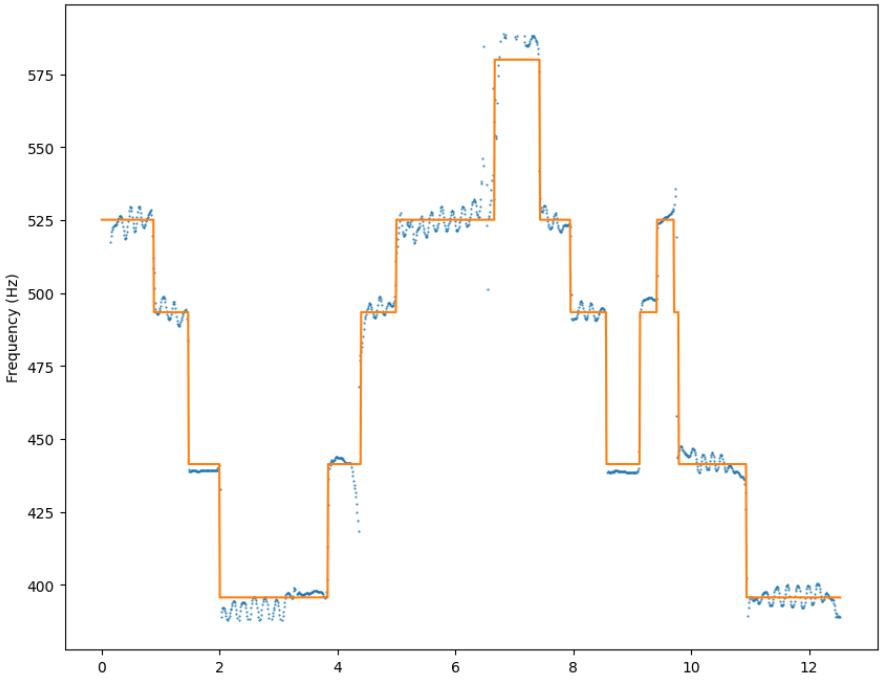


Figure 2.3: 5 bins

2.3 Feature/Target Distribution

In the remainder of this chapter, we will work with the *Titanic* dataset.[1]

2.3.1 Categorical (Discrete)

To visualize a categorical feature distribution we can use **bar charts.** Fig. 2.4 shows the number of objects (Titanic's passengers) in each category of the feature *Sex*.[2] Fig. 2.4 shows the normalized (probability) distribution with total sum equal to one.[3] The normalized bar chart can be viewed as *probability mass function* (pmf).

2.3.2 Numerical (Continuous)

The distribution of continuous data is described by the *probability density function* (pdf). In order to approximate pdf, we split the full range of the data into small intervals

[1] https://www.kaggle.com/c/titanic/, we use its version https://raw.githubusercontent.com/iad34/seminars/master/materials/data_sem1.csv

[2] `sns.barplot(Data['Sex'].value_counts(dropna=False))`

[3] `sns.barplot(Data['Sex'].value_counts(dropna=False, normalize=True))`

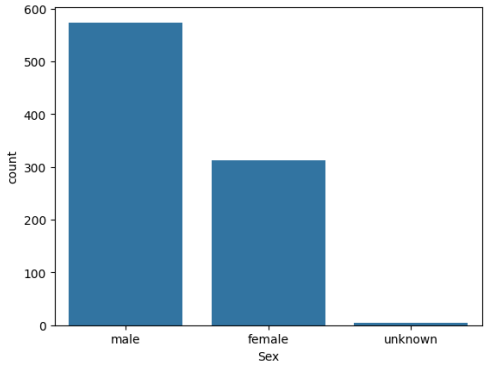


Figure 2.4: Counts (or Frequencies)

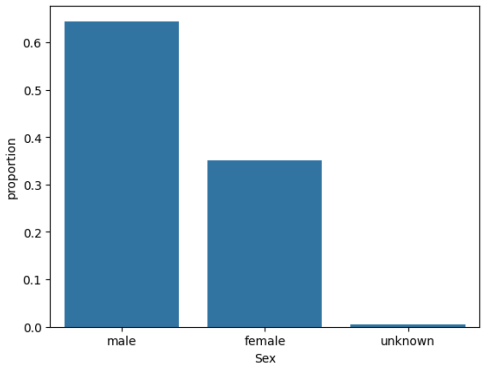


Figure 2.5: Normalized Counts (of Relative Frequencies)

(bins) and count the number of data points in these bins (see Fig. 2.6),[4] thus we get the *histogram.* The normalized histogram (divided by the total number of samples) approximates pdf. To get a more smooth curve the *kernel density estimation* can be used:[5]

$$p(x) \approx p_h(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x^{(i)}}{h}\right), \tag{2.4}$$

where K is a kernel function (see kNN regression). In Fig. 2.7

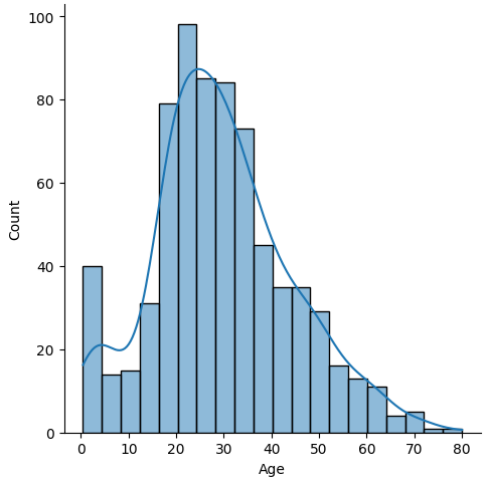


Figure 2.6: Histogram

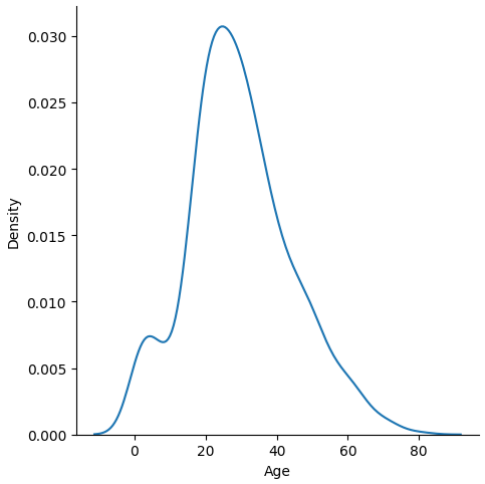


Figure 2.7: Kernel Density Estimation (over Normalized Histogram)

The following topics are discussed in the class notebook.

[4] `sns.displot(Data.Age, kde=True)`
[5] `sns.displot(Data.Age, kind='kde')`

2.4 Feature Importance

2.4.1 One-Factor Analysis

2.4.2 Two-Factor Analysis

2.4.3 Correlation

2.4.4 Mutual Information

2.4.5 By Model

2.5 Feature Creation

2.5.1 Scaling and Functional Transformations

2.5.2 Binning

2.5.3 One-Hot Encoding for Nominal Features

2.5.4 Dealing with Missing Values

2.5.5 Combining Features

2.5.6 Dimensionality Reduction

2.6 Transformations of the Target

3 Linear Regression

Consider Galton's data about the height of parents and children[1]

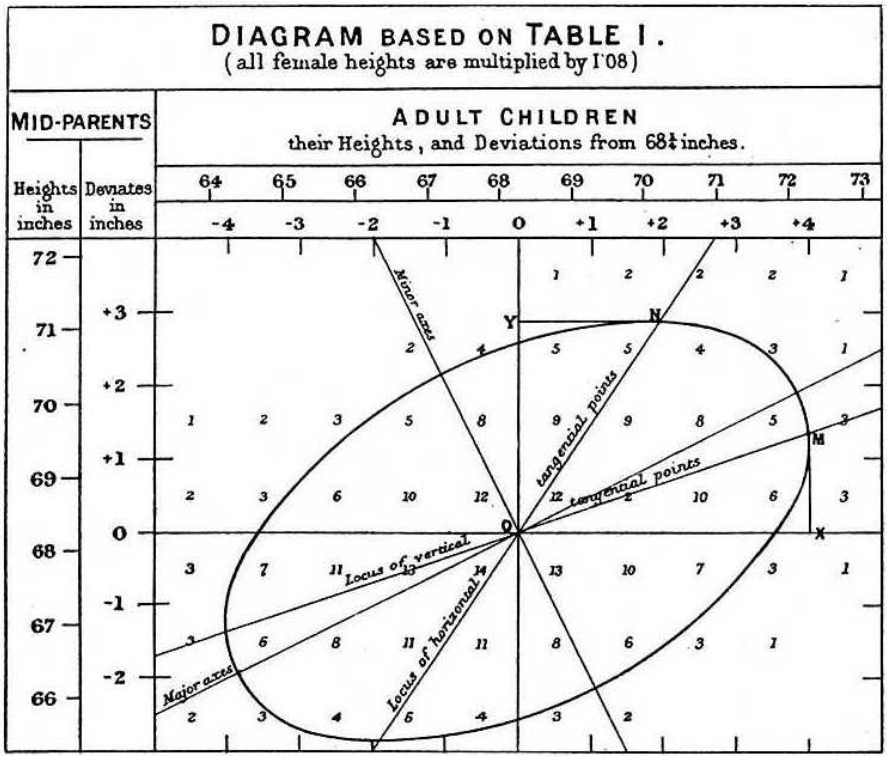


Figure 3.1: The plot from F. Galton's paper

The simplest assumption one can make: the data lie along a line. This data approximation is called *linear regression*[2]

F. Galton introduced the term *regression.* In Fig. 3.1, you can see that "extremely tall parents tend to have children who are taller than average and extremely small parents tend to have children who are smaller than average, but in both cases the children tend to be closer to the average than were their parents."[3]

We plotted the data from the Galton dataset[4] in Fig. 3.2

[1] Francis Galton. Regression towards mediocrity in hereditary stature. The Journal of the Anthropological Institute of Great Britain and Ireland. 15: 246–263 (1886).

[2] When we talk about regression, we assume some law behind the data (compare with Remark 1.2).

[3] Stephen Senn. Francis Galton and regression to the mean. Significance, Vol. 8, Iss. 3.

[4] https://www.randomservices.org/random/data/Galton.html

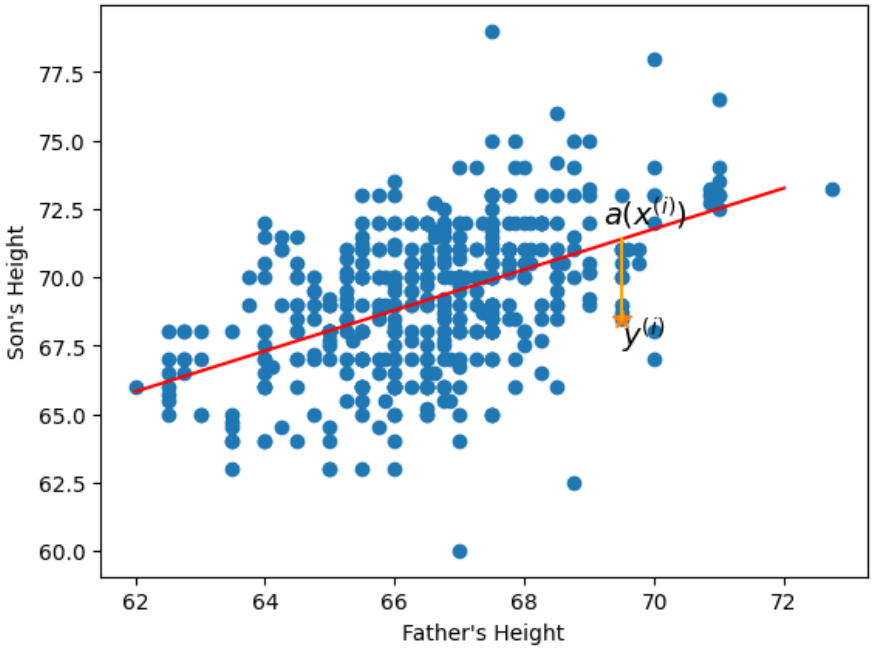


Figure 3.2: Scatter plot of the sons' height VS fathers' height from Galton's notebook

If we want to predict the children's height using the father's height using a linear function

$$a(x) = w_1 x + w_0, \tag{3.1}$$

we may want to minimize an average error or the sum of (squared) distances from this line to the real height:

$$\frac{1}{N}\sum_{i=1}^{N}\left(a\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(w_1 x^{(i)} + w_0 - y^{(i)}\right)^2 \to \min_{w_0, w_1}. \tag{3.2}$$

The error (3.2) depends only on two parameters w_0 and w_1, in other words, it is a function of two variables.

3.1 Optimization of Smooth Convex Functions

Definition 3.1

A point x^* is said to be a **local minimum** of a function $f\colon \mathbb{R}^d \to \mathbb{R}$, if there exists a neighborhood of some radius ε $B_\varepsilon(x^*)$, such that

$$f(x^*) \le f(x), \quad \forall x \in B_\varepsilon(x^*). \tag{3.3}$$

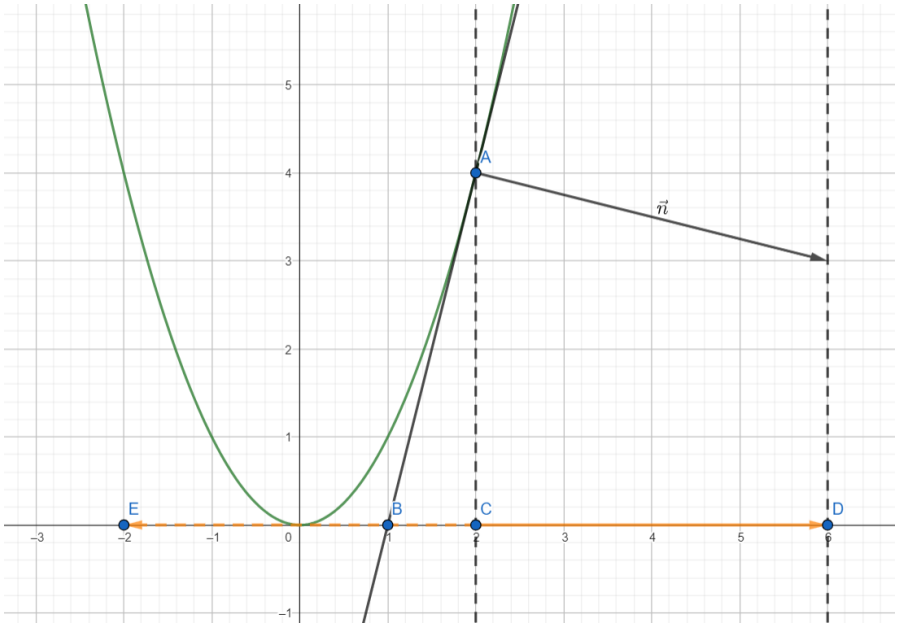


Figure 3.3: Illustration of the normal vector.

Definition 3.2

A function f is said to be **convex** if for two points x and y

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y), \quad \lambda \in [0,1]. \tag{3.4}$$

A smooth convex function has a unique minimum.

3.2 Gradient Decent in 1D

Consider a function

$$f(x) = x^2. \tag{3.5}$$

This function is differentiable and convex. And it has a unique minimum at $x_* = 0$ (see Fig. 3.3).

Recall the equation of a line with slope k

$$y = kx + b. \tag{3.6}$$

If we know that it goes through the point (x_0, y_0), then

$$y_0 = kx_0 + b. \tag{3.7}$$

Subtracting these equations, we get the equation of the tangent line:

$$y - y_0 = k(x - x_0) \qquad \text{or} \qquad \frac{x - x_0}{1} = \frac{y - y_0}{k}, \tag{3.8}$$

where $k = f'(x_0)$. And $\vec{\ell} = (1, k)$ is the vector along this line. The unnormalized normal vector can be chosen as[5]

$$\vec{n} = (k, -1). \tag{3.9}$$

Consider a point $x_0 = 2$, the derivative at this point is equal to $f'(x_0) = 4$, and the normal vector $n = (4, -1)$ (see Fig. 3.3). The projection of the vector $\vec{n}$ indicates the direction of the growth of the function f. This means that if we go from the point $x_0 = 2$ to the point $x_1 = x_0 + h = 6$, we will get a higher value of f.

This observation gives us the idea to go in the opposite direction, i.e., form x_0 to $x_0 - h$. In this case the new point is

$$x_1 = x_0 - f'(x_0) = 2 - 4 = -2. \tag{3.10}$$

If we continue this process and chose $x_2 = x_1 - f'(x_1) = -2 - (-4) = 2$, we will bounce between two points. To avoid this, we can multiply $f'(x_0)$ by a scaling factor $0 < \eta < 1$. This small modification of the rule (3.10) can resolve this problem. Now the update rule for every iteration takes form

$$x^{new} = x^{old} - \eta f'\big(x^{old}\big). \tag{3.11}$$

If we use $\eta = 0.5$, then $x_1 = x_0 - 0.5 f'(x_0) = 2 - 0.5 \cdot 4 = 0$, and we reach the minimum of f in one iteration.

Mathematically, by definition of a differentiable function,

$$f(x + h) - f(x) = f'(x)h + o(h) \tag{3.12}$$

It means, that if we move from the point x to point $x + h$, the function will increase by $f'(x)h$ plus a term that vanishes, when $h \to 0$.

Let us set $h = -\eta f'(x)$, then

$$f(x + h) - f(x) = -\eta\big(f'(x)\big)^2 + o\big(\eta f'(x)\big) \tag{3.13}$$

or

$$f(x + h) = f(x) - \eta\big(f'(x)\big)^2 + o\big(\eta f'(x)\big). \tag{3.14}$$

Choosing η small enough, we can make the last term smaller than the previous one, then we have

$$f(x + h) < f(x). \tag{3.15}$$

This means that the value of the function f will decrease in each iteration and eventually it will reach the global minimum.

3.3 Optimization and Gradient Decent in 2D

3.3.1 Partial Derivatives and Gradient Vector

Consider a function

$$z = x^2 + (y - 1)^2. \tag{3.16}$$

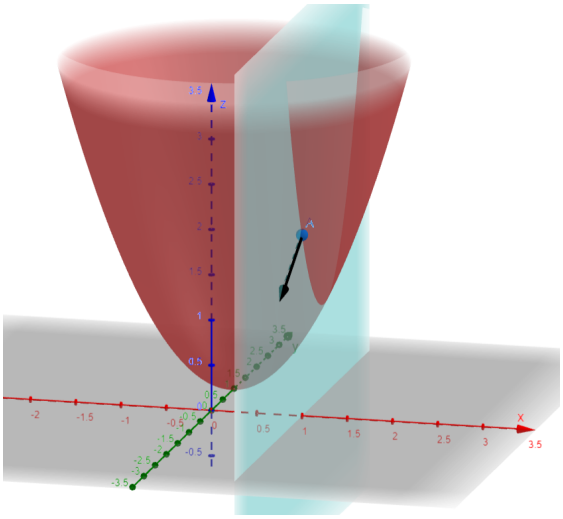


Figure 3.4: The normal vector in the section $x = 1$

Consider point $(x_0, y_0) = (1, 0)$. The value at this point is $z(1, 0) = 2$. If we fix $x = 1$, then in the section we will see a parabola:

$$z(1, y) = 1 + (y - 1)^2. \tag{3.17}$$

Because it is a function of one variable, the normal vector is

$$\vec{n}_y = \left(0, \left.\frac{\partial z}{\partial y}\right|_{(x,y)=(1,0)}, -1\right). \tag{3.18}$$

When we calculate the partial derivative with respect to y, we treat x as a constant.[6] In our case,

$$\frac{\partial z}{\partial y} = 2(y - 1) \quad \text{and} \quad \vec{n}_y = (0, -2, -1). \tag{3.19}$$

Now, we fix another coordinate: $y = 0$ (see Fig.). Then we have a function of one variable x :

$$z(x, 0) = x^2 + 1. \tag{3.20}$$

The normal vector at $x = 1$ is

$$\vec{n}_x = \left(\left.\frac{\partial z}{\partial x}\right|_{(x,y)=(1,0)}, 0, -1\right). \tag{3.21}$$

[5] Because the dot product $\vec{\ell} \cdot \vec{n} = 0$.

[6] By definition, $\dfrac{\partial z}{\partial y} = \lim\limits_{\Delta y \to 0} \dfrac{z(x, y + \Delta y) - z(x, y)}{\Delta y}$.

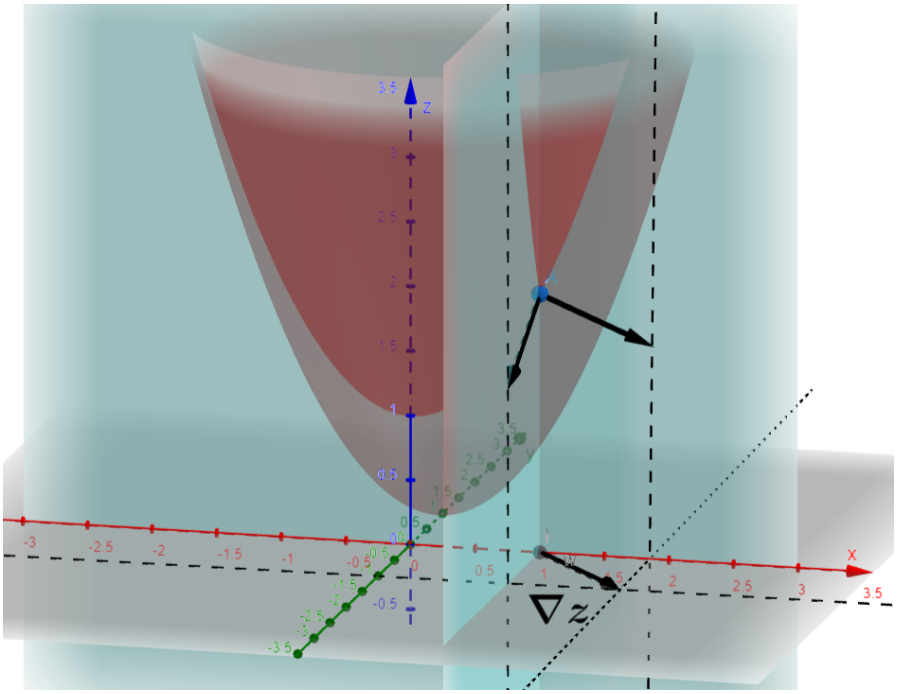


Figure 3.5: The gradient at $(x, y) = (1, 0)$ indicates the direction of the speediest increase of the function z

When we calculate the partial derivative with respect to x, we treat y as a constant. In our case,

$$\frac{\partial z}{\partial x} = 2x \quad \text{and} \quad \vec{n}_x = (0, 1, -1). \tag{3.22}$$

Definition 3.3

The vector of partial derivatives

$$\operatorname{grad} z = \nabla z = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} \right) \tag{3.23}$$

is called a **gradient** of the function of two variables z.

3.3.2 Minimization of a Smooth Convex Function in 2D

Because in the minimum the normal vector to the function $z = f(x, y)$

$$\vec{n} = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1 \right) \tag{3.24}$$

should point down, that is equal to the zero gradient:

$$\nabla z = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\right) = 0. \tag{3.25}$$

In case of our function $z = x^2 + (y-1)^2$, we have two equations:

$$\begin{cases} \frac{\partial z}{\partial x} = 2x = 0, \\ \frac{\partial z}{\partial y} = 2(y-1) = 0. \end{cases} \tag{3.26}$$

The solution gives the point of minimum:

$$x^* = 0,\ y^* = 1, \quad \text{and} \quad z(x^*, y^*) = 0. \tag{3.27}$$

3.3.3 Gradient Descent Algorithm

In two-dimensional case the gradient descent algorithm looks as follows.

1. Choose the initial point (x_0, y_0)
2. Update the points by the rule

$$\begin{pmatrix} x^{new} \\ y^{new} \end{pmatrix} = \begin{pmatrix} x^{old} \\ y^{old} \end{pmatrix} - \eta \begin{pmatrix} \frac{\partial z}{\partial x}\Big|_{(x,y)=(x^{old},y^{old})} \\ \frac{\partial z}{\partial y}\Big|_{(x,y)=(x^{old},y^{old})} \end{pmatrix} \tag{3.28}$$

or in vector notation

$$\boldsymbol{x}^{new} = \boldsymbol{x}^{old} - \eta \nabla z\big(\boldsymbol{x}^{old}\big). \tag{3.29}$$

There are theorems about the upper bound for the learning rate η in the case of a smooth function (see, for example, Theorem 3.4 in https://arxiv.org/abs/2301.11235). But often we don't know the properties of the loss function, then we choose the learning rate small enough, and if the algorithm does not converge, we try smaller learning rate value.

For the loss function (3.2) the algorithm takes form

Algorithm 1 GD

$T \leftarrow$ Number of epochs ▷ The number of times we go through the entire dataset
$w \leftarrow$ Initialize (randomly)
for *epoch* $= 1..T$ **do**
  $w^{new} = w^{old} - \eta \nabla_w L(w^{old})$
  You might want to update the learning rate η
end for

3.4 Least Mean Squares for Linear Regression

3.4.1 Simple Linear Regression

Assume, the data have only one feature. Then the model takes form

$$a(x) = w_0 + w_1 x. \tag{3.30}$$

We want to minimize the sum of squared residuals:

$$SSR(w_0, w_1) = \sum_{i=1}^{N} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right)^2. \tag{3.31}$$

In order to find optimal values w_0^* and w_1^*, we should solve the following equations:

$$\begin{cases} \dfrac{\partial SSR}{\partial w_0} = -2 \sum\limits_{i=1}^{N} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right) = 0, \\ \dfrac{\partial SSR}{\partial w_1} = -2 \sum\limits_{i=1}^{N} x^{(i)} \left(y^{(i)} - w_0 - w_1 x^{(i)}\right) = 0. \end{cases} \tag{3.32}$$

After division by -2 we get

$$\begin{cases} \sum\limits_{i=1}^{N} y^{(i)} - N w_0 - w_1 \sum\limits_{i=1}^{N} x^{(i)} = 0, \\ \sum\limits_{i=1}^{N} x^{(i)} y^{(i)} - w_0 \sum\limits_{i=1}^{N} x^{(i)} - w_1 \sum\limits_{i=1}^{N} x^{(i)2} = 0. \end{cases} \tag{3.33}$$

or

$$\begin{cases} \overline{y} - w_0 - w_1 \overline{x} = 0, \\ \overline{xy} - w_0 \overline{x} - w_1 \overline{x^2} = 0, \end{cases} \tag{3.34}$$

where

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}, \quad \overline{y} = \frac{1}{N} \sum_{i=1}^{N} y^{(i)}, \quad \overline{xy} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)} y^{(i)}, \quad \overline{x^2} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)2}. \tag{3.35}$$

After solution of this system with respect to w_0 and w_1, we will get

$$a(x) = \overline{y} + \rho_{XY} \frac{\sigma_X}{\sigma_Y} \left(x - \overline{x}\right), \tag{3.36}$$

where

$$\rho_{XY} = \frac{\frac{1}{N} \sum\limits_{i=1}^{N} \left(x^{(i)} - \overline{x}\right) \left(y^{(i)} - \overline{y}\right)}{\sigma_X \sigma_Y}, \quad \sigma_X^2 = \frac{1}{N} \sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)^2, \quad \sigma_Y^2 = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - \overline{y}\right)^2 \tag{3.37}$$

are correlation and variances correspondingly.

3.5 Closed Form Solution for Linear Regression

Consider the following model for data $\left(x^{(i)}, y^{(i)}\right)$:

$$y^{(i)} \approx \hat{y}^{(i)} = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + \ldots + w_d x_d^{(i)}, \quad i = 1,\ 2, \ldots,\ N. \tag{3.38}$$

Using vector notation, it can be represented as follows:

$$y^{(i)} \approx \hat{y}^{(i)} = \tilde{x}^{(i)} w^T, \tag{3.39}$$

where $\tilde{x}^{(i)} = \left(1,\ x_1^{(i)},\ x_2^{(i)}, \ldots,\ x_d^{(i)}\right)^T$ and $w = (w_0,\ w_1,\ w_2,\ \ldots,\ w_d)^T$.

The sum of squared residuals can be written using the dot product:

$$SSR = \left(y^{(1)} - \hat{y}^{(1)}, \ldots,\ y^{(N)} - \hat{y}^{(N)}\right) \begin{pmatrix} y^{(1)} - \hat{y}^{(1)} \\ \vdots \\ y^{(N)} - \hat{y}^{(N)} \end{pmatrix}. \tag{3.40}$$

Let us introduce more convenient notation for the vector of errors:

$$\begin{aligned} \begin{pmatrix} y^{(1)} - \hat{y}^{(1)} \\ \vdots \\ y^{(N)} - \hat{y}^{(N)} \end{pmatrix} &= \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{pmatrix} - \begin{pmatrix} w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)} + \ldots + w_d x_d^{(1)} \\ \vdots \\ w_0 + w_1 x_1^{(N)} + w_2 x_2^{(N)} + \ldots + w_d x_d^{(N)} \end{pmatrix} \\ &= \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{pmatrix} - \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \ldots & x_d^{(1)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_1^{(N)} & x_2^{(N)} & \ldots & x_d^{(N)} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix} = y - Xw. \end{aligned} \tag{3.41}$$

Thus,

$$SSR = (y - Xw)^T (y - Xw). \tag{3.42}$$

To minimize this function, we can calculate its gradient and equate it to zero.[7]

$$\begin{aligned} \frac{\partial SSR}{\partial w} &= \frac{\partial}{\partial w}\left((y^T - w^T X^T)(y - Xw)\right) \\ &= \frac{\partial}{\partial w}\left(y^T y - w^T X^T y - y^T X w + w^T X^T X w\right) \\ &= -2X^T y + 2X^T X w = 0. \end{aligned} \tag{3.43}$$

From the last equation we find

$$w^* = \left(X^T X\right)^{-1} X^T y. \tag{3.44}$$

And the model looks as follows:

$$a(x) = w_0^* + w_1^* x_1 + \ldots + w_d^* x_d = \left(w^*, \tilde{x}\right) = w^{*T} \tilde{x} = \tilde{x}^T w^*, \tag{3.45}$$

where $\tilde{x}$ is the vector x extended by a constant one.

[7] Note that the by the derivative $\frac{\partial f}{\partial A}$ of a scalar function $f(A)$ with respect to a matrix A we mean the derivatives with respect to each component of A, i.e., the resulted shape should be the same the shape of the matrix A, it is referred as **denominator layout.** The formulas of matrix derivatives can be found at https://en.wikipedia.org/wiki/Matrix_calculus.

3.6 Regression Metrics

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(a(x^{(i)}) - y^{(i)}\right)^2 \tag{3.46}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(a(x^{(i)}) - y^{(i)}\right)^2} \tag{3.47}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|a(x^{(i)}) - y^{(i)}\right| \tag{3.48}$$

$$Huber = \frac{1}{N}\sum_{i=1}^{N}\phi_\varepsilon\left(a(x^{(i)}) - y^{(i)}\right), \tag{3.49}$$

where

$$\phi_\varepsilon(z) = \begin{cases} \frac{1}{2}z^2, & |z| < \varepsilon, \\ \varepsilon\left(|z| - \frac{1}{2}\varepsilon\right), & |z| \geq \varepsilon. \end{cases}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y^{(i)} - a(x^{(i)})}{y^{(i)}}\right| \tag{3.50}$$

It allows for larger errors for large values of the target and penalizes smaller errors more heavily for small values of the target. E.g., $\frac{1000-900}{1000} = 0.1$ and $\frac{10-9}{9} = 0.1$. The first case has an absolute error of 100, while the second has an absolute error of 1. MAPE is often used in stock planning.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(y^{(i)} - a(x^{(i)})\right)^2}{\sum_{i=1}^{N}\left(y^{(i)} - \bar{y}\right)^2} \tag{3.51}$$

If the model has a constant prediction $a(x) = \overline{y}$, $R^2 = 0$. When the prediction is 100% correct, $R^2 = 1$. This metric is returned by scikit-learn model, when you call `model.score(X,y)`.

$$Quantile = \frac{1}{N}\sum_{i=1}^{N}\rho_\tau\left(y^{(i)} - a(x^{(i)})\right), \tag{3.52}$$

where

$$\rho_\tau(z) = \begin{cases} \tau z, & z > 0, \\ (\tau - 1)\, z, & z \leq 0. \end{cases}$$

$$MSLE = \frac{1}{N}\sum_{i=1}^{N}\left(\log\left(a(x^{(i)}) + 1\right) - \log\left(y^{(i)} + 1\right)\right)^2 \tag{3.53}$$

This metric is used when we are more interested in the relative magnitude or order of the error rather than the precise absolute error.

3.7 Statistical Linear Regression

The optimal predictor for MSE loss

$$a^* = \underset{a\in\mathcal{A}}{\operatorname{argmin}}\ Risk(a;\ Data) = \underset{a\in\mathcal{A}}{\operatorname{argmin}}\ E\big[Loss(a(X), Y)\big].$$

is given by

$$a^*(x) = E[Y|X = x]$$

The conditional expectation $E[Y|X = x]$ is called a **regression line.**
Under assumption that X and Y have joint Gaussian distribution

$$(X,\ Y) \sim N(\mu,\ \Sigma), \qquad \mu = \begin{pmatrix}\mu_X\\ \mu_Y\end{pmatrix}, \quad \Sigma = \begin{pmatrix}\sigma_X & \rho\\ \rho & \sigma_Y\end{pmatrix}$$

the conditional distribution is also Gaussian

$$Y|X = x\ \sim\ N\left(\mu_Y + \frac{\sigma_Y}{\sigma_X}\rho(x-\mu_X),\ (1-\rho^2)\sigma_Y\right)$$

Therefore, the regression line is

$$a^*(x) = E[Y|X = x] = \mu_Y + \frac{\sigma_Y}{\sigma_X}\rho(x-\mu_X) = w_0 + w_1 x,$$

where

$$w_0 = \mu_Y - \frac{\sigma_Y}{\sigma_X}\rho\mu_X, \quad w_1 = \frac{\sigma_Y}{\sigma_X}\rho.$$

This means that in the case when the data $(x^{(i)}, y^{(i)})$ come from an underlying normal distribution, the ordinary least squares (OLS) estimator coincides with the statistical regression estimator.

3.8 Exercises

Problem 1 (Linear Model). Assume that we have three points:

x	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$
y	$y^{(1)}$	$y^{(2)}$	$y^{(3)}$

and we want to draw a line

$$y = w_0 + w_1 x$$

minimizing the sum of squared errors

$$L(w_0, w_1) = \left(y^{(1)} - w_0 - w_1 x^{(1)}\right)^2 + \left(y^{(2)} - w_0 - w_1 x^{(2)}\right)^2 + \left(y^{(3)} - w_0 - w_1 x^{(3)}\right)^2.$$

This is a quadratic function; therefore, it has a unique minimum.

Calculate the derivatives:

$$\frac{\partial L}{\partial w_0} =$$
$$\frac{\partial L}{\partial w_1} =$$

Factor out w_0 and w_1 and equate the derivatives to zero:

$$\frac{\partial L}{\partial w_0} =$$
$$\frac{\partial L}{\partial w_1} =$$

Divide the equations by 6 and rewrite them using the following notation:

$$\overline{x} = \frac{x^{(1)} + x^{(2)} + x^{(3)}}{3}, \qquad \overline{y} = \frac{y^{(1)} + y^{(2)} + y^{(3)}}{3},$$

$$\overline{x^2} = \frac{x^{(1)2} + x^{(2)2} + x^{(3)2}}{3}, \qquad \overline{xy} = \frac{x^{(1)}y^{(1)} + x^{(2)}y^{(2)} + x^{(3)}y^{(3)}}{3}.$$

$Eq.1:$

$Eq.2:$

Solve the equations for w_0 and w_1

$w_0 =$

$w_1 =$

Problem 2. Given the following data

x	1	2	3
y	1	$\frac{3}{2}$	$\frac{5}{2}$

Calculate weights (result of Problem 1) and plot the line (draw the given points as well).

$w_0 =$

$w_1 =$

Problem 3. Prove formula (3.36).

Solution. From (3.34)

$w_0 = \overline{y} - w_1\overline{x}$

$$w_1 = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - (\overline{x})^2}$$

Let's rewrite the last equality:

$$\begin{aligned}
w_1 &= \frac{\frac{1}{N}\sum_{i=1}^{N} x^{(i)}y^{(i)} - \left(\frac{1}{N}\sum_{i=1}^{N} x^{(i)}\right)\left(\frac{1}{N}\sum_{i=1}^{N} y^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N} {x^{(i)}}^2 - \left(\frac{1}{N}\sum_{i=1}^{N} x^{(i)}\right)^2} = \\
&= \frac{\frac{1}{N}\sum_{i=1}^{N} x^{(i)}y^{(i)} - \overline{y}\frac{1}{N}\sum_{i=1}^{N} x^{(i)} - \overline{x}\frac{1}{N}\sum_{i=1}^{N} y^{(i)} + \overline{x}\frac{1}{N}\sum_{i=1}^{N} y^{(i)}}{\frac{1}{N}\sum_{i=1}^{N} {x^{(i)}}^2 - \overline{x}\frac{1}{N}\sum_{i=1}^{N} x^{(i)} - \overline{x}\frac{1}{N}\sum_{i=1}^{N} x^{(i)} + (\overline{x})^2} \\
&= \frac{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)}y^{(i)} - \overline{y}x^{(i)} - \overline{x}y^{(i)} + \overline{x}y^{(i)}\right)}{\frac{1}{N}\sum_{i=1}^{N} \left({x^{(i)}}^2 - 2\overline{x}x^{(i)} + (\overline{x})^2\right)} \\
&= \frac{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)\left(y^{(i)} - \overline{y}\right)}{\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)^2}\right)^2} \\
&= \frac{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)\left(y^{(i)} - \overline{y}\right)}{\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)^2}\right)^2} \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \overline{y}\right)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \overline{y}\right)^2}} \\
&= \frac{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)\left(y^{(i)} - \overline{y}\right)}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)^2}\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \overline{y}\right)^2}} \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \overline{y}\right)^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(x^{(i)} - \overline{x}\right)^2}} = \rho_{XY}\frac{\sigma_Y}{\sigma_X}.
\end{aligned} \tag{3.54}$$

Problem 4. From expression (3.43) derive the gradient descent formula (3.29) in matrix notation:

$$w^{new} = function(X, y, w^{old}, \eta) \tag{3.55}$$

4 Stochastic Gradient Descent

4.1 SGD

The minimization of SSR is equivalent to the minimization of MSE:

$$L(w) = \frac{1}{N} SSR = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - w^T \tilde{x}^{(i)} \right)^2 = \frac{1}{N} \sum_{i=1}^{N} L^{(i)}(w). \tag{4.1}$$

In the full-batch gradient descent, we update the weights by the formula

$$w^{new} = w^{old} - \eta \nabla_w L(w^{old}). \tag{4.2}$$

In stochastic gradient descent, for each update we are using only one term of the loss function, that means we use only one data point on each iteration.

Algorithm 2 SGD

$T \leftarrow$ Number of epochs ▷ The number of times we go through the entire dataset
$w \leftarrow$ Initialize (randomly)
for $epoch = 1..T$ **do**
  for $n = 1..N$ **do**
    $i \leftarrow$ Sample $i \sim U\{1, \ldots, N\}$
    $w^{new} = w^{old} - \eta \nabla_w L^{(i)}(w^{old})$
  end for
  You might want to update the learning rate η
end for

From the next statement it follows that the stochastic gradient is an unbiased estimator for the full-batch gradient.

Proposition 4.1

If we have a function represented as a sum of functions

$$f(w) = \sum_{i=1}^{N} f_i(w), \tag{4.3}$$

then the function $\hat{f}(\mathcal{I};\ w) = N f_{\mathcal{I}}(w)$ is an unbiased estimator of $f(w)$. Here we assume that the random variable $\mathcal{I}$ can take any of the values $1, \ldots, N$ with probability $1/N$.

In the above statement, the assumption is that the only randomness is contained in the index $\mathcal{I}$, which has a uniform distribution on the set $\{1, 2, \ldots, N\}$. Then expected value of $Nf_{\mathcal{I}}(w)$ is

$$E\big[Nf_{\mathcal{I}}(w)\big] = \sum_{i=1}^{N} \frac{1}{N} N f_i(w) = f(w). \tag{4.4}$$

To make it more precise, consider the data $\big(x^{(1)}, y^{(1)}\big), \ldots, \big(x^{(N)}, y^{(N)}\big)$ and function

$$L(w) = \sum_{i=1}^{N} L^{(i)}(w) = \sum_{i=1}^{N} \Big(y^{(i)} - w^T x^{(i)}\Big)^2. \tag{4.5}$$

Consider a random variable $\mathcal{I}$ uniformly distributed on $\{1, 2, \ldots, N\}$ and the estimator

$$\hat{L}(\mathcal{I};\ w) = NL^{(\mathcal{I})} = N\Big(y^{(\mathcal{I})} - w^T x^{(\mathcal{I})}\Big)^2 \tag{4.6}$$

with corresponding estimate

$$\hat{L}(\hat{\imath};\ w) = N\Big(y^{(\hat{\imath})} - w^T x^{(\hat{\imath})}\Big)^2, \qquad \hat{\imath} \sim U\{1, 2, \ldots, N\}. \tag{4.7}$$

Then its expected value is equal to

$$E[\hat{L}(\mathcal{I};\ w)] = \sum_{i=1}^{N} \frac{1}{N} N\Big(y^{(i)} - w^T x^{(i)}\Big)^2 = L(w). \tag{4.8}$$

4.2 Mini-Batch SGD

In the gradient descent algorithm, we use all the data points for one step, in the stochastic gradient descent, we use one point at a time. There is a compromise solution: we can use some amount M of data points for one step. This approach is known as *mini batch SGD.*

$$Loss_{SGD} = \Big(a\big(x^i\big) - y^{(i)}\Big)^2$$

$$Loss_{mini-batch} = \frac{1}{M} \sum_{i=1}^{M} \Big(a\big(x^i\big) - y^{(i)}\Big)^2$$

If we minimize

$$Loss_{batch} = \frac{1}{N} \sum_{i=1}^{N} \Big(a\big(x^i\big) - y^{(i)}\Big)^2,$$

it is called full-batch SGD or simply gradient descent (GD).

The theorems about convergence of SGD and Minibatch SGD can be found in Sec. 5 and 6 in https://arxiv.org/abs/2301.11235.

Algorithm 3 Minibatch SGD

$T \leftarrow$ Number of epochs ▷ The number of times we go through the entire dataset
$M \leftarrow$ Batch size
$w \leftarrow$ Initialize (randomly)
for $epoch = 1..T$ **do**
  You may shuffle the data
  for $batch = 1..N/M$ **do**
    $L_{batch} \leftarrow \frac{1}{M} \sum_{i=(batch-1)M}^{batch \cdot M} L^{(i)}$
    $w^{new} \leftarrow w^{old} - \eta \nabla_w L_{batch}(w^{old})$
  end for
  You might want to update the learning rate η
end for

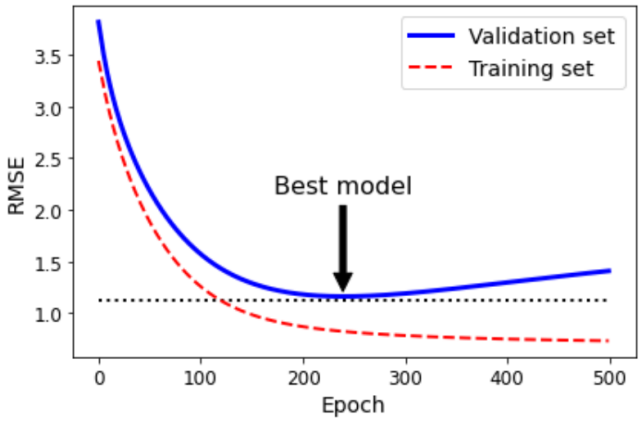


Figure 4.1: Typical plots for training and validation loss

4.3 Early Stopping

In SGD algorithm, the iterations go on for the number of epochs. But we can check the performance of our model on the validation set. Typically, it will be improving, but at some point it will grow up again (sii Fig. 4.1).[1] This is the time to stop training the model, because it will fit the training data more accurately, but loose the performance on the unseen data. This is called *memorization* of the data or *overfitting.*

4.4 Regularization

The idea of regularization appeared in the problem of inversion of matrices. For example, a matrix A with all zero elements is not invertible, but $A + \varepsilon I$ will be invertible if ε is greater than the machine's epsilon.

[1] https://github.com/ageron/handson-ml2/blob/master/04_training_linear_models.ipynb

Consider the following model for the apartments price prediction:

$$a(x) = 10 + 100 \cdot number\_rooms. \tag{4.9}$$

It can work correctly for one bedroom apartments, but it will give an unreasonable increment if the number of bedrooms is changed.

We can add a so called *regularization term* to our loss function, which will force the weights to be smaller.

In **ridged regression** we add ℓ_2 norm of the weights:

$$Loss + \alpha\|w_{-0}\|_2^2 \to min$$

In **Lasso regression**[2] ℓ_1 norm of the weights is used:

$$Loss + \beta\|w_{-0}\|_1 \to min$$

Elastic regression combines both ℓ_2 and ℓ_1 terms:

$$Loss + \alpha\|w_{-0}\|_2^2 + \beta\|w_{-0}\|_1 \to min$$

Usually, the bias term w_0 is not included, i.e., $\|w_{-0}\|_1 = |w_1| + \ldots + |w_d|$ and $\|w_{-0}\|_2^2 = w_1^2 + \ldots + w_d^2$. Otherwise, the regression line (or plain) would go through the origin.

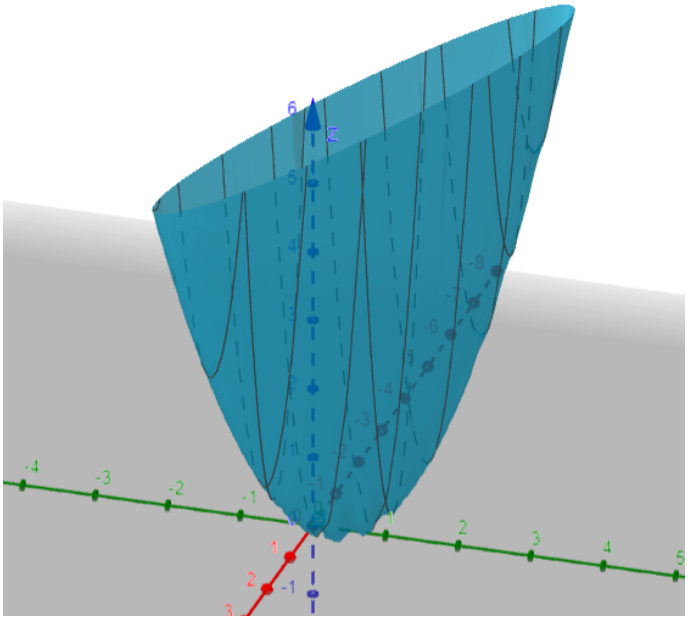
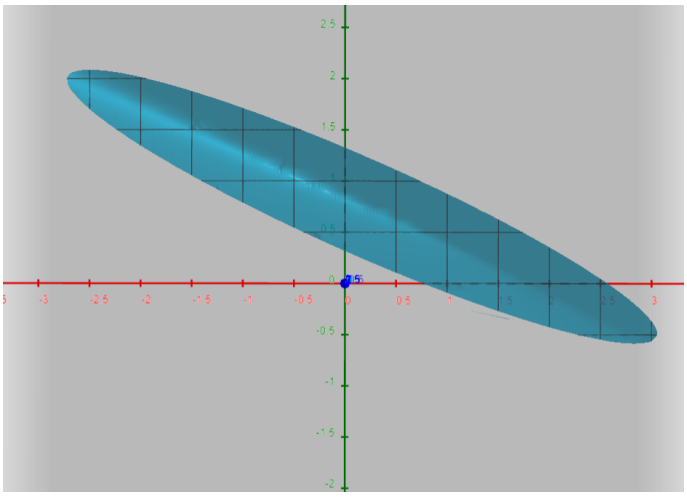
See class notebook for the choice of the regularization and parameters α and β.

4.5 Excersises

Problem 3. The plots of the SSR for Problem 2 is equal to

$$L(w_0, w_1) = L_1(w_0, w_1) + L_2(w_0, w_1) + L_3(w_0, w_1).$$

[2] From Least Absolute Shrinkage and Selection Operator. See the next chapter for the explanation of this name.

Problem 3. Apply one epoch of SGD. Initialize $w_0^{old} = w_1^{old} = 0$.

Calculate the derivatives of $L_1(w_0, w_1) = \left(y^{(1)} - w_0 - w_1 x^{(1)}\right)^2$ (at the first point):

$\frac{\partial L_1}{\partial w_0} =$

$\frac{\partial L_1}{\partial w_1} =$

Update the weights (use the learning rate $\eta = 1$):

$w_0^{new} = w_0^{old} - \eta \frac{\partial L_1}{\partial w_0} =$

$w_1^{new} =$

Calculate the derivatives of $L_2(w_0, w_1) = \left(y^{(2)} - w_0 - w_1 x^{(2)}\right)^2$:

$\frac{\partial L_2}{\partial w_0} =$

$\frac{\partial L_2}{\partial w_1} =$

Update the weights:

$w_0^{new} =$

$w_1^{new} =$

Calculate the derivatives of $L_3(w_0, w_1) = \left(y^{(3)} - w_0 - w_1 x^{(3)}\right)^2$:

$\frac{\partial L_3}{\partial w_0} =$

$\frac{\partial L_3}{\partial w_1} =$

Update the weights:

$w_0^{new} =$

$w_1^{new} =$

Plot each point in the figure on the right.

Problem 4. Can you think of how to improve convergence?

5 Model Selection

By *model selection* we usually mean the choice of the model including hyper-parameters. For example, in case of regression we can use kNN regression and tune the number of neighbors k, we can try the ridged regression with loss function

$$L(w) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - w^T\tilde{x}^{(i)}\right)^2 + \alpha\|w\|_2^2. \tag{5.1}$$

Then we should tune the parameter α.

This can be done by *grid search:* we train the model for different values of parameters and compare the error on the validation data set or by cross-validation. We create a set of values for parameter $\alpha : \alpha_1,\ \alpha_2, \ldots, \alpha_{S_1}$, a set of values for another parameter: $\beta_1,\ \beta_2, \ldots, \beta_{S_2}$ and so on. All possible combinations of those values create a grid.

Sometimes, the choice of features is also considered as part of the model selection.

5.1 Feature Selection

5.1.1 By Criterion

We can use correlation

$$\mathrm{Corr}(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}} \tag{5.2}$$

or mutual information (MI)[1]

$$\mathrm{I}(X, Y) = \mathrm{D}_{\mathrm{KL}}(p(x, y)||p(x)p(y)) \tag{5.3}$$

of each feature with the target and take the features with the criterion value higher than a predefined threshold.

You can try different thresholds by using a model and evaluating the error on the validation set.

Mutual information works better because correlation indicates linear dependence only.

[1] The Kullback — Leibler divergence measure the discrepancy between two distributions. Therefore, mutual information shows the strength of dependence of two random variables. It is zero, when they are independent, i.e., when $p(x, y) = p(x)p(y)$.

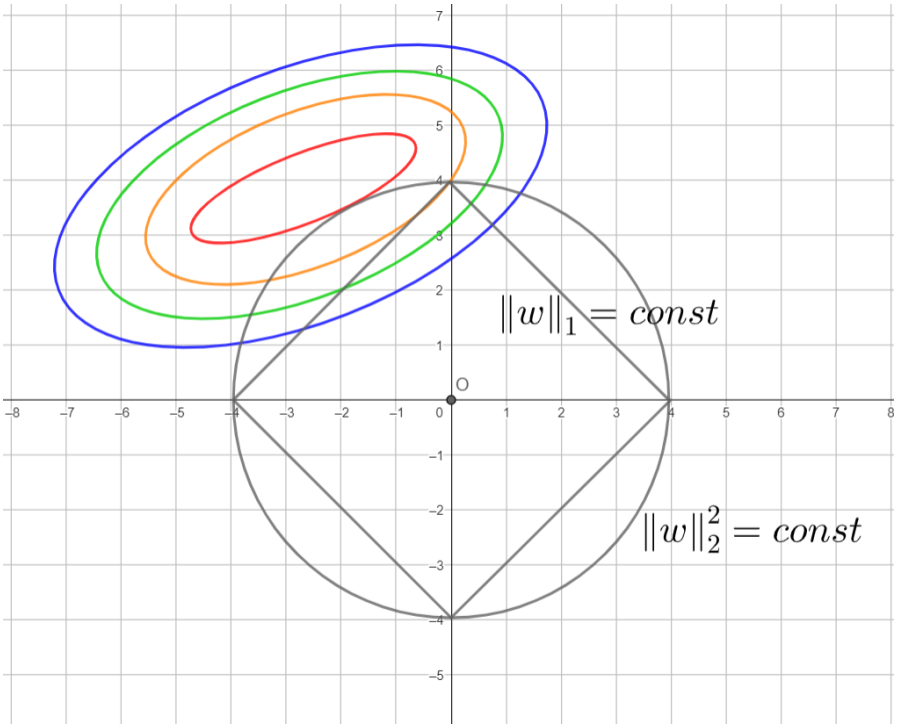


Figure 5.1: Counter plots of the loss function and ℓ_1 and ℓ_2 regularization terms with the same value

5.1.2 By Model

Lasso regularization is sometimes referred to as a *sparse* regression:

$$L(w) = MSE(w) + \alpha\|w\|_1. \tag{5.4}$$

If the position of the MSE loss function is not symmetric around the origin (see Fig. 5.1), then for a fixed value of the regularization term

$$\alpha\|w\|_1 = C_2, \tag{5.5}$$

its minimum may lie in the corner of the ℓ_1 ball (which has a constant value C_2 on its boundary). Assume that the loss function has some value C :

$$L(w) = MSE(w) + \alpha\|w\|_1 = C_1 + C_2 = C. \tag{5.6}$$

If we consider the ℓ_2 regularization, because the ℓ_2 ball is symmetric in any direction, typically, the minimum will be reached at some point with non-zero w_1 and w_2 components. In case of the ℓ_1 ball, the minimum will be reached with $w_1 = 0$. This means that the feature with index 1 is not important and can be omitted.

We can train Lasso regression and choose the features with the highest absolute values of the weights.

5.1.3 By (Recursive) Elimination

In this approach, we try all possible combinations of r features. The number r can be chosen by the model error on the validation set.

6 Linear Models for Binary Classification

Assume that we are given points $\left(x^{(i)},\ y^{(i)}\right)$, $i = 1,\ 2,\ \ldots,\ N$ (see Fig. 6.1), and $y^{(i)} \in \{-1,\ 1\}$. In 3D, those points will lie in the plane $z = -1$ for one class and on the planes $z = \pm 1$ (see Fig. 6.2).

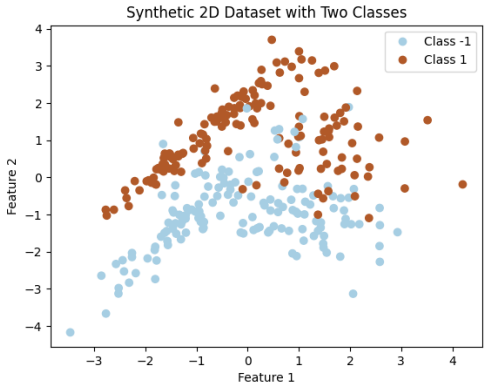


Figure 6.1: Caption

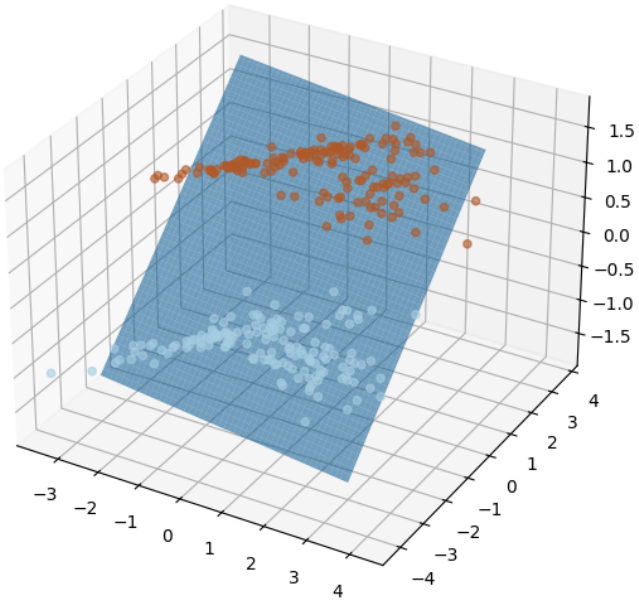


Figure 6.2: Caption

We have a linear model

$$a(x) = w_0 + w_1 x_1 + w_2 x_2.$$

which returns values from $(-\infty,\ \infty)$. We can modify it to return -1 or 1:

$$a(x) = \text{sign}(w_0 + w_1 x_1 + w_2 x_2) = \text{sign}\, w^T \tilde{x} = \text{sign}\langle w, \tilde{x}\rangle, \qquad \tilde{x} = (1, x_1, x_2). \tag{6.1}$$

6.1 Geometry of Linear Classifier

Equation

$$w^T \tilde{x} = 0 \tag{6.2}$$

defines a line in the plain $z = w^T \tilde{x}$. If we look at this figure from above, the projection of this line on the feature space will look like a separator. Every point below this line is classified as class -1 and every point above is classified as class $+1$.

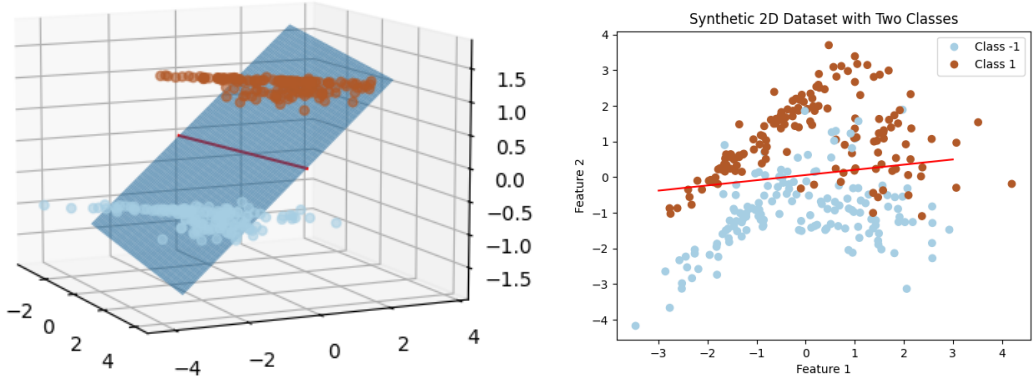


Figure 6.3: Threshold is set to 0

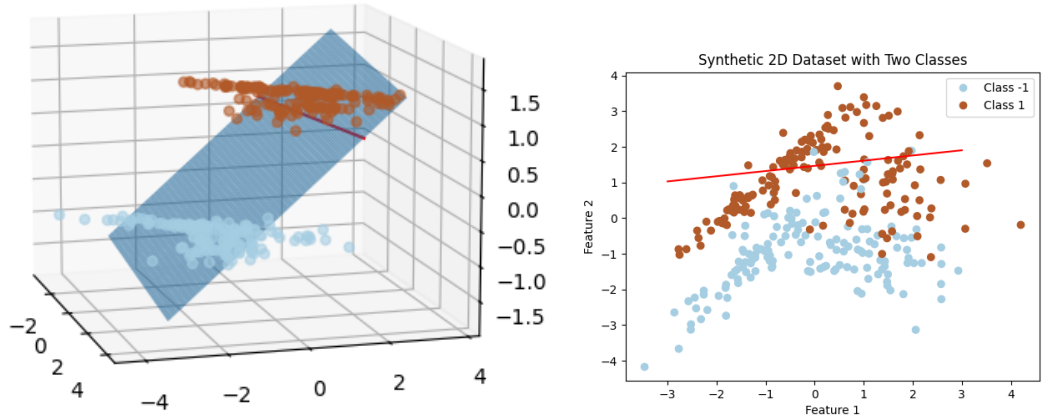


Figure 6.4: Threshold is set to 0.7

We can consider a different equation:

$$w^T \tilde{x} = th, \tag{6.3}$$

that corresponds to a change of the bias term $w_0' = w_0 - th$:

$$w_0 - th + w_1 x_1 + x_2 x_2 = 0. \tag{6.4}$$

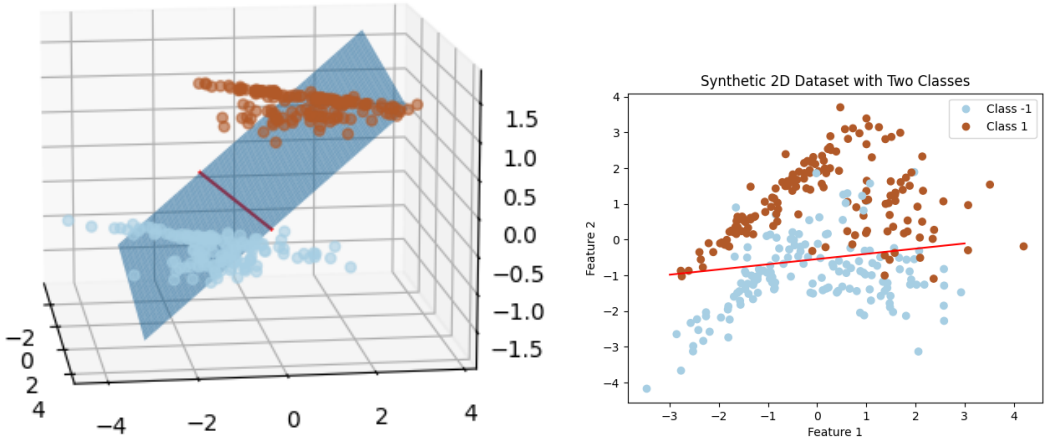
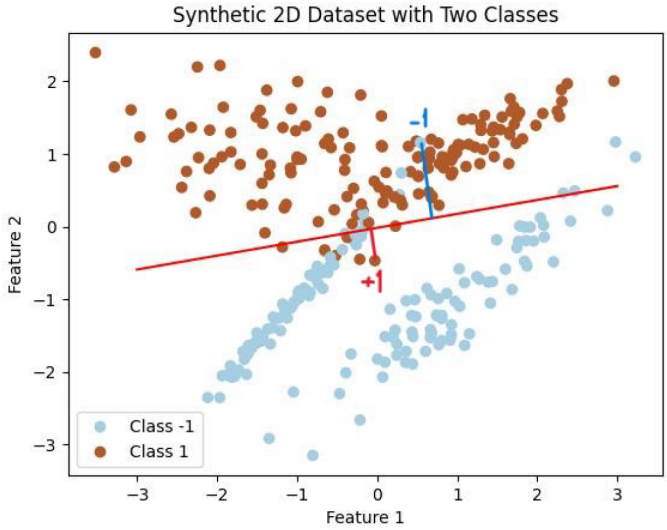


Figure 6.5: Threshold is set to -0.3

6.2 Margin

The distance from a point $(x_1,\ x_2)$ above the line to the line is given by (see Problem 4)

$$d = \frac{w_0 + w_1 x_1 + w_2 x_2}{\sqrt{w_1^2 + w_2^2}} = \frac{\langle w,\ \tilde{x} \rangle}{\sqrt{w_1^2 + w_2^2}},$$



By **margin** of $x^{(i)}$ we mean the distance multiplied by the sign $y^{(i)}$:

$$y\frac{\langle w,\ \tilde{x}\rangle}{\sqrt{w_1^2+w_2^2}}, \tag{6.5}$$

If $y=-1$ and $\langle w,\ \tilde{x}\rangle > 0$, then $y\langle w,\ \tilde{x}\rangle < 0$
If $y=-1$ and $\langle w,\ \tilde{x}\rangle < 0$, then $y\langle w,\ \tilde{x}\rangle > 0$

If $y=1$ and $\langle w,\ \tilde{x}\rangle < 0$, then $y\langle w,\ \tilde{x}\rangle < 0$
If $y=1$ and $\langle w,\ \tilde{x}\rangle > 0$, then $y\langle w,\ \tilde{x}\rangle > 0$

For the correct classification the margin is positive. That is why sometimes we call by **margin** the value

$$y\langle w,\ \tilde{x}\rangle, \tag{6.6}$$

which is positive for correct predictions and negative otherwise.

6.3 Support Vector Machine (SVM)

In this section, we use a slightly different notation: $w = (w_1, w_2)$ instead of $w = (w_0, w_1, w_2)$.

6.3.1 Separable Case

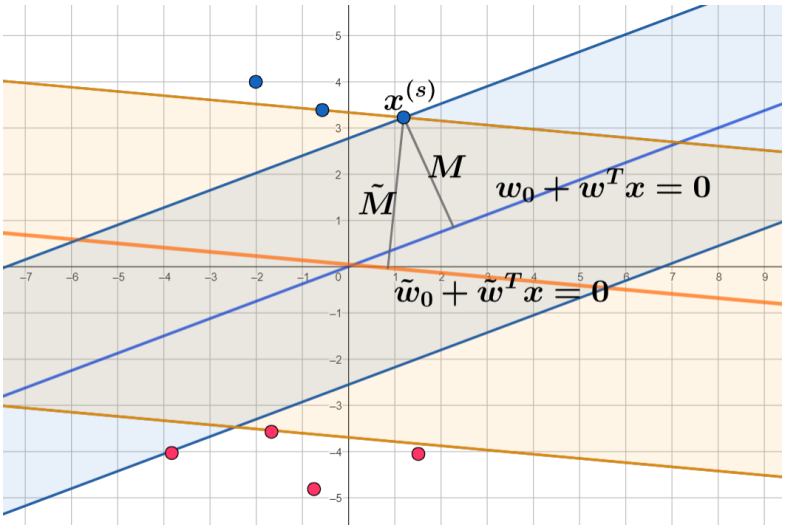


Figure 6.6: Two different separatos

In Fig. 6.6, there are two separating lines:

$$w_0 + w^T x = 0 \qquad \text{and} \qquad \tilde{w}_0 + \tilde{w}^T x = 0. \tag{6.7}$$

If we move the separating line up, at some point we will reach a point from our data and we won't be able to move the line further. The points that do not allow to move the separating line are called **supporting** points (vectors).

For a separable case the margin should be positive:

$$\frac{y^{(i)}\big(w_0 + w^T x^{(i)}\big)}{\|w\|} > 0, \qquad i = 1, \dots, N. \tag{6.8}$$

We assume that the best separator has the maximal possible margin to for the closest point from the decision line:

$$M(w_0, w) = \min_{i=1,\dots,N} \frac{y^{(i)}\big(w_0 + w^T x^{(i)}\big)}{\|w\|} = \frac{1}{\|w\|} \min_{i=1,\dots,N} y^{(i)}\big(w_0 + w^T x^{(i)}\big) \to \max_{w_0, w}. \tag{6.9}$$

In addition, we assume the normalization constrain: for the support boundary (the line containing supporting points) equation looks as[1]

$$w_0 + w^T x = 1. \tag{6.10}$$

Under this assumption problem (6.9) takes form

$$M(w_0, w) = \frac{1}{\|w\|} \to \max. \tag{6.11}$$

From normalization condition (6.11):

$$\min_{i=1,\dots,N} y^{(i)}\big(w_0 + w^T x^{(i)}\big) = 1. \tag{6.12}$$

Thus, we have the constrained optimization problem

$$\begin{cases} M(w_0, w) \to \max, \\ y^{(i)}\big(w_0 + w^T x^{(i)}\big) \geq 1, \qquad i = 1, \dots, N. \end{cases} \tag{6.13}$$

or, equivalently,

$$\begin{cases} \frac{1}{2}\|w\|^2 \to \min\limits_{w_0, w}, \\ y^{(i)}\big(w_0 + w^T x^{(i)}\big) \geq 1, \qquad i = 1, \dots, N. \end{cases} \tag{6.14}$$

[1] For fixed separator $w_0 + w^T x = 0$ this scaling can be done by division by $w_0 + w^T x^{(s)}$: $\tilde{w}_0 + \tilde{w}^T x = 0$, with $\tilde{w}_0 = \dfrac{1}{w_0 + w^T x^{(s)}}$ and $\tilde{w} = \dfrac{1}{w_0 + w^T x^{(s)}}$ describes the same line, but $\tilde{w}_0 + \tilde{w}^T x^{(s)} = 1$.

6.3.2 Non-Separable Case

We can allow for some points to lie inside the margin (see Fig. 6.7) by introducing non-negative *slack* variables:

$$\begin{cases} \frac{1}{2}\|w\|^2 \to \min\limits_{w_0,w}, \\ y^{(i)}\left(w_0 + w^T x^{(i)}\right) \geq 1 - \xi^{(i)}, \qquad i = 1, \ldots, N, \\ \xi^{(i)} \geq 0, \qquad i = 1, \ldots, N. \end{cases} \tag{6.15}$$

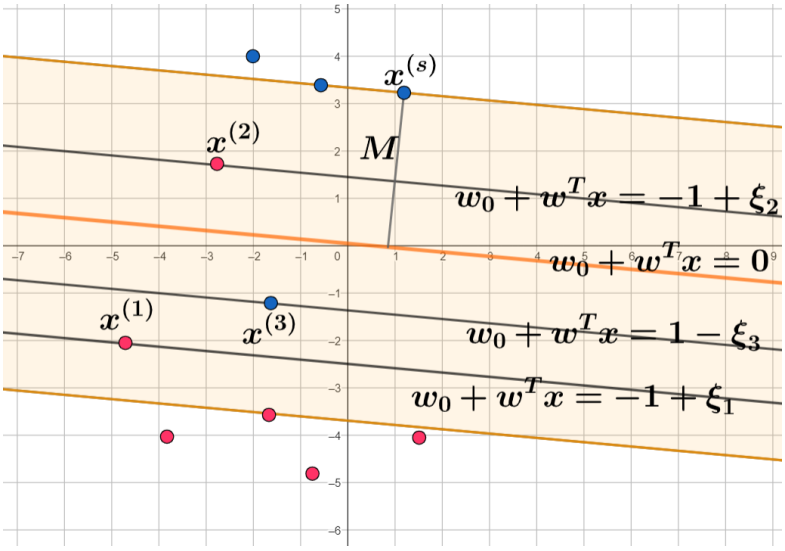


Figure 6.7: Separation with slack variables

But we can run into a problem like in Fig. 6.8, where decision boundary goes through the lowest point of one class and allows large negative margins for the other class. In order to avoid this, the additional term in the minimizing function is introduced:

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{N} \xi^{(i)} \to \min\limits_{w_0,w,\xi^{(i)}}, \\ y^{(i)}\left(w_0 + w^T x^{(i)}\right) \geq 1 - \xi^{(i)}, \qquad i = 1, \ldots, N, \\ \xi^{(i)} \geq 0, \qquad i = 1, \ldots, N. \end{cases} \tag{6.16}$$

It forces the slack variables to be as small controlling by the constant $C > 0$.

To solve this problem KKT theorem will be used.

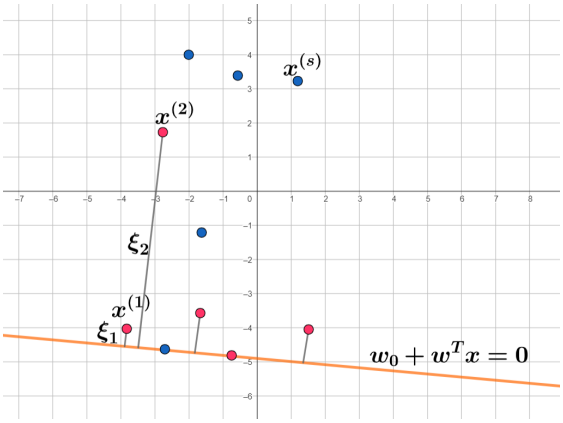


Figure 6.8: Separation without minimizing the slack variables

6.3.3 Karush — Kuhn — Tucker Theorem

Primal Problem

$$\begin{cases} f(x) \to \min\limits_{x}, \\ g_i(x) \leq 0, \\ h_j(x) = 0. \end{cases} \tag{6.17}$$

The Dual Problem

$$\begin{cases} \inf\limits_{x} \mathcal{L}(x,\ \lambda,\ \mu) \to \max\limits_{\lambda,\mu \geq 0}, \\ \sum\limits_{i} \lambda_i g_i(x) = 0. \end{cases} \tag{6.18}$$

where $\mathcal{L}$ is called a Lagrangian:

$$\mathcal{L}(x,\ \lambda,\ \mu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x). \tag{6.19}$$

If the functions f, g_i, and h_j are convex and there exists a point $x^{interior}$ satisfying the conditions given by inequalities and equalities, then the saddle point of the dual problem gives the solution to the primal problem and the slackness condition is satisfied:

$$\sum_i \lambda_i g_i(x) = 0.$$

We apply the Karush — Kuhn — Tucker theorem to problem (6.16):

$$\mathcal{L}(w_0, w, \xi, \lambda, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi^{(i)} - \sum_{i=1}^{N} \lambda_i \left(y^{(i)} \left(w_0 + w^T x^{(i)} \right) - 1 + \xi^{(i)} \right) - \sum_{i=1}^{N} \mu_i \xi^{(i)} \tag{6.20}$$

Calculate the derivatives:

$$\frac{\partial \mathcal{L}}{\partial w_0} = -\sum_{i=1}^{N} \lambda_i y^{(i)} = 0, \tag{6.21}$$

$$\frac{\partial \mathcal{L}}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y^{(i)} x^{(i)} = 0, \tag{6.22}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \quad \Rightarrow \quad \lambda_i + \mu_i = C. \tag{6.23}$$

And slackness conditions

$$\lambda_i \left(y^{(i)} \left(w_0 + w^T x^{(i)} \right) - 1 + \xi_i \right) = 0, \tag{6.24}$$

$$\mu_i \xi_i = 0. \tag{6.25}$$

1. $\lambda_i = 0 \xRightarrow{(6.23)} \mu_i = C \xRightarrow{(6.25)} \xi^{(i)} = 0 \Rightarrow x^{(i)}$ doesn't influence separation, because not included in (6.22).

2. $0 < \lambda_i < C \xRightarrow{(6.23)} \mu_i \neq 0 \Rightarrow \xi^{(i)} = 0$. (Or $\lambda_i = C$ and $\xi^{(i)} = 0$.) Therefore,

$$\left(y^{(i)} \left(w_0 + w^T x^{(i)} \right) - 1 \right) = 0$$

   and $x^{(i)}$ is a support vector.

3. $\lambda_i = C$ and $\xi^{(i)} > 0$, then $x^{(i)}$ is on the other side of the support boundary $w_0 + w^T x^{(i)} = \pm 1$.

From equality (6.22) it follows that problem (6.16) can be reduced to the following quadratic optimization problem in a cube with side C:

$$\begin{cases} \frac{1}{2} \sum_{i,j=1}^{N} \lambda_i \lambda_j y^{(i)} y^{(j)} \left\langle x^{(i)}, x^{(j)} \right\rangle \to \max\limits_{\lambda_i}, \\ 0 \leq \lambda_i \leq C, \quad i = 1, \ldots, N, \\ \sum\limits_{i=1}^{N} \lambda_i y^{(i)} = 0. \end{cases} \tag{6.26}$$

There are efficient solvers for such problems. To determine w_0 one can use equation (6.24).

For smaller values of C the model can make larger errors allowing bigger $\xi^{(i)}$ (see Fig. 6.9-6.11)

6.3.4 Minimization with a Hinge Loss

The other way to train SVM is to use SGD with so called *hinge loss*

$$L(M) = \max(0, 1 - M) \tag{6.27}$$

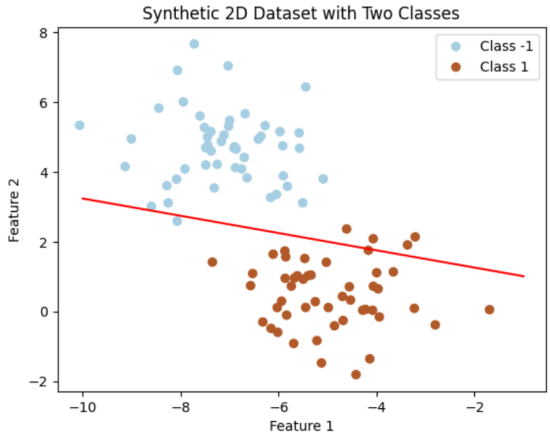


Figure 6.9: $C = 0.1$

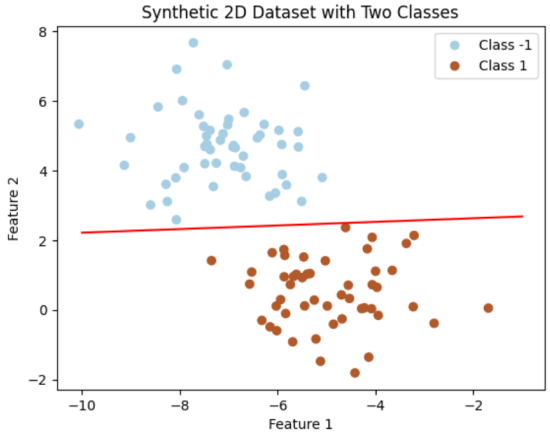


Figure 6.10: $C = 1$

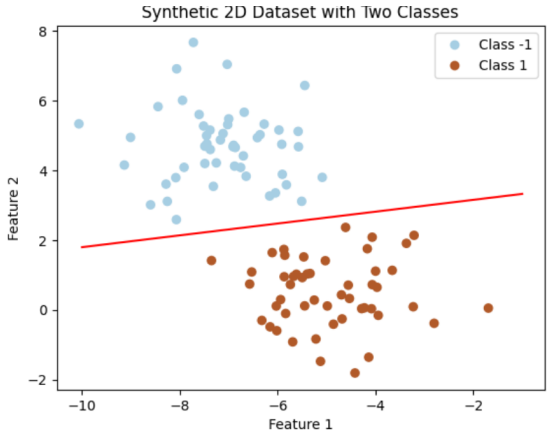


Figure 6.11: $C = 5$

with ℓ_2 regularization:

$$C\sum_{i=1}^{N}\max\left\{0,\ 1-y^{(i)}\left(w_0+\langle w,\ x^{(i)}\rangle\right)\right\}+\frac{1}{2}\|w\|^2\to\min_{w_0,w}. \tag{6.28}$$

It follows from the second constrain in (6.16) that

$$\xi^{(i)}\geq 1-y^{(i)}\left(w_0+\langle w,\ x^{(i)}\rangle\right) \tag{6.29}$$

And because $\xi^{(i)}\geq 0$,

$$\xi^{(i)}\geq\max\left\{0,\ 1-y^{(i)}\left(w_0+\langle w,\ x^{(i)}\rangle\right)\right\}. \tag{6.30}$$

The result of minimization of the hinge loss (6.28) is different from the solution of problem (6.26), but as practice shows, this difference is small.

6.4 F. Rosenblatt's Perceptron (1962)

C. Bishop, Pattern Recognition and Machine Learning, Sec. 4.1.7
https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

6.4.1 Perceptron Algorithm

$$Loss=\frac{1}{N}\sum_{i=1}^{N}\Big[\text{sign }a(x^{(i)})\neq y^{(i)}\Big]=\frac{1}{N}\sum_{i=1}^{N}\Big[y^{(i)}\langle w,x^{(i)}\rangle<0\Big]$$

Perceptron:

$$Loss=\sum_{i:y^{(i)}\langle w,x^{(i)}\rangle<0}y^{(i)}\langle w,x^{(i)}\rangle<0\to\max$$

or

$$Q(w)=-\sum_{i:y^{(i)}\langle w,x^{(i)}\rangle<0}y^{(i)}\langle w,x^{(i)}\rangle<0\to\min.$$

$$Q(w)=-\sum_{i:y^{(i)}\langle w,x^{(i)}\rangle<0}y^{(i)}\langle w,x^{(i)}\rangle<0\to\min.$$

SGD implies the update, if ith point is missclassified:

$$w^{new}=w^{old}+y^{(i)}x^{(i)}$$

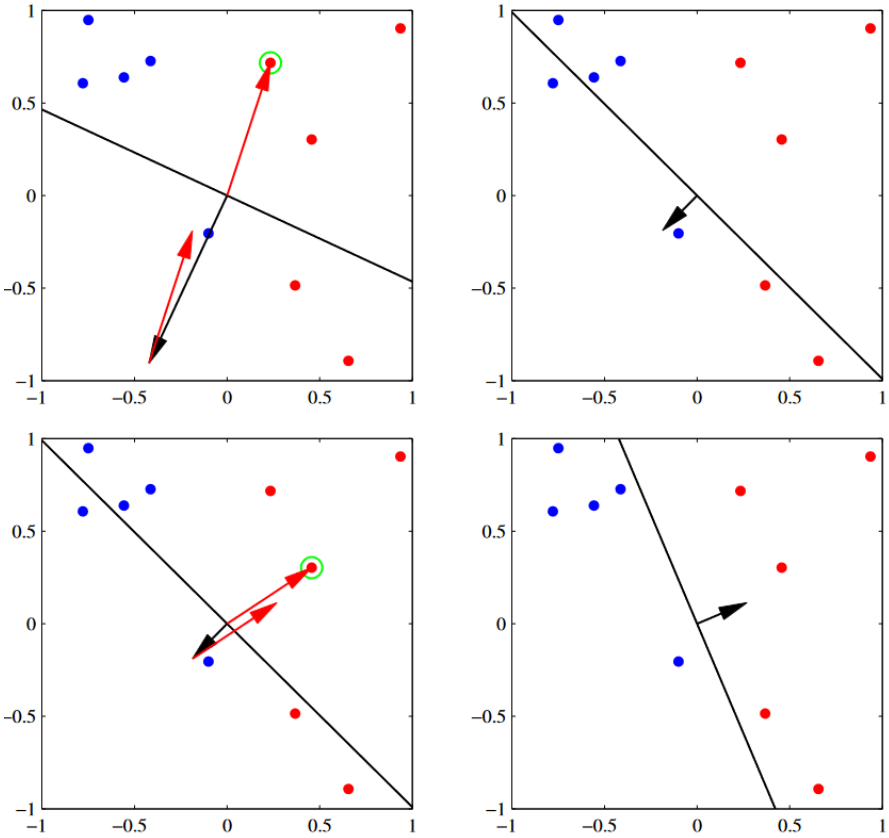


Figure 6.12: Perceptron updates

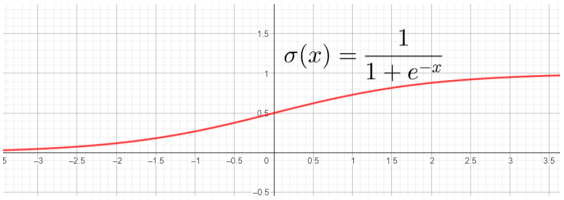


Figure 6.13: Sigmoid function

6.5 Logistic Regression

6.5.1 Sigmoid Function

In previous sections, we introduced the notion of the distance with a sign:

$$d(x) = \frac{w_0 + w^T x}{\|w\|}. \tag{6.31}$$

Let's consider a function that maps $(-\infty,\ +\infty)$ onto the interval $(0, 1)$:

$$\sigma \colon (-\infty,\ +\infty) \to (0, 1) \tag{6.32}$$

$$\sigma(d) = \frac{e^d}{1+e^d} = \frac{1}{1+e^{-d}} \tag{6.33}$$

Then we can interpret this value as probability of the positive class:

- For large values of $d > 0$ $\sigma(d) \approx 1$.
- For $d = 0$, $\sigma(d) = 0.5$.
- For large negative d, $\sigma(d) \approx 0$.

But for now, this transform doesn't have any probabilistic model behind it.
We can solve equation (6.33) with respect to d :

$$d = \log \frac{\sigma}{1-\sigma}. \tag{6.34}$$

Because σ represents the probability of the positive class, this value is the log ratio of the probabilities of two classes. It is known as log-odds or **logit.**

6.5.2 Probabilistic Model

We are given the data $\left(x^i, y^{(i)}\right)$, where $y^{(i)} \in \{-1, +1\}$. For every x consider a random variable Y that takes two values: $+1$ with probability $p(x)$ and -1 with probability $1-p(x)$. And assume

Y	-1	1
$P(Y)$	$1-p$	p

Table 6.1: Variant of Bernoulli random variable

that this probability is given by[2]

$$p(x) = \sigma(w_0 + w_1 x_1 + \ldots + w_d x_d). \tag{6.35}$$

If the margin is positive and $y = 1$,

$$P(Y = 1; x,\ w) = \sigma(\langle w,\ \tilde{x}\rangle) = \frac{1}{1 + e^{-\langle w,\ \tilde{x}\rangle}} = \frac{1}{1 + e^{-y \cdot \langle w,\ \tilde{x}\rangle}}.$$

If the margin is positive, but $y = -1$, we have

$$P(Y = -1; x,\ w) = 1 - \sigma(\langle w,\ \tilde{x}\rangle) = 1 - \frac{1}{1 + e^{-\langle w,\ \tilde{x}\rangle}} = \frac{1}{1 + e^{\langle w,\ \tilde{x}\rangle}} = \frac{1}{1 + e^{y \cdot (-\langle w,\ \tilde{x}\rangle)}}.$$

The model takes the form as in Table 6.2. The features x are assumed to be known, the

Y	-1	1
$P(Y)$	$\sigma(-\langle w,\ \tilde{x}\rangle)$	$\sigma(\langle w,\ \tilde{x}\rangle)$

Table 6.2: Logistic regression model

weights w are parameters that should be tuned.

6.5.3 Maximum Likelihood

The Likelihood function is defined as follows:

$$L(y^{(1)},\ y^{(2)}, \ldots,\ y^{(N)}; \mathbf{X},\ w) = \prod_{i=1}^{N} P\left(Y^{(i)} = y^{(i)}; x^{(i)}, w\right) \to \max_w \tag{6.36}$$

Using the representation of probabilities from Table 6.2, the likelihood takes form

$$L(w) = \prod_{i=1}^{N} \sigma\left(y^{(i)} \langle w,\ x^{(i)}\rangle\right) = \prod_{i=1}^{N} \frac{1}{1 + e^{-y^{(i)} \langle w,\ x^{(i)}\rangle}} \to \max_w. \tag{6.37}$$

We can maximize its logarithm instead:

$$\log L(w) = \sum_{i=1}^{N} \log \sigma\left(y^{(i)} \langle w,\ x^{(i)}\rangle\right) = -\sum_{i=1}^{N} \log\left(1 + e^{-y^{(i)} \langle w,\ x^{(i)}\rangle}\right) \to \max_w, \tag{6.38}$$

or minimize the negative log-loss:

$$-\log L(w) = \sum_{i=1}^{N} \log\left(1 + e^{-y^{(i)} \langle w,\ x^{(i)}\rangle}\right) \to \min_w. \tag{6.39}$$

[2] This is a particular case of generalized linear models (GLM), when we assume that the mean value is represented by a function of a linear function. If the labels were 0 and 1 (instead of -1 and 1), then p represents the expected value of the Bernoulli random variable.

Definition 6.1

The **binary cross-entropy loss**

$$Loss = \frac{1}{N}\sum_{i=1}^{N} \log\left(1 + e^{-y^{(i)}\langle w,\ x^{(i)}\rangle}\right) \tag{6.40}$$

is typically used in classification tasks.

The definition of entropy, cross-entropy, and KL-divergence are as follows

$$H(p) = -\sum_{i=1}^{K} p_i \log p_i, \qquad H(p,q) = -\sum_{i=1}^{K} p_i \log q_i, \qquad D_{KL}(p,q) = \sum_{i=1}^{K} p_i \log \frac{q_i}{p_i}. \tag{6.41}$$

If follows that

$$H(p,q) = H(p) - D_{KL}(p,q). \tag{6.42}$$

Therefore, the minimization of the negative cross-entropy is equivalent to the minimization of KL-divergence (we assume that the distribution p is given).

6.5.4 Other Metrics for Logistic Regression: Deviance, AIC, BIC

$$D(w) = -2(\log L(w) - \log L(w_s)) = -2\log L(w)$$

$$AIC = -\frac{2}{N}\log L + \frac{2d}{N}$$

$$BIC = -2\log L + d\log N$$

6.6 Classification Metrics

$$\sum_{i=1}^{N}\left[a(x^{(i)} \neq y^{(i)})\right] \to min$$

$$Accuracy = \sum_{i=1}^{N}\left[a(x^{(i)} = y^{(i)})\right] \to max$$

This metric depends on the balance of classes. For example, consider $N = 100$ data points with 95 points labeled $+1$ and 5 points labeled -1. If our model predicts every point as positive class, then the accuracy will be 95%.

Therefore, if q is the fraction of the largest class, we want

$$Accuracy \in [q,\ 1].$$

Another way to access the model's performance is the error matrix.

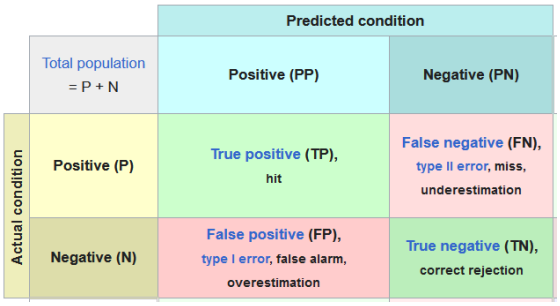


Figure 6.14: The matrix of errors

6.6.1 Confusion Matrix

From this matrix we can calculate accuracy as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6.43}$$

This matrix contains false positive (FP) and false negative (FN) errors.

Example (Medical exam). If the test is positive and the person is sick, we can look at the following fraction

$$presicion = \frac{TP}{TP + FP}.$$

This ratio says how accurate our model is when it predicts a positive class. If precision is 100%, it means that model predicts positive class without errors. If precision is 10%, it means that from 100 positive predictions, in 90 cases the person didn't have the disease.

The other type of errors arises when the model gives negative result, while the person is sick. To measure these errors, another metric is used:

$$recall/sensitivity/TPR = \frac{TP}{TP + FN} = \frac{TP}{Total\ positive} = \frac{TP}{P}.$$

It shows how well the model covers the positive class.

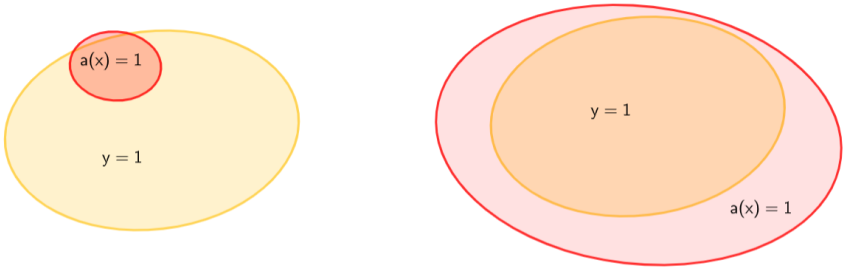


Figure 6.15: Large Precision VS Large Recall

6.6.2 F-score

If we want to combine the information about precision and recall in one metric, we can use so-called F_β-score.

$$F_\beta = (1+\beta)\frac{Precision \times Recall}{\beta(Precision + Recall)}, \quad \beta \in (0, \infty). \tag{6.44}$$

Small β optimises more Precision and larger β emphasizes Recall. Recall is considered β times more important than Precision (see Problem 7). Typically, $\beta = 1$ is used:

$$F_1 = 2\frac{Precision \times Recall}{(Precision + Recall)}. \tag{6.45}$$

6.6.3 Quality Improvement Reporting

old r_1	0.8	0.5	0.001
new r_2	0.9	0.75	0.01
Absolute $r_2 - r_1$	0.1	0.25	0.009
Relative $\frac{r_2 - r_1}{r_1}$	12%	50%	900%

6.7 Model Selection for Classification

6.7.1 Precision — Recall (PR) Curve

We can plot

6.7.2 Receiver Operating Characteristic (ROC) Curve

6.7.3 ROC-AUC

6.8 Exercises

Problem 1. Consider the line through the origin:

$$w_1 x_1 + w_2 x_2 = 0.$$

Prove that the vector $w = (w_1,\ w_2)^T$ is perpendicular to this line.

Problem 2. Prove that a distance from a point $(x_1,\ x_2)$ above the line to the line is given by

$$d = \frac{\langle w, x\rangle}{|w|} = \frac{w_1 x_1 + w_2 x_2}{\sqrt{w_1^2 + w_2^2}}.$$

Hint: by definition, $\langle \vec{a}, \vec{b}\rangle = |\vec{a}||\vec{b}|\cos\alpha$. Therefore, $\mathrm{Pr}_{\vec{a}}\,\vec{b} = |\vec{b}|\cos\alpha = \dfrac{\langle \vec{a}, \vec{b}\rangle}{|\vec{a}|}$.

Problem 3. The following value is called **margin**

$$y^{(i)} \left\langle w, x^{(i)} \right\rangle .$$

Prove that for correctly classified point $x^{(i)}$ the margin is positive and for an incorrectly classified point it's negative.

Problem 4. Assuming the separating line has a bias term

$$w_0 + w_1 x_1 + w_2 x_2 = 0,$$

prove that a distance from a point $(x_1,\ x_2)$ above the line to the line is given by

$$d = \frac{w_0 + \langle w, x \rangle}{|w|} = \frac{\langle \tilde{w}, \tilde{x} \rangle}{|w|} = \frac{w_0 + w_1 x_1 + w_2 x_2}{\sqrt{w_1^2 + w_2^2}},$$

where $w = (w_1, w_2)^T$, $\tilde{w} = (w_0, w_1, w_2)^T$, $x = (x_1, x_2)^T$, and $\tilde{x} = (1, x_1, x_2)^T$

Problem 5. Given the following predictions and threshold $th = 700$ draw the confusion matrix and calculate precision and recall

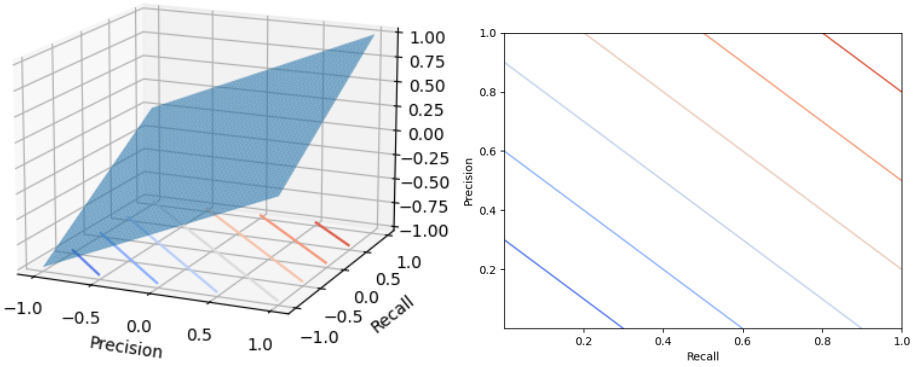
$a(x) = \langle w, x \rangle$	y	$\hat{y}$
1000	1	1
900	−1	1
800	1	1
700	1	−1
300	−1	−1
100	−1	−1
1	−1	−1
−10	1	−1
−200	−1	−1
−500	−1	−1

Problem 6. Given the following predictions and threshold $th = -200$ draw the confusion matrix and calculate precision and recall

$a(x) = \langle w, x \rangle$	y	$\hat{y}$
1000	1	1
900	−1	1
800	1	1
700	1	1
300	−1	1
100	−1	1
1	−1	1
−10	1	1
−200	−1	−1
−500	−1	−1

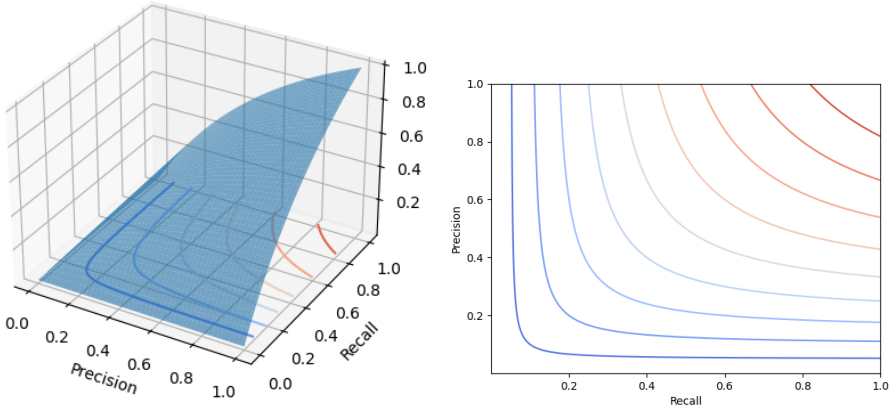
Problem 7. We can get better precision or recall by choosing the threshold. What if we want to combine them and maximize them together? Consider mean and harmonic mean. The surfaces and level curves are below. Why using mean is not a good idea?

$$\frac{1}{2}(Precision + Recall)$$



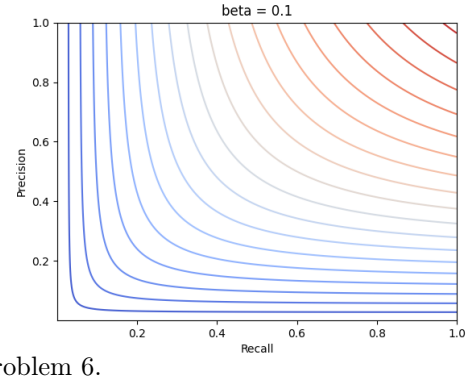
F-score

$$\frac{2(Precision \times Recall)}{Precision + Recall}$$



F_β-score. Small β optimises more Precision and larger β emphasises Recall. Recall is considered β times more important than Precision.

$$(1+\beta)\frac{Precision \times Recall}{\beta(Precision + Recall)}$$

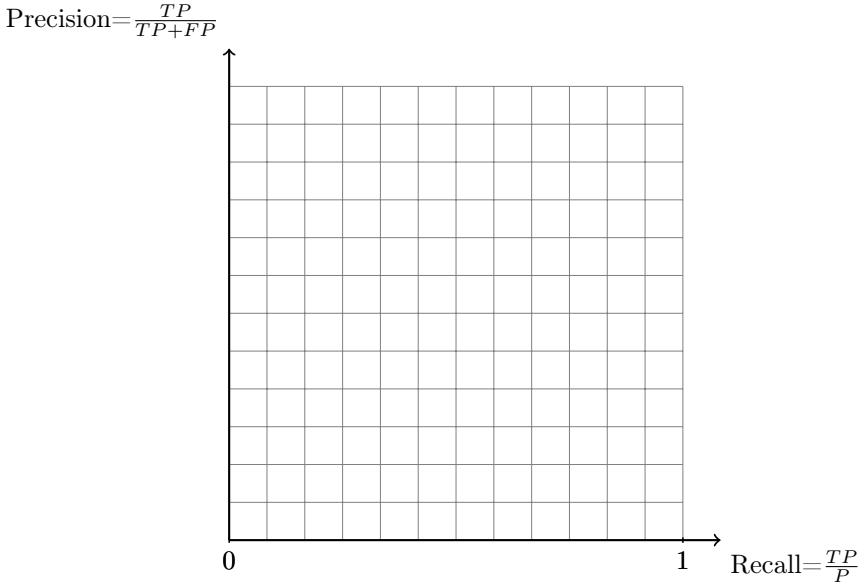


Problem 8. Calculate F_1-score for Problem 5 and Problem 6.

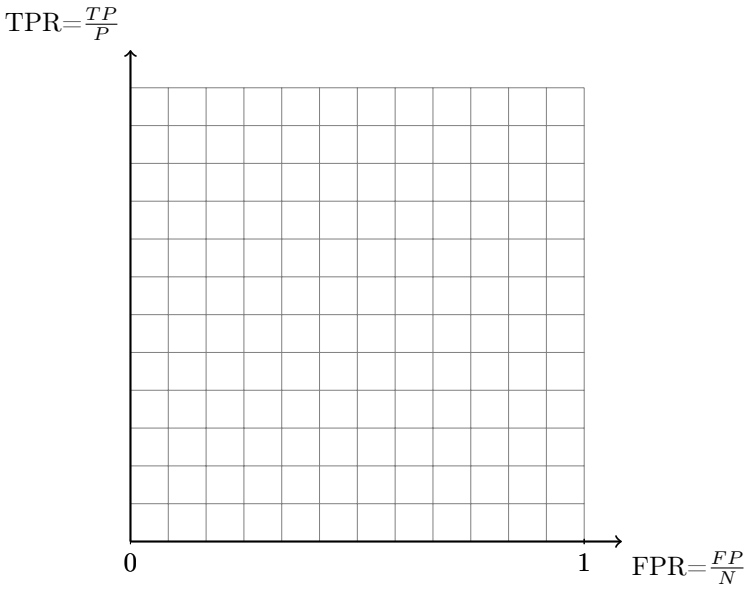
Problem 9. Given the following data

$\langle w, x \rangle$	0.45	−0.1	2	0.3	−0.5	0.7	0
y	+1	−1	+1	+1	−1	−1	+1

Plot the PR curve.



Problem 10. Using the data from Problem 9 plot ROC curve and calculate AUC.



AUC ROC =

7 Multiclass Classification

7.1 One-VS-All and All-VS-All

7.1.1 One - VS - all

We have
$\left(X^{(i)},\ y^{(i)}\right)$, where $y \in \mathbb{Y}$
$\mathbb{Y} = \{1,\ 2,\ \ldots,\ C\}$

- Train C models a_c on $\left(X^{(i)},\ \left[y^{(i)} = c\right]\right)$
- a_c returns distance $\langle w,\ x\rangle$ or probability
- The final model is

$$a(x) = \underset{c=1,2,\ldots,C}{\operatorname{argmax}} a_c(x).$$

7.1.2 All - VS - all

- For every c_1 and c_2 take $\left(X^{(i)},\ y^{(i)}\right)$, where $y \in \{c_1,\ c_2\}$
- Train model a_{c_1,c_2} on the set above
- a_{c_1,c_2} returns class c_1 or c_2
- The final model is

$$a(x) = \underset{c_1=1,2,\ldots,C}{\operatorname{argmax}} \sum_{c_2=1}^{C} [a_{c_1,c_2}(x) = c_1].$$

Figure 7.1: Caption

Figure 7.2: Caption

Figure 7.3: Caption

Week6/S5.png

7.2 Micro- and Macro-Averaging

Quality Metrics

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\left[a\big(x^{(i)}\big) = y^{(i)}\right]$$

Micro Averaging	Macro Averaging
$TP_c,\ TN_c,\ FP_c,\ FN_c$	
$TP = \sum_{c=1}^{C} TP_c, \dots$	$Precision_c = \frac{TP_c}{TP_c + FP_c}$
$Precision = \frac{TP}{TP + FP}$	$Precision = \frac{1}{C}\sum_{c=1}^{C} Precision_c$
Depends on the largest class	Ignores the class size

7.3 Multiple Logistic Regression

Multinomial Logistic Regression. Softmax Consider C linear models:

$$\begin{gathered} a_1(x) = w_{1,0} + w_{1,1}x_1 + w_{1,2}x_2 + \ldots + w_{1,d}x_d \\ a_2(x) = w_{2,0} + w_{2,1}x_1 + w_{2,2}x_2 + \ldots + w_{2,d}x_d \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_C(x) = w_{C,0} + w_{C,1}x_1 + w_{C,2}x_2 + \ldots + w_{C,d}x_d \end{gathered}$$

Convert the predictions into a vector of probabilities with components

$$p_c = \frac{e^{a_c}}{\sum\limits_{k=1}^{C} e^{a_k}}$$

Multinomial Logistic Regression. Cross-entropy Loss

$$L(y^{(1)}, \ldots, y^{(N)};\ p_1, \ldots, p_C) = \prod_{i=1}^{N} p_1^{[y^{(i)}=1]} \cdot \ldots \cdot p_C^{[y^{(i)}=C]}$$

$$\log L = N_1 \log p_1(w) + N_2 \log p_2(w) + \ldots + N_C \log p_C(w).$$

$$Loss = -\frac{1}{N} \log L = -\left(\frac{N_1}{N} \log p_1(w) + \frac{N_2}{N} \log p_2(w) + \ldots + \frac{N_C}{N} \log p_C(w) \right) = H(p, q),$$

where p and q are discrete distributions

Y	1	2	$\ldots$	C
p	$\frac{N_1}{N}$	$\frac{N_2}{N}$	$\ldots$	$\frac{N_C}{N}$

Y	1	2	$\ldots$	C
$q(w)$	p_1	p_2	$\ldots$	p_C

$$H(p, q) = -E_p[\log q]$$

7.3.1 Cross-Entropy Loss

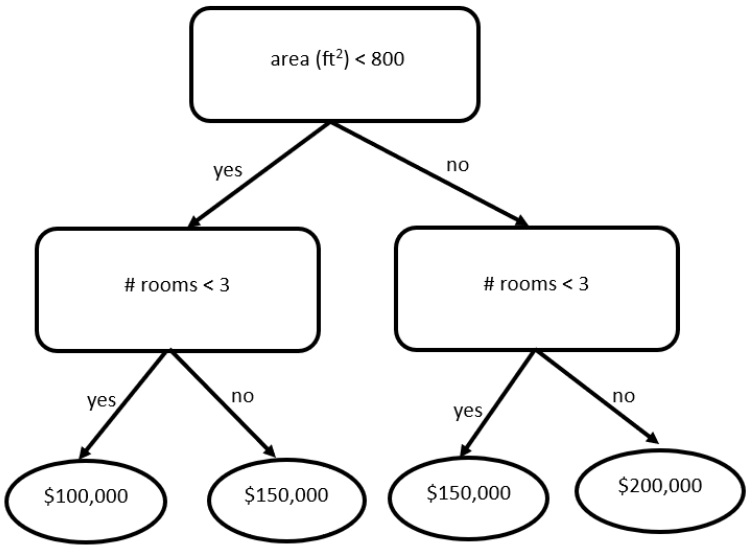
7.4 Bayessian Classification

7.4.1 Naïve Bayes

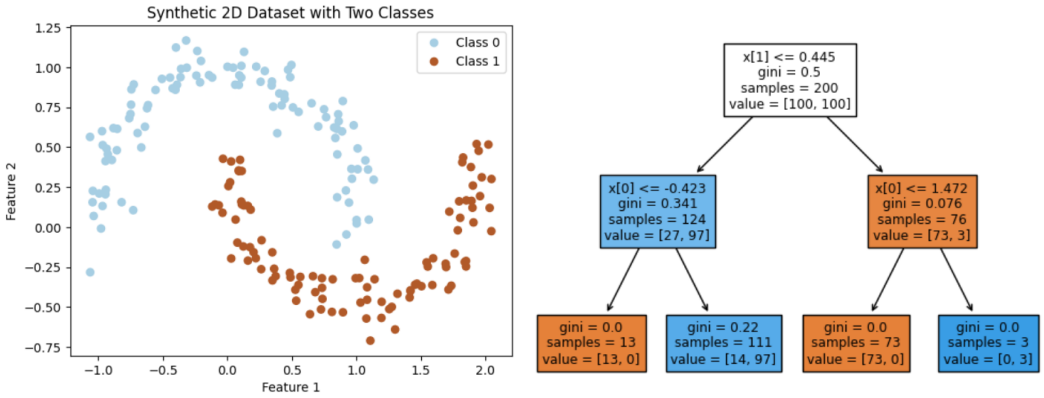
8 Decision Trees

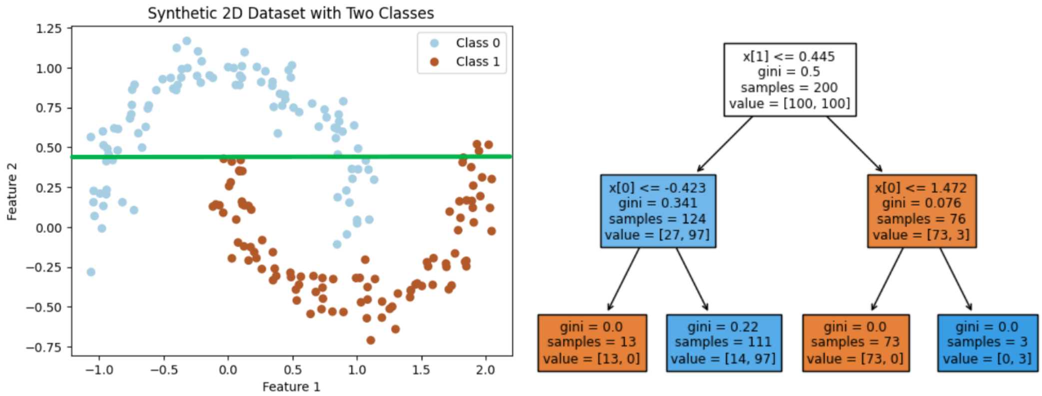
Creating new features

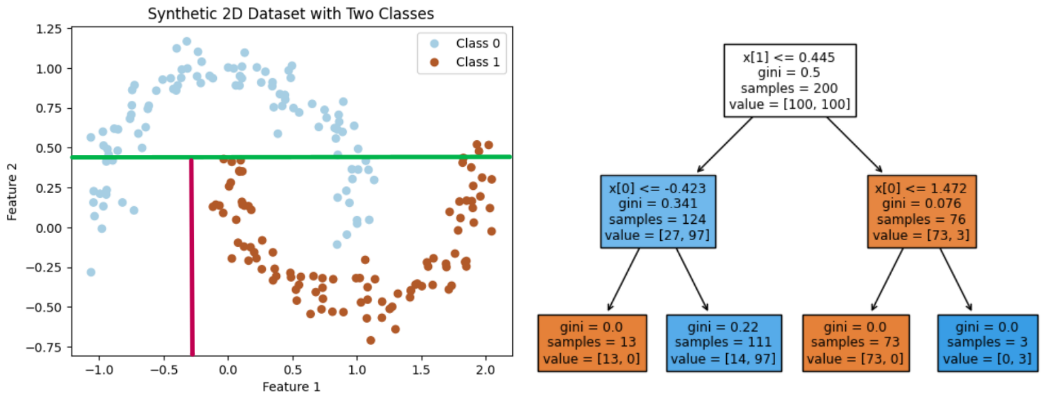
$$[700 < \text{area } (ft^2) < 800][1 < \#\text{of rooms} < 3]$$

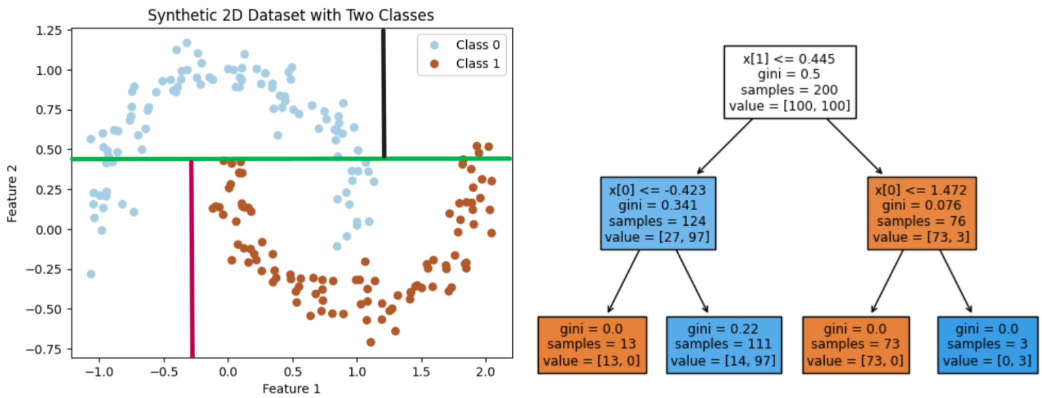


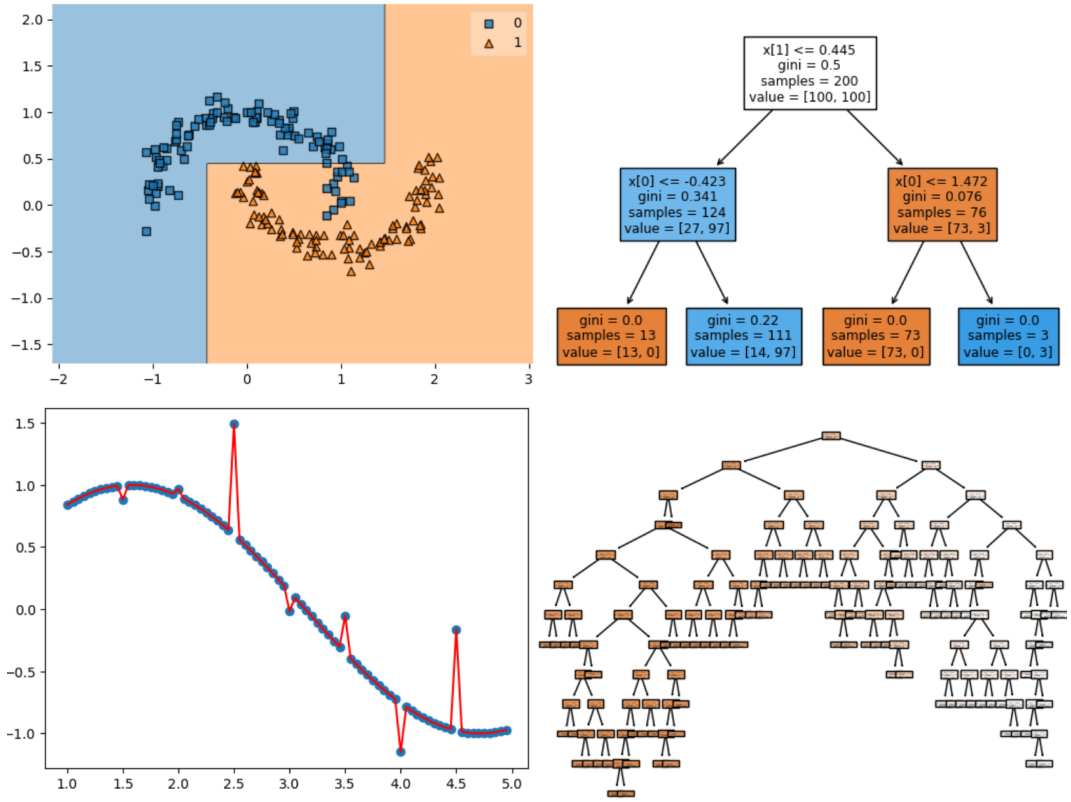
8.1 Geometry of Decision Tree Classifier




Synthetic 2D Dataset with Two Classes
Class 0
Class 1
Feature 1
Feature 2
x[1] <= 0.445
gini = 0.5
samples = 200
value = [100, 100]
x[0] <= -0.423
gini = 0.341
samples = 124
value = [27, 97]
x[0] <= 1.472
gini = 0.076
samples = 76
value = [73, 3]
gini = 0.0
samples = 13
value = [13, 0]
gini = 0.22
samples = 111
value = [14, 97]
gini = 0.0
samples = 73
value = [73, 0]
gini = 0.0
samples = 3
value = [0, 3]


Synthetic 2D Dataset with Two Classes
Class 0
Class 1
Feature 1
Feature 2
x[1] <= 0.445
gini = 0.5
samples = 200
value = [100, 100]
x[0] <= -0.423
gini = 0.341
samples = 124
value = [27, 97]
x[0] <= 1.472
gini = 0.076
samples = 76
value = [73, 3]
gini = 0.0
samples = 13
value = [13, 0]
gini = 0.22
samples = 111
value = [14, 97]
gini = 0.0
samples = 73
value = [73, 0]
gini = 0.0
samples = 3
value = [0, 3]


Synthetic 2D Dataset with Two Classes
Class 0
Class 1
Feature 1
Feature 2
x[1] <= 0.445
gini = 0.5
samples = 200
value = [100, 100]
x[0] <= -0.423
gini = 0.341
samples = 124
value = [27, 97]
x[0] <= 1.472
gini = 0.076
samples = 76
value = [73, 3]
gini = 0.0
samples = 13
value = [13, 0]
gini = 0.22
samples = 111
value = [14, 97]
gini = 0.0
samples = 73
value = [73, 0]
gini = 0.0
samples = 3
value = [0, 3]



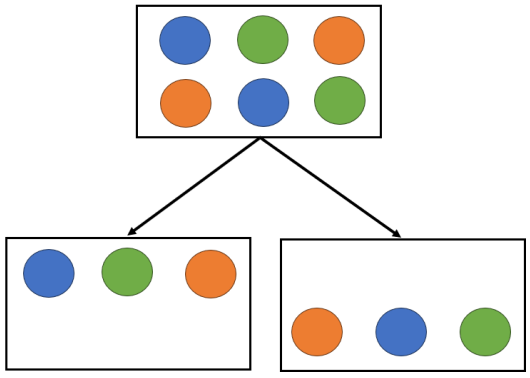
8.2 Leaf's Values

Classification:

$$\underset{c}{\operatorname{argmax}} \sum_{x^{(i)} \in V} \left[y^{(i)} = c \right] = \underset{c}{\operatorname{argmax}} \frac{1}{|V|} \sum_{x^{(i)} \in V} \left[y^{(i)} = c \right].$$

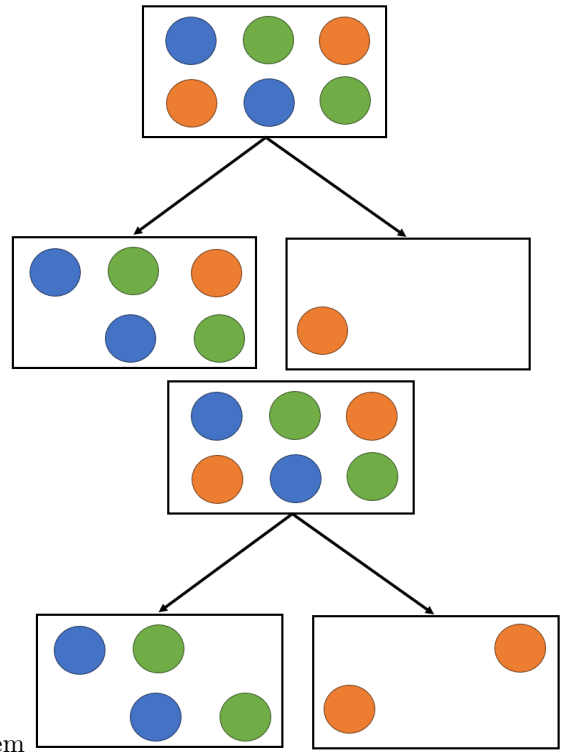
Regression

$$\frac{1}{|V|} \sum_{x^{(i)} \in V} y^{(i)}$$



-2em-2em

8.2.1 How to split one node? Classification



-2em-2em

-2em-2em

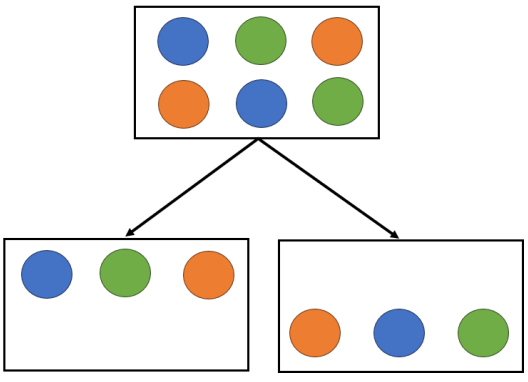
Information is defined by

$$I(p) = -\log p.$$

Entropy (work 'ergon' transformation 'tropy')

$$H(p) = \mathrm{E}[I(p)] = -\sum_i p_i \log p_i.$$

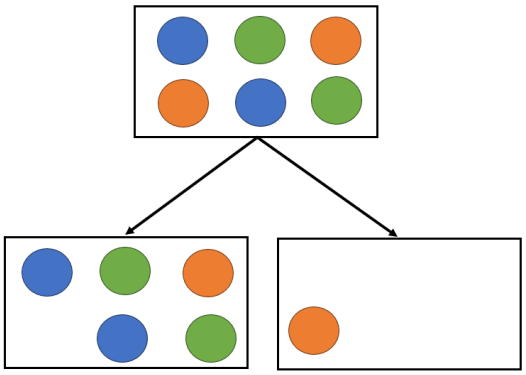
8.2.2 Impurity



-2em-2em

$$H(V) = -\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} \approx 1.0986$$

$$H(V_L) = H(V_R) = -\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} \approx 1.0986$$

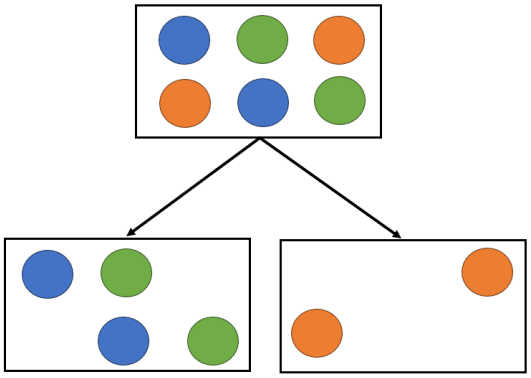


-2em-2em

$$H(V) = -\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} \approx 1.0986$$

$$H(V_L) = -\frac{2}{5}\log\frac{2}{5} - \frac{2}{5}\log\frac{2}{5} - \frac{1}{5}\log\frac{1}{5} \approx 1.055$$

$$H(V_R) = 1\log 1 = 0$$

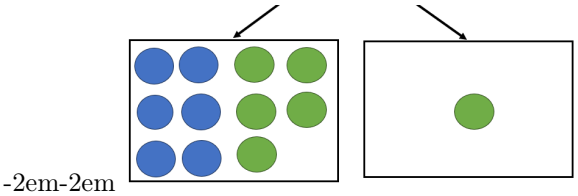


-2em-2em

$$H(V) = -\frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} - \frac{1}{3}\log\frac{1}{3} \approx 1.0986$$

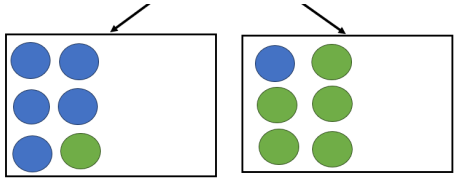
$$H(V_L) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2} \approx 0.693$$

$$H(V_R) = 1\log 1 = 0$$



-2em-2em

$$H(V_L) = -\frac{6}{11}\log\frac{6}{11} - \frac{5}{11}\log\frac{5}{11} \approx 0.689, \qquad H(V_R) = 1\log 1 = 0.$$



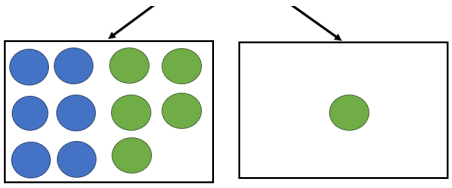
$$H(V_L) = -\frac{5}{6}\log\frac{5}{6} - \frac{1}{6}\log\frac{1}{6} \approx 0.45, \qquad H(V_R) = H(V_L)$$

8.2.3 Impurity

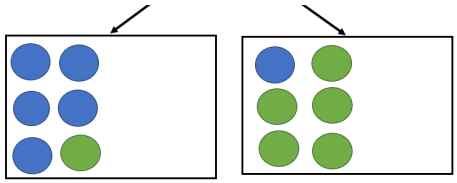
$$Q = H(V) - \left(\frac{|V_L|}{|V|}H(V_L) + \frac{|V_R|}{|V|}H(V_R)\right) \to \max$$

or

$$Q = \frac{|V_L|}{|V|}H(V_L) + \frac{|V_R|}{|V|}H(V_R) \to \min$$

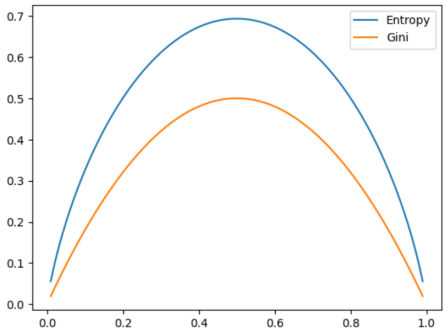
-2em-2em 

$$H(V_L) \approx 0.689, \quad H(V_R) = 1\log 1 = 0, \quad Q = \frac{11}{12}H(V_L) + \frac{1}{12}H(V_R) \approx 0.632.$$



$$H(V_L) = H(V_R) \approx 0.45, \quad Q = \frac{1}{2}H(V_L) + \frac{1}{2}H(V_R) \approx 0.45.$$

8.2.4 Gini VS Entropy



In practice we use

$$H(V) = \sum_{i=1} p_i(1 - p_i)$$

instead of

$$H(V) = -\sum_{i=1} p_i \log p_i$$

8.3 Impurity for Regression

L_2

$$H(V) = \frac{1}{V} \sum_{x^{(i)} \in V} \left(y^{(i)} - \bar{y}\right)^2$$

or L_1

$$H(V) = \frac{1}{V} \sum_{x^{(i)} \in V} \left|y^{(i)} - \text{median}(y)\right|.$$

8.3.1 Decision Tree Formula

If we have leaves $V_1,\ V_2,\ \ldots,\ V_J$, and $a_1,\ a_2,\ \ldots,\ a_J$ are the predictions in these nodes, then

$$a(x) = \sum_{j=1}^{J} a_j [x \in V_j].$$

9 Dimensionality Reduction

9.1 PCA

9.1.1 Problem Formulation

In this notes, we derive PCA from the dimensionality reduction perspective.

We assume that we have n data points with p features:

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_p^{(i)} \end{bmatrix}, \qquad i = 1, \dots, n. \tag{9.1}$$

We want to find a low-dimensional representation of the data

$$z^{(i)} = \begin{bmatrix} z_1^{(i)} \\ \vdots \\ z_\ell^{(i)} \end{bmatrix}, \qquad i = 1, \dots, n, \tag{9.2}$$

with a linear reconstruction

$$\hat{x}^{(i)} = W z^{(i)}, \qquad W \in M(p \times \ell). \tag{9.3}$$

We denote by $M(n_1 \times n_2)$ the space of real-valued matrices with n_1 rows and n_2 columns.

We want to minimize the reconstruction loss

$$L(W, z^{(1)}, \dots, z^{(n)}) = \sum_{i=1}^{n} \|x^{(i)} - \hat{x}^{(i)}\|^2 = \sum_{i=1}^{n} \|x^{(i)} - W z^{(i)}\|^2 \tag{9.4}$$

under the additional assumption that the columns W_k of the matrix W are orthonormal:

$$\|W_i\|^2 = 1, \qquad (W_i, W_j) = 0, \qquad i, j = 1, \dots, \ell, \quad i \neq j. \tag{9.5}$$

Denote by X a matrix with vectors $x^{(i)}$ in rows:

$$X = \begin{bmatrix} -x^{(1)T}- \\ -x^{(2)T}- \\ \cdots \\ -x^{(n)T}- \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \cdots & \cdots & \cdots & \cdots \\ x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{bmatrix}. \tag{9.6}$$

Similarly, denote by Z a matrix with vectors $z^{(i)}$ in its rows. Then the loss function (9.4) can be written as follows[1]:

$$L(W, Z) = \|X^T - W Z^T\|_F^2. \tag{9.7}$$

[1] Recall that the Frobenius norm of a matrix A is the square root from the sum of squared elements of this matrix: $\|A\|_F = \sqrt{\sum\limits_{i,j} a_{ij}^2}$

Condition (9.5) is equivalent to the following equations:

$$W_i^T W_j = \delta_{ij}, \quad i, j = 1, \ldots, \ell, \tag{9.8}$$

or in matrix notation

$$W^T W = I. \tag{9.9}$$

We arrive to the minimization problem:

$$\begin{cases} L(W, Z) = \|X^T - WZ^T\|_F^2 \to \min\limits_{W,Z}, \\ W^T W = I. \end{cases} \tag{9.10}$$

Because it contains only equality constrains, we can use the Lagrange multipliers method. Compose the Lagrange function:

$$\mathcal{L}(W, Z, \Lambda) = \|X^T - WZ^T\|_F^2 - \sum_{i=1}^{\ell} \lambda_{ii}(W_i^T W_i - 1) - \sum_{\substack{i,j=1 \\ i<j}}^{\ell} \lambda_{ij} W_i^T W_j. \tag{9.11}$$

Fist, we take the derivatives[2] with respect to $z^{(i)}$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z^{(i)}} &= \frac{\partial}{\partial z^{(i)}} \sum_{k=1}^{\ell} \|x^{(k)} - Wz^{(k)}\|^2 \\ &= \frac{\partial}{\partial z^{(i)}} \sum_{k=1}^{\ell} (x^{(k)} - Wz^{(k)})^T (x^{(k)} - Wz^{(k)}) \\ &= \frac{\partial}{\partial z^{(i)}} \sum_{k=1}^{\ell} (x^{(k)T} - z^{(k)T} W^T)(x^{(k)} - Wz^{(k)}) \\ &= \frac{\partial}{\partial z^{(i)}} \sum_{k=1}^{\ell} (x^{(k)T} x^{(k)} - 2x^{(k)T} Wz^{(k)} + z^{(k)T} W^T W z^{(k)}) \\ &= -2W^T x^{(i)} + 2z^{(i)} = 0. \end{aligned} \tag{9.12}$$

We find that

$$z^{(i)} = W^T x^{(i)}. \tag{9.13}$$

Substituting this representation in loss function (9.7), we obtain a new optimization problem:

$$\begin{cases} L(W) = L(W, Z(W)) = \|X^T - WW^T X^T\|_F^2 \to \min\limits_{W}, \\ W^T W = I. \end{cases} \tag{9.14}$$

Rewrite the target function:[3]

$$\begin{aligned} \|X^T - WW^T X^T\|_F^2 &= Tr\left((X^T - WW^T X^T)^T (X^T - WW^T X^T)\right) \\ &= Tr\left((X - XWW^T)(X^T - WW^T X^T)\right) \\ &= Tr\left(XX^T - 2XWW^T X^T + XWW^T WW^T X^T\right) \\ &= Tr\left(XX^T - 2XWW^T X^T + XWW^T X^T\right) \\ &= Tr\left(XX^T - XWW^T X^T\right). \end{aligned} \tag{9.15}$$

[2] We use the following equalities $\frac{\partial a^T x}{\partial x} = a$ and $\frac{\partial x^T A x}{\partial x} = 2Ax$. See, e.g., https://en.wikipedia.org/wiki/Matrix_calculus

[3] We use the following identity $\|A\|_F^2 = Tr(A^T A)$. The trace of a matrix is defined as the sum of its diagonal elements.

Because the first term does not depend on W, problem (9.14) can be reformulated as follows:

$$\begin{cases} L(W) = Tr\left(-XWW^TX^T\right) \to \min\limits_{W}, \\ W^TW = I. \end{cases} \tag{9.16}$$

Or equivalently[4] as

$$\begin{cases} L(W) = Tr\left(\frac{1}{n}X^TXWW^T\right) \to \max\limits_{W}, \\ W^TW = I. \end{cases} \tag{9.17}$$

The matrix $\frac{1}{n}X^TX$ is known as a covariance matrix and will be denoted by

$$\Sigma = \frac{1}{n}X^TX. \tag{9.18}$$

Compose the Lagrange function for this problem:

$$\mathcal{L}(W, \underset{\ell\times\ell}{\Lambda}) = Tr\left(\Sigma WW^T\right) - \sum_{i=1}^{\ell}\lambda_{ii}(W_i^TW_i - 1) - \sum_{\substack{i,j=1\\ i<j}}^{\ell}\lambda_{ij}W_i^TW_j. \tag{9.19}$$

Here, we used a lower-triangular matrix $\underset{\ell\times\ell}{\Lambda} \in M(\ell\times\ell)$ to store the Laplace multipliers:

$$\underset{\ell\times\ell}{\Lambda} = \begin{bmatrix} \lambda_{11} & 0 & \cdots & 0 \\ \lambda_{21} & \lambda_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_{\ell 1} & \lambda_{\ell 2} & \cdots & \lambda_{\ell\ell} \end{bmatrix}. \tag{9.20}$$

Taking the derivatives with respect to columns W_k and λ_{ij}, we get the following system of equations:[5]

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial W_k} = 2\Sigma W_k - 2\lambda_{kk}W_k - \sum\limits_{i=1}^{k-1}\lambda_{ik}W_i - \sum\limits_{j=k+1}^{\ell}\lambda_{kj}W_j = 0, \quad k = 1,\ldots,\ell, \\ \dfrac{\partial\mathcal{L}}{\partial\lambda_{kk}} = W_k^TW_k - 1 = 0, \quad k = 1,\ldots,\ell, \\ \dfrac{\partial\mathcal{L}}{\partial\lambda_{ij}} = W_i^TW_j = 0, \quad i,j = 1,\ldots,\ell,\ i < j. \end{cases} \tag{9.21}$$

If we multiply the first equation by W_k^T from the left, we will get

$$2W_k^T\Sigma W_k - 2\lambda_{kk}W_k^TW_k - \sum_{i=1}^{k-1}\lambda_{ik}W_k^TW_i - \sum_{j=k+1}^{\ell}\lambda_{kj}W_k^TW_j = 0, \quad k = 1,\ldots,\ell. \tag{9.22}$$

Using the orthogonality conditions, we get

$$W_k^T\Sigma W_k - \lambda_{kk} = 0, \quad k = 1,\ldots,\ell. \tag{9.23}$$

[4] We use the following property of the trace: $Tr(AB) = Tr(BA)$.

[5] To calculate the derivative $\dfrac{\partial\mathcal{L}}{\partial W_k}$ we use formula $\dfrac{\partial Tr(\Sigma WW^T)}{\partial W} = 2\Sigma W$ and then take kth column.

And multiplying the last equation by W_k from the left, it takes form

$$\Sigma W_k = \lambda_{kk} W_k, \quad k = 1, \ldots, \ell, \tag{9.24}$$

which is the eigenvalue problem for the matrix Σ.

Because the matrix $\Sigma \in M(p \times p)$ is symmetric,[6] it has p orthogonal eigenvectors[7] and corresponding eigenvalues are real and nonnegative.[8] We can choose $\ell \leq p$ columns of the matrix W as normalized eigenvectors of the matrix Σ and λ_{kk} as corresponding eigenvalues. They satisfy equation (9.24) and conditions (9.9).

Due to the linear independence of the eigenvectors W_k from the first equation of (9.21) it follows that off-diagonal elements $\lambda_{ij} = 0$ with $i < j$. Therefore, matrix Λ has form

$$\underset{\ell \times \ell}{\Lambda} = \begin{bmatrix} \lambda_{11} & 0 & \cdots & 0 \\ 0 & \lambda_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \lambda_{\ell\ell} \end{bmatrix}. \tag{9.25}$$

If we substitute this solution to the target function, we will get

$$L(W) = Tr\left(\Sigma W W^T\right) = Tr\left(W^T \Sigma W\right) = Tr(\underset{\ell \times \ell}{\Lambda}) = \lambda_{11} + \ldots + \lambda_{\ell\ell}. \tag{9.26}$$

To get the maximal value, we should use the first ℓ maximal eigenvalues of the matrix Σ.

9.1.2 Statistical Interpretation

Before applying PCA algorithm, i.e., calculating ℓ eigenvectors of the matrix Σ corresponding to the ℓ greatest eigenvalues, we it is recommended to subtract the mean:

$$x \to x - \bar{x}, \tag{9.27}$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)} = \begin{bmatrix} \frac{1}{n} \sum_{i=1}^{n} x_1^{(i)} \\ \vdots \\ \frac{1}{n} \sum_{i=1}^{n} x_p^{(i)} \end{bmatrix}. \tag{9.28}$$

If we assume the zero mean of the points $x^{(i)}$, then the low-dimensional points

$$z^{(i)} = W^T x^{(i)} = \begin{bmatrix} -W_1^T - \\ \cdots \\ -W_\ell^T - \end{bmatrix} \begin{bmatrix} x_1^{(i)} \\ \vdots \\ x_p^{(i)} \end{bmatrix} = \begin{bmatrix} W_1^T x^{(i)} \\ \cdots \\ W_\ell^T x^{(i)} \end{bmatrix} \tag{9.29}$$

[6] For $A = X^T X$ $A^T = (X^T X)^T = X^T (X^T)^T = X^T X = A$.

[7] Theorem

[8] If h is an eigenvector of the matrix $X^T X$ with the eigenvalue λ, $X^T X h = \lambda h$. Therefore, $\lambda h^T h = h^T X^T X h = (Xh)^T (Xh) = \|Xh\|^2 \geq 0$. By definition of the eigenvector, $h \neq 0$, that means that $h^T h = \|h\|^2 > 0$; this leads to $\lambda \geq 0$.

will have zero mean as well. Then their variance can be calculated as follows:

$$\begin{aligned} S^2_{z_k} &= \frac{1}{n}\sum_{i=1}^{n}\left(z_k^{(i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(W_k^T x^{(i)}\right)^2 = \frac{1}{n}\sum_{i=1}^{n} W_k^T x^{(i)} x^{(i)T} W_k \\ &= W_k^T \left(\frac{1}{n}\sum_{i=1}^{n} x^{(i)} x^{(i)T}\right) W_k = W_k^T \Sigma W_k = W_k^T \lambda_{kk} W_k = \lambda_{kk}. \end{aligned} \tag{9.30}$$

The total variance of the data points $z^{(i)}$ is equal to

$$\sum_{k=1}^{n} S^2_{z_k} = \lambda_{11} + \ldots + \lambda_{\ell\ell}. \tag{9.31}$$

The total variance of the data points $x^{(i)}$ is equal to[9]

$$\sum_{k=1}^{n} S^2_{x_k} = Tr(\Sigma) = Tr(W \underset{p\times p}{\Lambda} W^T) = Tr(\underset{p\times p}{\Lambda}) = \lambda_{11} + \ldots + \lambda_{pp}. \tag{9.32}$$

This means that if we take $\ell = p$ components, then we reconstruct the data without any losses. It will correspond to the rotation of the original data. The fraction

$$\frac{\lambda_{11} + \ldots + \lambda_{\ell\ell}}{\lambda_{11} + \ldots + \lambda_{pp}} \tag{9.33}$$

represents the fraction of the explained variance.

9.1.3 Geometric Interpretation

We will show that reconstruction of the projection $z^{(i)}$ is the orthogonal projection of the point $x^{(i)}$ on the subspace spanned by vectors $W_1, \ldots, W_\ell$.

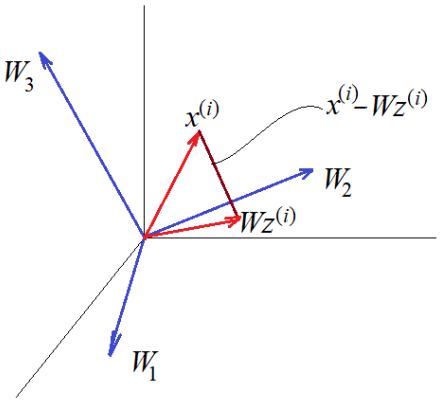


Figure 9.1: Orthogonal projection of a 3D vector $x^{(i)}$ on 2D plane of two principal components W_1 and W_2

It is sufficient to show that the residuals are orthogonal for each component:

$$x^{(i)} - Wz^{(i)} \perp W_k, \qquad k = 1, \ldots, \ell. \tag{9.34}$$

[9] We assume that the eigenvalues are sorted in descending order.

Equivalently, their dot product should be zero:

$$\begin{aligned} W_k^T(x^{(i)} - Wz^{(i)}) &= W_k^T x^{(i)} - W_k^T \begin{bmatrix} | & & | \\ W_1 & \cdots & W_\ell \\ | & & | \end{bmatrix} z^{(i)} \\ &= W_k^T x^{(i)} - [\underset{1}{0}, \ldots, \underset{k}{1}, \ldots \underset{\ell}{0}] z^{(i)} \\ &\overset{(9.29)}{=} W_k^T x^{(i)} - [\underset{1}{0}, \ldots, \underset{k}{1}, \ldots \underset{\ell}{0}] \begin{bmatrix} W_1^T x^{(i)} \\ \cdots \\ W_\ell^T x^{(i)} \end{bmatrix} \\ &= W_k^T x^{(i)} - W_k^T x^{(i)} = 0. \end{aligned} \tag{9.35}$$

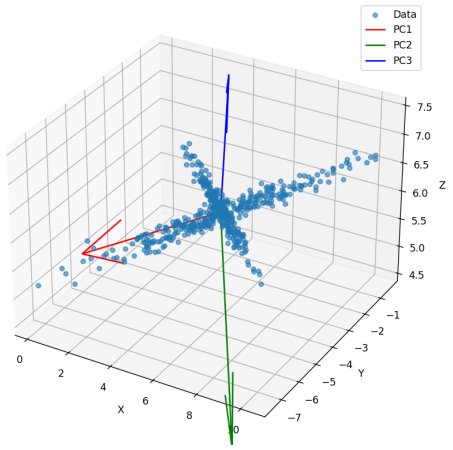
The explained variance ratio (9.33) provides another algorithm of PCA. Find the first component as a vector, such that projection of the data on it has the highest variance (see Fig. 9.2b):

$$\lambda_{11} = \max_{\|W_1\|=1} W_1^T \Sigma W_1. \tag{9.36}$$

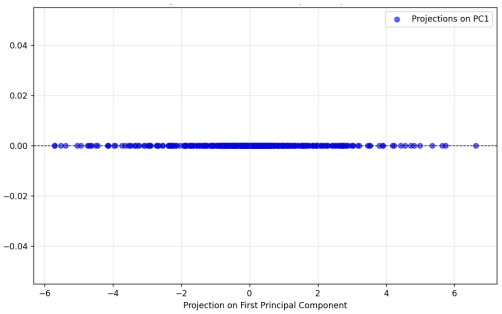
After obtaining the first component W_1, we should find the second component orthogonal to W_1, such that it captures the highest variance of the residuals (see Fig. 9.2c) $\tilde{x}^{(i)} = x^{(i)} - W_1 z^{(i)}$. This can be formulated as the following maximization problem:

$$\lambda_{22} = \max_{\|W_2\|=1} W_2^T \tilde{\Sigma} W_2, \tag{9.37}$$

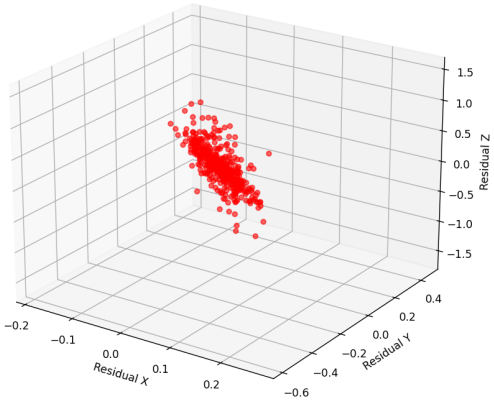
where $\tilde{\Sigma} = \frac{1}{n} \tilde{X}^T \tilde{X}$. And so on.



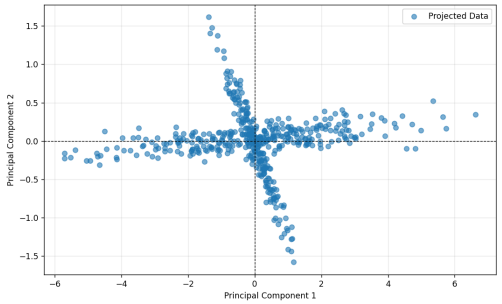
(a) Data with principal components.



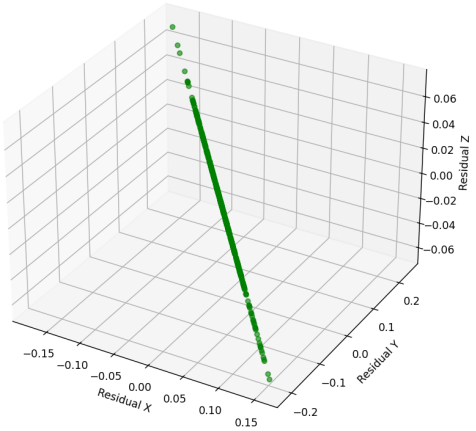
(b) Projection on W_1.



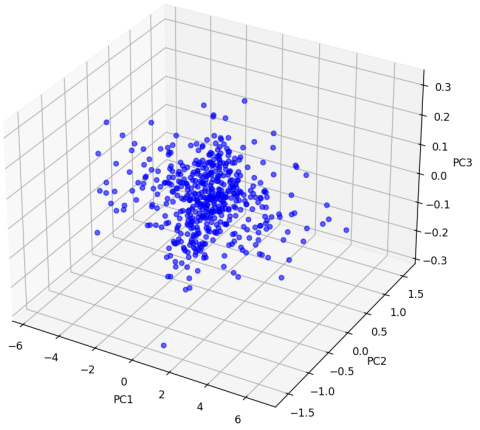
(c) Residuals lie on a plane.



(d) Projection on W_1 and W_2.



(e) Residuals lie on a line.



(f) Projection on W_1, W_2, and W_3.

Figure 9.2: Illustration to sequential building of PCA.

9.1.4 Connection with Singular Value Decomposition (SVD)

Each matrix $A \in M(n_1 \times n_2)$ allows the singular value decomposition:

$$\underset{n_1 \times n_2}{A} = \underset{n_1 \times n_1}{U}\,\underset{n_1 \times n_2}{\Sigma}\,\underset{n_2 \times n_2}{V^T}, \tag{9.38}$$

with depending on $n_1 \geq n_2$ or $n_1 \leq n_2$

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_{n_2} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \quad \text{or} \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_{n_1} & 0 & \cdots & 0 \end{bmatrix}. \tag{9.39}$$

Where σ_i are eigenvalues of AA^T or A^TA. Columns of U are normalized eigenvectors of AA^T and olumns of V are normalized eigenvectors of A^TA.

Consider the singular value decomposition of the matrix X :

$$X = USV^T. \tag{9.40}$$

Then

$$\frac{1}{n}X^TX = \frac{1}{n}VSU^TUSV^T = V\left(\frac{1}{\sqrt{n}}S\right)^2V^T. \tag{9.41}$$

Comparison with

$$\frac{1}{n}X^TX = W\underset{\ell \times \ell}{\Lambda}W^T \tag{9.42}$$

demonstrates that variances can be calculated via squared singular values of the matrix X

$$\lambda_{ii} = \frac{\sigma_{ii}^2}{n}, \quad i = 1, \ldots, \ell, \tag{9.43}$$

and the principle components W are its right singular vectors V.

9.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

In some cases PCA doesn't reflect the data structure. For example, in Fig. 9.3, the projections of the data on the first component overlap. To address this problem, we can use linear discriminant analysis (LDA), but for data visualization the t-distributed stochastic neighbor embedding algorithm (t-SNE) is typically used.

https://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

9.2.1 One Degree of Freedom in t-Distribution

Introduce the distribution on the discrete space of pairwise distances. The easiest way to do it is for the point $x^{(j)}$ to find k nearest neighbors (excluding itself)

$$x^{(j)}, x^{(j_1)}, \ldots, x^{(j_k)}, \quad j_0 = j,$$

and define numbers

$$p_{j|j} = 0, \quad p_{j_1|j} = \ldots = p_{j_k|j} = \frac{1}{k}, \quad p_{j_{k+1}|j} = \ldots = p_{j_{n-1}|j} = 0. \tag{9.44}$$

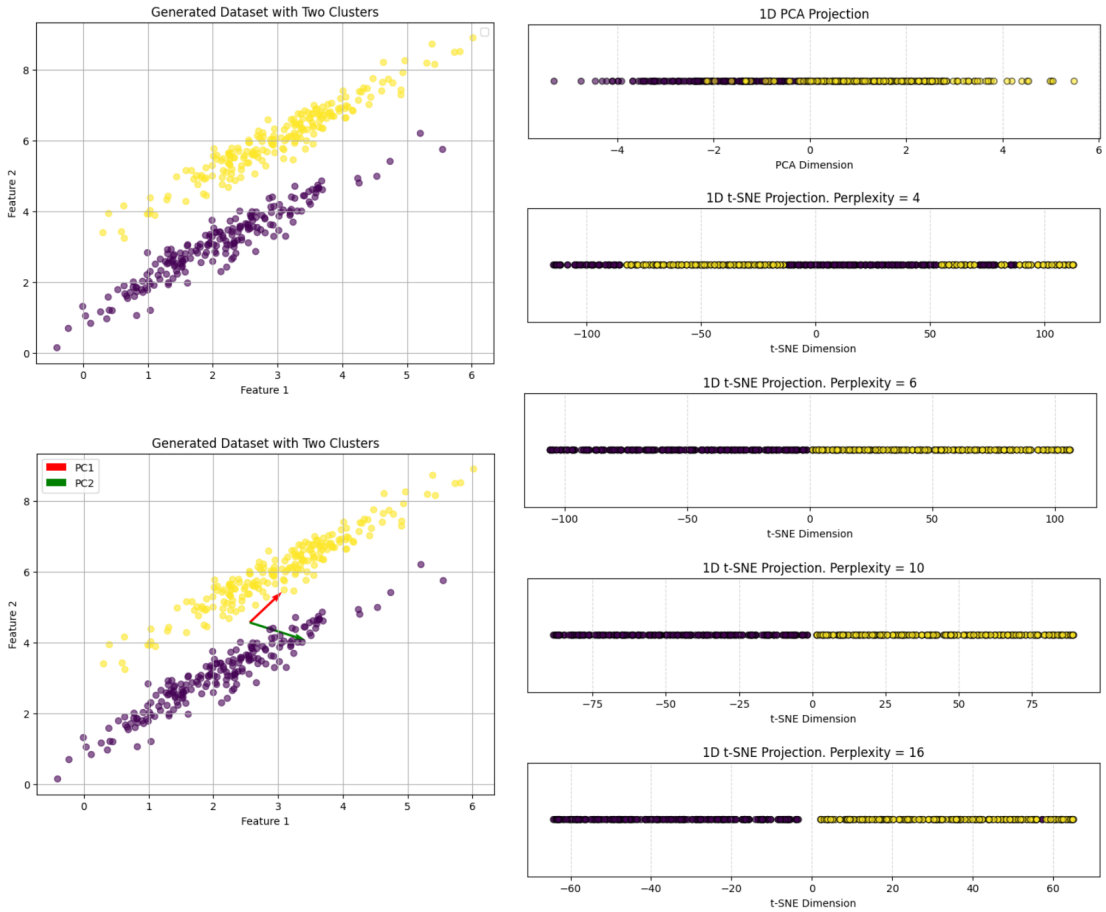


Figure 9.3: Comparison of PCA and t-SNE

Then for each pair $x^{(i)}$ and $x^{(j)}$ define probabilities

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}. \tag{9.45}$$

It is easy to check that

$$\sum_{i,j=1}^{n} p_{ij} = \frac{1}{2n}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} p_{j|i} + \sum_{j=1}^{n}\sum_{i=1}^{n} p_{i|j}\right) = 1. \tag{9.46}$$

By *perplexity* we mean

$$Perplexity = 2^{H}, \tag{9.47}$$

where H is the Shannon's entropy.

For one point $x^{(j)}$ with probabilities () the perplexity is equal to the number of the neighbors:

$$Perplexity = 2^{-\sum\limits_{i=1}^{n} p_{i|j} \log_2 p_{i|j}} = k. \tag{9.48}$$

The common choice for the numbers $p_{i|j}$ is

$$p_{i|j} = \frac{e^{-\frac{\left|x^{(i)}-x^{(j)}\right|^2}{2\sigma_j^2}}}{\sum_{k=1}^{n} e^{-\frac{\left|x^{(k)}-x^{(j)}\right|^2}{2\sigma_k^2}}}, \tag{9.49}$$

where σ_j is chosen such that distribution $p_{i|j}$ has the desired perplexity.[10] The standard choice of perplexity is $5 - 50$.

The corresponding points $z^{(i)}$ in the lower-dimensional space are initialized randomly or using PCA.

The pairwise density is defined as follows:

$$q_{ij} = \frac{1}{Z} \frac{1}{1 + \left|z^{(i)} - z^{(j)}\right|^2}, \tag{9.50}$$

where Z is the normalizing constant:

$$Z = \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \left(1 + \left|z^{(i)} - z^{(j)}\right|^2\right)^{-1}. \tag{9.51}$$

The positions of the points $z^{(i)}$ are defined by minimization of the Kullback — Leibler divergence

$$KL(P||Q) = \sum_{r,s=1}^{n} p_{rs} \log \frac{p_{rs}}{q_{rs}}, \tag{9.52}$$

using gradient[11] descent:

$$z^{(i)} = z^{(i)} - \eta \nabla_{z^{(i)}} KL(P||Q), \quad i = 1, \ldots, n. \tag{9.53}$$

To calculate the gradient with respect to $z^{(i)}$ denote by

$$d_{ij} = \|z^{(i)} - z^{(j)}\|^2 = z^{(i)T} z^{(i)} - 2 z^{(i)T} z^{(j)} + z^{(j)T} z^{(j)} \tag{9.54}$$

and rewrite the minimizing function:

$$\begin{aligned} KL(P||Q) &= \sum_{r,s=1}^{n} p_{rs} \log p_{rs} - \sum_{r,s=1}^{n} p_{rs} \log q_{rs} \\ &= \sum_{r,s=1}^{n} p_{rs} \log p_{rs} - \sum_{r,s=1}^{n} p_{rs} \log(1 + d_{rs})^{-1} + \sum_{r,s=1}^{n} p_{rs} \log Z \\ &= \sum_{r,s=1}^{n} p_{rs} \log p_{rs} - \sum_{r,s=1}^{n} p_{rs} \log(1 + d_{rs})^{-1} + \log Z. \end{aligned} \tag{9.55}$$

[10] In practice, perplexity is chosen in the range 5—50.

[11] By definition, the gradient is a vector of partial derivatives: $\nabla_{z^{(i)}} F(z^{(i)}) = \begin{bmatrix} \frac{\partial F}{\partial z_1^{(i)}} \\ \vdots \\ \frac{\partial F}{\partial z_\ell^{(i)}} \end{bmatrix}$. We use for it also notation of a matrix derivative $\frac{\partial F}{\partial z^{(i)}}$.

Then

$$\frac{\partial KL(P||Q)}{\partial z^{(k)}} = \sum_{i,j=1}^{n} \frac{\partial KL(P||Q)}{\partial d_{ij}} \frac{\partial d_{ij}}{\partial z^{(k)}}. \tag{9.56}$$

From (9.54) it follows that

$$\frac{\partial d_{ij}}{\partial z^{(k)}} = \begin{cases} 0, & i = j \text{ or } i \neq k, j \neq k, \\ 2z^{(k)} - 2z^{(j)}, & i = k, j \neq k, \\ -2z^{(i)} + 2z^{(k)}, & i \neq k, j = k. \end{cases} \tag{9.57}$$

From (9.56) it follows that

$$\begin{aligned} \frac{\partial KL(P||Q)}{\partial d_{ij}} &= \frac{p_{ij}}{1+d_{ij}} + \frac{1}{Z}\frac{\partial (1+d_{ij})^{-1}}{\partial d_{ij}} = p_{ij}(1+d_{ij})^{-1} - \frac{1}{Z}(1+d_{ij})^{-2} \\ &= \frac{p_{ij}(1+d_{ij})^{-1}}{Z} Z - q_{ij}(1+d_{ij})^{-1} = p_{ij}q_{ij}Z - q_{ij}^2 Z \\ &= (p_{ij} - q_{ij})q_{ij}Z. \end{aligned} \tag{9.58}$$

$$\begin{aligned} \frac{\partial KL(P||Q)}{\partial z^{(k)}} &= \sum_{i \neq k} \frac{\partial KL(P||Q)}{\partial d_{ik}} \left(-2z^{(i)} + 2z^{(k)}\right) + \sum_{j \neq k} \frac{\partial KL(P||Q)}{\partial d_{kj}} \left(2z^{(k)} - 2z^{(j)}\right) \\ &= \sum_{i \neq k} (p_{ik} - q_{ik})q_{ik}Z \left(-2z^{(i)} + 2z^{(k)}\right) + \sum_{j \neq k} (p_{kj} - q_{kj})q_{kj}Z \left(2z^{(k)} - 2z^{(j)}\right) \\ &= 4 \sum_{i \neq k} (p_{ik} - q_{ik})q_{ik}Z \left(z^{(k)} - z^{(i)}\right). \end{aligned} \tag{9.59}$$

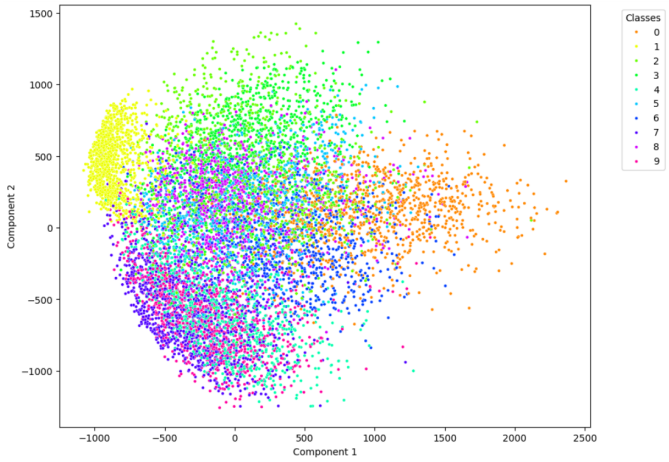
9.2.2 Other Degrees of Freedom

The pairwise density is defined as follows:

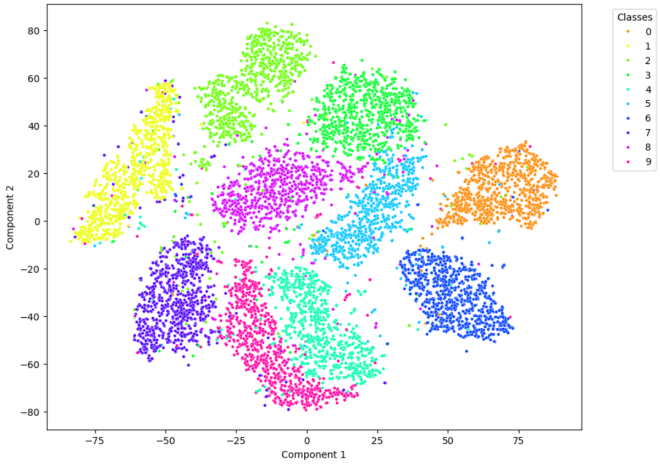
$$q_{ij} = \frac{1}{Z} \frac{1}{\left(1 + \frac{\left|z^{(i)} - z^{(j)}\right|^2}{\nu}\right)^{\frac{\nu+1}{2}}}, \tag{9.60}$$

where Z is the normalizing constant:

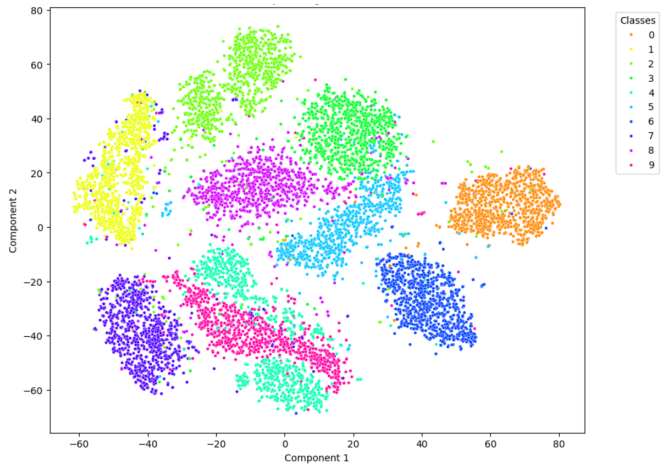
$$Z = \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \left(1 + \frac{\left|z^{(i)} - z^{(j)}\right|^2}{\nu}\right)^{-\frac{\nu+1}{2}}. \tag{9.61}$$



(a) PCA.



(b) t-SNE with perplexity 30.



(c) t-SNE with perplexity 50.

Figure 9.4: Comparison of different dimensionality reduction algorithms on MNIST.

9.3 Orthogonal Procrustes Problem

This problem arises when we need to align two sets of points using only rotations and reflections. As in PCA we assume the data is centered.

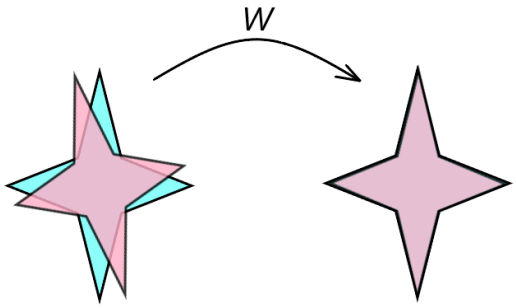


Figure 9.5: Illustration of the orthogonal alignment

Mathematically we want to find an orthogonal matrix W that maps a set of points X onto the points Y:

$$\begin{cases} L(W) = \sum\limits_{i=1}^{n} \|Wx^{(i)} - y^{(i)}\|^2 = \|WX^T - Y^T\|_F^2 \to \min\limits_W, \\ W^TW = I. \end{cases} \tag{9.62}$$

Rewrite the target function using trace:

$$\begin{aligned} L(W) &= Tr\left((WX^T - Y^T)^T(WX^T - Y^T)\right) \\ &= Tr\left((XW^T - Y)(WX^T - Y^T)\right) \\ &= Tr\left(XW^TWX^T - XW^TY^T - YWX^T + YY^T\right) \\ &= Tr(XX^T) - Tr(XW^TY^T) - Tr(YWX^T) + Tr(YY^T). \end{aligned} \tag{9.63}$$

Thus, the minimization problem (9.62) can be reformulated as a maximization problem:[12]

$$\begin{cases} L(W) = Tr(WX^TY) \to \max\limits_W, \\ W^TW = I. \end{cases} \tag{9.64}$$

If we consider SVD decomposition of the matrix X^TY

$$X^TY = U\Sigma V^T, \tag{9.65}$$

then[13]

$$Tr(WX^TY) = Tr(WU\Sigma V^T) \le Tr(V\Sigma V^T) = Tr(\Sigma). \tag{9.66}$$

Therefore, the matrix W^* delivering the maximum satisfies equation

$$W^*U = V \tag{9.67}$$

or

$$W^* = VU^T. \tag{9.68}$$

[12]We use the fact $Tr(A^T) = \sum\limits_{i=1}^{m} a_{ii} = Tr(A)$ for a matrix $A \in M(m \times m)$.

[13]Theorem