Machine Learning And Neural Networks (CM3015)

Course Notes

Felipe Balbi

May 21, 2021

Contents

Week 9

Week 1

Key Concepts

- explain the concepts of clustering and dimensionality reduction
- Describe various types of machine learning problem
- Describe various applications of machine learning

1.101 Applications of Machine Learning

Machine Learning is a branch of artificial intelligence that enables machines to learn by example. Carried by the increase in data availability and computational power, we can already experiences applications of machine learning in our everyday lives: mobile phones, personal assistants, language translators, etc.

One application of machine learning are the e-passport gates at some airports which rely on face recognition to identify passengers with high probability.

Computer Vision systems can also be used to detect and classify human posture and facial expressions. Machine Learning can also be applied to other types of data such as text (handwriting recognition) or audio (speech recognition).

These systems collect and process vast amounts of data and the issue of privacy arises. We must be conscious about what data has been recorded, who has access to it, and how it can be used.

Autonomous Vehicles are a focus in machine learning research. They pose interesting and complex challenges both technically and ethically. Vehicles need to be able to detect and avoid pedestrians and other objets on the road. In the case of the accident, who's to blame? The owner of the vehicle? The company who made the car? The software engineers who built the system?

Another common system in our daily lives are recommender systems. We encounter them in streaming services, online shopping experiences, MOOC education providers, and many more. The main goal of these systems is to recommend other items similar to what we have already *consumed*. Because these systems are also used to suggest similar content to what we already watch, they may end up skewing our view of the world.

Generative Machine Learning are models that can generate new data based on a sample, for example given a sample of someone's handwriting, we want to produce more text in the same style.

Another application is related to Sensor-based Activity Recognition. Here the goal is to detect what activity the user is executing (sitting, walking, running, playing footbal) based on the data from sensors the user's wearing.

1.102 Types of ML

Machine Learning is used when we want to learn from data rather than hardcode a solution. There are two types of machine learning

Supervised Learning in supervised learning, the label y is associated with every sample x. We're trying to learning mapping from x to y

Unsupervices Learning here the goal is usually about clustering data in subgroups. For example, given a dataset containing pictures of animals, separate the images by animal.

We can use the decision tree depicted in figure 1 to decide which type of Machine Learning application to apply:

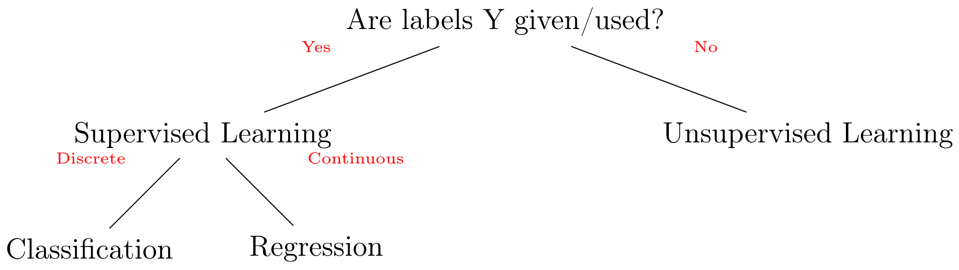

Figure 1: Decision Tree

One final type is Reinforcement Learning where the goal is to learn a sequence of actions that entail some reward. This can be used to teach a machine how to play a specific video game.

Week 2

Key Concepts

- explain the concepts of clustering and dimensionality reduction
- Describe various types of machine learning problem
- Describe various applications of machine learning

1.202 Further reading

- Chapter 1, sections 1.1 to 1.2 of the course textbook (Chollet).
- Chapter 1 of Alpaydin, E. Introduction to machine learning. (Cambridge, MA: MIT Press, 2014) 3rd edition [ISBN 9780262028189].
- Chapter 1, Introduction, up to and including section 1.3, of the following textbook gives a good introduction to the topic of ML: Murphy, K. Machine learning: a probabilistic perspective. (Cambridge, MA: MIT Press, 2012) [ISBN 9780262018029]

Week 3

Key Concepts

- Explain how a simple nearest neighbour algorithm works
- Describe the Decision Tree Classifier
- Evaluate a supervised classification algorithm on a dataset

2.101 Introduction to supervised Learning

Classification is a type of supervised learning where the labels on a data are discrete and categorical.

2.201 K-Nearest Neighbours Classification

The k-NN algorithm works on the premise that things are similar if they are closer together. Essentially, we measure the distance from a *test* sample X^* to every sample in the *training* set X, in other words, we compute $(X^* - X_i)^2$.

We can employ Euclidean distance ($\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$) or the Manhattan distance ($|(x_2 - x_1)| + |(y_2 - y_1)|$) for this computation.

The two parameters for the algorithm are the number k and the distance algorithm used.

K-Nearest Neighbours is known as a lazy learning algorithm, which means that we don't generalize on the training dataset until we want to make a query.

2.301 Decision tree

Decision Tree is a very versatile machine learning algorithm, because they are capable of handling both classification and regressions tasks. They can also handle non-linear datasets.

This algorithm can be used a the basic classifier in Random Forests, which is among the most poweruﬂ class of machine learning algorithm.

To illustrate how Decision Tree algorithm, we will look at how we can classify Hares vs Rabbits. Figure 1 shows and example of how the decision tree could look like. When applying the algorithm to a new data to be classified, we descend through the tree until we get to a leaf node.

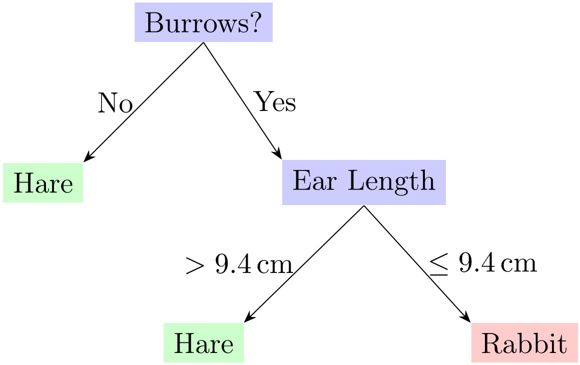

Figure 2: Decision Tree

Decision Trees are referred to as *White Box* models, this means they are easy to interpret, because of the hierarchical nature of the classification rules which is easy to visualize. Conversely, *Black Box* models make decisions in a more opaque process.

There are several types of Decision Tree algorithms, one of which is known as Classification And Regression Tree, or CART. It's a binary tree which can be used, as the name suggests, for both classification and regression.

A decision tree is prone to overfitting, which is when the model fits the training dataset perfectly; this makes it harder to generalize the trained model to other sets of data. Finding the optimal split and feature combination is an NP-complete problem.

Week 4

Key Concepts

- Explain how a simple nearest neighbour algorithm works
- Describe the Decision Tree Classifier
- Evaluate a supervised classification algorithm on a dataset

2.501 Classifier evaluation

One way of evaluating the performance of a classifier is to measure its accuracy, this, however, is not always a good measure of the quality of a classifier.

Another approach to measuring quality of the classifier is to employ a Confusion Matrix. This matrix lets us compare a true condition vs a predicted condition, Like shown in table 1.

Table 1: Confusion Matrix

	Condition Positive	**Condition Negative**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

The *True Positive Rate*, also known as Recall or Sensitivity, tells us how likely the model is to predict the correct value. It's computed as shown below:

$$TPR = \frac{TP}{TP + FN}$$

The *Precision* or *Positive Predictive Value* tells us how likely the prediction is to be correct, given a positive prediction. It's computed as shown below:

$$PPV = \frac{TP}{TP + FP}$$

The *True Negative Rate*, also known as Specificity or Selectivity, is computed as:

$$TNR = \frac{TN}{TN + FP}$$

The *False Negative Rate*, also known as Miss Rate, is computed as:

$$FNR = \frac{FN}{FN + TP}$$

The *False Positive Rate*, also known as Fall-out is computed as:

$$FPR = \frac{FP}{FP + TN}$$

2.602 Further reading

- `http://scikit-learn.org/stable/modules/neighbors.html`
- `https://scikit-learn.org/stable/modules/tree.html`
- Chapter 1, section 1.2 of the course textbook (Chollet), also briefly mentions decision tree classifiers.
- Sections 2.1 and 2.5 of Ethem Alpaydin's book provide a good overview of supervised classification: Alpaydin, E. Introduction to machine learning. (Cambridge, MA: MIT Press, 2014) 3rd edition [ISBN 9780262028189].
- Alpaydin's book also discusses decision trees in depth in Chapter 9, sections 9.1 to 9.3.

Week 5

Key Concepts

- Explain the idea behind gradient descent
- Apply linear regression on a dataset.
- Explain the concept of linear regression and interpret results.

3.102 Linear regression

Linear Regression is a method for predicting output based on a linear combination of the input. Figure 3 shows an example of this.

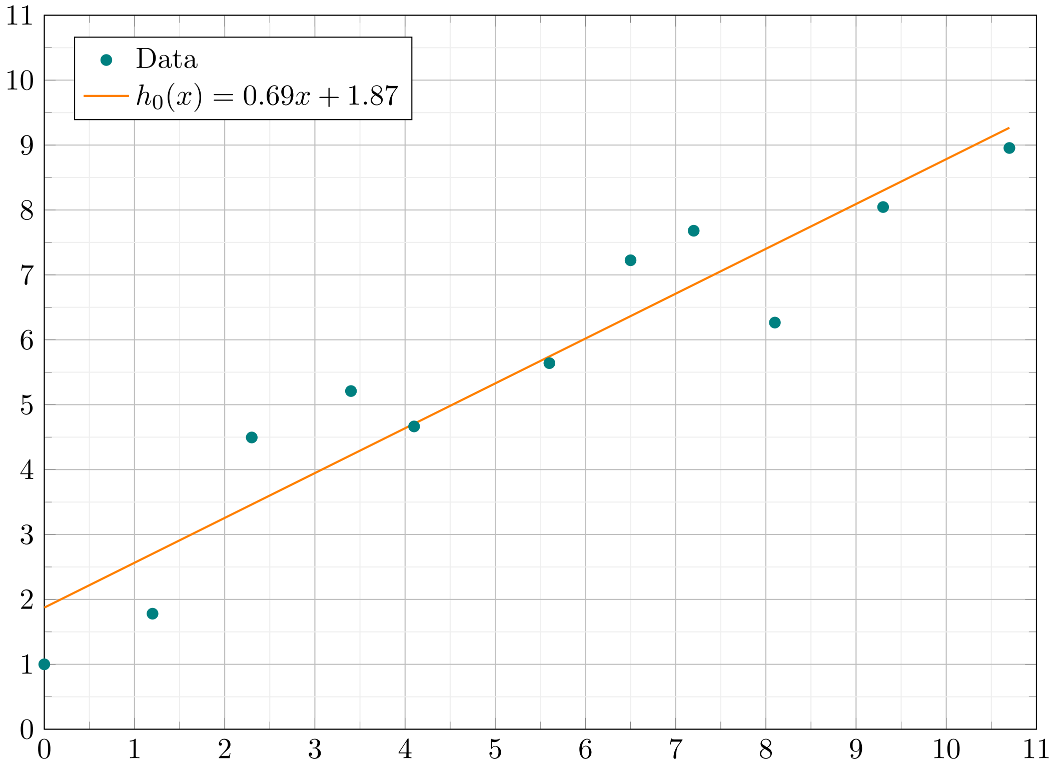

Figure 3: Linear Regression Example

The idea of Linear Regression is exactly that: try to fit a line through the data to predict new y values based on x input. There are two important parameters in the regression line: θ_0 is the y intercept point and θ_1 determines the gradient. With these two parameters we can construct the line's equation $h_0(x) = \theta_1 x_1 + \theta_0$. We call it $h_0(x)$ because it represents our *hypothesis*.

The hypothesis can also be represented as a summation:

$$h_0(x) = \sum_{j=0}^{1} \theta_j x_j$$

Where $x_0 = 1$. The same equation can be represented in vector notation:

$$h_0(x) = \begin{bmatrix} 1 & x \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

When modelling a linear regression, what we can do is plot the line anywhere and measure the *error* from the line to each of the points in the plot. The function shown below is the L2 loss or *Mean Squared Error*, it's a common function for computing error between regression line and x inputs.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)} - y^{(i)} \right)^2$$

The goal is to minimize the function $J(\theta)$, thus minimizing the error of the regression line.

3.202 Further reading

- Linear Algebra with Python
- Goodfellow, I., Y. Bengio and A. Courville Deep learning. (Cambridge, MA: MIT Press, 2017) [ISBN 9780262035613] Chapter 2 Linear Algebra

Week 6

Key Concepts

- Explain the idea behind gradient descent
- Apply linear regression on a dataset.
- Explain the concept of linear regression and interpret results.

3.301 Gradient descent in 1D

Revisiting previous lectures, we learned that we can approximate a solution for a linear regression problem by starting a random regression and trying to iteratively minimize a Loss function given by:

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

As an example, we can use a simple 3-point data as shown in figure 4. Initial θ is 0.25.

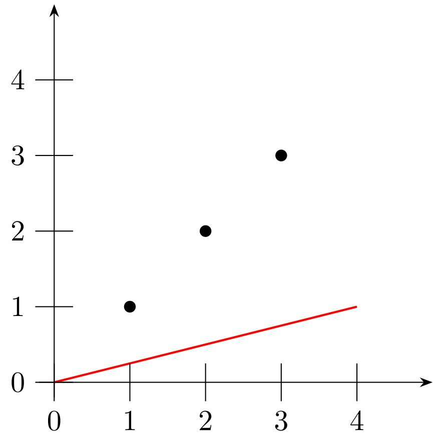

Figure 4: Input Data

We measure the distance from each point to the random regression line as shown in figure 5.

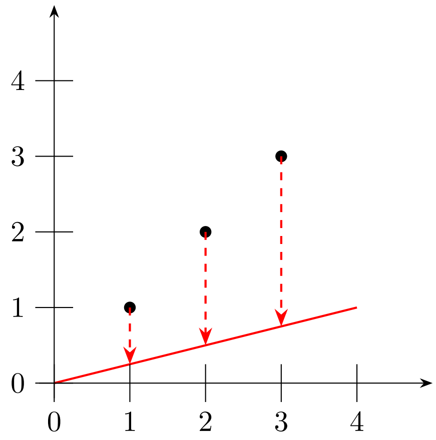

Figure 5: Input Data

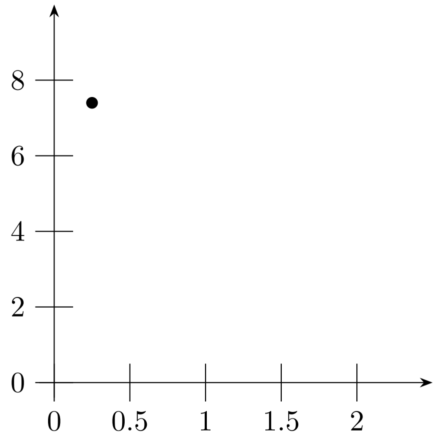

Figure 6: Loss $J(\theta)$

From this, we can compute $J(\theta_1) = 7.4$. As we compute the losses, we can plot the result in another graph, shown in figure 6.

Increasing our θ to 0.5, we get a new error, shown in figure 7. And that results in a new error $J(\theta_2) = 3.3$, which we update in our plot as shown in figure 8.

We repeat the process again with $\theta = 0.75$, which gives an error of $\theta_3 = 0.82$. The results are shown in figures 9 and 10.

We repeat this process until we find a *Global Minimum* and this is work of *Gradient Descent* algorithm. Given a convex loss function, we **know** there has to be a minimum value and gradient descent tries to find it by modifying the parameters of the loss function.

At each step of the Gradient Descent algorithm, the gradient or slope of the function is computed in order for the algorithm to make a decision of where to go next. In order to do that, we must calculate the derivative of the Loss Function.

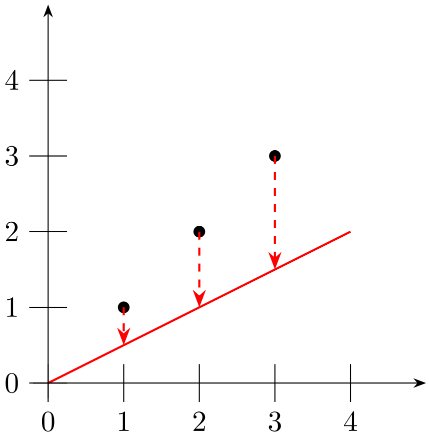

Figure 7: Input Data

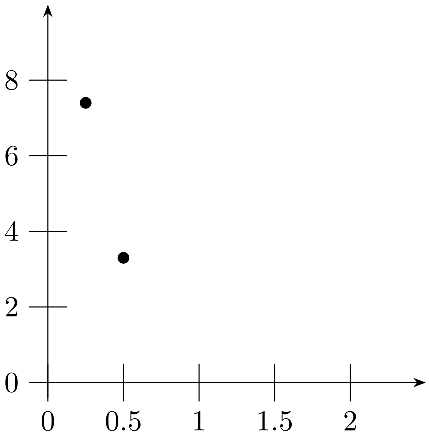

Figure 8: Loss $J(\theta)$

$$
\begin{aligned}
J_1'(\theta) &= \frac{\partial J(\theta)}{\partial \theta_1} \\
&= 2 \cdot \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^{(i)} - y^{(i)}) \cdot x_1^{(i)} \\
&= \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^{(i)} - y^{(i)}) \cdot x_1^{(i)}
\end{aligned}
$$

Gradient Descent applies a convergence rate α to the slope computed by the derivative of the loss function and uses that to compute the new value for θ_1, i.e. $\theta_1^{(2)} = \theta_1^{(1)} - \alpha J_1'(\theta_1^{(1)})$

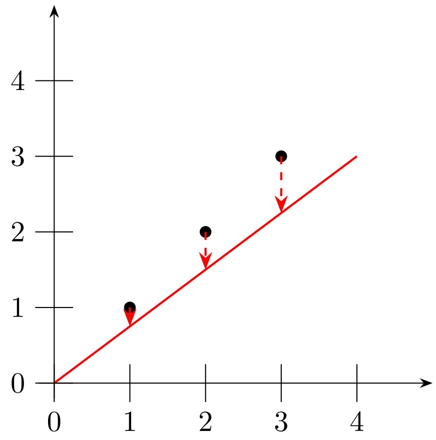

Figure 9: Input Data

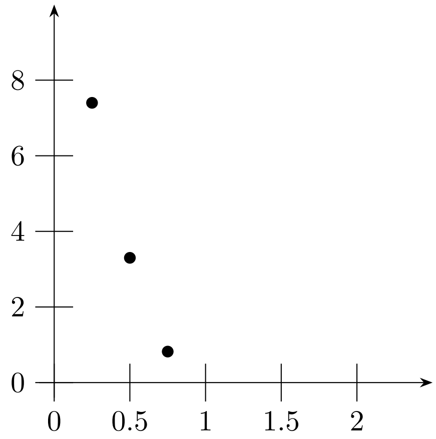

Figure 10: Loss $J(\theta)$

3.303 Gradient descent in 2D

We skipped θ_0 in previous video. The equation is shown below.

$$\begin{aligned} J_0'(\theta) &= \frac{\partial J(\theta)}{\partial \theta_0} \\ &= 2 \cdot \frac{1}{2m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^{(i)} - y^{(i)}) \\ &= \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x_1^{(i)} - y^{(i)}) \end{aligned}$$

When running gradient in higher dimensions we just compute more partial derivatives. For 3D we must compute partial derivatives with respect to θ_0, θ_1, and θ_2. For more dimensions, just add more θ_n parameters.

To summarise, the multivariate linear model is given by:

$$\begin{aligned} h_\theta(x) &= \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n \\ &= \sum_{j=0}^{n} \theta_j x_j && \text{with } x_0 = 1 \\ &= \theta^\mathsf{T} x && \text{with } x_0 = 1 \end{aligned}$$

3.305 Data scaling

When using multivariate data we can run into situations where the scale for each of the features in the data can be vastly different, as shown in table 2 below.

Table 2: Data Scale Can be Different

x_1	x_2	x_3	x_4	y
0	0	10000	1	13.1
16	12000	7000	0	17.8
8	500	7000	0	10.3
45	10000	5000	1	23.0
65	1000	12000	1	30.8

We can sort this out by scaling each feature so they all sit in a similar range. This is referred to as *Feature Scaling.* A common way of achieving this is *min-max normalisation*, which can be achieved with the equation below:

$$x_j^s = \frac{x_j - min(x_j)}{max(x_j) - min(x_j)}$$

Applying this to the previous table 2 results in the scaled table 3 shown below.

What this means in practice is that Gradient Descent runs faster when every dimension is of comparable range.

There are other normalisation techniques, such as Range Normalisation, which achieve the same thing by slightly different method. With Range Normalisation we *center* the data around the mean.

$$x_j^s = \frac{x_j - mean(x_j)}{max(x_j) - min(x_j)}$$

Table 3: Data Scale Corrected With Min-Max Normalisation

x_1^s	x_2^s	x_3^s	x_4^s	y
0	0	0.7	1	13.1
0.24	1	0.3	0	17.8
0.12	0.04	0.3	0	10.3
0.69	0.83	0	1	23.0
1	0.08	1	1	30.8

A third approach is called Standardization, or z-score.

$$x_j^s = \frac{x_j - mean(x_j)}{std(x_j}$$

3.306 Polynomial regression

If a linear regression doesn't fit the data very well, we can try increasing the number of θ terms to try to better fit the data.

3.402 Further reading

- Chapter 2, section 2.4 of the course textbook (Chollet)

Week 7

Key Concepts

- Explain how regularisation works.
- Explain the concept of cross-validation.
- Explain the effect of overfitting.

4.101 Overfitting and underfitting

A regression that's too simple to fit the data is said to *underfit* the data, or has a High Bias. An example of which is shown in figure 11.

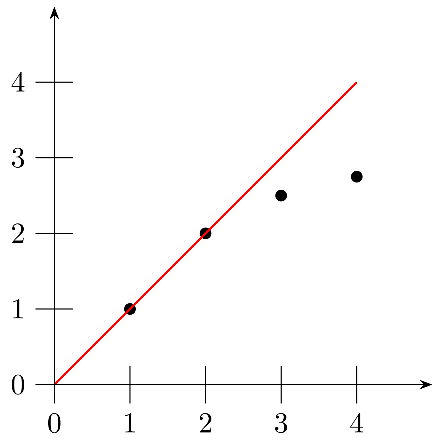

Figure 11: Underfit

A regression with a high degree that fits the data too perfectly is said to overfit the data[1], or has a high variance.

We can evaluate a model with the Bias-variance curve, as shown in figure 12. At the left side of the graph, we have underfitting (high bias), at the right side we have overfitting (high variance). We want to find a model that sits in the middle of the Bias-variance curve.

To ensure a model is generalisable, we can employ a number of techniques:

1. Reduce the number of features

[1] Not drawing that

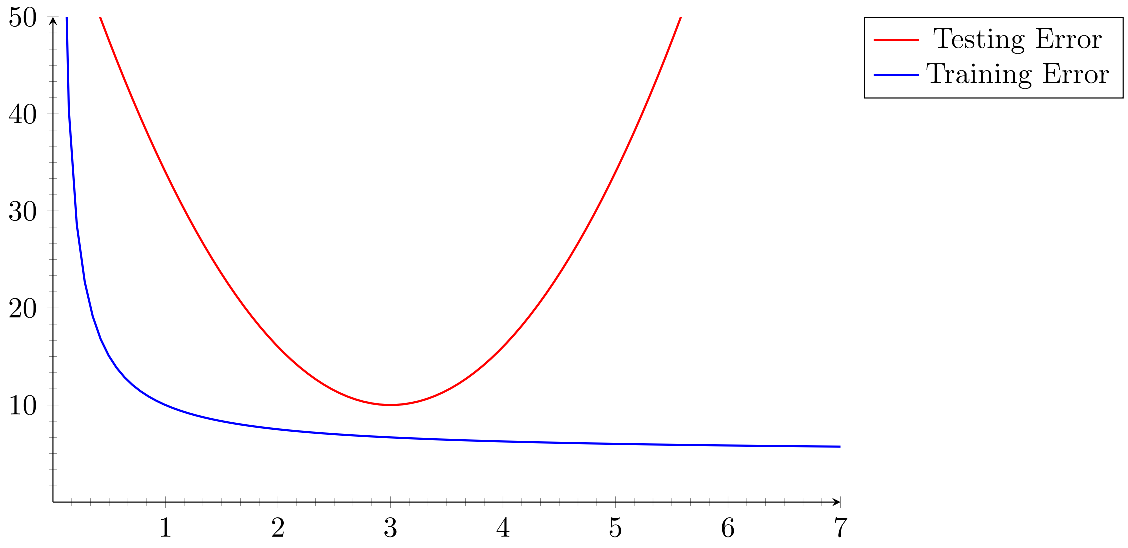

Figure 12: Bias-variance Curve

- Manually select which features to keep
- Use model selection algorithm (e.g. cross-validation)

2. Regularisation
    - Keep all features, but reduce the values of θ_j
    - Works well when we have a lot of features

4.201 Regularisation

Regularisation is a method of penalizing complexity in a Machine Learning Model. There are several regularisation methods, one of them, called, L2 regularisation, involves squaring and summing up all the θ parameters. In other words

$$\sum \theta^2$$

computes the *penalty* in question, the goal being reducing the penalty.

Alternatively, L1 regularisation

$$\sum |\theta|$$

involves adding up the absolute values of the θ parameters. We will rely on L2 for the time being.

Looking back at our linear regression Loss function

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

to regularize this loss function, we add a new regularisation term, thus

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2$$

Note that the regularisation hyperparamter λ must be tuned:

λ **too big** results in **underfitting**

λ **too small** results in **overfitting**

4.301 Cross-validation

In order to avoid overfitting, it's important that the data used to train the algorithm is **not** the same the data used to test the algorithm. In practice, from our input data we create two disjoint sets called *Training Data* and *Test Data*; one dedicated to training the model and the other dedicated to evaluating the performance of the model.

Usually, we don't have enough data to carry out this process, an approach that maximizes the use of our daata is called *N-fold Cross-Validation.* In summary, we will run the process of splitting the data into test and training sets, train and evaluate the model multiple times, an example of this is depicted in figure 13.

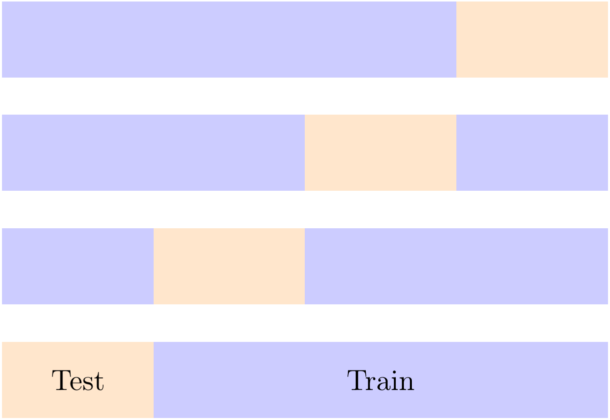

Figure 13: N-fold Cross Valication

The total error in this case is the sum of each of the errors for each of the N splits. In other words

$$e_{total} = \frac{1}{n}\sum_{i=1}^{n} e_i$$

is the total error.

Week 8

Key Concepts

- Discuss the difference between generative and discriminative models
- Describe the Naive Bayes classifier
- Explain Bayes' rule

5.101 Bayesian classification

Bayesian modelling is centered around the concept that new evidence can change decisions. The basic components of a base model are:

Prior initial degree of belief in some preposition

Posterior degree of belief after seeing some evidence

Ultimately, we want to calculate the posterior given evidence. This is, however, difficult to compute directly. We can use an indirect approach using a generative model of the likelihood that some outcome leads to a particular observation. Bayes Theorem helps in this case.

Bayes Theorem is given by:

$$Posterior = \frac{Likelihood \cdot Prior}{MarginalProbability}$$

Some probability rules:

Inverse $P(\bar{A}) = 1 - P(A)$

Conditional $P(B \mid A)$

Product Rule $P(B, A) = P(B \mid A)P(A) = P(A \mid B)P(B)$

Sum Rule $P(B) = P(B, A) + P(B, \bar{A}) = P(B \mid A)P(A) + P(B \mid \bar{A})P(\bar{A})$

Bayes Theorem $P(B \mid A) = \dfrac{P(A \mid B)P(B)}{P(A)}$

5.103 The Naive Bayes Classifier

Given by:

$$P(Y \mid X_1, X_2, \dots, X_F) \alpha P(Y) \cdot \prod_{i=1}^{F} P(X_i \mid Y)$$

This results in very small numbers which are susceptible to underflow. A solution is use a logarithm scale:

$$\log(P(Y \mid X_1, X_2, \dots, X_F)) \alpha \log(P(Y)) + \sum_{i=1}^{F} \log(P(X_i \mid Y))$$

The final formulation for the Naive Bayes Classifier is:

$$NB = argmax_Y(\log(P(Y)) + \sum_{i=1}^{F} \log(P(X_i \mid Y)))$$

5.202 Further reading

- Probabilisitic modelling is covered in sections 3.1 and 3.2 of Ethem Alpaydin's book: Alpaydin, E. Introduction to machine learning. (Cambridge, MA: MIT Press, 2014) 3rd edition [ISBN 9780262028189].

Week 9

Key Concepts

- explain the concepts of clustering and dimensionality reduction
- implement the k-means algorithm
- explain principal component analysis (PCA) and its properties.

6.102 Clustering

With Unsupervised Learning, the goal is to learn patterns from the data without having any labels assigned. In applications of clustering, the goal is to separate our data points into groups based on some sort of similarit index. The K-Means algorithm is one implementation of this basic concept.

Given figure 14 below, we want separate the points into disjoint sets similarly to he one shown in 15 that follows.

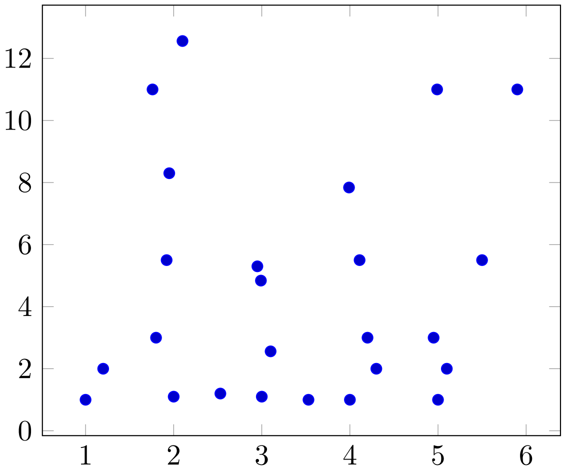

Figure 14: Scatter plot

6.103 K-Mean

The K-Means algorithm works similarly to the steps below:

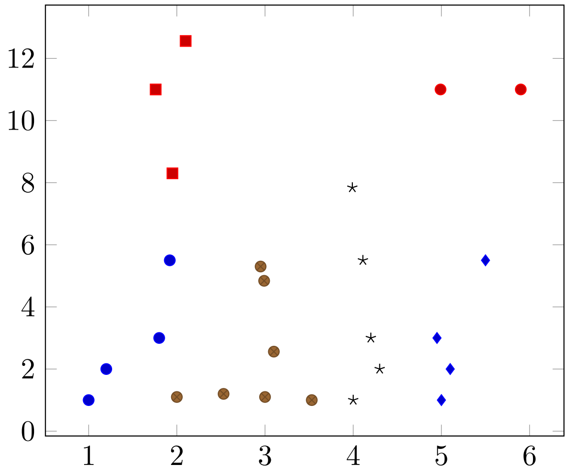

Figure 15: Scatter plot groupped

1. Initialize k centroids randomly
2. Calculate the distance from every data point to each of the k centroids
3. Assign data points to the nearest k_i centroid
4. Re-assign centroids as the mean of each cluster's data
5. Repeat 2–4 until convergence

Scaling the data with a min-max scaling algorithm is usually required to avoid certain phenomena that depend on scale of each of the axis. Min-max is given by

$$x = \frac{x - min(x)}{max(x) - min(x)}$$

which will make sure that every axis falls within the interval $[0, 1]$.

Week 10

Key Concepts

- explain the concepts of clustering and dimensionality reduction
- implement the k-means algorithm
- explain principal component analysis (PCA) and its properties.

6.301 Dimensionality reduction

Dimensionality reduction is particularly useful when dealing with images, because they take a lot of space. Principal Component Analysis is a common algorithm for dimensionality reduction. In summary, PCA is the process of computing the main components of the input data and working with the first few of them, rather than the full data input.

6.302 PCA

Given data X with F features and N samples

$$\mathbf{X} = \big[x^{(1)}, x^{(2)}, \ldots, x^{(N)}\big],$$

where $\mathbf{X} \in \mathbb{R}^{F \times N}$, the sample mean is then

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)},$$

where $x^{(i)} \in \mathbb{R}^{F \times 1}$. The covariance is

$$cov(\mathbf{X}) = \mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}\big(x^{(i)} - \bar{x}\big)\big(x^{(i)} - \bar{x}\big)^{\mathsf{T}}$$

The basic idea of PCA is that we're trying to linearly transform a set of data from one dimension into another.

$$\mathbf{Y} = \mathbf{W}^{\mathsf{T}}\mathbf{X}^{2}$$

For example, if we're mapping from 3 dimensions to 2 dimensions, could train our algorithm to produce the matrix

$$\mathbf{W} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}$$

and the data

$$\mathbf{X} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

, then

$$\begin{aligned} \mathbf{Y} &= \mathbf{W}^\mathsf{T}\mathbf{X} \\ &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \\ &= \begin{bmatrix} 1 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 \\ 0 \cdot 1 + 2 \cdot 2 + 0 \cdot 3 \end{bmatrix} \\ &= \begin{bmatrix} 4 \\ 4 \end{bmatrix} \end{aligned}$$

We have the relationship that

$$cov(\mathbf{Y}) = cov(\mathbf{W}^\mathsf{T}\mathbf{X}) = \mathbf{W}^\mathsf{T} cov(\mathbf{X})\mathbf{W}$$

We can let $\mathbf{S} = cov(\mathbf{X})$ which simplifies our notation to $cov(\mathbf{Y}) = \mathbf{W}^\mathsf{T}\mathbf{S}\mathbf{W}$. The goal of our model to train PCA is the maximize the covariance of $\mathbf{Y}$, in order words

$$\arg\max_{\mathbf{W}}(\mathbf{W}^\mathsf{T}\mathbf{S}\mathbf{W})$$

is our goal, such that $\mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}$. One one to solve this problem is to use *eigenanalysis*, which will give us the optimal values for $\mathbf{W}$.

$$[\mathbf{W}, \mathbf{\Lambda}] = eig(\mathbf{S}),$$

where $\mathbf{S} = cov(\mathbf{X})$.

6.402 Further reading

- Sections 6.1 and 6.3 from Chapter 6 of Ethem Alpaydin's book discuss dimensionality reduction and PCA. Sections 7.1 to 7.3 from Chapter 7 discuss clustering (including k-means): Alpaydin, E. Introduction to machine learning. (Cambridge, MA: MIT Press, 2014) 3rd edition [ISBN 9780262028189].