Lecture 2

We will begin with linear methods, studying one example from supervised learning and one example from unsupervised learning.

2 Introduction to (linear) kernels and supervised learning

We want to define a notion of similarity between two points $x, x' \in \mathcal{X}$. To start we shall consider one of the form:

$$\begin{aligned} k :& \mathcal{X} \times \mathcal{X} \to \mathbb{R} \\ & (x, x') \mapsto k(x, x'). \end{aligned}$$

The function k is called a *kernel*. It returns a real number that measures the similarity between x and x'. Unless otherwise stated, we assume k is symmetric so that:

$$k(x, x') = k(x', x).$$

2.1 Dot product kernel

Adapted from [1, Chapter 1.1].

Let's start with a simple example which we will generalize in several directions. Suppose $\mathcal{X} = \mathbb{R}^p$, so that:

$$x = (x[1], \ldots, x[p]) \in \mathbb{R}^p.$$

A natural measure of similarity is the *inner product*. In $\mathbb{R}^p$, the canonical inner product is the *dot product*, which is given by:

$$\langle x, x' \rangle = x \cdot x' = \sum_{i=1}^{p} x[i]x'[i].$$

Recall from linear algebra that we can:

1. Compute the length of x, also known as the *norm* $\|x\|$:

$$\langle x, x \rangle = \|x\|^2.$$

2. Compute the angle θ between x, x',

$$\langle x, x' \rangle = \|x\| \|x'\| \cos\theta.$$

Thus the kernel k defined as:

$$k(x, x') = \langle x, x' \rangle,$$

allows us to consider geometric properties of $\mathcal{X}$ that can be formulated with angles, lengths, and distances. It is simultaneously a simple kernel, but one that later we will see serves as the foundation of many kernel learning algorithms. For now we will use it in a simple classification algorithm, to illustrate how kernels are used in supervised learning.

2.2 Simple binary classification algorithm

Adapted from [1, Chapter 1.2]

We are going to define a binary classification algorithm that assigns a new data point x to the class with closer mean.

Let $\mathcal{Y} = \{-1, +1\}$ and define the mean of each class as:

$$\mu_\epsilon = \frac{1}{n_\epsilon} \sum_{\{i : y_i = \epsilon\}} x_i, \quad \epsilon = +1, -1,$$

and where n_ϵ is number of examples with the ϵ label. The algorithm will assign a label to x based on its proximity to these means. In particular the class with the closer mean is the assigned label:

$$y(x) = \arg \min_{\epsilon = \{+1, -1\}} \|x - \mu_\epsilon\|. \tag{1}$$

We formulate (1) in terms of the dot product. Let c be the point between μ_+ and μ_-:

$$c = (\mu_+ + \mu_-)/2.$$

We formulate (1) by checking weather the vector $x - c$ encloses an angle smaller than $\pi/2$ with the vector

$$w = \mu_+ - \mu_-.$$

This leads to:

$$\begin{aligned} y(x) &= \operatorname{sgn}\langle x - c, w\rangle \\ &= \operatorname{sgn}\langle x - (\mu_+ + \mu_-)/2, \mu_+ - \mu_-\rangle \\ &= \operatorname{sgn}(\langle x, \mu_+\rangle - \langle x, \mu_-\rangle + b) \\ &= \operatorname{sgn}(\langle x, w\rangle + b), \end{aligned} \tag{2}$$

where $b \in \mathbb{R}$ is the offset defined as:

$$b = \frac{1}{2}\left(\|\mu_-\|^2 - \|\mu_+\|^2\right).$$

Note that (2) induces a decision boundary which has the form of a hyperplane. See Figure 1 for the geometric picture of the above calculation.

We can rewrite (2) in terms of the kernel $k(x, x') = \langle x, x'\rangle$ and the samples x_i as:

$$y(x) = \operatorname{sgn}\left(\frac{1}{n_+}\sum_{\{i\,:\,y_i=+1\}} k(x, x_i) - \frac{1}{n_-}\sum_{\{i\,:\,y_i=-1\}} k(x, x_i) + b\right), \tag{3}$$

where b can also be rewritten as:

$$b = \frac{1}{2}\left(\frac{1}{n_-^2}\sum_{\{(i,j)\,:\,y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{n_+^2}\sum_{\{(i,j)\,:\,y_i=y_j=+1\}} k(x_i, x_j)\right).$$

As we shall later, it will be (extremely) useful to take kernels k other than the dot product in $\mathbb{R}^p$. To illustrate the connection of the above with statistics, for now let us assume that $\|\mu_+\| = \|\mu_-\|$ so that $b = 0$, and that the kernel k is bi-stochastic, meaning:

$$\forall\, x' \in \mathcal{X}, \quad \int_{\mathcal{X}} k(x, x')\, dx = 1.$$

Thus $k(\cdot, x')$ is a probability density for any $x' \in \mathcal{X}$. Now suppose that the two class patterns were generated by sampling from two probability distributions, p_+ and p_-. The function

$$\widetilde{p}_+(x) = \frac{1}{n_+}\sum_{\{i\,:\,y_i=+1\}} k(x, x_i)$$

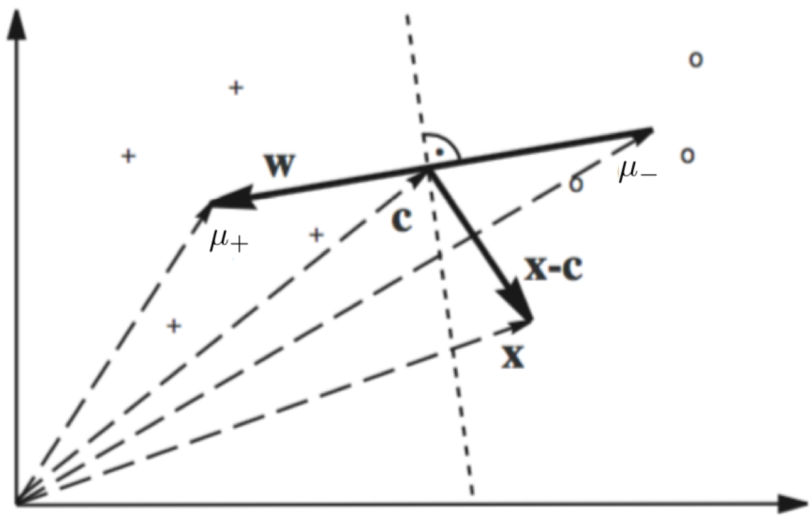

Figure 1: [1, Figure 1.1] A simple geometric classification algorithm: given two classes of points (depicted by $+$ and $\circ$), compute their means μ_+ and μ_- and assign a test point x to the one whose mean is closer. This can be done by looking at the dot product between $x - c$ (where $c = (\mu_+ + \mu_-)/2$) and $w = \mu_+ - \mu_-$, which changes sign as the enclosed angle passes through $\pi/2$. Note that the corresponding decision boundary is a hyperplane (the dotted line) orthogonal to w.

is the *kernel density estimator* of the probability distribution p_+, and similarly

$$\widetilde{p}_-(x) = \frac{1}{n_-} \sum_{\{i\,:\,y_i=-1\}} k(x, x_i)$$

is the kernel density estimator for p_-. Equation (3) then reads:

$$y(x) = \operatorname{sgn}(\widetilde{p}_+(x) - \widetilde{p}_-(x)).$$

Thus the label is computed by simply checking which of the two values of $\widetilde{p}_+(x)$ or $\widetilde{p}_-(x)$ is larger. This is known as a *Bayes classifier*, and it is the best decision we can make with no prior information about the two underlying distributions.

The classifier (3) is a particular case of the more general kernel classifier:

$$y(x) = \operatorname{sgn}\left(\sum_{i=1}^{n} \alpha_i k(x, x_i) + b\right).$$

We will study these in more detail later. As we shall see, the key is in the selection of the kernel k, although there are various ways of selecting the weights α_i which also affect performance.

Exercises

Exercise 1. Implement the binary classifier, either directly with (1) or through the kernelized form (3). Submit the script or function.

Exercise 2. Download the MNIST data set from D2L and load it into MATLAB. Notice that it contains two variables - one consisting of all of the images $\mathcal{X}$, the other consisting of the labels $\mathcal{Y}$. Visualize a few of the images to convince yourself the data is loaded properly (you can use, for example, `imagesc`), and check the label. Now extract all of the "one" images and all of the "two" images. Use 80% of the "ones" and 80% of the "twos" to train your binary classifier. Test the binary classifier on the remaining images. What is the performance on each of the two classes?

Exercise 3. Now try some other pairs of numbers. Which pairs does the classifier do better on? Which pairs does it do worse? Plot a few of the digits that are misclassified (make sure to say which two you are using). Can you explain the performance and misclassified images?

Exercise 4. Describe in words how you might generalize your binary classifier to a multiclass classifier that works on the whole MNIST data set at once.

References

[1] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. The MIT Press, 2002.

[2] Afonso S. Bandeira. Ten lectures and forty-two open problems in the mathematics of data science. MIT course *Topics in Mathematics of Data Science*, 2015.

[3] Jon Shlens. A tutorial on principal component analysis. arXiv:1404.1100, 2014.

[4] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine, Series 6*, 2(11):559–572, 1901.