MA4270: Classification, The Perceptron Algorithm, Its Variants, and the Geometric Margin

Vincent Y. F. Tan

August 14, 2021

I describe the classification problem here as well as some variations of the standard perceptron algorithm I mentioned in Lecture 1 and 2. You should know Sections 1, 2 and 3 here. Other sections are for interest. Some notation: Given two vectors $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{R}^d$, define their inner product and ℓ_2-norm in the usual way, i.e.,

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{d} a_i b_i \tag{1}$$

and

$$\|\mathbf{a}\| = \sqrt{\langle \mathbf{a}, \mathbf{a} \rangle} = \sqrt{\sum_{i=1}^{d} a_i^2}. \tag{2}$$

These notes are heavily adapted from Jaakkola's MIT notes.

1 Classification

The most canonical task is classification. This problem takes the following form. We have a *dataset* $\mathcal{D}$ which consists of a certain number m of *data examples* $(\mathbf{x}_t, y_t), t = 1, \ldots, n$. Each data example $(\mathbf{x}_t, y_t)$ consists of a *feature vector* (also called *training sample* or *training example*)

$$\mathbf{x}_t = \begin{bmatrix} x_{t,1} \\ x_{t,2} \\ \vdots \\ x_{t,d} \end{bmatrix} \quad \text{or} \quad \mathbf{x}_t = \begin{bmatrix} x_{t,1} & x_{t,2} & \ldots & x_{t,d} \end{bmatrix}^\top, \tag{3}$$

(usually an element of d-dimensional Euclidean space $\mathbb{R}^d$) and a *label* y_t. The crux of classification is that the label y_t can only take on *finitely many values* so $y_t \in \{1, 2, \ldots, c\}$ for some integer $c \geq 2$, where c is the number of classes. We do not allow y_t to take on infinitely many values.

The classification problem consists in finding a *classifier* which is a function $f : \mathbb{R}^d \to \{1, 2 \ldots, c\}$ such that it accurately predicts labels given new samples, called *test samples*. This function f is constructed or learned based on the dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t) : 1 \leq t \leq n\}$.

A couple of examples will illustrate this point.

- We have records of $n = 100$ emails. Let $d = 2$. Thus, there are two components in each $\mathbf{x}_t$. Furthermore the first component of $\mathbf{x}_t$, denoted as $x_{t,1}$, counts the number of times the word "love" appears. The second component of $\mathbf{x}_t$, denoted as $x_{t,2}$, counts the number of times the word "money" appears. Each label $y_t \in \{-1, 1\}$ where $y_t = 1$ means that the i^{th} email is spam while $y_t = -1$ means that the t^{th} email is non-spam. Based on the dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t) : 1 \leq t \leq 100\}$ of $n = 100$ emails, we are tasked to learn a *classifier* $f : \mathbb{R}^2 \to \{-1, 1\}$ such that we can accurately predict whether the next email you

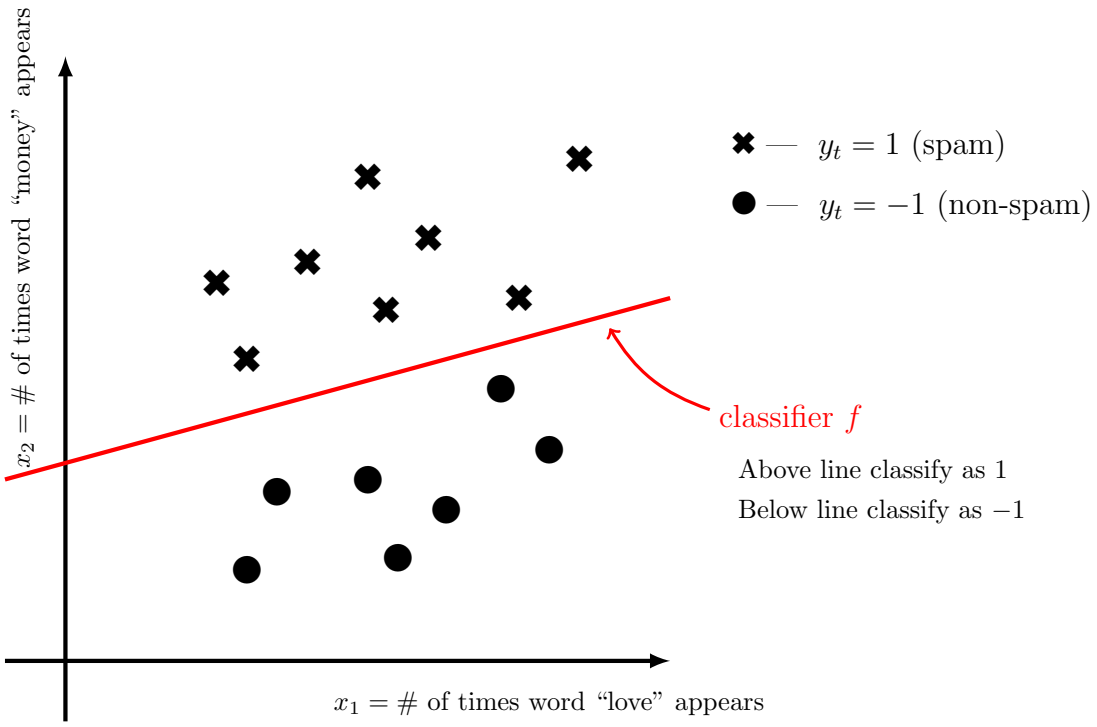


Figure 1: Spam classification

receive (which is of course not in the training dataset) is spam or not. See Fig. 1. In this case in which there are only two classes, we refer to this as *binary* classification.

- Consider an image classification problem. We are given a dataset of size $n = 10^6$, namely, $\mathcal{D} = \{(\mathbf{x}_t, y_t) : 1 \le i \le 10^6\}$ where each $\mathbf{x}_t$ represents an image that is vectorized into a vector. For example, each image contains $5 \times 5 = 25$ pixels and for the sake of simplicity, each pixel value is 0 or 1. Thus $\mathbf{x}_t \in \{0,1\}^{5\times 5} \cong \{0,1\}^{25}$ and $d = 25$. Of course, we can have more complicated images that have multiple quantization levels (not restricted to binary) and colors as well, but the treatment will be the same. To each image, there is a label y_t which can take on 10 values so $c = 10$. These 10 values signify what object is in the image, e.g., $y_t \in \{\text{dog}, \text{cat}, \ldots, \text{snake}\}$. We would now like to design a image classifier $f : \{0,1\}^{25} \to \{\text{dog}, \text{cat}, \ldots, \text{snake}\}$ that "does well" (in some sense) on new images (or test images) that contain one of the 10 animals. Since there are more than two classes in this scenario, we refer to this as *multi-class* classification.

Classification belongs to the class of *supervised* learning methods.

1.1 Linear Classifiers and Training Error

An important family of classifiers is *linear classifiers* take the form

$$\text{Predict Positive Label} \quad \iff \quad f_{\boldsymbol{\theta}}(\mathbf{x}) = \langle \boldsymbol{\theta}, \mathbf{x} \rangle > 0 \tag{4}$$

for some $\boldsymbol{\theta} \in \mathbb{R}^d$.

For any classifier, say linear, the *training error* is

$$\hat{E}(\boldsymbol{\theta}) := \frac{1}{n} \sum_{t=1}^{n} \text{Loss}(y_t, f_{\boldsymbol{\theta}}(\mathbf{x}_t)), \tag{5}$$

where the *zero-one loss* is

$$\text{Loss}(y_t, \hat{y}_t) := \mathbb{1}\{y_t \neq \hat{y}_t\} = \begin{cases} 1 & y_t \neq \hat{y}_t \\ 0 & y_t = \hat{y}_t \end{cases} . \tag{6}$$

The word "training" refers to the fact that we are evaluating on the data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ that is used for finding $\boldsymbol{\theta}$. Later, we will introduce the notion of "test error", in which we evaluate the error on different data that we haven't seen previously (this is what we are ultimately interested in being able to do! The linearly separable assumption means there exists $\boldsymbol{\theta}$ such that $\hat{E}(\boldsymbol{\theta}) = 0$ where $f_{\boldsymbol{\theta}}(\mathbf{x})$ is a linear function.

2 Standard Perceptron Algorithm

We assume that the dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \{-1, +1\} : t = 1, \ldots, n\}$ is *linearly separable*. This means that there exists $\boldsymbol{\theta}^* \in \mathbb{R}^d$ and $\gamma > 0$ such that

$$y_t \langle \boldsymbol{\theta}^*, \mathbf{x}_t \rangle \geq \gamma \tag{7}$$

for all $t = 1, \ldots, n$. If (7) holds for a dataset $\mathcal{D}$, we also say that $\mathcal{D}$ is γ-*linearly separable*. For convenience, we may define

$$\gamma := \min_{t \in \{1,\ldots,n\}} y_t \langle \boldsymbol{\theta}^*, \mathbf{x}_t \rangle. \tag{8}$$

Note that γ depends on $\boldsymbol{\theta}^*$ but we do not make this dependence explicit as it does not affect the arguments below. Furthermore, we also assume that the dataset is R-*bounded*, i.e.,

$$\max_{t \in \{1,\ldots,n\}} \|\mathbf{x}_t\| = R < \infty. \tag{9}$$

All finite datasets are obviously bounded, i.e., R-bounded for some $R \in (0, \infty)$. But if we have an infinite dataset (i.e., $|\mathcal{D}| = \infty$), we need to use a sup in (9) and $\mathcal{D}$ obviously may not be bounded. We will explore this in the tutorial.

The perceptron algorithm works on a dataset $\mathcal{D}$ that is γ-linearly separable for some $\gamma > 0$. We seek a parameter vector $\boldsymbol{\theta}$ such that $y_t \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle \geq 0$ for all t so that all the training samples are classified correctly. Thus, the classification function or classifier mentioned in Section 1 is

$$f(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sgn}(\langle \boldsymbol{\theta}, \mathbf{x} \rangle). \tag{10}$$

where sgn outputs 1 (resp. -1) if the argument is positive (resp. negative). We do not worry what happens when the argument is 0. This is called a *linear classifier* as it partitions the data space $\mathbb{R}^d$ into two halfspaces. We will study *linear classifiers* extensively in the following lectures.

The perceptron algorithm proceeds as follows:

1. Arbitrarily initialize $\boldsymbol{\theta}^{(0)}$ (say to $\mathbf{0}$). This is the initial estimate of the parameter vector.
2. Cycle through all the training samples. If $y_t \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle \leq 0$ (i.e., there is disagreement between the current prediction and the true label), then
$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + y_t \mathbf{x}_t. \tag{11}$$

Matlab code is provided in Appendix A.

A natural question is whether the algorithm above will converge in a finite number of steps such that the update in (11) will no longer be necessary.

Before we state and prove an important theorem, let us look at the case in which there is only *one* (non-zero) data sample in $\mathcal{D} = \{(\mathbf{x}_1, y_1)\}$. If we initialize $\boldsymbol{\theta}^{(0)} \neq \mathbf{0}$, then the criterion $y_1 \langle \boldsymbol{\theta}^{(0)}, \mathbf{x}_1 \rangle < 0$ is not satisfied; hence, we need to update using (11). Now, we will have

$$\begin{aligned} y_1 \langle \boldsymbol{\theta}^{(1)}, \mathbf{x}_1 \rangle &= y_1 (\langle \boldsymbol{\theta}^{(0)} + y_1 \mathbf{x}_1), \mathbf{x}_1 \rangle & (12) \\ &= y_1 (\langle \boldsymbol{\theta}^{(0)}, \mathbf{x}_1 \rangle + y_1 \langle \mathbf{x}_1, \mathbf{x}_1 \rangle) & (13) \\ &= y_1 \langle \boldsymbol{\theta}^{(0)}, \mathbf{x}_1 \rangle + \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & (14) \\ &= y_1 \langle \boldsymbol{\theta}^{(0)}, \mathbf{x}_1 \rangle + \|\mathbf{x}_1\|^2 & (15) \end{aligned}$$

Thus, $y_1\langle \boldsymbol{\theta}^{(t)}, \mathbf{x}_1\rangle$ increases by $\|\mathbf{x}_1\|^2$, a fixed positive quantity at each iteration. Thus, it is easy to see that after

$$\left\lceil \frac{|\langle \boldsymbol{\theta}^{(0)}, \mathbf{x}_1\rangle|}{\|\mathbf{x}_1\|^2} \right\rceil \tag{16}$$

iterations, we will have $y_1\langle \boldsymbol{\theta}^{(t)}, \mathbf{x}_1\rangle \geq 0$, which means that $\mathbf{x}_1$ is eventually classified correctly.

Somewhat surprisingly, we have the following general guarantee due to Novikoff [1].

Theorem 1 (Novikoff (1962))**.** *Initialize $\boldsymbol{\theta}^{(0)}$ to be the zero vector $\mathbf{0}$. The total number of updates of the perceptron algorithm is bounded as*

$$k \leq \frac{R^2\|\boldsymbol{\theta}^*\|^2}{\gamma^2}. \tag{17}$$

Proof. We show that the sequence of parameter vectors $\{\boldsymbol{\theta}^{(k)}\}$ becomes "increasingly aligned" with $\boldsymbol{\theta}^*$. We do this in two steps. First, we show that the inner product $\langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k)}\rangle$ increases *at least linearly* with each update. Second, we show that the squared norm $\langle \boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)}\rangle$ increases at most linearly in the number of updates k. By combining the two we can show that the cosine of the angle between $\boldsymbol{\theta}^*$ and $\boldsymbol{\theta}^{(k)}$ has to increase by a finite increment due to each update. Since cosine is bounded by one, it follows that we can only make a finite number of updates.

Consider their inner product when a mistake is made (or equivalently an update is performed):

$$\begin{aligned}
\langle \boldsymbol{\theta}^{(k+1)}, \boldsymbol{\theta}^*\rangle &= \langle \boldsymbol{\theta}^{(k)} + y_t\mathbf{x}_t, \boldsymbol{\theta}^*\rangle && (18)\\
&= \langle \boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^*\rangle + y_t\langle \mathbf{x}_t, \boldsymbol{\theta}^*\rangle && (19)\\
&\geq \langle \boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^*\rangle + \gamma && (20)
\end{aligned}$$

where in the last line, we used (7). Since $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, this means that

$$\langle \boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^*\rangle \geq k\gamma. \tag{21}$$

Next consider the square of the norm of the sequence of parameter vectors $\{\boldsymbol{\theta}^{(k)}\}$. We have

$$\begin{aligned}
\|\boldsymbol{\theta}^{(k+1)}\|^2 &= \|\boldsymbol{\theta}^{(k)} + y_t\mathbf{x}_t\|^2 && (22)\\
&= \|\boldsymbol{\theta}^{(k)}\|^2 + 2\langle \boldsymbol{\theta}^{(k)}, y_t\mathbf{x}_t\rangle + \|\mathbf{x}_t\|^2 && (23)\\
&\leq \|\boldsymbol{\theta}^{(k)}\|^2 + \|\mathbf{x}_t\|^2 && (24)\\
&\leq \|\boldsymbol{\theta}^{(k)}\|^2 + R^2 && (25)
\end{aligned}$$

where the second equality follows because $y_t \in \{-1, +1\}$, the first and second inequalities follow be due to the fact that an update occurs iff $y_t\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_t\rangle \leq 0$ and (9) respectively. As a result and due the fact that $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, one has

$$\|\boldsymbol{\theta}^{(k)}\|^2 \leq kR^2. \tag{26}$$

Now, the Cauchy-Schwarz inequality states that for any two vectors a, b, we have $\langle \mathbf{a}, \mathbf{b}\rangle \leq \|\mathbf{a}\|\|\mathbf{b}\|$. Applying this inequality to $\boldsymbol{\theta}^{(k)}$ and $\boldsymbol{\theta}^*$, we obtain

$$\frac{\langle \boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^*\rangle}{\|\boldsymbol{\theta}^{(k)}\|\|\boldsymbol{\theta}^*\|} \leq 1. \tag{27}$$

Now applying the bounds in (21) and (26), we obtain

$$\frac{k\gamma}{\sqrt{kR^2}\|\boldsymbol{\theta}^*\|} \leq 1. \tag{28}$$

Rearranging this establishes the claim in (17). □

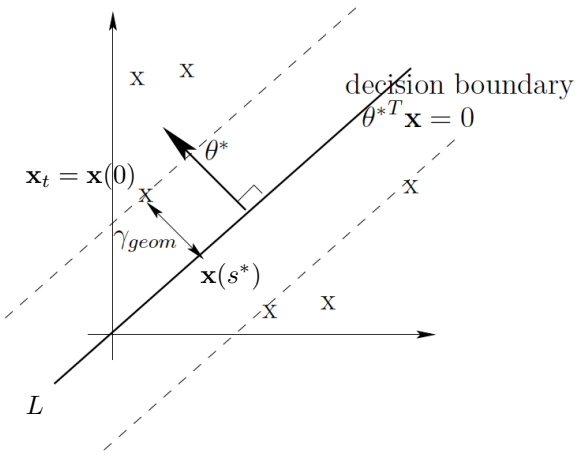


Figure 2: Illustration of the geometric margin

3 Geometric Margin

From Theorem 1, it seems like $\|\boldsymbol{\theta}^*\|/\gamma$ controls the "difficulty" of the classification problem. Is this indeed so? We now claim that its inverse $\gamma/\|\boldsymbol{\theta}^*\|$ is the smallest distance over all the training samples to the line $L = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{x}, \boldsymbol{\theta}^* \rangle = 0\}$. This line is called the *decision boundary*. In other words, $\gamma/\|\boldsymbol{\theta}^*\|$ serves as a measure of how well the two classes of training samples are separated (by a linear boundary). We will call this the *geometric margin* or

$$\gamma_{\text{geom}} = \frac{\gamma}{\|\boldsymbol{\theta}^*\|}. \tag{29}$$

See Fig. 2 Thus, $\gamma_{\text{geom}}^{-1}$ is then a fair measure of how difficult the problem is.

Proposition 2. *The distance between the training sample* $\mathbf{x}_t$ *that is closest to the decision boundary* L *and* L *is* γ_{geom}*, i.e.,*

$$\min_{t=1,\ldots,n} \min_{\mathbf{x} \in L} \|\mathbf{x}_t - \mathbf{x}\| = \gamma_{\text{geom}}. \tag{30}$$

Proof. We measure the distance from each sample $\mathbf{x}_t$ such that

$$y_t \langle \mathbf{x}_t, \boldsymbol{\theta}^* \rangle = \gamma \tag{31}$$

to the line L. Thus, any such $\mathbf{x}_t$ is a training samples that is closest to the decision boundary as $\gamma = \min_{t=1,\ldots,n} y_t \langle \mathbf{x}_t, \boldsymbol{\theta}^* \rangle$. We define a line segment from $\mathbf{x}(0) = \mathbf{x}_t$, parallel to $\boldsymbol{\theta}^*$, towards the boundary. This is given by

$$\mathbf{x}(s) = \mathbf{x}(0) - s \frac{y_t \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}^*\|}. \tag{32}$$

(Convince yourself that this formula is correct by looking at Fig. 2.) We want to find the $s > 0$ such that $\mathbf{x}(s)$ satisfies $y_t \langle \mathbf{x}(s), \boldsymbol{\theta}^* \rangle = 0$. This is a matter of algebra. Indeed, we have

$$y_t \langle \mathbf{x}(s), \boldsymbol{\theta}^* \rangle = y_t \left\langle \mathbf{x}_t - s \frac{y_t \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}^*\|}, \boldsymbol{\theta}^* \right\rangle \tag{33}$$

$$= y_t \langle \mathbf{x}_t, \boldsymbol{\theta}^* \rangle - s y_t^2 \left\langle \boldsymbol{\theta}^*, \frac{\boldsymbol{\theta}^*}{\|\boldsymbol{\theta}^*\|} \right\rangle \tag{34}$$

$$\overset{(a)}{=} \gamma - s\|\boldsymbol{\theta}^*\| = 0, \tag{35}$$

where (a) is because $y_t^2 = 1$. This means that the distance $s = s^*$ that makes $\mathbf{x}(s)$ lie on L is exactly $s^* = \gamma/\|\boldsymbol{\theta}^*\|$ as desired. □

Thus the bound on the number of errors in the perceptron algorithm can be rewritten as

$$k \leq \left(\frac{R}{\gamma_{\text{geom}}}\right)^2. \tag{36}$$

with the understanding that γ_{geom} is the largest geometric margin that could be achieved by a linear classifier for this problem. Note that the result does not depend (directly) on the dimension $d$of the examples,[1] nor the number of training examples n.[2] It is nevertheless tempting to interpret R/γ_{geom} as a measure of difficulty (or complexity) of the problem of learning linear classifiers in this setting. You will see that this is exactly the case, cast in terms of a measure known as VC dimension.

In the following lectures, we will explicitly optimize (maximize) the geometric margin to obtain the so-called *maximum margin classifier*. This leads to a well-known classifier known as the *Support Vector Machines*.

4 Margin Perceptron Algorithm (Optional)

Often times, we not only want the trainings samples to be classified correctly. We want find a large-margin separator for them. One approach is to directly solve for the maximum-margin separator using convex programming (which is what is done in the SVM algorithm). However, if we only need to approximately maximize the margin, then another approach is to use Perceptron. In particular, suppose we cycle through the data using the Perceptron algorithm, updating not only on mistakes, but also on examples x that our current hypothesis gets correct by margin less than $\gamma/2$. Assuming our data is separable by margin γ, then we can show that this is guaranteed to halt in a number of rounds that is polynomial in $1/\gamma$. (In fact, we can replace $\gamma/2$ with $(1-\varepsilon)\gamma$ and have bounds that are polynomial in $1/(\varepsilon\gamma)$.)

The Margin Perceptron Algorithm proceeds as follows

1. Assume again that all examples are normalized to have Euclidean length 1. Initialize $\boldsymbol{\theta}^{(1)} = y_1\mathbf{x}_1$, where $\mathbf{x}_1$ is the first example seen and initialize the iteration counter k to 1.

2. Predict positive if
$$\frac{\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}\rangle}{\|\boldsymbol{\theta}^{(k)}\|} \geq \frac{\gamma}{2} \tag{37}$$
and predict negative if
$$\frac{\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}\rangle}{\|\boldsymbol{\theta}^{(k)}\|} \leq -\frac{\gamma}{2} \tag{38}$$
and consider an example to be a margin mistake when
$$\frac{\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}\rangle}{\|\boldsymbol{\theta}^{(k)}\|} \in \Big(-\frac{\gamma}{2}, \frac{\gamma}{2}\Big). \tag{39}$$

3. On a mistake (incorrect prediction or margin mistake), update as in the standard Perceptron algorithm
$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + y_t\mathbf{x}_t. \tag{40}$$

One natural problem with this algorithm is that we need to know γ, which is not known in practice. However suppose we do, it can be shown that with a slightly larger number of iterations, we obtain a linear classifier with larger margin than the standard perceptron updates. In fact, we have the following:

[1] There is a subtlety here as γ_{geom} depends on the input vectors $\mathbf{x}_t \in \mathbb{R}^d$, so it is unclear how to compare two different d values.

[2] Adding more samples to a data set, even in a way that is sure to preserve linear separability, could decrease γ_{geom}.

Theorem 3. *Assume that $\|\mathbf{x}_t\| = 1$ for all $1 \le t \le n$ (so $R = 1$) and $\boldsymbol{\theta}^*$, the normal vector of a decision boundary that separates the positively and negatively labelled samples, satisfies $\|\boldsymbol{\theta}^*\| = 1$. Let $\gamma := \min_{1\le t\le n} y_t\langle \boldsymbol{\theta}^*, \mathbf{x}_t\rangle$ and note that $\gamma \in (0, 1]$ (why?). Then, the maximum number of perceptron updates for the "Margin Perceptron Algorithm" described above is bounded above as*

$$k \le \frac{12}{\gamma^2}. \tag{41}$$

The reader is invited to improve on the constant 12. The bounds in the proof below are probably too loose/conservative. *The first person to refine the argument below to improve on the constant 12 gets a reward of $5 from me.*

Proof. One can check that as usual,

$$\begin{aligned}
\langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k)}\rangle &= \langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k-1)} + y_t\mathbf{x}_t\rangle && (42)\\
&= \langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k-1)}\rangle + y_t\langle \boldsymbol{\theta}^*, \mathbf{x}_t\rangle && (43)\\
&\ge \langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k-1)}\rangle + \gamma. && (44)
\end{aligned}$$

Hence,

$$\langle \boldsymbol{\theta}^*, \boldsymbol{\theta}^{(k)}\rangle \ge k\gamma. \tag{45}$$

However, the calculation of the norm of $\boldsymbol{\theta}^{(k)}$ needs a bit more care. This is a little more complicated since we are updating on some examples where the angle is less than 90 degrees. Consider

$$\begin{aligned}
\|\boldsymbol{\theta}^{(k)}\|^2 &= \|\boldsymbol{\theta}^{(k-1)} + y_t\mathbf{x}_t\|^2 && (46)\\
&= \|\boldsymbol{\theta}^{(k-1)}\|^2 + 2y_t\langle \boldsymbol{\theta}^{(k-1)}, \mathbf{x}_t\rangle + y_t^2\|\mathbf{x}_t\|^2 && (47)\\
&= \|\boldsymbol{\theta}^{(k-1)}\|^2 + 2y_t\langle \boldsymbol{\theta}^{(k-1)}, \mathbf{x}_t\rangle + 1 && (48)\\
&\le \|\boldsymbol{\theta}^{(k-1)}\|^2 + \gamma\|\boldsymbol{\theta}^{(k-1)}\| + 1 && (49)
\end{aligned}$$

where the last inequality holds because we update iff $y_t\langle \boldsymbol{\theta}^{(k-1)}, \mathbf{x}_t\rangle \le \frac{\gamma}{2}\|\boldsymbol{\theta}^{(k-1)}\|$. Now, it is easy to verify that for $a, \gamma \ge 0$, the following inequality is true: $a^2 + a\gamma + 1 \le (a + \frac{1}{2a} + \frac{\gamma}{2})^2$. Applying this to (49) yields

$$\|\boldsymbol{\theta}^{(k)}\| \le \|\boldsymbol{\theta}^{(k-1)}\| + \frac{1}{2\|\boldsymbol{\theta}^{(k-1)}\|} + \frac{\gamma}{2}. \tag{50}$$

Now, if $\|\boldsymbol{\theta}^{(k)}\| \ge \frac{2}{\gamma}$, then $\|\boldsymbol{\theta}^{(k+1)}\| \le \|\boldsymbol{\theta}^{(k)}\| + \frac{3\gamma}{4}$. Thus, after k updates, it is easy to see that

$$\|\boldsymbol{\theta}^{(k+1)}\| \le 1 + \frac{2}{\gamma} + \frac{3k\gamma}{4}. \tag{51}$$

Why? We prove (51) by induction. Let $\mathcal{P}_k$ be the proposition that $\|\boldsymbol{\theta}^{(k)}\| \le 1 + \frac{2}{\gamma} + \frac{3(k-1)\gamma}{4}$. The base case $k = 1$ is easy. Now consider the general case. If $\|\boldsymbol{\theta}^{(k)}\| \ge \frac{2}{\gamma}$, we know that $\|\boldsymbol{\theta}^{(k+1)}\| \le \|\boldsymbol{\theta}^{(k)}\| + \frac{3\gamma}{4}$. Combining this with $\mathcal{P}_k$ yields (51). On the other hand if $\|\boldsymbol{\theta}^{(k)}\| < \frac{2}{\gamma}$, since $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + y_t\mathbf{x}_t$, by the triangle inequality, $\|\boldsymbol{\theta}^{(k+1)}\| \le \|\boldsymbol{\theta}^{(k)}\| + 1 < \frac{2}{\gamma} + 1 \le 1 + \frac{2}{\gamma} + \frac{3k\gamma}{4}$, (51) is also true.

Since $\|\boldsymbol{\theta}^*\| = 1$, (45) yields

$$\gamma k \le \|\boldsymbol{\theta}^{(k)}\| \le 1 + \frac{2}{\gamma} + \frac{3k\gamma}{4} \le \frac{3}{\gamma} + \frac{3k\gamma}{4} \tag{52}$$

where the second inequality follows from (51) where upper bounded $k-1$ by k and 1 by $\frac{1}{\gamma}$ (recall $\gamma \in (0, 1]$). Solving this inequality for k yields the bound in (41). □

5 Variable-Increment Perceptron with Margin (Optional)

For the *Variable-Increment Perceptron with Margin* update rule, we perform the update

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \eta(k) y_t \mathbf{x}_t \tag{53}$$

if

$$y_t \langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_t \rangle \leq \frac{\gamma}{2}. \tag{54}$$

The *learning rate* $\eta(k)$ is chosen to decrease with k to reflect our belief that as the iterations progress we get closer and closer to the ground truth $\boldsymbol{\theta}^*$ so we update less. It is known that if the samples are γ-linearly separable and

$$\eta(k) \geq 0 \tag{55}$$

$$\lim_{m \to \infty} \sum_{k=1}^{m} \eta(k) = \infty \tag{56}$$

$$\lim_{m \to \infty} \frac{\sum_{k=1}^{m} \eta(k)^2}{(\sum_{k=1}^{m} \eta(k))^2} = 0, \tag{57}$$

then $\boldsymbol{\theta}^{(k)}$ converges to a solution satisfying $y_t \langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_t \rangle \geq \gamma/2$ for all t. In particular these conditions are satisfied if we take $\eta(k) = 1/k$. Please attempt to prove this.

6 Batch Variable Increment Perceptron (Optional)

Another variant of our original perceptron algorithm is the *Batch Variable Increment Perceptron* update rule

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \eta(k) \sum_{t \in \mathcal{Y}_k} y_t \mathbf{x}_t \tag{58}$$

where

$$\mathcal{Y}_k := \{t : y_t \langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_t \rangle \leq 0\} \tag{59}$$

denotes the set of indices of samples that are currently mis-classified according to $\boldsymbol{\theta}^{(k)}$. We terminate when $\mathcal{Y}_k = \emptyset$.

The benefit of the batch algorithm is that the trajectory of the weight vector is smooth compared to that in the corresponding single-sample algorithms, because at each update the full set of mis-classified samples is used—the local statistical variations in the mis-classified patterns tend to cancel while the large-scale trend does not. Thus, if the samples are linearly separable, all the possible correction vectors form a linearly separable set and if $\eta(k)$ satisfies the conditions in (55)–(57), the sequence $\boldsymbol{\theta}^{(k)}$ will converge to a solution vector.

There may be other variations of the perceptron algorithm that are potential exam questions.

A Code for perceptrons

```
close all;
rng(1);
n = 100;
mu = [1 1];
Sigma = .5*[1 0; 0 1]; R = chol(Sigma);
x_plus = repmat(mu,n/2,1) + randn(n/2,2)*R;
x_minus = repmat(-mu,n/2,1) + randn(n/2,2)*R;
```

```
x = [x_plus; x_minus];
y = [ones(n/2,1) -ones(n/2,1)];

scatter(x_plus(:,1),x_plus(:,2), 'bo'); hold on;
scatter(x_minus(:,1), x_minus(:,2),'rx');
xlim([-2,2]);
ylim([-2,2]);
xlabel('x_1');
ylabel('x_2');
grid;

x_plot = linspace(-2,2,100);

n_iter = 100;
theta = zeros(2,1);

for i = 1:n_iter
    for t = 1:n
       if(y(t)*dot(x(t,:)', theta)<=0)
          theta = theta + y(t) * x(t,:)';
          plot(x_plot, -theta(1)/theta(2)*x_plot, 'k');
          pause(.5);
       end
    end

end

plot(x_plot, -theta(1)/theta(2)*x_plot, 'm', 'Linewidth', 3);
```

References

[1] A. B. Novikoff. On convergence proofs on perceptron. In *Symposium on the Mathematical Theory of Automatas*, pages 615–622. Polytechnic Institute of Brooklyn, 1962.

MA4270 : Support Vector Machines in the Primal Form and Examples

Vincent Y. F. Tan

August 19, 2021

Here are some aspects of the primal formulation of the Support Vector Machine (SVM). We also provide several examples to illustrate the optimization problems.

These notes are heavily adapted from Jaakkola's MIT notes.

1 Support vector machines in the primal form

Assume that we have a dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ that is γ-linearly separable for some $\gamma > 0$. This means that there exists some $\boldsymbol{\theta}^*$ such that

$$y_t \langle \boldsymbol{\theta}^*, \mathbf{x}_t \rangle \geq \gamma \qquad \forall\, t = 1, 2, \ldots, n. \tag{1}$$

We have seen from the previous lecture that the geometric margin $\gamma_{\text{geom}} = \gamma / \|\boldsymbol{\theta}^*\|$ measure the difficulty of the binary classification problem. Can we maximize this geometric margin directly? This is known as the *maximum margin linear classifier*. In principle, we can do this by finding a classifier that correctly classifies all the samples, then increasing the geometric margin until the classifier "locks in place" at the point where we cannot increase the margin any further (Fig. 1). The solution is unique as we prove in Section 2.

Thus, consider the following optimization problem in which we maximize γ_{geom} subject to the constraints in (1). This can be written formally as

$$\max_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{\gamma}{\|\boldsymbol{\theta}\|} \quad \text{s.t.} \quad y_t \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle \geq \gamma \qquad \forall\, t = 1, \ldots, n. \tag{2}$$

Usually, it is more conventional to consider minimization in optimization problems. Maximizing $f(x)$ is the same as minimizing $1/f(x)$ (as long as $f(x)$ is non-zero). Thus, the basic formulation in (2) can be rewritten

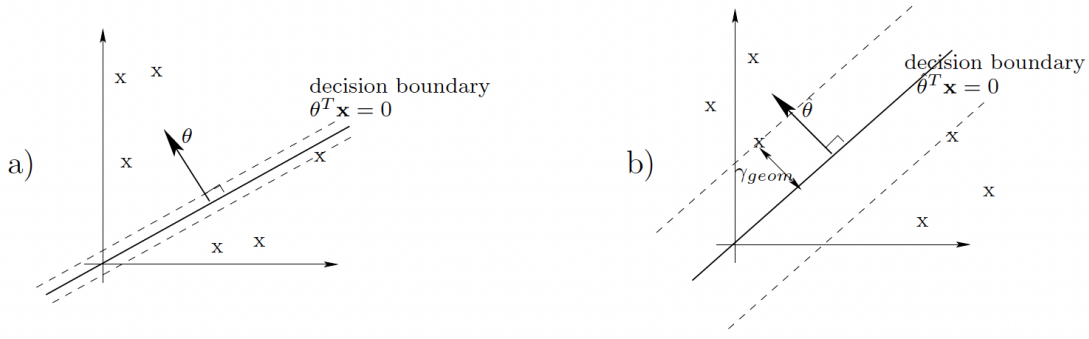


Figure 1: a) A linear classifier with a small geometric margin, b) maximum margin linear classifier.

as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{\|\boldsymbol{\theta}\|}{\gamma} \quad \text{s.t.} \quad y_t\langle \boldsymbol{\theta}, \mathbf{x}_t\rangle \geq \gamma \qquad \forall\, t = 1, \ldots, n. \tag{3}$$

Since $\gamma > 0$, this can again be rewritten as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \left\| \frac{\boldsymbol{\theta}}{\gamma} \right\| \quad \text{s.t.} \quad y_t \left\langle \frac{\boldsymbol{\theta}}{\gamma}, \mathbf{x}_t \right\rangle \geq 1 \qquad \forall\, t = 1, \ldots, n. \tag{4}$$

So you can see that our classification problem (the data, setup) only tells us something about the ratio $\boldsymbol{\theta}/\gamma$, not $\boldsymbol{\theta}$ or γ separately. For example, the geometric margin is defined only on the basis of this ratio. Scaling $\boldsymbol{\theta}$ by a constant also doesn't change the decision boundary. We can therefore fix $\gamma = 1$ and solve for $\boldsymbol{\theta}$ from

$$\text{(Primal-SVM)} \qquad \min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{2}\|\boldsymbol{\theta}\|^2 \quad \text{s.t.} \quad y_t \langle \boldsymbol{\theta}, \mathbf{x}_t\rangle \geq 1 \qquad \forall\, t = 1, \ldots, n. \tag{5}$$

Notice here that the minimization of $\|\boldsymbol{\theta}\|$ is the same as the minimization of $\frac{1}{2}\|\boldsymbol{\theta}\|^2$ (the square makes the objective differentiable everywhere on $\mathbb{R}^d$ which makes life easy for us subsequently). We now have (5), which is known as the *primal form* of the SVM without offset.

1.1 Incorporating an offset

Till now, we have only considered *linear* classifiers, i.e., classifiers that take the form $f_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sgn}(\langle \boldsymbol{\theta}, \mathbf{x}\rangle)$. This is very restrictive data may not be "centered". For example, we might want to classify male and female students based on their height H and weight W. Typically, H and W are in intervals $[140, 200]$ (centimeters) and $[45, 90]$ (kg). Hence, the data is not centered at zero. It is thus worthwhile to consider *affine* classifiers that incorporate an offset $\theta_0 \in \mathbb{R}$. Such classifiers take the form

$$f_{\boldsymbol{\theta},\theta_0}(\mathbf{x}) = \text{sgn}(\langle \boldsymbol{\theta}, \mathbf{x}\rangle + \theta_0). \tag{6}$$

Here, the decision boundary is the line

$$L = \{\mathbf{x} \in \mathbb{R}^d : \langle \boldsymbol{\theta}, \mathbf{x}\rangle + \theta_0 = 0\}. \tag{7}$$

Clearly, the line L passes through the origin if and only if $\theta_0 = 0$.

What's the real advantage of allowing $\theta_0 \neq 0$? Well, it can lead to a classifier with a larger geometric margin. This is illustrated in Fig. 2. Note that the vector $\hat{\boldsymbol{\theta}}$ corresponding to the maximum margin solution is different in the two figures.

How do we find the maximum margin classifier if we allow for an offset? The optimization problem in (5) is modified to

$$\text{(Primal-SVM with Offset)} \qquad \min_{(\boldsymbol{\theta},\theta_0)\in\mathbb{R}^d\times\mathbb{R}} \frac{1}{2}\|\boldsymbol{\theta}\|^2 \quad \text{s.t.} \quad y_t \left(\langle \boldsymbol{\theta}, \mathbf{x}_t\rangle + \theta_0\right) \geq 1 \qquad \forall\, t = 1, \ldots, n. \tag{8}$$

Note that the offset parameter only appears in the constraints. This is different from simply modifying the linear classifier through origin by feeding it with examples that have an additional constant component, i.e., $\mathbf{x}' = [\mathbf{x}; 1]$. In the above formulation, we do not bias in any way where the separating hyperplane should appear, only that it should maximize the geometric margin.

Exercise 1. *Assume the dataset is γ-affinely separable, i.e., there exists $(\boldsymbol{\theta}, \theta_0)$ such that $y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0) \geq \gamma$ for all t. Define the line L as in (7). Prove that with the offset, the minimum distance over all training samples to L is still the geometric margin $\gamma/\|\boldsymbol{\theta}\|$. The moral of the story here is that θ_0 doesn't influence the geometric margin, i.e., the inherent difficulty of the problem instance.*

See Section 3 for the effect of the offset on a toy example.

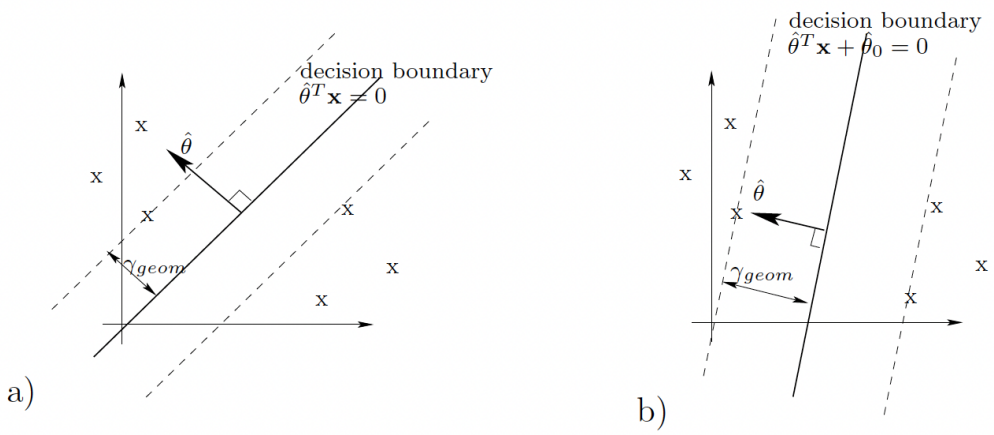


Figure 2: a) Maximum margin linear classifier through origin; b) Maximum margin linear classifier with an offset parameter

1.2 Robustness of the max margin classifier

There are several advantages of the support vector machine, also called the *maximum margin classifier*. These include the fact that the solution is unique (see Section 2) and its robustness property, which we describe below.

Drawing the boundary as far away from the training examples as possible makes it robust to noisy examples. The maximum margin linear boundary also has the curious property that the solution depends only on a subset of the examples, those that appear exactly on the margin (the dashed lines parallel to the boundary in the figures). The examples that lie exactly on the margin are called *support vectors* (see Fig. 3). The rest of the training samples could lie anywhere outside the margin without affecting the solution. We would therefore get the same classifier if we had only received the support vectors as training examples.

To assess the robustness of the SVM, let us define the leave-one-out cross-validation error as follows.

$$\text{LOOCV} = \frac{1}{n} \sum_{t=1}^{n} \text{Loss}\left(y_t, f(\mathbf{x}_t, (\boldsymbol{\theta}^{-t}, \theta_0^{-t}))\right), \tag{9}$$

where $(\boldsymbol{\theta}^{-t}, \theta_0^{-t})$ is the parameter vector and offset learned from the dataset with the t^{th} sample left out, i.e., $\mathcal{D}^{-t} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{t-1}, y_{t-1}), (\mathbf{x}_{t+1}, y_{t+1}), \ldots, (\mathbf{x}_n, y_n)\}$ and $\text{Loss}(y, y') = \mathbb{1}\{y \neq y'\}$ is the zero-one loss. That is

$$(\boldsymbol{\theta}^{-t}, \theta_0^{-t}) \in \underset{(\boldsymbol{\theta}, \theta_0) \in \mathbb{R}^d \times \mathbb{R}}{\arg\min} \left\{ \frac{1}{2} \|\boldsymbol{\theta}\|^2 : y_s \left(\langle \boldsymbol{\theta}, \mathbf{x}_s \rangle + \theta_0 \right) \geq 1 \quad \forall\, s \in [n] \setminus \{t\} \right\}. \tag{10}$$

We are effectively trying to gauge how well the classifier would generalize to each training example if it had not been part of the training set. A classifier that has a low leave-one-out cross-validation error is likely to generalize well though it is not guaranteed to do so.

We will see that LOOCV is intimately related to the number of support vectors. To be formal, say that $\mathbf{x}_t$ is a *support vector* if its distance to the decision boundary is exactly $\gamma / \|\hat{\boldsymbol{\theta}}\|$ where $\hat{\boldsymbol{\theta}}$ is the learned parameter vector based on the entire dataset $\mathcal{D}$.

Proposition 1. *Let the number of support vectors be $N \in \{0, 1, \ldots, n\}$. Then,*

$$\text{LOOCV} \leq \frac{N}{n}. \tag{11}$$

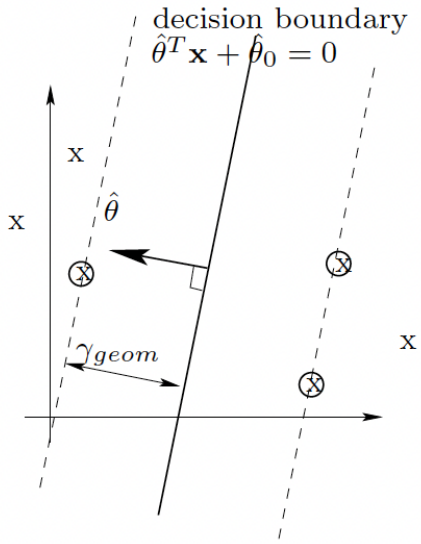


Figure 3: Support vectors (circled) associated the maximum margin linear classifier

This proposition says that if the number of support vectors is small, then the LOOCV is also small, which means that the SVM is likely to generalize well, i.e., work well on test samples (not in $\mathcal{D}$).

Proof. If $\mathbf{x}_t$ is not a support vector, we know that it cannot influence the solution of the primal SVM. Thus $\text{Loss}\left(y_t, f(\mathbf{x}_t, (\boldsymbol{\theta}^{-t}, \theta_0^{-t}))\right) = 0$ for all t such that $\mathbf{x}_t$ is not a support vector. However, $\text{Loss} \leq 1$. Thus, the only potential contributions to the sum in (9) are from the α support vectors. □

1.3 Relaxed primal SVM

What if the data is not linearly- or affinely-separable? The simplest way to permit errors in the maximum margin linear classifier is to introduce "slack" variables for the classification/margin constraints in the optimization problem. In other words, we measure the degree to which each margin constraint is violated and associate a cost for the violation. The costs of violating constraints are minimized together with the norm of the parameter vector

(Primal-SVM with slack)

$$\min_{(\boldsymbol{\theta},\theta_0,\boldsymbol{\xi})\in\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}^n_+} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^{n}\xi_t \quad \text{s.t.} \quad y_t\left(\langle\boldsymbol{\theta},\mathbf{x}_t\rangle + \theta_0\right) \geq 1 - \xi_t, \quad \xi_t \geq 0 \qquad \forall\, t = 1,\ldots,n. \tag{12}$$

Here, $C > 0$ is some pre-specified constant and $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_n) \in \mathbb{R}^n_+$ denotes the vector of *slack variables*. The margin constraint is violated when we have to set $\xi_t > 0$ for some example $\mathbf{x}_t$. The penalty for this violation is $C\xi_t > 0$ and it is traded-off with the possible gain in minimizing the squared norm of the parameter vector or $\|\boldsymbol{\theta}\|$. If we increase the penalty C for margin violations then at some point all $\xi_t = 0$ and we get back the maximum margin linear separator (if possible).

Let's try to understand the setup a little further. For example, what is the resulting margin when some of the constraints are violated? We can still take $1/\|\hat{\boldsymbol{\theta}}\|$ as the geometric margin. This is indeed the geometric margin based on examples for which $\xi_t^* = 0$ where "*" indicates the optimized value. So, is it the case that we get the maximum margin linear classifier for the subset of examples for which $\xi_t^* = 0$? No, we don't. The examples that violate the margin constraints, including those training examples that are

actually *misclassified*, do affect the solution. In other words, the parameter vector $\hat{\boldsymbol{\theta}}$ is defined on the basis of examples

1. right on the margin ($\xi_t^* = 0$ but $y_t(\langle \hat{\boldsymbol{\theta}}, \mathbf{x}_t \rangle + \hat{\theta}_0) = 1$);
2. those that violate the constraint but not enough to be misclassified ($0 < \xi_t^* < 1$);
3. and those that are misclassified ($\xi_t^* \geq 1$).

All of these are support vectors in this sense. We will study this in great detail in the following using convex duality.

See Section 4 for the effect of the slack variables on a toy example.

2 Uniqueness of the minimum length solution vector

As we have seen, one of the problems of the perceptron algorithm is that it outputs a vector defining the decision boundary $\boldsymbol{\theta}$ that is potentially non-unique. It may depend on the order that we consider and update the samples. We now show that this problem is no longer present if we consider the primal SVM without offset. Try doing the same thing for the the SVM with offset and with slack.

The primal SVM formulation without slack and without offset is

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{2} \|\boldsymbol{\theta}\|^2 \quad \text{s.t.} \quad y_t \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle \geq 1 \quad \forall\, t = 1, \ldots, n \tag{13}$$

Now, we claim that the solution in (13) is unique, i.e., there is only one vector

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\boldsymbol{\theta}\|^2 : y_t \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle \geq 1,\ \forall\, t = 1, \ldots, n \right\} \tag{14}$$

Suppose, to the contrary, there were two distinct minimum length solution vectors $\boldsymbol{\theta}_1 \neq \boldsymbol{\theta}_2$ with

$$y_t \langle \boldsymbol{\theta}_j, \mathbf{x}_t \rangle \geq 1 \quad \forall\, t = 1, \ldots, n \tag{15}$$

for each $j = 1, 2$. Then necessarily

$$\|\boldsymbol{\theta}_1\| = \|\boldsymbol{\theta}_2\| \tag{16}$$

(otherwise the longer of the two vectors would not be a minimum length solution). Consider the averaged vector

$$\bar{\boldsymbol{\theta}} = \frac{1}{2}(\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2). \tag{17}$$

By linearity of the inner product,

$$y_t \langle \bar{\boldsymbol{\theta}}, \mathbf{x}_t \rangle \geq 1 \quad \forall\, t = 1, \ldots, n \tag{18}$$

so the constraints are satisfied. Furthermore by the triangle inequality,

$$\|\bar{\boldsymbol{\theta}}\| = \left\| \frac{1}{2}(\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2) \right\| \leq \frac{1}{2}(\|\boldsymbol{\theta}_1\| + \|\boldsymbol{\theta}_2\|) = \|\boldsymbol{\theta}_1\| = \|\boldsymbol{\theta}_2\| \tag{19}$$

so the norm of the vector $\bar{\boldsymbol{\theta}}$ is no larger than that of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. Now, there are two possibilities:

1. $\|\bar{\boldsymbol{\theta}}\| < \|\boldsymbol{\theta}_1\| = \|\boldsymbol{\theta}_2\|$. In this case, there is already a contradiction since we have found a solution $\bar{\boldsymbol{\theta}}$ whose length is strictly smaller than $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$ and satisfies the constraints.
2. $\|\bar{\boldsymbol{\theta}}\| = \|\boldsymbol{\theta}_1\| = \|\boldsymbol{\theta}_2\|$. In this case, the triangle inequality in (19) holds with equality. This can only happen if $\boldsymbol{\theta}_1$ is proportional to $\boldsymbol{\theta}_2$. Since they are of the same lengths, $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$, another contradiction.

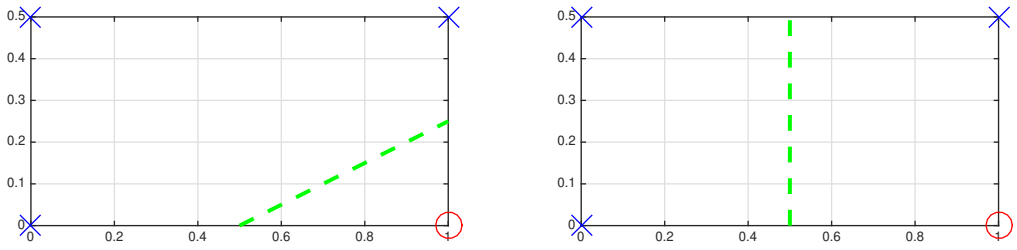


Figure 4: Different $(\boldsymbol{\theta}, \theta_0)$ for the same dataset. Blue crosses and red circles respectively denote points from the positive and negative class in the dataset defined in (24) with $\epsilon = 0.5$. On the left and right, we have $(\boldsymbol{\theta}^{(1)}, \theta_0^{(1)})$ and $(\boldsymbol{\theta}^{(2)}, \theta_0^{(2)})$ as defined in (25) and (26) respectively. Note that on the left all points are classified correctly but with small margin. On the right the top-right point is classified wrongly but the margin is larger than on the left.

3 Example of computation of an SVM with offset

Suppose our one-dimensional dataset consists of two points

$$\mathcal{D} = \{(x_-, -1), (x_+, +1)\} \tag{20}$$

where $0 < x_- < x_+ < \infty$ are scalars on the positive real line. So the SVM with offset is formulated as

$$\min_{(\theta,\theta_0)\in\mathbb{R}\times\mathbb{R}} \frac{1}{2}\theta^2 \quad \text{s.t.} \quad -1(\theta x_- + \theta_0) \geq 1, \quad +1(\theta x_+ + \theta_0) \geq 1. \tag{21}$$

Adding the two constraints yields

$$\theta(x_+ - x_-) \geq 2, \qquad \Rightarrow \qquad \theta \geq \frac{2}{x_+ - x_-}. \tag{22}$$

Since we want to minimize the length of θ, the optimal value is the lower bound, i.e.,

$$\hat{\theta} = \frac{2}{x_+ - x_-}. \tag{23}$$

Observe that $1/\hat{\theta}$ is half the distance between x_+ and x_- as intuition suggests of the margin.

Exercise 2. *Find the optimal θ if there is no offset, i.e., $\theta_0 = 0$. Discuss your result.*

4 Example of computation of an SVM with offset and slack

Suppose our two-dimensional linearly-separable dataset (even without offset) consists of four points

$$\mathcal{D} = \left\{(\mathbf{x}_1, y_1) = ((0,0)^\top, +1), (\mathbf{x}_2, y_2) = ((0,\epsilon)^\top, +1), (\mathbf{x}_3, y_3) = ((1,\epsilon)^\top, +1), (\mathbf{x}_4, y_4) = ((1,0)^\top, -1)\right\} \tag{24}$$

where $\epsilon > 0$ is some parameter. See Fig. 4. Consider two solutions. First the solution[1]

$$\boldsymbol{\theta}^{(1)} = (-\epsilon, 1)^\top, \quad \theta_0^{(1)} = \epsilon/2. \tag{25}$$

[1]The superscipt (1) in $\theta^{(1)}$ does not mean iteration number like the perceptron lectures; rather it is the index of the solution.

This solution classifies all training points correctly. However, we see that if ϵ is small, so is the geometric margin γ_1, i.e., the distance from the closest training point in $\mathcal{D}$ to the line $x_2 = +\epsilon x_1 - \epsilon/2$. Consider another solution

$$\boldsymbol{\theta}^{(2)} = (-1, 0)^\top, \quad \theta_0^{(2)} = 1/2 \tag{26}$$

then the point at the top right corner $(\mathbf{x}_3, y_3) = ((1, \epsilon)^\top, +1)$ is classified wrongly because the decision boundary is given by the vertical line $x_1 = 1/2$. In fact it is on the "other" margin boundary incurring a cost in the objective of $2C$. However, now the geometric margin γ_2 is large. Since the objective function in (primal) regularized SVM is $\frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^n \xi_t$, we prefer the first solution to the second iff

$$\frac{1}{2\gamma_1^2} \leq \frac{1}{2\gamma_2^2} + 2C. \tag{27}$$

Now note that if C is large, we revert to the first solution in which there are no slack parameters (effectively $\xi_t = 0$ for all t) and all training points are classified correctly. Conversely, when C is small, the second solution is preferred and training points are potentially mis-classified.

Exercise 3. *Discuss the merits of the third solution*

$$\boldsymbol{\theta}^{(3)} = (0, \delta)^\top, \quad \theta_0^{(2)} = 1 \tag{28}$$

where δ is very small.

```
clear all;
close all;
eps = .5;
% training samples
x=[0 0; 0 eps; 1 eps; 1 0];
% labels
y = [1 1 1 -1]';

% setting up the objective function
H = [1 0 0 0 0  0 0; 0 1 0 0 0 0 0; zeros(5,7)];
C = 20; % if C is large, all points are classified correctly
f = [0 0 0 C C C C]';

% setting up the constraints
% there is a better way to do this!
A = [-y(1)*x(1,:) -y(1); -y(2)*x(2,:) -y(2); -y(3)*x(3,:) -y(3); -y(4)*x(4,:) -y(4)];
A = [A -eye(4,4)]; % impose that y_t ( <theta, x_t> + theta_0) >= 1-xi_t
A = [A; zeros(4,3) -eye(4)]; % impose that xi_t >= 0
b = [-ones(4,1); zeros(4,1)];

h2 = figure('Position',[0 0 600 300]);
set(gcf,'PaperPositionMode','auto');
scatter(x(1:3,1),x(1:3,2),500,'bx'); hold on;
scatter(x(4,1),x(4,2),500,'ro');
grid;

[params, fval] = quadprog(H, f, A,b);
theta = params(1:3);
xi = params(4:end);

x_plot = linspace(0,1);
```

```
plot(x_plot, -1/theta(2).*(theta(3)+theta(1)*x_plot), 'k--', 'Linewidth', 3);
title('Primal SVM', 'Interpreter', 'Latex', 'Fontsize',14);
xlabel('$x_1$', 'Interpreter', 'Latex', 'Fontsize',14);
ylabel('$x_2$', 'Interpreter', 'Latex', 'Fontsize',14);
```

MA4270 : Regularization in the SVM, Logistic Regression, and Maximum Likelihood

Vincent Y. F. Tan

September 9, 2021

Here, we discuss regularization in the context of the SVM with slack parameters. This then leads to a probabilistic model of labels known as *logistic regression*. We discuss how to maximize the likelihood of the parameters in the logistic regression model, showing its similarity to the updates for the perceptron algorithm. We consider more advanced but optional topics such as statistical properties of the maximum likelihood estimator.

These notes are heavily adapted from Jaakkola's MIT notes and Jonathan Scarlett's NUS notes.

You need to know Section 1 and 2 and *possibly* Section 3, but the rest of the sections are optional.

1 SVM with Slack and Regularization

We have seen from the previous chapter that given a dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ that is not linearly- or affinely-separable, the primal SVM with slack optimization problem reads

(Primal-SVM with slack)

$$\min_{(\boldsymbol{\theta},\theta_0,\boldsymbol{\xi})\in\mathbb{R}^d\times\mathbb{R}\times\mathbb{R}_+^n} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^n \xi_t \quad \text{s.t.} \quad y_t\left(\langle\boldsymbol{\theta},\mathbf{x}_t\rangle+\theta_0\right) \geq 1-\xi_t, \quad \xi_t \geq 0 \qquad \forall\, t=1,\ldots,n. \tag{1}$$

Here, $C>0$ is some pre-specified constant and $\boldsymbol{\xi} = (\xi_1,\xi_2,\ldots,\xi_n) \in \mathbb{R}_+^n$ denotes the vector of *slack variables*.

Here, we try to understand Primal-SVM with slack from the perspective of *regularization*, a key concept in machine learning. Often machine learning problems boil down to optimization problems of the form

$$\min_{\boldsymbol{\theta},\theta_0} \sum_{t=1}^n \text{Loss}\big(y_t, f_{\boldsymbol{\theta},\theta_0}(\mathbf{x}_t)\big) + \lambda R(\boldsymbol{\theta}) \tag{2}$$

where $R(\boldsymbol{\theta})$ is a *regularization term* that "stabilizes" the solution or inject prior knowledge about the problem. For example, we might somehow guess before any data is observed that $\boldsymbol{\theta}$ has low ℓ_q norm for some $q\geq 1$, in which case an appropriate regularizer may be $R(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_q^q = \sum_{i=1}^d \theta_i^q$. We will see that (1) can be cast in the form of (5) even though they look quite different.

We define a loss function that corresponds to individually optimized ξ_t values and specifies the cost of violating each of the margin constraints. We are effectively solving the optimization problem with respect to the $\boldsymbol{\xi}$ values for a fixed $\boldsymbol{\theta}$ and θ_0. This will lead to an expression of $C\xi_t$ as a function of $\boldsymbol{\theta}$ and θ_0.

The loss function that we need at this point is the *hinge loss* defined as $\text{Loss}_\text{h}(z) = (1-z)^+ = \max\{0,1-z\}$; see Fig. 1(a). The Primal-SVM with slack can be written as

$$\min_{(\boldsymbol{\theta},\theta_0)\in\mathbb{R}^d\times\mathbb{R}} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^n \underbrace{\big(1-y_t(\langle\boldsymbol{\theta},\mathbf{x}_t\rangle+\theta_0)\big)^+}_{=:\hat{\xi}_t} \tag{3}$$

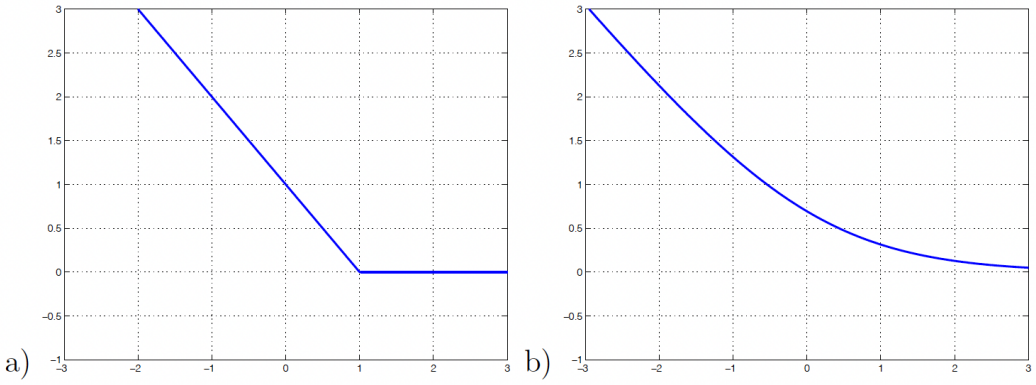


Figure 1: a) The hinge loss $(1-z)^+$ as a function of z. b) The logistic loss $\log(1+\exp(-z))$ as a function of z.

Notice that we have converted the constraints in (1) into the objective. Notice that when sample t does not violate the constraint, we have $y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0) \geq 1$. Hence, by the definition of the hinge loss $\hat{\xi}_t = (1 - y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0))^+ = 0$. One the other hand, if sample t violates the constraint, then $y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0) < 1$. In this case, we are in the linear part of the definition of the hinge loss, and we incur a cost of $C\hat{\xi}_t = C(1 - y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0)) > 0$ to the objective function.

Here $\|\boldsymbol{\theta}\|^2/2$, the inverse squared geometric margin, is viewed as a regularization penalty that helps stabilize the objective

$$C \sum_{t=1}^{n} \big(1 - y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0)\big)^+ = C \sum_{t=1}^{n} \mathrm{Loss_h}\big(y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0)\big). \tag{4}$$

Note that now, $y_t(\langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0)$ plays the role of the argument z in the hinge loss. The regularization parameter λ in (5) is $1/C$.

In other words, when no margin constraints are violated (zero loss), the regularization penalty helps us select the solution with the largest geometric margin.

2 Logistic Regression and Maximum Likelihood Estimation

2.1 Logistic Regression

Often, we want to model our *confidence* in the labels, which may be noisy. For example, "I predict Team A has a 55% chance of beating Team B" is much more useful (if accurate) than just "I predict Team A will beat Team B". (Even more so for applications in medicine, law, etc.) One way to do this is through a probability distribution distribution over the two labels. Samples that are further away from the decision boundary are "more correct". The way to do this is through the *logistic regression model*. We define the *logistic function* (see Fig. 2) as

$$g(z) = \frac{1}{1+\exp(-z)} \qquad z \in \mathbb{R}. \tag{5}$$

Then, given a sample $\mathbf{x}$ and the parameters $(\boldsymbol{\theta}, \theta_0)$, we set the probability of label $y = 1$ as

$$\Pr(y = 1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = g\big(\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0\big) = \frac{1}{1+\exp(-(\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0))} \tag{6}$$

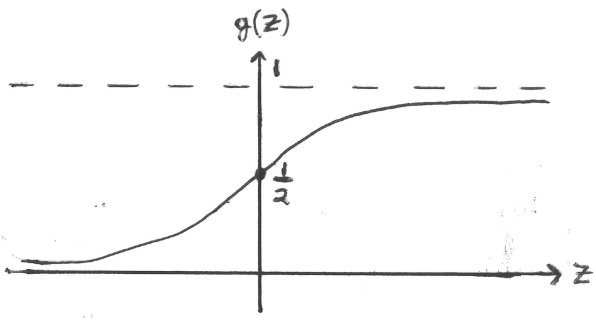


Figure 2: Sketch of the logistic function g

This is known as the *class conditional probability*. One way to derive the form of the logistic function is to say that the *log-odds* of the predicted class probabilities should be a linear function of the inputs

$$\log \frac{\Pr(y=1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0)}{\Pr(y=-1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0)} = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0. \tag{7}$$

Now since $\Pr(y=1 \mid E) = 1 - \Pr(y=-1 \mid E)$ (probabilities add to one), we have

$$\log \frac{\Pr(y=1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0)}{1 - \Pr(y=1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0)} = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0. \tag{8}$$

Solving this for $\Pr(y=1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0)$ yields the logistic regression model in (6) with logistic function in (5). Notice that if a sample $\mathbf{x}$ sits on the decision boundary, then $\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0 = 0$ and we are maximally uncertain about its label, i.e., $\Pr(y=1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = \Pr(y=-1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = 1/2$. The logistic regression is known as a *generalized linear model* because the class conditional probabilities are a function (namely g) of a linear function of the sample $\mathbf{x}$.

Let's write this in a form that is similar to the SVM with slack in which the loss is a function of the product of the label y_t and the linear term $\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0$. Notice that

$$\Pr(y \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = g\big(y(\langle \mathbf{x}, \boldsymbol{\theta} \rangle + \theta_0)\big). \tag{9}$$

Why is this so? If $y=1$, we recover (6). If instead $y=-1$, we have

$$\Pr(y=-1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = g\big(-(\langle \mathbf{x}, \boldsymbol{\theta} \rangle + \theta_0)\big) \overset{(a)}{=} 1 - g\big(\langle \mathbf{x}, \boldsymbol{\theta} \rangle + \theta_0\big) = 1 - \Pr(y=1 \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) \tag{10}$$

where (a) holds because $g(-z) = 1 - g(z)$ (check!).

2.2 Maximum Likelihood Estimation of the Logistic Regression Model and Relation to Perceptron Update

Now, given the logistic regression probabilistic model, we want to learn the parameters given the dataset $\mathcal{D}$. One sensible way to do this is to maximize the probability of seeing the dataset $\mathcal{D}$ over all the parameters $(\boldsymbol{\theta}, \theta_0)$. The probability of seeing one sample $(\mathbf{x}_t, y_t)$ given that the current parameters are $(\boldsymbol{\theta}, \theta_0)$ is

$$\Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0) = g\big(y_t(\langle \mathbf{x}_t, \boldsymbol{\theta} \rangle + \theta_0)\big). \tag{11}$$

We call this the *likelihood* of $(\mathbf{x}_t, y_t)$ given $(\boldsymbol{\theta}, \theta_0)$ and denote this by $L(\boldsymbol{\theta}, \theta_0 \mid \mathbf{x}_t, y_t)$. Usually, there is little reason that data points are correlated, so we assume that the samples are *independent* this leads to the likelihood of the dataset $\mathcal{D}$ being

$$L(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) = \prod_{t=1}^{n} L(\boldsymbol{\theta}, \theta_0 \mid \mathbf{x}_t, y_t) = \prod_{t=1}^{n} \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0). \tag{12}$$

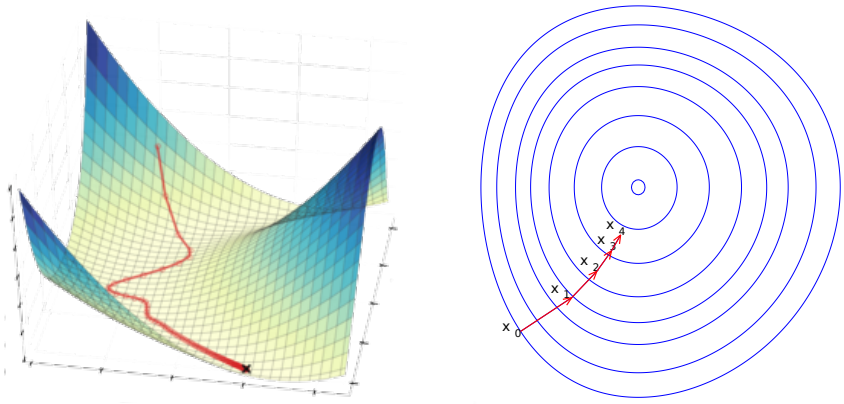


Figure 3: An illustration of minimizing a generic function (not necessarily the logistic log-likelihood) using gradient descent

This is called the *maximum likelihood* formulation as we are maximizing the overall likelihood of seeing the dataset given the parameters. By maximizing this conditional likelihood with respect to $\boldsymbol{\theta}$ and θ_0 we obtain *maximum likelihood estimates* (MLE) of the parameters. Maximum likelihood estimators have many nice properties. For example, assuming we have selected the right model class (logistic regression model) and certain regularity conditions hold, then the ML estimator is a) consistent (we will get the right parameter values in the limit of a large number of training examples), b) asymptotically normal, and c) efficient (no other estimator will converge to the correct parameter values faster in the mean squared sense). Some properties of MLEs are examined in the optional reading in Section 4.

Usually, one works with the logarithm of the MLE problem and so we formulate the following optimization problem

$$\max_{(\boldsymbol{\theta},\theta_0)\in\mathbb{R}^d\times\mathbb{R}} \sum_{t=1}^{n} \log \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0) = \sum_{t=1}^{n} \log g\big(y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0)\big). \tag{13}$$

As mentioned previously, we usually convert optimization problems to minimizations. Hence, we consider

$$\min_{(\boldsymbol{\theta},\theta_0)\in\mathbb{R}^d\times\mathbb{R}} \ell(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) \equiv \min_{(\boldsymbol{\theta},\theta_0)\in\mathbb{R}^d\times\mathbb{R}} \sum_{t=1}^{n} -\log \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0) \tag{14}$$

This is equivalent to minimizing

$$\sum_{t=1}^{n} -\log g\big(y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0)\big) = \sum_{t=1}^{n} \log\big[1 + \exp(-y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0))\big]. \tag{15}$$

This can be interpreted similarly to the unregularized form of the SVM in as the hinge loss is replaced by the logistic loss $\text{Loss}_{\text{logistic}}(z) = \text{Loss}_{\text{logistic}}(y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0))$ where $\text{Loss}_{\text{logistic}}(z) = \log[1 + \exp(-z)]$; See Fig. 4(b). This loss only depends on the value of the margin $y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0)$ for each example. The difference here is that we have a clear probabilistic interpretation of the "strength" of the prediction, i.e., how high $\Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)$ is for any particular example.

We can optimize (15) using calculus leading to a stochastic gradient descent algorithm; see Fig. 3. Taking the gradients with respect to $\boldsymbol{\theta}$ and θ_0 separately, we get

$$\frac{\mathrm{d}}{\mathrm{d}\theta_0} \log\big[1 + \exp(-y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0))\big] = -y_t \frac{\exp\big(-y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0)\big)}{1 + \exp\big(-y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0)\big)} \tag{16}$$

$$= -y_t\big[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)\big] \tag{17}$$

and

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \log \left[1 + \exp(-y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0))\right] = -y_t \mathbf{x}_t \left[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)\right]. \tag{18}$$

Check that you can derive these on your own. The parameters are updated by selecting training examples at random and moving the parameters in the opposite direction of the derivatives, i.e.,

$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \eta \cdot y_t \mathbf{x}_t \left[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)\right] \tag{19}$$

$$\theta_0 \longleftarrow \theta_0 + \eta \cdot y_t \left[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)\right], \tag{20}$$

where $\eta > 0$ is a small learning rate. Note that $\Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)$ is the probability that we predict the training label correctly and $[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}, \theta_0)]$ is the probability of *making a mistake*. The stochastic gradient descent updates in the logistic regression context therefore strongly resemble the perceptron mistake driven updates. The key difference here is that the updates are *graded* or *soft*, made in proportion to the probability of making a mistake.

The stochastic gradient descent algorithm leads to no significant change on average when the gradient of the full objective equals zero. Setting the gradient to zero is also a necessary condition for the optimality of $(\boldsymbol{\theta}^*, \theta_0^*)$

$$\frac{\mathrm{d}}{\mathrm{d}\theta_0} \ell(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) = \sum_{t=1}^{n} -y_t \left[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] = 0 \tag{21}$$

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \ell(\boldsymbol{\theta}, \theta_0 \mid \mathcal{D}) = \sum_{t=1}^{n} -y_t \mathbf{x}_t \left[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] = 0 \tag{22}$$

The sum in (21) is the difference between mistake probabilities associated with positively and negatively labeled examples. The optimality of θ_0^* therefore ensures that the mistakes are balanced in this (soft) sense. Another way of understanding this is that the vector of mistake probabilities is orthogonal to the vector of labels. Similarly, the optimal setting of $\boldsymbol{\theta}^*$ is characterized by mistake probabilities that are orthogonal to all linear functions of the inputs. This can be seen by mapping the ± 1 labels to $\{0, 1\}$ as $\tilde{y}_t = (1+y_t)/2$. Then the optimality conditions in (21) and (22) can be rewritten in terms of the *prediction errors* $\tilde{y}_t - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)$ (rather than mistake probabilities $[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)]$) as

$$\sum_{t=1}^{n} \left[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] = 0 \tag{23}$$

$$\sum_{t=1}^{n} \mathbf{x}_t \left[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] = 0. \tag{24}$$

It can be seen from these two optimality conditions that prediction errors are orthogonal to any linear function of the inputs, i.e., for any $(\tilde{\boldsymbol{\theta}}, \tilde{\theta}_0)$,

$$\tilde{\theta}_0 \sum_{t=1}^{n} \left[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] + \left\langle \tilde{\boldsymbol{\theta}}, \sum_{t=1}^{n} \mathbf{x}_t \left[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] \right\rangle = 0 \tag{25}$$

or

$$\sum_{t=1}^{n} \left(\tilde{\theta}_0 + \langle \tilde{\boldsymbol{\theta}}, \mathbf{x}_t \rangle\right) \left[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\right] = 0. \tag{26}$$

We elaborate on this in the optional Section 3.

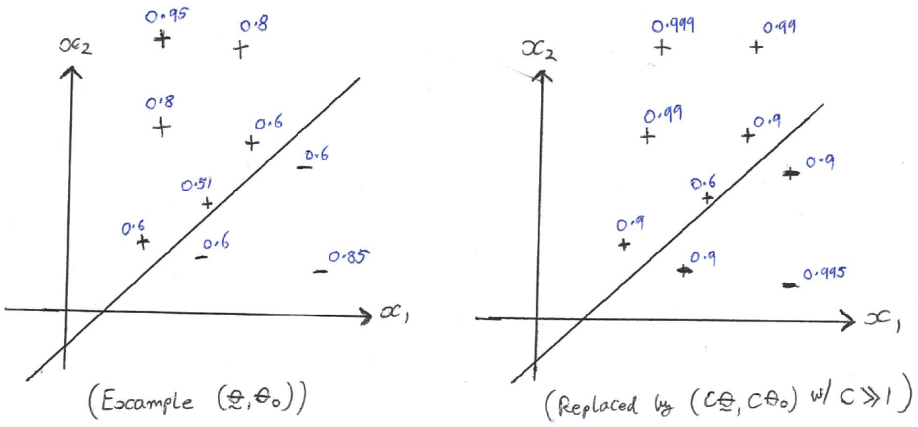


Figure 4: For linearly separable data, scaling up the parameters results in probabilities that go to one

Proof of (23). The negative of (21) can be expressed as

$$\sum_t y_t \big[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big]$$

$$= \sum_{t:y_t=+1} y_t \big[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] + \sum_{t:y_t=-1} y_t \big[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] \tag{27}$$

$$= \sum_{t:y_t=+1} \big[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] + \sum_{t:y_t=-1} (-1) \big[1 - \Pr(y_t \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] \tag{28}$$

$$= \sum_{t:\tilde{y}_t=+1} \big[1 - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] - \sum_{t:\tilde{y}_t=0} \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*) \tag{29}$$

$$= \sum_{t:\tilde{y}_t=+1} \big[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] + \sum_{t:\tilde{y}_t=0} \big[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big] \tag{30}$$

$$= \sum_t \big[\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*)\big]. \tag{31}$$

□

2.3 Final Remarks and Regularization

We end with three final remarks. If the training data are linearly- or affinely-separable, we can find parameters $(\boldsymbol{\theta}^*, \theta_0^*)$ such that $y_t(\langle \boldsymbol{\theta}^*, \mathbf{x}_t \rangle + \theta_0^*) \geq 0$ for all t. By scaling up the parameters, we make these values larger and larger. This is beneficial as far as the likelihood model is concerned since the log of the logistic function is strictly increasing as a function of $y_t(\boldsymbol{\theta}^*, \mathbf{x}_t\rangle + \theta_0^*)$. Thus, as a result, the maximum likelihood parameter values would become unbounded, and infinite scaling of any parameters corresponding to a perfect linear classifier would attain the highest likelihood (likelihood of exactly one or the loss exactly zero). The resulting probability values, predicting each training label correctly with probability one, are hardly accurate in the sense of reflecting our uncertainty about what the labels might be. See Fig. 4.

To mitigate this issue we have to resort to *regularization* again. Recall that the objective function in the minimization form for the logistic model is (14) and (15). Thus, we may consider modifying it to

$$\frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 + \sum_{t=1}^{n} \log\big[1 + \exp(-y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}\rangle + \theta_0))\big]. \tag{32}$$

where the first term ensures that $\boldsymbol{\theta}$ cannot be unbounded as we penalize its ℓ_2 norm.

Finally, we mention that logistic regression has a very natural multi-class counterpart: When we have $M \geq 2$ classes, replace (6) by the *soft-max* function

$$\Pr(y = c \mid \mathbf{x}, \{\boldsymbol{\theta}_c, \theta_{0,c}\}_{c=1}^M) = \frac{\exp(\langle \boldsymbol{\theta}_c, \mathbf{x} \rangle + \theta_{0,c})}{\sum_{c'=1}^M \exp(\langle \boldsymbol{\theta}_{c'}, \mathbf{x} \rangle + \theta_{0,c'})} \tag{33}$$

where we now have a different pair $(\boldsymbol{\theta}_c, \theta_{0,c})$ for each class. Without loss of generality, we can assume that one of the classes say the first one has $(\boldsymbol{\theta}_1, \theta_{0,1}) = (\mathbf{0}, 0)$ (why?), which is useful for showing that (6) is a special case of (33).

3 Orthogonality of Error and Data (Optional)

The vector of optimal prediction errors $\mathbf{e} := (\tilde{y}_t - \Pr(y = 1 \mid \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*))_{t=1}^n$ is orthogonal to any linear function of the inputs (data), say $\mathbf{d} := (\langle \tilde{\boldsymbol{\theta}}, \mathbf{x}_t \rangle + \tilde{\theta}_0)_{t=1}^n$ and thus there is no further linearly available information in the training samples $\{\mathbf{x}_t\}_{t=1}^n$ to improve the prediction probabilities (or mistake probabilities).

Let us try to understand this a little better. Suppose, to the contrary, there exists $\bar{\boldsymbol{\theta}}, \bar{\theta}_0$ such that $\mathbf{z} := (\langle \bar{\boldsymbol{\theta}}, \mathbf{x}_t \rangle + \bar{\theta}_0)_{t=1}^n$ *is not* orthogonal to the optimal prediction errors $\mathbf{e}$, i.e.,

$$\langle \mathbf{z}, \mathbf{e} \rangle \neq 0. \tag{34}$$

Then, I claim I can improve the ℓ_2 norm of the vector of prediction errors. Here's how. Construct the new error vector

$$\mathbf{e}' := \mathbf{e} - \frac{\langle \mathbf{z}, \mathbf{e} \rangle}{\langle \mathbf{z}, \mathbf{z} \rangle} \mathbf{z}. \tag{35}$$

Hence, we are subtraction the projection of $\mathbf{e}$ onto the subspace spanned by $\mathbf{z}$ (draw a picture to visualize this). Then, we claim that $\mathbf{e}'$ is orthogonal to $\mathbf{z}$. This follows by straightforward calculation:

$$\langle \mathbf{z}, \mathbf{e}' \rangle = \left\langle \mathbf{z}, \mathbf{e} - \frac{\langle \mathbf{z}, \mathbf{e} \rangle}{\langle \mathbf{z}, \mathbf{z} \rangle} \mathbf{z} \right\rangle = \langle \mathbf{z}, \mathbf{e} \rangle - \frac{\langle \mathbf{z}, \mathbf{e} \rangle}{\langle \mathbf{z}, \mathbf{z} \rangle} \langle \mathbf{z}, \mathbf{z} \rangle = 0. \tag{36}$$

Furthermore (and this is the important point),

$$\|\mathbf{e}'\|^2 = \left\| \mathbf{e} - \frac{\langle \mathbf{z}, \mathbf{e} \rangle}{\langle \mathbf{z}, \mathbf{z} \rangle} \mathbf{z} \right\|^2 \tag{37}$$

$$= \|\mathbf{e}\|^2 - 2\langle \mathbf{z}, \mathbf{e} \rangle \frac{\langle \mathbf{z}, \mathbf{e} \rangle}{\langle \mathbf{z}, \mathbf{z} \rangle} + \frac{\langle \mathbf{z}, \mathbf{e} \rangle^2}{\langle \mathbf{z}, \mathbf{z} \rangle^2} \|\mathbf{z}\|^2 \tag{38}$$

$$= \|\mathbf{e}\|^2 - \frac{\langle \mathbf{z}, \mathbf{e} \rangle^2}{\langle \mathbf{z}, \mathbf{z} \rangle}. \tag{39}$$

Hence

$$\|\mathbf{e}'\| < \|\mathbf{e}\| \tag{40}$$

a contradiction to the optimality of $\boldsymbol{\theta}^*, \theta_0^*$ leading to the optimality of the prediction errors $\mathbf{e}$.

So in sum, at the optimal solution, there is no linearly available information in the examples for us to form a non-zero inner product $\langle \mathbf{z}, \mathbf{e} \rangle$ so as to improve the overall prediction error in terms of its ℓ_2 norm. We can state this more formally as a theorem.

Theorem 1. *At the optimal solution for the parameter and offset vector $\boldsymbol{\theta}^*, \theta_0^*$, the prediction error vector $\mathbf{e} := (\tilde{y}_t - \Pr(y = 1 | \mathbf{x}_t, \boldsymbol{\theta}^*, \theta_0^*))_{t=1}^n$ is orthogonal to any linear function of the data, i.e., for any $\bar{\boldsymbol{\theta}}, \bar{\theta}_0$,*

$$\langle \mathbf{z}, \mathbf{e} \rangle = 0, \quad \textit{where} \quad \mathbf{z} := (\langle \bar{\boldsymbol{\theta}}, \mathbf{x}_t \rangle + \bar{\theta}_0)_{t=1}^n. \tag{41}$$

Furthermore if a prediction vector $\mathbf{e}$ satisfies (41), it has the minimal ℓ_2 norm of $(\tilde{y}_t - \Pr(y = 1 | \mathbf{x}_t, \boldsymbol{\theta}, \theta_0))_{t=1}^n$ over all $\boldsymbol{\theta}, \theta_0$.

4 The Maximum Likelihood Estimator in General Form (Optional)

Here, we discuss some properties of the MLE. As in usual statistics books, we denote random samples/variables as upper case letters X_i. We now assume that $X_1, X_2, \ldots$ are sampled from p_{θ^*} (we consider scalar parameters here for brevity) and define the likelihood function as $L_n(\theta) = L_n(\theta \mid X_1, \ldots, X_n) = \prod_{i=1}^n p_\theta(X_i)$. A maximum likelihood estimator (MLE) is defined as any function $\hat{\theta}_n = \hat{\theta}_n(X_1, \ldots, X_n)$ such that

$$L_n(\hat{\theta}_n) = \sup_{\theta \in \Theta} L_n(\theta). \tag{42}$$

The MLE maximizes

$$\frac{1}{n} \log L_n(\theta) - \frac{1}{n} \log L_n(\theta^*) = \frac{1}{n} \sum_{i=1}^n \log \frac{p_\theta(X_i)}{p_{\theta^*}(X_i)} \xrightarrow{a.s} -D(p_{\theta^*} \| p_\theta) \leq 0 \tag{43}$$

which converges almost surely by the strong law of large numbers to an expression that is maximized at 0 iff $\theta^* = \theta$ because the *relative entropy*

$$D(p\|q) = \mathbb{E}_p \left[\log \frac{p(X)}{q(X)} \right] \geq 0 \tag{44}$$

with equality iff $p(X) = q(X)$ almost surely. Why is this so? Assume that $p(x)$ and $q(x)$ are probability mass functions for simplicity. Consider the following chain of inequalities

$$\begin{aligned} -D(p\|q) &= \sum_x p(x) \log \frac{q(x)}{p(x)} && (45) \\ &= \sum_x p(x) \log \left[1 + \frac{q(x) - p(x)}{p(x)} \right] && (46) \\ &\leq \sum_x p(x) \left[\frac{q(x) - p(x)}{p(x)} \right] = 0 && (47) \end{aligned}$$

where the inequality follows from the fact that $\log(1 + x) \leq x$. Hence $D(p\|q) \geq 0$ and equality holds iff $p(x) = q(x)$ for all x.

4.1 Consistency of MLE (Optional)

We say that any estimator $\hat{\theta}_n = \hat{\theta}_n(X_1, \ldots, X_n)$ is *consistent* if

$$\hat{\theta}_n \xrightarrow{p} \theta^*, \quad n \to \infty. \tag{48}$$

The symbol $\xrightarrow{p}$ denotes *convergence in probability*. In other words, for every $\varepsilon > 0$ and any $\theta \in \Theta$,

$$\mathbb{P}_{\theta^*} \left(|\hat{\theta}_n - \theta^*| > \varepsilon \right) \to 0, \quad n \to \infty. \tag{49}$$

This means that as the amount of data increases the estimator is better and better. We hope that the MLE above is consistent but this is not so easy to prove. We start with an easy result.

Proposition 2 (Consistency for finite parameter set)**.** *Assume that Θ is a finite set and there exists $\varepsilon > 0$ such that*

$$D(p_{\theta_i} \| p_{\theta_j}) > \varepsilon, \qquad \forall \text{ distinct } i, j \in \Theta. \tag{50}$$

Then the MLE is consistent.

Proof. Fix $\delta > 0$ and assume without loss of generality that $\Theta = \{\theta^* = \theta_0, \theta_1, \ldots, \theta_k\}$. By the almost sure and hence in probability convergence in (43), we know that for each $j \in \Theta$,

$$\mathbb{P}_{\theta^*}\left(\left|\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} - D(p_{\theta^*}\|p_{\theta_j})\right| \geq \frac{\varepsilon}{2}\right) < \delta \tag{51}$$

for all $n > N_j \in \mathbb{N}$. By the union-of-events bound,

$$\mathbb{P}_{\theta^*}\left(\max_{j=0,1,\ldots,k}\left|\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} - D(p_{\theta^*}\|p_{\theta_j})\right| \geq \frac{\varepsilon}{2}\right) < (k+1)\delta \tag{52}$$

for all $n > N := \max_j N_j \in \mathbb{N}$. In other words,

$$\mathbb{P}_{\theta^*}\left(\max_{j=0,1,\ldots,k}\left|\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} - D(p_{\theta^*}\|p_{\theta_j})\right| < \frac{\varepsilon}{2}\right) \geq 1 - (k+1)\delta. \tag{53}$$

Define the event

$$\mathcal{A} := \left\{\max_{j=0,1,\ldots,k}\left|\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} - D(p_{\theta^*}\|p_{\theta_j})\right| < \frac{\varepsilon}{2}\right\}. \tag{54}$$

Conditioned on $\mathcal{A}$, we have

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} \\
&= \frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} - D(p_{\theta^*}\|p_{\theta_j}) + D(p_{\theta^*}\|p_{\theta_j}) &(55)\\
&\geq D(p_{\theta^*}\|p_{\theta_j}) - \left|\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} - D(p_{\theta^*}\|p_{\theta_j})\right| &(56)\\
&\geq \frac{\varepsilon}{2}, &(57)
\end{aligned}$$

where the last step follows from the hypothesis of the Proposition. However, for $j = 0$, we have $D(p_{\theta^*}\|p_{\theta_j}) = 0$ and so

$$\frac{1}{n}\sum_{i=1}^{n}\log\frac{p_{\theta^*}(X_i)}{p_{\theta_j}(X_i)} < \frac{\varepsilon}{2} \tag{58}$$

therefore the minimum of the log-likelihood function is attained for $j = 0$. Therefore, with probability exceeding $1 - (k+1)\delta$, we have $\hat{\theta}_n = \theta^*$ for $n > N$. Since δ is arbitrary,

$$\hat{\theta}_n \xrightarrow{p} \theta^*, \quad n \to \infty \tag{59}$$

which means the MLE is consistent. □

This proof worked because the class of possible models Θ was finite and we could use a union of events bound. To generalize this result to bigger classes of models we need to make more assumptions. You can look at any decent statistics text to generalize the above ideas to the case where Θ is infinite but the intuition is as follow:

1. $\hat{\theta}_n$ is the maximizer of $\frac{1}{n}\log L_n(\theta)$;
2. θ^* is the maximizer of $\mathbb{E}_{\theta^*}\log p_\theta(X)$ (relative entropy is non-negative);
3. With X_i distributed as p_{θ^*}, $\frac{1}{n}\log L_n(\theta)$ converges to $\mathbb{E}_{\theta^*}\log p_\theta(X)$ in probability;
4. So $\hat{\theta}_n$ converges to θ^* in probability.

Obviously, to make the above argument rigorous, the likelihood function ought to be sufficiently smooth (differentiable).

4.2 Asymptotic Normality of MLE (Optional)

We would like to understand the rate at which the MLE $\hat{\theta}_n$ converges to the true parameter θ^*. One way to quantify this is to try to understand the asymptotic distribution of $\hat{\theta}_n$. Let

$$\ell(\theta) = \ell(\theta; X_1) := \log L_1(\theta) = \log p_\theta(X_1) \tag{60}$$

be the log-likelihood function and assume it is twice differentiable wrt θ. Define the *Fisher information* to be

$$I(\theta^*) = \mathbb{E}_{\theta^*}[\ell'(\theta^*)^2]. \tag{61}$$

There is often an easier way to compute Fisher information.

Lemma 3. *We have*

$$I(\theta^*) = -\mathbb{E}_{\theta^*}\left[\ell''(\theta^*)\right]. \tag{62}$$

Proof. First of all,

$$\ell'(\theta) = \frac{p'_\theta(X)}{p_\theta(X)} \tag{63}$$

where throughout, the $'$ refers to differentiation with respect to θ. Also,

$$(\log p_\theta(X))'' = \frac{p''_\theta(X)}{p_\theta(X)} - \frac{(p'_\theta(X))^2}{(p_\theta(X))^2}. \tag{64}$$

Since the density integrates to one,

$$\int p_\theta(x)\,\mathrm{d}x = 1 \tag{65}$$

if we take derivatives of this equation with respect to θ (and interchange derivative and integral, which can usually be done) we will get,

$$\int \frac{\partial}{\partial\theta} p_\theta(x)\,\mathrm{d}x = 0 \qquad \int \frac{\partial^2}{\partial\theta^2} p_\theta(x)\,\mathrm{d}x = 0. \tag{66}$$

Putting the above pieces together,

$$\begin{aligned}
\mathbb{E}_{\theta^*}\ell''(\theta^*) &= \mathbb{E}_{\theta^*}\left[\frac{\partial^2}{\partial\theta^2}\log p_{\theta^*}(X)\right] && (67)\\
&= \int (\log p_{\theta^*}(x))'' p_{\theta^*}(x)\,\mathrm{d}x && (68)\\
&= \int \left(\frac{p''_{\theta^*}(x)}{p_{\theta^*}(x)} - \frac{(p'_{\theta^*}(x))^2}{(p_{\theta^*}(x))^2}\right) p_{\theta^*}(x)\,\mathrm{d}x && (69)\\
&= \int p''_{\theta^*}(x)\,\mathrm{d}x - \mathbb{E}_{\theta^*}[\ell'(\theta^*)^2] = -I(\theta^*). && (70)
\end{aligned}$$

□

Now we are ready to state the result concerning asymptotic normality of the MLE.

Theorem 4 (Asymptotic normality)**.** *Assume that the MLE $\hat{\theta}_n$ is consistent. We have the convergence in distribution*

$$\sqrt{n}\left(\hat{\theta}_n - \theta^*\right) \xrightarrow{d} \mathcal{N}(0, 1/I(\theta^*)). \tag{71}$$

As we can see, the asymptotic variance/dispersion of the estimate around true parameter will be smaller when Fisher information is larger.

Proof. Since the MLE is a maximizer of $\ell_n(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(\theta; X_i)$, we have

$$\ell_n'(\hat{\theta}_n) = 0. \tag{72}$$

Recall the mean-value theorem, which says that for any differentiable function f,

$$f(a) = f(b) + f'(c)(a-b) \tag{73}$$

for some $c \in [a, b]$. Applying the MVT to the log-likelihood function, we obtain

$$0 = \ell_n'(\hat{\theta}_n) = \ell_n'(\theta^*) + \ell_n''(\theta_1)(\hat{\theta}_n - \theta^*) \tag{74}$$

for some $\theta_1 \in [\hat{\theta}_n, \theta^*]$. Hence,

$$\sqrt{n}(\hat{\theta}_n - \theta^*) = -\frac{\sqrt{n}\ell_n'(\theta^*)}{\ell_n''(\theta_1)}. \tag{75}$$

Note that since θ^* maximizes the likelihood, $\mathbb{E}_{\theta^*}\ell'(\theta^*; X_i) = 0$. Now, consider the numerator

$$\sqrt{n}\ell_n'(\theta^*) = \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^n \ell'(\theta^*; X_i)\right) \tag{76}$$

$$= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^n [\ell'(\theta^*; X_i) - \mathbb{E}_{\theta^*}\ell'(\theta^*; X_i)]\right) \tag{77}$$

$$\xrightarrow{d} \mathcal{N}(0, \mathsf{Var}_{\theta^*}(\ell'(\theta^*; X_i)) \tag{78}$$

where the last step follows from the central limit theorem. Since $\theta_1 \in [\hat{\theta}_n, \theta^*]$ and by consistency, $\hat{\theta}_n \to \theta^*$ in probability, $\theta_1 \to \theta^*$ in probability. Hence, the denominator behaves as

$$\ell_n''(\theta_1) \to \mathbb{E}_{\theta^*}\ell''(\theta^*, X_1) = -I(\theta^*) \tag{79}$$

where the last equality is by Lemma 3. Since $\mathsf{Var}_{\theta^*}(\ell'(\theta^*; X_i)) = I(\theta^*)$, convergence in distribution is established by uniting (78) and (79). □

MA4270 : Linear Regression and the Bias-Variance Tradeoff

Vincent Y. F. Tan

September 9, 2021

Here, we discuss linear regression and the bias-variance tradeoff.
These notes are heavily adapted from Jaakkola's MIT notes and Jonathan Scarlett's NUS notes.
You need to know everything except Example 2.

1 Regression

Regression is just like classification except that y_t are no longer restricted to belong to a finite set. Rather it can take on uncountably[1] many values. In the case of regression, we typically do not say that y_t is the label. Rather we use the term *target variable* or *outcome variable* or *dependent variable* to refer to the y_t's. For instance y_t could take any value in the interval $[0, 1]$ (which is uncountable) or it could take values in $\mathbb{R}$. In either case, the data examples in the dataset $\mathcal{D}$, denoted as $(\mathbf{x}_t, y_t)$ belong to $\mathbb{R}^d \times \mathbb{R}$, which means that each feature vector or training sample $\mathbf{x}_t \in \mathbb{R}^d$ (d-dimensional Euclidean space) and each target variable $y_t \in \mathbb{R}$ is a real number. We wish to learn a *regressor* $f : \mathbb{R}^d \to \mathbb{R}$ which is a function that brings us from the feature space to the set of real numbers. The regressor f should be designed such that given a new feature vector $\mathbf{x}'$, its corresponding target value y' is "well predicted", in some precise mathematical sense, by $f(\mathbf{x}')$.

Again some examples would make this clear.

- Say each feature vector $x_i \in \mathbb{R}$ is scalar (so $d = 1$) and represents the height of a student. The target variable y_t represents the weight of the same student. Given the data of $m = 600$ students in MA4270 captured in the dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t) : 1 \le t \le 600\}$, I am tasked to learn a regressor $f : \mathbb{R} \to \mathbb{R}$ so that I can use the height of Alice x' in a new class to predict her weight $y' = f(x')$. See Fig. 1.
- Say each feature vector $\mathbf{x}_t \in \mathbb{R}^3$ is a 3-dimensional vector and $x_{i,1}$ represents the air pressure at location i, $x_{t,2}$ represents the amount of rainfall at location i and $x_{i,3}$ represents the "amount of greenery" at location i. Each y_t corresponds to the temperature at location i. The temperature can take on uncountably many values—it is a real number. Given the dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t) : 1 \le t \le n\}$, we would like to learn a regressor $f : \mathbb{R}^3 \to \mathbb{R}$ such that if I give you the feature vector of new location $\mathbf{x}'$, you can tell me the temperature $y' = f(\mathbf{x}')$.

Regression also belongs to the class of *supervised* learning methods.

2 Probabilistic Modeling via Normal Distributions

We consider linear (or more accurately, affine) functions of the input $\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0$. By treating this as a mean prediction more formally, we are stating that the expected value of the response variable (what we are trying to predict), conditioned on the input being $\mathbf{x}$ is $\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0$. More succinctly,

$$\mathbb{E}[y \mid \mathbf{x}] = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0. \tag{1}$$

[1] A set $\mathcal{A}$ is *countable* if there is a bijection between $\mathcal{A}$ and the set of natural numbers $\mathbb{N}$. A set $\mathcal{A}$ is *uncountable* if its cardinality is strictly larger than that of the natural numbers. Check that the set of rationals is countable. A well-known, but non-trivial, fact (due to Cantor) is that the interval $[0, 1]$ is uncountable.

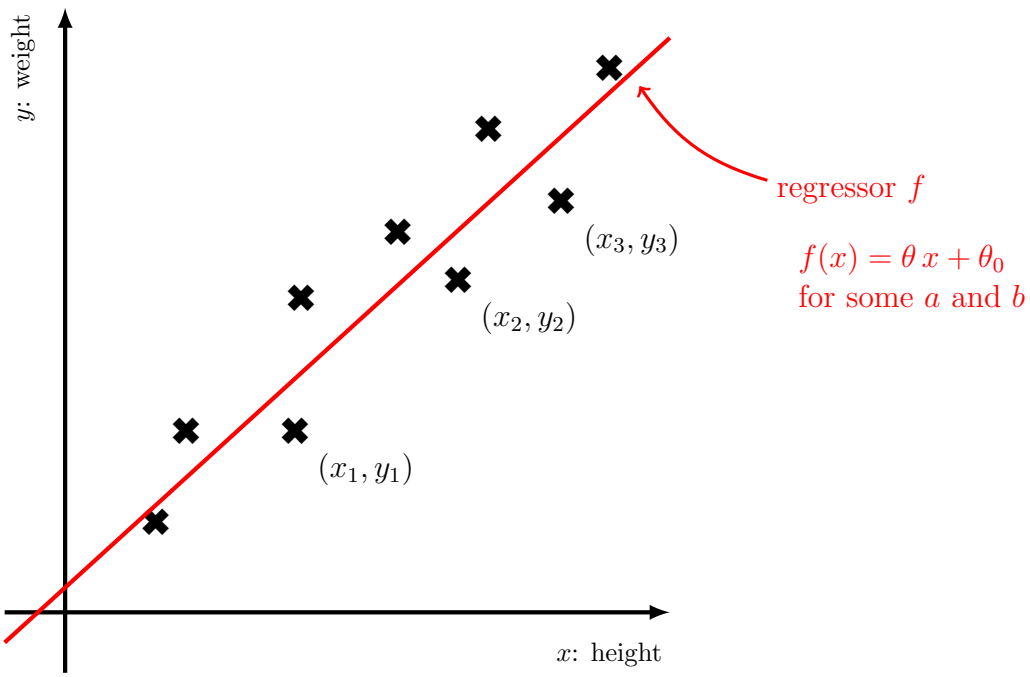


Figure 1: Height and weight regression example

It remains to associate a distribution over the responses around such mean prediction. The simplest symmetric distribution is the normal (Gaussian) distribution. In other words, we say that the responses y follow the normal pdf

$$\mathcal{N}(y; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right). \tag{2}$$

where $\mu = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0$. Our response model is then

$$P(y \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = \mathcal{N}(y; \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0, \sigma^2). \tag{3}$$

This can also be written in a more explicit manner as follows

$$y = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0 + \epsilon \quad \text{where} \quad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{4}$$

When we have n training samples and targets in the dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, the statistical model governing them is

$$y_t = \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle + \theta_0 + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2), \tag{5}$$

and the ϵ_t's are mutually independent.

3 Maximizing the Likelihood

Now that we have a statistical model, we can use the dataset $\mathcal{D}$ to find the "best" parameters $(\boldsymbol{\theta}, \theta_0, \sigma^2)$. What constitutes "best"? Well we have seen that one way to learn the parameters is by maximum likelihood estimation. The likelihood of the data given the parameters is

$$L(\boldsymbol{\theta}, \theta, \sigma^2) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle - \theta_0)^2}{2\sigma^2}\right). \tag{6}$$

Note that in addition to $(\boldsymbol{\theta}, \theta_0)$, the noise variance σ^2 is also a parameter we wish to estimate. It accounts for errors not captured by the linear model. In terms of the log-likelihood, we try to maximize

$$\ell(\boldsymbol{\theta}, \theta, \sigma^2) = \sum_{t=1}^{n} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle - \theta_0)^2}{2\sigma^2} \right) \right] \tag{7}$$

$$= \sum_{t=1}^{n} \left[-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\left(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle - \theta_0\right)^2 \right] \tag{8}$$

$$= \text{const} - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{t=1}^{n} \left(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle - \theta_0\right)^2. \tag{9}$$

where const absorbs terms that do not depend on the parameters. Now, the problem of estimating the parameters $\boldsymbol{\theta}$ and θ_0 is nicely decoupled from estimating σ^2. In other words, we can find the maximizing $\boldsymbol{\theta}$ and θ_0 by simply minimizing the mean squared error

$$\text{MSE} = \sum_{t=1}^{n} \left(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle - \theta_0\right)^2. \tag{10}$$

At this point, it is more convenient for us to write things in matrix form. Let $\mathbf{X} \in \mathbb{R}^{n \times (d+1)}$ be a matrix whose rows index training samples with 1 added at the end, i.e.,

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \\ \mathbf{x}_n^\top & 1 \end{bmatrix} \in \mathbb{R}^{n \times (d+1)} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n. \tag{11}$$

The matrix $\mathbf{X}$ is known as the *design matrix* and $\mathbf{y}$ the *target vector*. We see that the MSE can be rewritten as

$$\text{MSE} = \sum_{t=1}^{n} \left(y_t - \begin{bmatrix} \boldsymbol{\theta} \\ \theta_0 \end{bmatrix}^\top \begin{bmatrix} \mathbf{x}_t \\ 1 \end{bmatrix} \right)^2 = \sum_{t=1}^{n} \left(y_t - \begin{bmatrix} \mathbf{x}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \theta_0 \end{bmatrix} \right)^2 \tag{12}$$

$$= \left\| \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \\ \mathbf{x}_n^\top & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\theta} \\ \theta_0 \end{bmatrix} \right\|^2 = \left\| \mathbf{y} - \mathbf{X} \begin{bmatrix} \boldsymbol{\theta} \\ \theta_0 \end{bmatrix} \right\|^2 \tag{13}$$

For simplicity, we let $\widetilde{\boldsymbol{\theta}} := [\boldsymbol{\theta}^\top \ \theta_0]^\top$. Then, the MSE further reduces to

$$\text{MSE} = \left\| \mathbf{y} - \mathbf{X}\widetilde{\boldsymbol{\theta}} \right\|^2 = \mathbf{y}^\top \mathbf{y} - 2\widetilde{\boldsymbol{\theta}}^\top \mathbf{X}^\top \mathbf{y} + \widetilde{\boldsymbol{\theta}}^\top \mathbf{X}^\top \mathbf{X} \widetilde{\boldsymbol{\theta}}. \tag{14}$$

Differentiating this with respect to $\widetilde{\boldsymbol{\theta}}$ yields

$$\frac{\mathrm{dMSE}}{\mathrm{d}\widetilde{\boldsymbol{\theta}}} = -2\mathbf{X}^\top \mathbf{y} + 2(\mathbf{X}^\top \mathbf{X})\widetilde{\boldsymbol{\theta}}. \tag{15}$$

Setting this to be equal to zero and assuming that $\mathbf{X}$ *has full column rank*, we obtain

$$\widehat{\widetilde{\boldsymbol{\theta}}} = \begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{16}$$

Note that the optimal parameter values are linear functions of the observed responses $\mathbf{y}$. We will make use of this property later on. The dependence on the training samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ (or on the matrix $\mathbf{X}$) is non-linear, however. In fact, the symmetric positive semi-definite matrix $\mathbf{X}^\top\mathbf{X}$ has a name and is known as the *Gram matrix*.

Now we can do the same to find the optimal σ^2.

$$\widehat{\sigma}^2 = \frac{1}{n}\sum_{t=1}^{n}\left(y_t - \langle\widehat{\boldsymbol{\theta}}, \mathbf{x}_t\rangle + \widehat{\theta}_0\right)^2. \tag{17}$$

This is the average squared prediction error and is the residual that the linear model cannot explain.

4 Bias and variance of the parameter estimates

How good are the estimates of the parameters in (16) and (17)? To answer this question, we make the somewhat strong assumption that the data is generated according to a fixed linear model. The observed responses are thus

$$y_t = \langle\boldsymbol{\theta}^*, \mathbf{x}_t\rangle + \theta_0^* + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, (\sigma^2)^*), \tag{18}$$

for some fixed but unknown $(\boldsymbol{\theta}^*, \theta_0^*, (\sigma^2)^*)$. In matrix form

$$\mathbf{y} = \mathbf{X}\begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0\end{bmatrix} + \boldsymbol{\epsilon}, \tag{19}$$

where $\mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0}$ and $\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = (\sigma^2)^*\mathbf{I}$. If the data were generated from this model and we used the estimates in (16) and (17), we can calculate the bias and variance of these estimators.

Let's start with the bias, defined as

$$\text{Bias} := \mathbb{E}\left[\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} - \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0\end{bmatrix} \;\middle|\; \mathbf{X}\right]. \tag{20}$$

We claim that the bias is $\mathbf{0}$. Indeed, we have

$$\mathbb{E}\left[\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} \;\middle|\; \mathbf{X}\right] = \mathbb{E}\left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y} \;\middle|\; \mathbf{X}\right] = \mathbb{E}\left[\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\left(\mathbf{X}\begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0^*\end{bmatrix} + \boldsymbol{\epsilon}\right) \;\middle|\; \mathbf{X}\right] \tag{21}$$

$$= \mathbb{E}\left[\begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0^*\end{bmatrix} + (\mathbf{X}^\top\mathbf{X})\mathbf{X}^\top\boldsymbol{\epsilon} \;\middle|\; \mathbf{X}\right] = \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0^*\end{bmatrix} + (\mathbf{X}^\top\mathbf{X})\mathbf{X}^\top\mathbb{E}[\boldsymbol{\epsilon} \mid \mathbf{X}] = \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0^*\end{bmatrix}, \tag{22}$$

where the second equality holds due to (19) and the last equality because $\boldsymbol{\epsilon}$ has zero mean. Thus our parameter estimates are therefore *unbiased* or "correct on average" when averaging is over possible training sets we could generate. The averaging here is conditioned on the specific training inputs.

Now we consider the (conditional) covariance of the estimates where the expectation is again over the noise in the outputs:

$$\text{Cov}\left[\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} \;\middle|\; \mathbf{X}\right] = \mathbb{E}\left[\left(\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} - \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta^*\end{bmatrix}\right)\left(\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} - \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta^*\end{bmatrix}\right)^\top \;\middle|\; \mathbf{X}\right] \tag{23}$$

$$= \mathbb{E}\left[\left((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\epsilon}\right)\left((\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\epsilon}\right)^\top \mid \mathbf{X}\right] \tag{24}$$

$$= \mathbb{E}\left[(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1} \mid \mathbf{X}\right] \tag{25}$$

$$= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top \mid \mathbf{X}\right]\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1} \tag{26}$$

$$= (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top(\sigma^2)^*\mathbf{I}\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1} \tag{27}$$

$$= (\sigma^2)^*(\mathbf{X}^\top\mathbf{X})^{-1}. \tag{28}$$

Notice that the larger the noise variance of the model $(\sigma^2)^*$, the larger the covariance of the estimator. The covariance measures the spread of the estimates; how closely it is concentrated. Furthermore, notice that

$$\mathbf{X}^\top \mathbf{X} = \sum_{t=1}^{n} \begin{bmatrix} \mathbf{x}_t \\ 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_t & 1 \end{bmatrix} = \sum_{t=1}^{n} \mathbf{C}_t = n \left(\frac{1}{n} \sum_{t=1}^{n} \mathbf{C}_t \right) \approx n\mathbf{C} \tag{29}$$

so the conditional covariance in (28) drops as $O(1/n)$ as $n \to \infty$.

5 Analyzing the Mean Squared Error in terms of the Bias and Variance

Ultimately, we are most interested in the mean squared error (MSE) of the estimated parameters in (16). To do this, we first perform a decomposition of the MSE into a squared bias term and a variance term. Let $\mathbf{z}^*$ be the true parameters and $\mathbf{z}$ an estimate of $\mathbf{z}^*$. Then the MSE can be decomposed as

$$\begin{aligned}
\text{MSE} &= \mathbb{E}\left[\|\mathbf{z} - \mathbf{z}^*\|^2\right] && (30)\\
&= \mathbb{E}\left[\|(\mathbf{z} - \mathbb{E}[\mathbf{z}]) + (\mathbb{E}[\mathbf{z}] - \mathbf{z}^*)\|^2\right] && (31)\\
&= \mathbb{E}\left[\|\mathbf{z} - \mathbb{E}[\mathbf{z}]\|^2\right] + \|\mathbb{E}[\mathbf{z}] - \mathbf{z}^*\|^2 + 2\mathbb{E}\left[(\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top\right](\mathbb{E}[\mathbf{z}] - \mathbf{z}^*) && (32)\\
&= \mathbb{E}\left[\|\mathbf{z} - \mathbb{E}[\mathbf{z}]\|^2\right] + \|\mathbb{E}[\mathbf{z}] - \mathbf{z}^*\|^2 && (33)
\end{aligned}$$

where the last equality holds because $\mathbb{E}[\mathbf{z} - \mathbb{E}[\mathbf{z}]] = \mathbf{0}$. It also holds that

$$\begin{aligned}
\mathbb{E}\left[\|\mathbf{z} - \mathbb{E}[\mathbf{z}]\|^2\right] &= \mathbb{E}\left[(\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top (\mathbf{z} - \mathbb{E}[\mathbf{z}])\right] && (34)\\
&= \mathbb{E}\left[\text{Tr}\left((\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top (\mathbf{z} - \mathbb{E}[\mathbf{z}])\right)\right] && (35)\\
&= \mathbb{E}\left[\text{Tr}\left((\mathbf{z} - \mathbb{E}[\mathbf{z}])(\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top\right)\right] && (36)\\
&= \text{Tr}\left(\mathbb{E}\left[(\mathbf{z} - \mathbb{E}[\mathbf{z}])(\mathbf{z} - \mathbb{E}[\mathbf{z}])^\top\right]\right) && (37)\\
&= \text{Tr}\left(\text{Cov}(\mathbf{z})\right) && (38)
\end{aligned}$$

Let's try to understand the terms in (33). The second term is known as the bias squared. On average, our estimator behaves as $\mathbb{E}[\mathbf{z}]$. The deviation of this from the ground truth $\mathbf{z}^*$ is captured by the second term. The first term doesn't involve the ground truth $\mathbf{z}^*$. It is simply how much fluctuation there is in the estimate $\mathbf{z}$ and the fluctuation is captured by its covariance $\mathbb{E}\left[\|\mathbf{z} - \mathbb{E}[\mathbf{z}]\|^2\right]$, which is a generalization of the familiar notion of variance. Ideally, we would like both terms to be small. See Fig. 2 for an illustration.

Applying the decomposition in (33) to our setting, we obtain

$$\begin{aligned}
\text{MSE} &= \mathbb{E}\left[\left\|\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} - \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta^*\end{bmatrix}\right\|^2 \;\middle|\; \mathbf{X}\right] = \text{Tr}\left(\text{Cov}\left(\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} \;\middle|\; \mathbf{X}\right)\right) + \left\|\mathbb{E}\left[\begin{bmatrix}\widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0\end{bmatrix} \;\middle|\; \mathbf{X}\right] - \begin{bmatrix}\boldsymbol{\theta}^* \\ \theta_0^*\end{bmatrix}\right\|^2 && (39)\\
&= (\sigma^2)^* \text{Tr}\left((\mathbf{X}^\top \mathbf{X})^{-1}\right). && (40)
\end{aligned}$$

6 Penalized log-likelihood and Ridge regression

It seems inconceivable that we can do better than the MSE in (40). But as can be seen from the decomposition of the MSE, there are two components to it. When we use the maximum likelihood estimate, the bias term is identically equal to zero but the variance term $(\sigma^2)^* \text{Tr}\left((\mathbf{X}^\top \mathbf{X})^{-1}\right)$ is "large". Is there a way to "balance" these two terms so that we achieve the smallest possible MSE?

When the number of training examples is small, i.e., not too much larger than the number of parameters (dimension of the inputs), it is often beneficial to regularize the parameter estimates. We will derive the

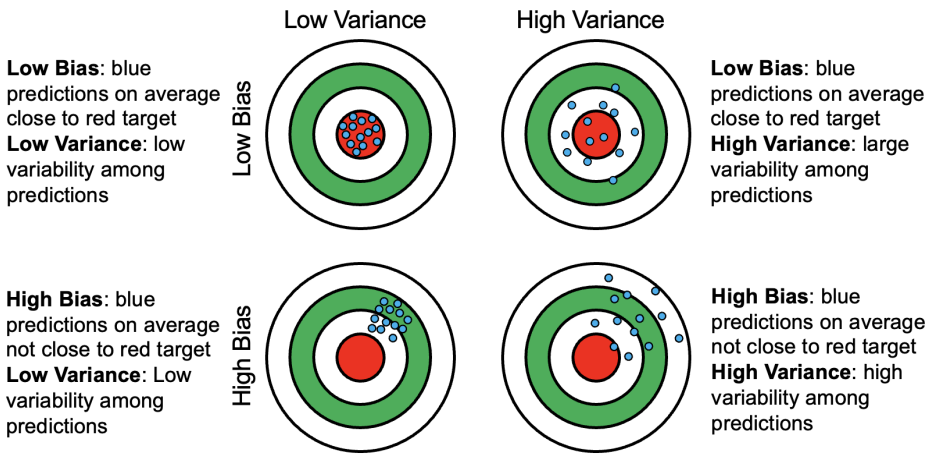


Figure 2: Illustration of bias and variance

form of regularization here by assigning a prior distribution over the parameters $P(\boldsymbol{\theta}, \theta_0)$. The purpose of the prior is to prefer small parameter values (predict values close to zero) in the absence of data. Specifically, we will look at simple zero mean Gaussian distribution

$$P(\boldsymbol{\theta}, \theta_0; \gamma) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\theta} \\ \theta_0\end{bmatrix}; \begin{bmatrix}\mathbf{0} \\ 0\end{bmatrix}, \gamma\mathbf{I}\right) = \mathcal{N}(\theta_0; 0, \gamma) \prod_{j=1}^{d} \mathcal{N}(\theta_j; 0, \gamma), \tag{41}$$

where the variance parameter γ in the prior distribution specifies how strongly we wish to bias the parameters towards zero.

By combining the log-likelihood criterion with the prior we obtain a *penalized log-likelihood function* (penalized by the prior):

$$\widetilde{\ell}(\boldsymbol{\theta}, \theta, \sigma^2) = \sum_{t=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t\rangle - \theta_0)^2}{2\sigma^2}\right)\right] + \log P(\boldsymbol{\theta}, \theta_0; \gamma) \tag{42}$$

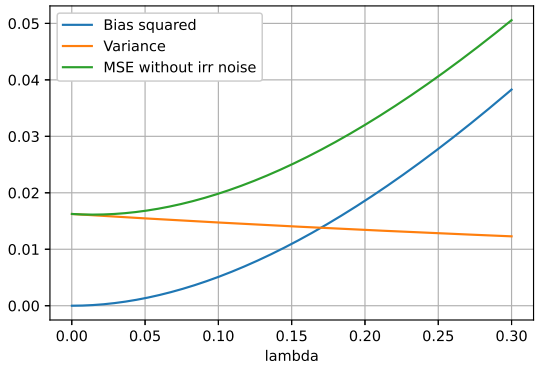
$$= \text{const} - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{n}(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t\rangle - \theta_0)^2 - \frac{1}{2\gamma}\left(\theta_0^2 + \sum_{j=1}^{d}\theta_j^2\right) - \frac{d+1}{2}\log\gamma. \tag{43}$$

It is convenient to tie the prior variance γ to the noise variance σ^2 according to $\gamma = \sigma^2/\lambda$ for some $\lambda > 0$. Incorporating this parametrization into the penalized log-likelihood, we obtain

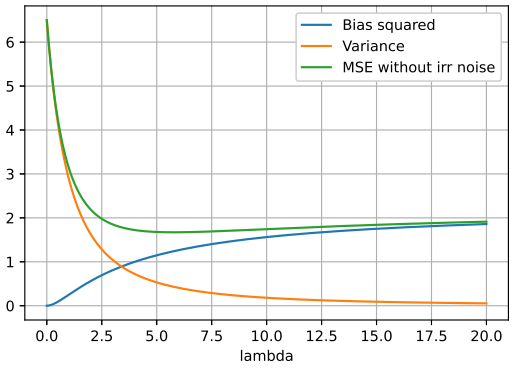
$$\widetilde{\ell}(\boldsymbol{\theta}, \theta, \sigma^2) = \text{const} - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{n}(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t\rangle - \theta_0)^2 - \frac{\lambda}{2\sigma^2}\left(\theta_0^2 + \sum_{j=1}^{d}\theta_j^2\right) - \frac{d+1}{2}\log\left(\frac{\sigma^2}{\lambda}\right) \tag{44}$$

$$= \text{const} - \frac{n+d+1}{2}\log(\sigma^2) + \frac{d+1}{2}\log\lambda - \frac{1}{2\sigma^2}\left[\sum_{t=1}^{n}(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t\rangle - \theta_0)^2 + \lambda\left(\theta_0^2 + \sum_{j=1}^{d}\theta_j^2\right)\right] \tag{45}$$

where again the estimation of $\boldsymbol{\theta}$ and θ_0 separates from setting the noise variance σ^2. Note that this separation is achieved because we tied the prior and noise variance parameters. The above regularized problem of finding the parameter estimates $\widehat{\boldsymbol{\theta}}$ and $\widehat{\theta}_0$ is known as *Ridge regression* or *Tikhonov regularization*.



Small noise



Large noise

Figure 3: Bias-Variance tradeoff for a prediction of a point at a test $x = 1.5$ for Example 1. There is a significant reduction in the MSE for positive λ for the large noise case.

In a similar way as before, we can differentiate this and set to zero and we obtain

$$\begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{46}$$

Note that in the presence of the regularization, we need not check that $\mathbf{X}$ has full column rank because $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ is always invertible; its minimum eigenvalue is at least $\lambda > 0$.

When $\lambda = 0$ (no regularization), we saw that the parameter estimates we unbiased; recall (22). Is the regularized solution in (46) unbiased for fixed $\lambda > 0$? No! It is not. We can do a simple calculation to check this. Indeed,

$$\mathbb{E}\left[\begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} \;\middle|\; \mathbf{X}\right] = (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X} \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} \tag{47}$$

$$= (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \left((\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X}) - \lambda \mathbf{I}\right) \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} \tag{48}$$

$$= \left(\mathbf{I} - \lambda (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1}\right) \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix}. \tag{49}$$

Note that for $\lambda > 0$, the matrix $\mathbf{I} - \lambda(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1}$ is not the identity matrix, so the estimator is *biased*. However, note that $\mathbf{I} - \lambda(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1}$ is a positive semi-definite matrix whose eigenvalues all less than one. The parameter estimates are therefore shrunk towards zero and more so the larger the value of λ. This is what we would expect since we explicitly favored small parameter values with the prior penalty. The bias is

$$\text{Bias} = -\lambda (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix}. \tag{50}$$

Exercise 1. *By using the same strategy as was done for the conditional expectation, show that the conditional covariance is*

$$\text{Cov}\left[\begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} \;\middle|\; \mathbf{X}\right] = (\sigma^2)^* (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} - \lambda (\sigma^2)^* (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-2}. \tag{51}$$

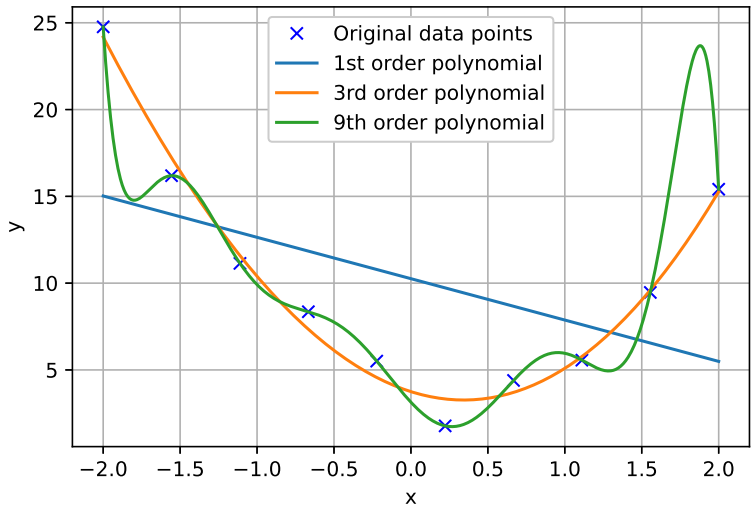


Figure 4: Polynomial regression with various orders

By combining the conditional expectation and conditional covariance, we see that

$$\mathrm{MSE}_\lambda = \left[\left\| \begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} \right\|^2 \,\middle|\, \mathbf{X} \right] \tag{52}$$

$$= (\sigma^2)^* \mathrm{Tr}\left[(\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} - \lambda (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-2} \right] + \lambda^2 \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix}^\top (\lambda \mathbf{I} + \mathbf{X}^\top \mathbf{X})^{-1} \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix}. \tag{53}$$

Somewhat surprisingly, this can be smaller than the MSE of the unbiased estimator!

Example 1. Consider the scalar case in which $x_1 = -1$, $x_2 = 1$ so the design matrix with offset is

$$\mathbf{X} = \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \tag{54}$$

After some calculations (which you should carry out), the MSE with regularization parameter $\lambda > 0$ in (53) can be evaluated to

$$\mathrm{MSE}_\lambda = \frac{4(\sigma^2)^*}{(2+\lambda)^2} + \frac{\lambda^2}{(2+\lambda)^2}((\theta^*)^2 + (\theta_0^*)^2). \tag{55}$$

Note that when $\lambda \to 0^+$ we have $\mathrm{MSE}_\lambda \to$ MSE for the unregularized case in (40). So the MSE is continuous at 0. In the noisy case $(\sigma^2)^* > (\theta^*)^2 + (\theta_0^*)^2$, we can set $\lambda = 2$ and obtain

$$\mathrm{MSE}_2 = \frac{4(\sigma^2)^*}{16} + \frac{4}{16}((\theta^*)^2 + (\theta_0^*)^2) < \frac{(\sigma^2)^*}{2} < \mathrm{MSE}. \tag{56}$$

See Fig. 3 for an illustration of the bias-variance tradeoff when we consider this model and aim to predict the value of y at the point $x = 1.5$.

Example 2. *(Optional)* Finally, we show an example of polynomial fitting. In Fig. 4, we generated $n = 10$ data points from the one-dimensional quadratic model

$$y = 4x^2 - 2x + 3 + e, \tag{57}$$

where e is a zero-mean unit variance Gaussian random variable. More precisely, the points on the abscissa x are equally spaced between -2 and 2, while the points on the ordinate are generated according to (57) given the x's. The 10 points are plotted as blue crosses in Fig. 4. If we only observed these points, in general, we will not be able to know the order of the polynomial and, of course, the coefficients. We found and plotted the least squares curves assuming polynomials with orders 1, 2 and 9.

- For the polynomial of order 1, we are using affine functions of the form $y = \theta_0 + \theta_1 x$. The fit is not good as the model is too simple and the error between the training points and the learned line is bad. If we have a new data point x_{new}, we expect that the model fit is also bad. For example, if $x_{\text{new}} = 1.8$, the prediction is around 6.2 which is very far from the ground truth $4(1.8)^2 - 2(1.8) + 3 = 12.36$. This can be quantified more precisely in terms of the mean-squared error (MSE), which will be high. This is a case of underfitting.

- For the polynomial of order 9, we are using functions of the form $y = \sum_{i=0}^{9} \theta_i x^i$. Since there are only 10 data points, it can seen that the fit on the training points is perfect.[2] However, if we have a new data point x_{new}, we also expect that the model fit is very bad. For example if $x_{\text{new}} = 1.8$, the prediction is around 23, which is again very far from the ground truth 12.36. This is a case of severe overfitting.

- For the polynomial of order 2, we are using quadratic functions $y = \theta_0 + \theta_1 x + \theta_2 x^2$. The fit to the training data points is not bad; the errors are all reasonably small. More importantly, on new data points, we can see that the prediction is also good. This is unsurprising as the model we posit, a quadratic, is the same as the true one.

[2] This is because the training points are distinct and so the 10 by 10 polynomial design matrix, a so-called Vandermonde matrix, is non-singular.

MA4270: Active Learning and Introduction to Kernels

Vincent Y. F. Tan

August 29, 2021

In this lecture, we discuss active learning in linear regression and introduce the notion of kernels in preparation for the dual formulation of SVMs. You need to know everything except Section 1.2 in this document.

1 Active Learning in Linear Regression

1.1 Formulation

Last time, we studied the linear model

$$y_t = \langle \boldsymbol{\theta}^*, \mathbf{x}_t \rangle + \theta_0^* + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, (\sigma^2)^*), \tag{1}$$

for some fixed but unknown $(\boldsymbol{\theta}^*, \theta_0^*, (\sigma^2)^*)$. We saw that the *maximum likelihood estimate* of $(\boldsymbol{\theta}, \theta_0)$ is

$$\begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}. \tag{2}$$

where the *design matrix* and *target vector* are

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \mathbf{x}_2^\top & 1 \\ \vdots & \\ \mathbf{x}_n^\top & 1 \end{bmatrix} \in \mathbb{R}^{n \times (d+1)} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n. \tag{3}$$

We saw that the mean squared error (MSE) is

$$\text{MSE} = \mathbb{E}\left[\left\| \begin{bmatrix} \widehat{\boldsymbol{\theta}} \\ \widehat{\theta}_0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} \right\|^2 \,\middle|\, \mathbf{X} \right] = (\sigma^2)^* \text{Tr}\left((\mathbf{X}^\top \mathbf{X})^{-1} \right), \tag{4}$$

where the conditional expectation is over how the responses y_t are generated given the training examples $\mathbf{x}_t$. Thus the only source of randomness in the expectation is the noise $\{\epsilon_t\}$.

The question we want to answer in the first half of this lecture is the following. We want to *actively* select the input points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ *sequentially* so as to minimize the MSE or estimation error. This is an active learning (experiment design) problem. By letting the method guide the selection of the training examples (inputs), we will generally need far fewer examples in comparison to selecting them at random from some underlying distribution, database, or trying available experiments at random. Since the noise variance $(\sigma^2)^*$ is unknown but constant, it appears as a multiplicative factor in (4) so we do not have to worry about it when we select the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to minimize the MSE.

Let us now solve the problem given that we have already selected n samples, resulting in the design matrix $\mathbf{X} \in \mathbb{R}^{n\times(d+1)}$ in (3). Write the inverse of the Gram matrix as $\mathbf{A} = (\mathbf{X}^\top\mathbf{X})^{-1}$. We are trying to select another *row* $[\mathbf{x}^\top \ 1]$ to $\mathbf{X}$ to form the new design matrix

$$\mathbf{X}_{n+1} = \begin{bmatrix} \mathbf{x}_1^\top & 1 \\ \vdots & \\ \mathbf{x}_n^\top & 1 \\ \mathbf{x}^\top & 1 \end{bmatrix} \tag{5}$$

In an applied context we are typically constrained by what $\mathbf{x}$ can be (e.g., due to the experimental setup). Taking this into account, let's calculate the effect of adding a new row (that satisfies the constraints) as follows:

$$\mathbf{X}_{n+1}^\top\mathbf{X}_{n+1} = (\mathbf{X}^\top\mathbf{X}) + \begin{bmatrix}\mathbf{x}\\1\end{bmatrix}^\top \begin{bmatrix}\mathbf{x}\\1\end{bmatrix} = \mathbf{A}^{-1} + \mathbf{v}\mathbf{v}^\top, \tag{6}$$

where $\mathbf{v} = [\mathbf{x}^\top \ 1]^\top$. We would like to find a valid $\mathbf{v}$ that minimizes the trace of (6), i.e., $\mathrm{Tr}[(\mathbf{A}^{-1} + \mathbf{v}\mathbf{v}^\top)^{-1}]$.

By the matrix inversion lemma.[1]

$$(\mathbf{A}^{-1} + \mathbf{v}\mathbf{v}^\top)^{-1} = \mathbf{A} - \frac{1}{1+\mathbf{v}^\top\mathbf{A}\mathbf{v}}\mathbf{A}\mathbf{v}\mathbf{v}^\top\mathbf{A} \tag{7}$$

so its trace becomes

$$\mathrm{Tr}\big[(\mathbf{A}^{-1} + \mathbf{v}\mathbf{v}^\top)^{-1}\big] = \mathrm{Tr}\big[\mathbf{A}\big] - \frac{1}{1+\mathbf{v}^\top\mathbf{A}\mathbf{v}}\mathrm{Tr}\big[(\mathbf{A}\mathbf{v})(\mathbf{v}^\top\mathbf{A})\big] \tag{8}$$

$$= \mathrm{Tr}\big[\mathbf{A}\big] - \frac{1}{1+\mathbf{v}^\top\mathbf{A}\mathbf{v}}\mathrm{Tr}\big[\mathbf{v}^\top\mathbf{A}\mathbf{A}\mathbf{v}\big] \tag{9}$$

$$= \mathrm{Tr}\big[\mathbf{A}\big] - \frac{\mathbf{v}^\top\mathbf{A}\mathbf{A}\mathbf{v}}{1+\mathbf{v}^\top\mathbf{A}\mathbf{v}}, \tag{10}$$

where (9) follows since $\mathrm{Tr}[\mathbf{B}\mathbf{C}] = \mathrm{Tr}[\mathbf{C}\mathbf{B}]$ and $\mathbf{A}$ is a symmetric matrix so $\mathbf{A}^\top = \mathbf{A}$ and (10) follows since the trace of a scalar is the scalar itself and $\mathbf{v}^\top\mathbf{A}\mathbf{A}\mathbf{v}$ is a scalar.

Note that since $\mathrm{Tr}[\mathbf{A}] \propto \mathrm{Tr}[(\mathbf{X}^\top\mathbf{X})^{-1}]$ is the mean squared error before adding the new example, any choice of $\mathbf{v}$, i.e., any additional example $\mathbf{x}$ will reduce the mean squared error since $\mathbf{v}^\top\mathbf{A}\mathbf{A}\mathbf{v}$ and $\mathbf{v}^\top\mathbf{A}\mathbf{v}$ are non-negative (because $\mathbf{A}^\top\mathbf{A}$ and $\mathbf{A}$ are positive semidefinite). We are interested in finding the one that reduces the error the most. This is the example $\mathbf{x}$, resulting in a $\mathbf{v}$, that maximizes

$$f(\mathbf{v}) = \frac{\mathbf{v}^\top\mathbf{A}\mathbf{A}\mathbf{v}}{1+\mathbf{v}^\top\mathbf{A}\mathbf{v}}. \tag{11}$$

We claim that this term is upper bounded by the largest eigenvalue of $\lambda_1(\mathbf{A})$ and it is achieved by $t\mathbf{u}_1$ where $\mathbf{u}_1$ is the eigenvector that corresponds to λ_1. We justify this in the next section. If we accept this result, the maximizing vector would be of infinite length and proportional to the eigenvector of $\mathbf{A}$ with the largest eigenvalue (all the eigenvalues of $\mathbf{A}$ are positive as it is an inverse of a positive definite matrix $\mathbf{X}^\top\mathbf{X}$). It is indeed advantageous in linear regression to have the input points as far from each other as possible (see Fig. 1). If we constrain $\|\mathbf{v}\| \le c$, then the maximizing $\mathbf{v}$ is the normalized eigenvector of $\mathbf{A}$ with the largest eigenvalue normalized such that $\|\mathbf{v}\| \le c$. Note that it may not be possible to select this eigenvector since $\mathbf{v} = [\mathbf{x}^\top \ 1]^\top$. Other constraints on $\mathbf{x}$ will further restrict $\mathbf{v}$.

[1] This is also called the *Woodbury matrix identity* and can be stated as $(\mathbf{A}+\mathbf{U}\mathbf{C}\mathbf{V})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$ for compatible matrices.

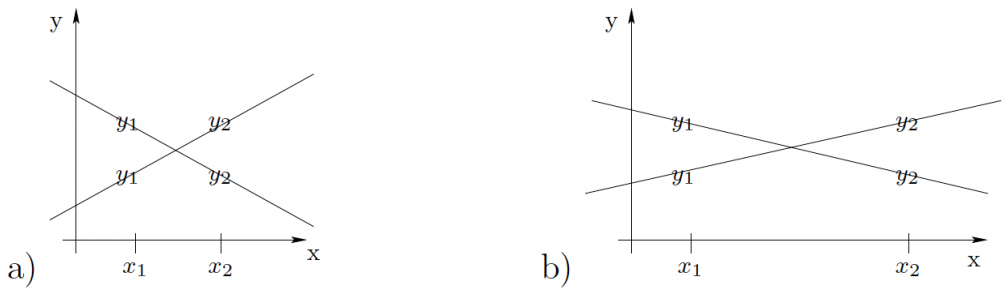


Figure 1: a) The effect of noise in the responses has a large effect on the parameters of the linear model when the corresponding inputs are close to each other; b) the effect is smaller when the inputs are further away.

1.2 Calculation of the Maximum Reduction of the MSE for Active Learning (Optional)

Recall that a positive definite symmetric matrix $\mathbf{A}$ has the property that all its eigenvalues $\lambda_i > 0$ (in particular, real) and the eigenvectors $\mathbf{u}_i$ are orthonormal. We let $\mathbf{A} = (\mathbf{X}^\top \mathbf{X})^{-1} \in \mathbb{R}^{(d+1)\times(d+1)}$ have eigenvalues $\lambda_1 > \lambda_2 > \ldots, \lambda_{d+1} > 0$ with corresponding eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{d+1}$ resp.

Lemma 1. *Let $\mathbf{A} \in \mathbb{R}^{(d+1)\times(d+1)}$ be a positive definite symmetric matrix. For any $\mathbf{v} \in \mathbb{R}^{d+1}$,*

$$f(\mathbf{v}) := \frac{\mathbf{v}^\top \mathbf{A}\mathbf{A}\mathbf{v}}{1 + \mathbf{v}^\top \mathbf{A}\mathbf{v}} \leq \lambda_1 \tag{12}$$

and the upper bound is achieved by taking $\mathbf{v} = t\mathbf{u}_1$ and letting $t \uparrow \infty$.

Proof. We first prove the upper bound in (12). Since $\{\mathbf{u}_i\}_{i=1}^{d+1}$ forms a basis, every vector $\mathbf{v}$ can be written as

$$\mathbf{v} = \sum_i \alpha_i \mathbf{u}_i \tag{13}$$

for some $\alpha_i, i = 1, \ldots, d+1$. We may thus compute

$$\mathbf{A}\mathbf{v} = \sum_i \alpha_i \mathbf{A}\mathbf{u}_i = \sum_i \alpha_i \lambda_i \mathbf{u}_i \tag{14}$$

$$\mathbf{v}^\top \mathbf{A}\mathbf{v} = \left(\sum_j \alpha_j \mathbf{u}_j\right)^\top \left(\sum_i \alpha_i \lambda_i \mathbf{u}_i\right)^\top = \sum_{i,j} \alpha_i \alpha_j \lambda_i \langle \mathbf{u}_i, \mathbf{u}_j \rangle = \sum_i \alpha_i^2 \lambda_i \tag{15}$$

$$\mathbf{v}^\top \mathbf{A}\mathbf{A}\mathbf{v} = \left(\sum_j \alpha_j \lambda_j \mathbf{u}_j\right)^\top \left(\sum_i \alpha_i \lambda_i \mathbf{u}_i\right)^\top = \sum_{i,j} \alpha_i \alpha_j \lambda_i \lambda_j \langle \mathbf{u}_i, \mathbf{u}_j \rangle = \sum_i \alpha_i^2 \lambda_i^2. \tag{16}$$

where the final equalities in (15) and (16) hold because $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 1$ if $i = j$ and 0 otherwise. Plugging this into the expression defining $f(\mathbf{v})$, we obtain

$$f(\mathbf{v}) \leq \frac{\mathbf{v}^\top \mathbf{A}\mathbf{A}\mathbf{v}}{\mathbf{v}^\top \mathbf{A}\mathbf{v}} = \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i} \leq \frac{\lambda_1 \sum_i \alpha_i^2 \lambda_i}{\sum_i \alpha_i^2 \lambda_i} = \lambda_1 \tag{17}$$

as desired.

Now, we take $\mathbf{v} = t\mathbf{u}_1$ for some $t > 0$. Then

$$f(t\mathbf{u}_1) = \frac{(t\mathbf{u}_1)^\top \mathbf{A}\mathbf{A}(t\mathbf{u}_1)}{1 + (t\mathbf{u}_1)^\top \mathbf{A}(t\mathbf{u}_1)} = \frac{t^2(\mathbf{A}\mathbf{u}_1)^\top(\mathbf{A}\mathbf{u}_1)}{1 + t^2 \mathbf{u}_1 \mathbf{A}\mathbf{u}_1} = \frac{t^2 \lambda_1^2 \|\mathbf{u}_1\|^2}{1 + t^2 \lambda_1 \|\mathbf{u}_1\|^2} \tag{18}$$

Now letting $t \uparrow \infty$, we see that $f(t\mathbf{u}_1) \uparrow \lambda_1$ as was to be shown. □

1.3 Example of Active Learning in Regression

As in the previous set of notes, we consider the one-dimensional linear regression example with offset so the model is

$$y = \theta^* x + \theta_0^* + \epsilon \tag{19}$$

where x is constrained to be in the interval $[-1, 1]$ and ϵ is Gaussian noise with some variance. Assume that we already have responses (y values) for the training examples $x_1 = 1$ and $x_2 = -1$. Which x should we choose next? We have the following matrices

$$\mathbf{X} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \quad \mathbf{X}^\top \mathbf{X} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{and} \quad \mathbf{A} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix}. \tag{20}$$

We parametrize $\mathbf{v}$ as $[x \;\; 1]^\top$ and therefore $\mathbf{v}^\top \mathbf{A} \mathbf{v} = (x^2 + 1)/2$ and $\mathbf{v}^\top \mathbf{A} \mathbf{A} \mathbf{v} = (x^2 + 1)/4$. The criterion to be maximized is

$$\frac{\mathbf{v}^\top \mathbf{A} \mathbf{A} \mathbf{v}}{1 + \mathbf{v}^\top \mathbf{A} \mathbf{v}} = \frac{(x^2 + 1)/4}{1 + (x^2 + 1)/2} \tag{21}$$

and this is clearly maximized at $|x| = 1$. Either choice would do but, after selecting one, the other one would be preferred at the next step. The result is consistent with the intuition that for linear models the inputs should be as far from each other as possible (cf. Fig. 1).

1.4 Another Criterion: Maximizing the Variance

We have so far used the mean squared error in the parameters as the selection criterion. What about the variance in the predictions? Let's try to find the point $\mathbf{x}$ whose response we are the most uncertain about.

Note that

$$\mathrm{Var}\left[y|\mathbf{x}, \mathbf{X}\right] = \mathbb{E}\left\{ \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \left(\begin{bmatrix} \hat{\boldsymbol{\theta}} \\ \hat{\theta}_0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} \right) \left(\begin{bmatrix} \hat{\boldsymbol{\theta}} \\ \hat{\theta}_0 \end{bmatrix} - \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} \right)^\top \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \,\middle|\, \mathbf{x}, \mathbf{X} \right] \tag{22}$$

Now, we know from Lecture 5 that

$$\begin{bmatrix} \hat{\boldsymbol{\theta}} \\ \hat{\theta}_0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}^* \\ \theta_0^* \end{bmatrix} + (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{e}. \tag{23}$$

Combining the above two equations yields

$$\begin{aligned}
\mathrm{Var}\left[y|\mathbf{x}, \mathbf{X}\right] &= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \mathbb{E}\left[(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{e} ((\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{e})^\top \,\middle|\, \mathbf{x}, \mathbf{X} \right] \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} && (24) \\
&= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \mathbb{E}\left[(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{e}\mathbf{e}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \,\middle|\, \mathbf{x}, \mathbf{X} \right] \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} && (25) \\
&= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbb{E}\left[\mathbf{e}\mathbf{e}^\top\right] \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} && (26) \\
&= \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \sigma^2 \mathbf{I} \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} && (27) \\
&= \sigma^2 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top (\mathbf{X}^\top \mathbf{X})^{-1} (\mathbf{X}^\top \mathbf{X}) (\mathbf{X}^\top \mathbf{X})^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} && (28) \\
&= \sigma^2 \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top (\mathbf{X}^\top \mathbf{X})^{-1} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}, && (29)
\end{aligned}$$

where the expectation is over responses for existing training examples, again assuming that there is a correct underlying linear model. So the largest variance corresponds to the input $\mathbf{x}$ that maximizes $\mathbf{v}^\top \mathbf{A} \mathbf{v}$ where

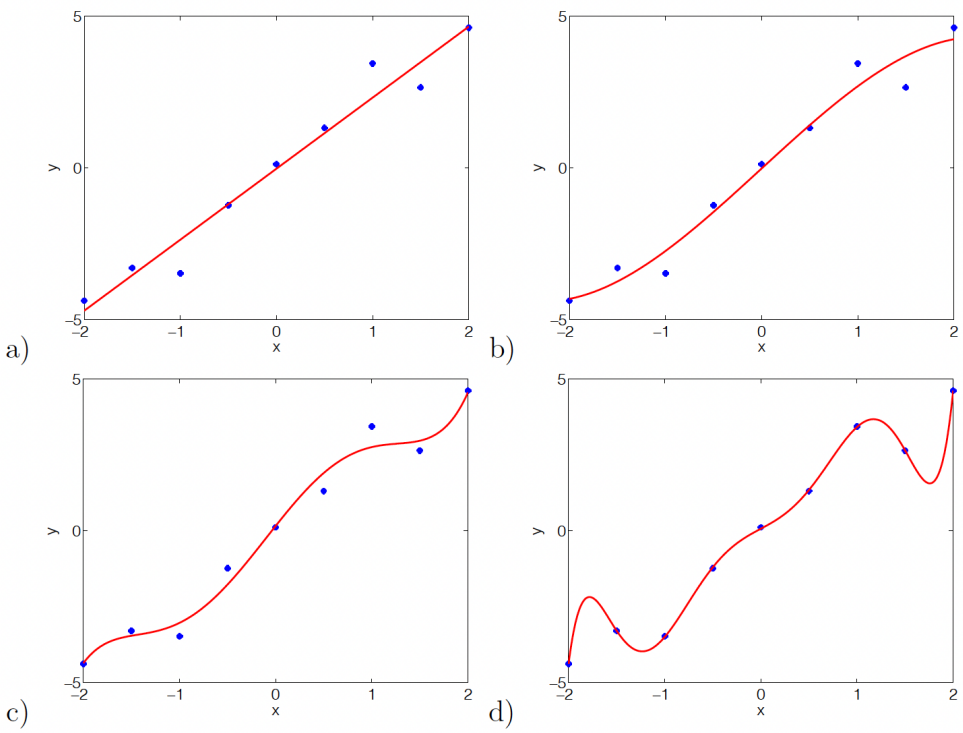


Figure 2: a) a linear model fit; b) a third order polynomial model fit to the same data; c) a fifth order polynomial model; d) a seventh order polynomial model.

$\mathbf{v} = [\mathbf{x}^\top \;\; 1]^\top$. In the unconstrained case where there are few or no restrictions on $\mathbf{v}$, this maximizing point is exactly the one we would query according the previous selection criterion based on the MSE. This is exactly the point that is aligned with the leading eigenvector of $\mathbf{A}$.

2 Introduction to Kernels

Thus far, we have restricted ourselves to linear models

$$y = \langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0 + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2), \tag{30}$$

or generalized linear models (in logistic regression)

$$P(y \mid \mathbf{x}, \boldsymbol{\theta}, \theta_0) = g\big(y(\langle \boldsymbol{\theta}, \mathbf{x} \rangle + \theta_0)\big). \tag{31}$$

where $g(z) = 1/(1 + \exp(-z))$ is the logistic function. These are reasonably rich models but cannot capture nonlinearities in the data. We would like to model non-linearities in the original feature space. This is achieved by mapping the input examples to a higher dimensional feature space where the dimensions in the new feature vectors include non-linear functions of the inputs. The simplest setting for demonstrating this is linear regression in one dimension. Consider therefore the one-dimensional linear model

$$y = \theta x + \theta_0 + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{32}$$

We can obtain a quadratic model by simply mapping the input x to a longer feature vector that includes a term quadratic in x, i.e., x^2. A third order model can be constructed by including all terms up to degree

three, and so on. Then we will be able to get the curves in Fig. 2. Explicitly, we would make linear predictions using feature vectors

$$x \xrightarrow{\phi} [1, \sqrt{2}x, x^2]^\top = \boldsymbol{\phi}(x) \tag{33}$$

$$x \xrightarrow{\phi} [1, \sqrt{3}x, \sqrt{3}x^2, x^3]^\top = \boldsymbol{\phi}(x) \tag{34}$$

$$x \xrightarrow{\phi} [1, \sqrt{4}x, \sqrt{6}x^2, \sqrt{4}x^3, x^4]^\top = \boldsymbol{\phi}(x) \tag{35}$$

$$\vdots$$

The role of $\sqrt{2}$ and other constants will become clear shortly. The new polynomial regression model is then given by

$$y = \boldsymbol{\theta}^\top \boldsymbol{\phi}(x) + \epsilon. \tag{36}$$

Note that if we instantiate $\boldsymbol{\phi}$ to be the function in (33), $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2]^\top$ is three-dimensional, and this can be expanded as

$$y = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}^\top \begin{bmatrix} 1 \\ \sqrt{2}x \\ x^2 \end{bmatrix} + \epsilon = \theta_0 + \theta_1\sqrt{2}x + \theta_2 x^2 + \epsilon. \tag{37}$$

One interesting feature is that equipped with the polynomial mapping $\boldsymbol{\phi}$, we no longer need the offset θ_0 in the affine model in (32). This is because each *expanded feature vector* $\boldsymbol{\phi}$ already includes the entry with a 1; this allows us to automatically incorporate the offset term θ_0 into the model.

Of course, we may choose higher and higher order feature vectors $\boldsymbol{\phi}$ that have more monomials like cx^n. For a 1-dimensional dataset with $n+1$ points, if we choose a polynomial of degree n containing monomials of the form cx^n, we are guaranteed to achieve 0 training error.

Exercise 1. *Prove the above statement. Namely, if we have n points $\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}$ and any n targets $\{y_1, y_2, \ldots, y_n\} \subset \mathbb{R}$, then we can achieve 0 training loss in terms of the mean-squared error if we use a polynomial of order $n-1$. When we we use a polynomial of a strictly smaller order and still achieve 0 training error? You may have to use some properties of Vandermonde matrices.*

But is this desirable? If we look at the bottom right plot of Fig. 2, we will see that we have *severely overfit* the data and are not likely to generalize well to unseen test data. Thus, kernelization, i.e., the map from x to $\boldsymbol{\phi}(x)$, is almost always used in conjunction with regularization.

What if our input data has dimension 2, i.e., $\mathbf{x} = (x_1, x_2)$. Then a map to quadratic features is

$$\mathbf{x} \xrightarrow{\phi} [1, x_1, x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2]^\top = \boldsymbol{\phi}(\mathbf{x}). \tag{38}$$

One limitation of explicating the feature vectors is that the dimensionality can increase rapidly with the degree of polynomial expansion, especially when the dimension of the input vector is already high. The table below gives some indicating of this though the effect is more dramatic with higher input dimensions.

$\dim(\mathbf{x})$	2				3			
degree p	2	3	4	5	2	3	4	5
number of features	6	10	15	21	10	20	35	56

Can we somehow avoid explicating the dimensions of the feature vectors? In the context of models we have discussed thus far, yes, we can. We can define the feature vectors implicitly by focusing on specifying the values of their inner products or kernel instead. To get a sense of the power of this approach, let's evaluate the *inner product* between two feature vectors corresponding to the specific cubic (third-order) expansions

of 1-dimensional inputs shown before:

$$\boldsymbol{\phi}(x) = [1, \sqrt{3}x, \sqrt{3}x^2, x^3]^\top \tag{39}$$

$$\boldsymbol{\phi}(x') = [1, \sqrt{3}x', \sqrt{3}(x')^2, (x')^3]^\top \tag{40}$$

$$\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x') \rangle = 1 + 3xx' + 3(xx')^2 + (xx')^3 = (1 + xx')^3 \tag{41}$$

So it seems we can compactly evaluate the inner products between polynomial feature vectors by evaluating their inner product in the original feature space and applying a simple function. In this case the inner product in the original feature space is xx' and the simple function is $z \mapsto (1+z)^3$. The effect is more striking with higher dimensional inputs and higher polynomial degrees. (We did have to specify the constants appropriately in the feature vectors to make this work). To shift the modeling from explicit feature vectors to inner products (kernels) we obviously have to first turn the estimation problem into a form that involves only inner products between feature vectors. This is what we do in the next lecture.

Exercise 2. *Prove that in the general case if* $\dim(\mathbf{x}) = d$ *and the polynomial order is* p*, then the number of features is* $\binom{d+p}{d}$*. Note that the inner product in the general case looks like* $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^p$*. [Hint: Google "number of monomials of a given degree".]*

MA4270 : Kernels

Vincent Y. F. Tan

September 9, 2021

These are the notes for Lecture 7 on kernels. You should know Sections 1 and 2. The rest are optional reading.

1 Linear Regression and the Representer Theorem

Consider the kernelized linear model

$$y = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}) \rangle + \epsilon. \tag{1}$$

As we mentioned last time, with the kernel map, there is no need to include an offset because typically there is a constant component in $\boldsymbol{\phi}(\mathbf{x})$. For example, when x is a scalar and $\boldsymbol{\phi}$ represents a map to cubic features, it maps x to $\boldsymbol{\phi}(x) = [1, \sqrt{3}x, \sqrt{3}x^2, x^3]^\top$, so the constant component is represented by the 1 in $\boldsymbol{\phi}(x)$.

We have emphasized that regularization is necessary in conjunction with mapping examples to higher-dimensional feature vectors. The regularized least squares problem is thus

$$J(\boldsymbol{\theta}) = \sum_{t=1}^{n} (y_t - \langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t) \rangle)^2 + \lambda \|\boldsymbol{\theta}\|^2. \tag{2}$$

This form can be derived from penalized log-likelihood estimation assuming that ϵ_t are independent Gaussian noises.

Exercise 1. *Make the sentence above precise and prove your claim.*

The effect of the regularization penalty is to pull all the parameters towards zero. So any linear dimensions in the parameters that the training feature vectors do not pertain to are set explicitly to zero. We would therefore expect the optimal parameters to lie in the span of the feature vectors corresponding to the training examples. This is indeed the case.

To see this, differentiate $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ and set the derivative to zero we can find the optimal parameters $\hat{\boldsymbol{\theta}}_\lambda$. We find that $\hat{\boldsymbol{\theta}}_\lambda$ satisfies

$$\nabla J(\boldsymbol{\theta})\Big|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_\lambda} = -2 \left(\sum_{t=1}^{n} (y_t - \langle \hat{\boldsymbol{\theta}}_\lambda, \boldsymbol{\phi}(\mathbf{x}_t) \rangle) \boldsymbol{\phi}(\mathbf{x}_t) \right) + 2\lambda \hat{\boldsymbol{\theta}}_\lambda = \mathbf{0}. \tag{3}$$

Now we can rename $\alpha_t := y_t - \langle \hat{\boldsymbol{\theta}}_\lambda, \boldsymbol{\phi}(\mathbf{x}_t) \rangle$ and see that

$$\hat{\boldsymbol{\theta}}_\lambda = \frac{1}{\lambda} \sum_{t=1}^{n} \alpha_t \boldsymbol{\phi}(\mathbf{x}_t). \tag{4}$$

While this is simple, the result in (4) is surprisingly deep and is a special instance of the *Representer Theorem* in statistical learning theory; see the seminal works by Kimeldorf and Wahba [KW70, Lemma 2.2] and Schölkopf, Herbrich and Smola [SHS01]. It says that no matter how high dimensional the kernelized data samples $\boldsymbol{\phi}(\mathbf{x}_t)$ are—they may even be infinite dimensional—we do not need to solve the high-dimensional

optimization problem in (2) to find the optimal $\boldsymbol{\theta}$. All that is required to solve for $\hat{\boldsymbol{\theta}}_\lambda$ within a ridge regression framework is the set of n coefficients $\{\alpha_t\}_{t=1}^n$! So we have reduced a d-dimensional optimization problem to one that is "only" n dimensional and n may be much smaller than d. Here, we note the importance of λ being positive. If $\lambda = 0$ (no regularization), this will no longer be true because (4) would not hold.

But how do we set α_t? The values for α_t can be found by insisting that they indeed can be interpreted as prediction differences:

$$\alpha_t = y_t - \langle \hat{\boldsymbol{\theta}}_\lambda, \boldsymbol{\phi}(\mathbf{x}_t) \rangle = y_t - \frac{1}{\lambda} \left\langle \sum_{t'=1}^{n} \alpha_{t'} \boldsymbol{\phi}(\mathbf{x}_{t'}), \boldsymbol{\phi}(\mathbf{x}_t) \right\rangle. \tag{5}$$

Now notice something magical, namely that α_t depends only on the actual responses y_t and the inner products between the training examples, the *Gram matrix*

$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \dots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \dots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \dots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix} \tag{6}$$

where $K(\mathbf{x}_t, \mathbf{x}_{t'}) = \langle \boldsymbol{\phi}(\mathbf{x}_t), \boldsymbol{\phi}(\mathbf{x}_{t'}) \rangle$ is the kernel function which only involves inner products of the kernelized feature vectors $\boldsymbol{\phi}(\mathbf{x}_t)$. It is more convenient to write (5) in vector form by defining

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}. \tag{7}$$

Then

$$\boldsymbol{\alpha} = \mathbf{y} - \frac{1}{\lambda} \mathbf{K} \boldsymbol{\alpha} \tag{8}$$

and the solution is

$$\hat{\boldsymbol{\alpha}} = \lambda (\lambda \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}. \tag{9}$$

Note that finding the estimates $\hat{\alpha}_t$ requires inverting a $n \times n$ matrix. This is the cost of dealing with inner products as opposed to handing feature vectors directly. In some cases, the benefit is substantial since the feature vectors in the inner products may be infinite dimensional but never needed explicitly.

As a result of finding $\hat{\alpha}_t$ we can cast the predictions for new examples $\mathbf{x}$ also in terms of inner products:

$$\hat{y} = \hat{\boldsymbol{\theta}}_\lambda^\top \boldsymbol{\phi}(\mathbf{x}) = \left\langle \frac{1}{\lambda} \sum_{t'=1}^{n} \alpha_{t'} \boldsymbol{\phi}(\mathbf{x}_{t'}), \boldsymbol{\phi}(\mathbf{x}) \right\rangle = \frac{1}{\lambda} \sum_{t'=1}^{n} \alpha_{t'} K(\boldsymbol{\phi}(\mathbf{x}_{t'}), \boldsymbol{\phi}(\mathbf{x}_t)). \tag{10}$$

Note that this computation can also be done in terms of the kernel matrix.

2 Kernels

So we have now successfully turned a regularized linear regression problem into a kernel form. This means that we can simply substitute different kernel functions $K(\mathbf{x}, \mathbf{x}')$ into the estimation/prediction equations. This gives us an easy access to a wide range of possible regression functions.

The most standard example of a kernel is the polynomial kernel of degree or order $p \in \mathbb{N}$. This is written as

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^p. \tag{11}$$

We have seen that this kernel is induced by feature maps such as $\boldsymbol{\phi}(x) = [1, \sqrt{2}x, x^2]^\top$ for $(p, d) = (2, 1)$ and $\boldsymbol{\phi}(x) = [1, \sqrt{3}x, \sqrt{3}x^2, x^3]^\top$ for $(p, d) = (3, 1)$ etc. To be formal, let's make the following definition.

Definition 1. *A (valid) kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a function such that there exists a feature map $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that $K(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle$ for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$.*

While we have written the original feature space as $\mathbb{R}^d$ and the kernelized feature space as $\mathbb{R}^{d'}$ where $d' > d$, neither of these spaces has to be finite-dimensional. In fact, a case in point is the *radial basis kernel* in which the feature maps sends vectors to functions (and no longer finite-dimensional vectors).

Proposition 1 (Algebra of Kernels)**.** *Let K_1 and K_2 be valid kernel functions. Let $f : \mathbb{R}^d \to \mathbb{R}$ be an arbitrary function. Then the following are also valid kernel functions.*

1. $K(\mathbf{x}, \mathbf{x}') = f(\mathbf{x}) K_1(\mathbf{x}, \mathbf{x}') f(\mathbf{x}')$;
2. $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}')$;
3. $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}')$;

Proof. Let $\boldsymbol{\phi}^{(i)}$ be the feature map associated to kernel K_i for $i = 1, 2$.

We first prove part 1. Consider the feature map $\boldsymbol{\phi}(\mathbf{x}) = f(\mathbf{x}) \boldsymbol{\phi}^{(1)}(\mathbf{x})$. We claim that this is the feature map that gives K in part 1. Consider,

$$\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle = \langle f(\mathbf{x}) \boldsymbol{\phi}^{(1)}(\mathbf{x}), f(\mathbf{x}') \boldsymbol{\phi}^{(1)}(\mathbf{x}') \rangle = f(\mathbf{x}) \langle \boldsymbol{\phi}^{(1)}(\mathbf{x}), \boldsymbol{\phi}^{(1)}(\mathbf{x}') \rangle f(\mathbf{x}') = f(\mathbf{x}) K(\mathbf{x}, \mathbf{x}') f(\mathbf{x}'). \tag{12}$$

Thus $K(\mathbf{x}, \mathbf{x}')$ can be written as $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle$ for some $\boldsymbol{\phi}$, showing that it is a valid kernel.

Next we prove part 2. Consider the feature map constructed by concatenating the kernelized feature vectors

$$\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \boldsymbol{\phi}^{(1)}(\mathbf{x}) \\ \boldsymbol{\phi}^{(2)}(\mathbf{x}) \end{bmatrix}. \tag{13}$$

We claim this does the job. Consider,

$$\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle = \left\langle \begin{bmatrix} \boldsymbol{\phi}^{(1)}(\mathbf{x}) \\ \boldsymbol{\phi}^{(2)}(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} \boldsymbol{\phi}^{(1)}(\mathbf{x}') \\ \boldsymbol{\phi}^{(2)}(\mathbf{x}') \end{bmatrix} \right\rangle \tag{14}$$

$$= \langle \boldsymbol{\phi}^{(1)}(\mathbf{x}), \boldsymbol{\phi}^{(1)}(\mathbf{x}') \rangle + \langle \boldsymbol{\phi}^{(2)}(\mathbf{x}), \boldsymbol{\phi}^{(2)}(\mathbf{x}') \rangle = K_1(\mathbf{x}, \mathbf{x}') + K_2(\mathbf{x}, \mathbf{x}'). \tag{15}$$

Thus $K(\mathbf{x}, \mathbf{x}')$ can be written as $\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle$ for some $\boldsymbol{\phi}$ (which is presented in (13)), showing that it is a valid kernel.

The third and the last rule is a little more complicated but not much. Suppose we use a double index i, j to index the components of $\boldsymbol{\phi}(\mathbf{x})$ where i ranges over the components of $\boldsymbol{\phi}^{(1)}(\mathbf{x})$ and j ranges over the components of $\boldsymbol{\phi}^{(2)}(\mathbf{x})$. Then,

$$\boldsymbol{\phi}_{i,j}(\mathbf{x}) = \boldsymbol{\phi}_i^{(1)}(\mathbf{x}) \boldsymbol{\phi}_j^{(2)}(\mathbf{x}). \tag{16}$$

We claim that this does the job. Consider,

$$\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \rangle = \sum_{i,j} \boldsymbol{\phi}_{i,j}(\mathbf{x}) \boldsymbol{\phi}_{i,j}(\mathbf{x}') \tag{17}$$

$$= \sum_{i,j} \boldsymbol{\phi}_i^{(1)}(\mathbf{x}) \boldsymbol{\phi}_j^{(2)}(\mathbf{x}) \boldsymbol{\phi}_i^{(1)}(\mathbf{x}') \boldsymbol{\phi}_j^{(2)}(\mathbf{x}') \tag{18}$$

$$= \left(\sum_i \boldsymbol{\phi}_i^{(1)}(\mathbf{x}) \boldsymbol{\phi}_i^{(1)}(\mathbf{x}') \right) \left(\sum_j \boldsymbol{\phi}_j^{(2)}(\mathbf{x}) \boldsymbol{\phi}_j^{(2)}(\mathbf{x}') \right) \tag{19}$$

$$= K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}'). \tag{20}$$

This $K(\mathbf{x}, \mathbf{x}') = K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}')$ can be expressed as the inner product of $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\phi}(\mathbf{x}')$, showing that it is a valid kernel. □

Now consider the rather esoteric "kernel", known as the *radial basis kernel*

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right). \tag{21}$$

We claim that this is valid kernel. Consider the functions $\phi_{\mathbf{z}}(\mathbf{x}) = c(\beta, d)\mathcal{N}(\mathbf{z}; \mathbf{x}, 1/(2\beta))$ for some $c(\beta, d)$ to be chosen. Note that $\phi_{\mathbf{z}}(\mathbf{x})$ corresponds to the $\mathbf{z}^{\text{th}}$ component of the function $\boldsymbol{\phi}(\mathbf{x})$. We will only prove the case $d = 1$. Roughly speaking, for this case,

$$K(\mathbf{x}, \mathbf{x}') = \lim_{\Delta z \to 0} \int_{\mathbb{R}} \Pr(X \in [z \pm \Delta z], X' \in [z \pm \Delta z])\, \mathrm{d}z \tag{22}$$

where $X \sim \mathcal{N}(x, 1/(2\beta))$ and $X' \sim \mathcal{N}(x', 1/(2\beta))$. If $x \approx x'$, then $K(x, x')$ will be high and vice versa.

To be precise, consider the inner product

$$\begin{aligned}
\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x') \rangle &= \int_{\mathbb{R}} \phi_z(x)\phi_z(x')\, \mathrm{d}z && (23)\\
&= \int_{\mathbb{R}} c(\beta, 1)^2 \mathcal{N}(z; x, 1/(2\beta))\mathcal{N}(z; x', 1/(2\beta))\, \mathrm{d}z && (24)\\
&\propto \int_{\mathbb{R}} \exp\left(-\beta(z - x)^2\right) \exp\left(-\beta(z - x')^2\right)\, \mathrm{d}z && (25)\\
&\propto \int_{\mathbb{R}} \exp\left(-\beta(2z^2 - 2z(x + x') + x^2 + (x')^2)\right)\, \mathrm{d}z && (26)\\
&= \exp(-\beta(x^2 + (x')^2)) \int_{\mathbb{R}} \exp\left(-\beta(2z^2 - 2z(x + x'))\right)\, \mathrm{d}z && (27)\\
&\propto \exp(-\beta(x^2 + (x')^2)) \exp\left((2\beta(x + x'))^2/(8\beta)\right) && (28)\\
&= \exp\left(-\frac{\beta}{2}(x - x')^2\right) \propto K(x, x'). && (29)
\end{aligned}$$

Thus the proposed feature maps (which maps a scalar in $\mathbb{R}$ to a function $c(\beta, 1)\mathcal{N}(\cdot; x, 1/(2\beta))$) indeed result in the radial basis kernel. The feature maps $x \mapsto c(\beta, 1)\mathcal{N}(\cdot; x, 1/(2\beta))$ are known as *radial basis functions* or RBFs.

Exercise 2. *Use Proposition 1 to provide an alternative proof that the radial basis kernel is indeed a kernel.*

Exercise 3. *Given any unlabelled dataset $\mathcal{D} = \{\mathbf{x}_t\}_{t=1}^n$, prove that the $n \times n$ matrix in (6) is positive semi-definite (PSD). Recall that a PSD matrix $\mathbf{M}$ is one in which $\mathbf{z}^\top \mathbf{M}\mathbf{z} \geq 0$ for al $\mathbf{z}$.*

3 String Kernels (Optional)

This is based on the seminal paper "Text Classification using String Kernels" by Lodhi, Saunders, Shawe-Taylor, Cristianini, and Watkins [LSST+12].

We describe a kernel between two text documents. The idea is to compare them by means of the substrings they contain: the more substrings in common, the more similar they are. An important part is that such substrings do not need to be contiguous, and the degree of contiguity of one such substring in a document determines how much weight it will have in the comparison.

3.1 An Example

For example: the substring **car** is present both in the word "card" and in the word "custard", but with different weighting. For each such substring there is a dimension of the feature space, and the value of such coordinate depends on how frequently and how compactly such string is embedded in the text. In order to

deal with non-contiguous substrings, it is necessary to introduce a decay factor $\lambda \in (0,1)$ that can be used to weight the presence of a certain feature in a text.

Consider - as simple documents - the words **cat**, **car**, **bat**, **bar**. If we consider only $k = 2$, we obtain an 8-dimensional feature space, where the words are mapped as in Table 1.

	ca	**ct**	**at**	**ba**	**bt**	**cr**	**ar**	**br**
ϕ(**cat**)	λ^2	λ^3	λ^2	0	0	0	0	0
ϕ(**car**)	λ^2	0	0	0	0	λ^3	λ^2	0
ϕ(**bat**)	0	0	λ^2	λ^2	λ^3	0	0	0
ϕ(**bar**)	0	0	0	λ^2	0	0	λ^2	λ^3

Table 1: Feature maps of the 8 features for 4 words

Hence, the unnormalized kernel between **car** and **cat** is $K(\mathbf{car}, \mathbf{cat}) = \lambda^4$, whereas the normalized version is obtained as follows: $K(\mathbf{car}, \mathbf{car}) = K(\mathbf{cat}, \mathbf{cat}) = 2\lambda^4 + \lambda^6$ and hence $\hat{K}(\mathbf{car}, \mathbf{cat}) = \lambda^4/(2\lambda^4 + \lambda^6) = 1/(2 + \lambda^2)$ (see Section 3.2). Note that in general the document will contain more than one word, but the mapping for the whole document is into one feature space: the concatenation of all the words and the spaces (ignoring the punctuation) is considered as a unique sequence.

3.2 Kernel Normalization

Once we have create such a kernel it is natural to normalise to remove any bias introduced by document length. We can produce this effect by normalising the feature vectors in the feature space. Hence, we create a new embedding $\hat{\phi}(s) = \phi(s)/\|\phi(s)\|$, which gives rise to the kernel

$$\hat{K}(s,t) = \langle \hat{\phi}(s), \hat{\phi}(t) \rangle = \left\langle \frac{\phi(s)}{\|\phi(s)\|}, \frac{\phi(t)}{\|\phi(t)\|} \right\rangle = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}. \tag{30}$$

3.3 String Subsequence Kernel (SSK)

We now define the notion of the String Subsequence Kernel (SSK). Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s, t, we denote by $|s|$ the length of the string $s = s_1 \ldots, s_{|s|}$, and by st the string obtained by concatenating the strings s and t. The string $s[i:j]$ is the substring $s_i \ldots s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_1, \ldots, i_{|u|})$, with $1 \le i_1 \le \ldots \le i_{|u|} \le |s|$, such that $u_j = s_{i_j}$, for $j = 1, \ldots, |u|$, or $u = s[\mathbf{i}]$ for short. The length $l(\mathbf{i})$ of the subsequence in s is $i_{|u|} - i_1 + 1$. We denote by Σ^n the set of all finite strings of length n, and by Σ^* the set of all strings

$$\Sigma^* = \bigcup_{n \ge 0} \Sigma^n. \tag{31}$$

As an example, let s be the string **segmentation** and let u be the subsequence **set**. Then $\mathbf{i} = (1, 2, 7)$ as the characters **s**, **e**, and **t** occur in locations 1, 2, and 7 in **segmentation**.

We now define feature spaces $F_n = \mathbb{R}^{|\Sigma^n|}$. The feature mapping ϕ for a string s is defining the u coordinate $\phi_u(s)$ for each $u \in \Sigma^n$. We define

$$\phi_u(s) = \sum_{\mathbf{i}: u = s[\mathbf{i}]} \lambda^{l(\mathbf{i})}. \tag{32}$$

for some $\lambda \le 1$. These features measure the number of occurrences of subsequences in the string s weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t give a sum over all common subsequences weighted according to their frequency of occurrence and lengths

$$K_n(s,t) = \sum_{u \in \Sigma^n} \langle \phi_u(s), \phi_u(t) \rangle = \sum_{u \in \Sigma^n} \sum_{\mathbf{i}: u = s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}: u = t[\mathbf{j}]} \lambda^{l(\mathbf{j})} = \sum_{u \in \Sigma^n} \sum_{\mathbf{i}: u = s[\mathbf{i}]} \sum_{\mathbf{j}: u = t[\mathbf{j}]} \lambda^{l(\mathbf{i}) + l(\mathbf{j})}. \tag{33}$$

A direct computation of these features would involve $O(|\Sigma|^n)$ time and space, since this is the number of features involved. It is also clear that most of the features will have non zero components for large documents. In order to derive an effective procedure for computing such kernel, we introduce an additional function which will aid in defining a recursive computation for this kernel. Let

$$K'_h(s,t) := \sum_{u\in\Sigma^h} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{|s|+|t|-i_1-j_1+2}, \qquad h=1,\ldots,n-1 \tag{34}$$

that is counting subsequence u of length h from the beginning of the particular sequence through to the end of the strings s and t instead of just lengths of the indices $l(\mathbf{i})$ and $l(\mathbf{j})$. This is called the *intermediate kernel*. We can now define a recursive computation for K'_h and hence compute K_n extremely efficiently.

3.4 Recursive Computation of SSK

$$K'_0(s,t) = 1, \qquad \forall s,t \tag{35}$$

$$K'_h(s,t) = 0, \qquad \text{if} \quad \min\{|s|,|t|\} < h \tag{36}$$

$$K_h(s,t) = 0, \qquad \text{if} \quad \min\{|s|,|t|\} < h \tag{37}$$

$$K'_h(sx,t) = \lambda K'_h(s,t) + \sum_{j:t_j=x} K'_{h-1}(s,t[1:j-1])\lambda^{|t|-j+2},$$

$$\forall\, h = 1,\ldots,n-1 \tag{38}$$

$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2. \tag{39}$$

In (35), we set the intermediate kernel for the null string to be 1. In (36) and (37), the target substring is shorter than the search substring. Notice that we need the auxiliary function K'_h since it is only the interior gaps in the subsequences that are penalised. The correctness of the recursion in (38) follows from observing how the length of the strings has increased, incurring a factor of λ for each extra length unit. Hence, in the formula for $K'_h(sx,t)$ in (38), the first term has one fewer character, so requiring a single λ factor, while the second has $|t|-j+2$ fewer characters. For the last formula in (39) the second term requires the addition of just two characters, one to s and one to $t[1:j-1]$, since x is the last character of the n-sequence. If we wished to compute $K_n(s,t)$ for a range of values of n, we would simply perform the computation of $K'_h(s,t)$ up to one less than the largest n required, and then apply the last recursion for each $K_n(s,t)$ that is needed using the stored values of $K'_h(s,t)$. We can of course create a kernel $K(s,t)$ that combines the different $K_n(s,t)$ giving different (positive) weightings for each n.

3.5 Improving the Computational Efficiency of SSK via Dynamic Programming

SSK measures the similarity between documents s and t in a time proportional to $O(n|s||t|^2)$ where n is the length of the sequence. It is evident from the description of the recursion in Section 3.4, as the outermost recursion is over the sequence length and for each length and each additional character in s and t a sum over the sequence t must be evaluated. However it is possible to to speed up the computation of SSK via dynamic programming, or divide-and-conquer. We now present an efficient recursive computation of SSK that reduce the complexity of the computation to $O(n|s||t|)$, by first evaluating

$$K''_h(sx,t) = \sum_{j:t_j=x} K'_{h-1}(s,t[1:j-1])\lambda^{|t|-j+2} \tag{40}$$

and observing that we can then evaluate $K'_h(s,t)$ with the $O(|s||t|)$ recursion,

$$K'_h(sx,t) = \lambda K'_h(s,t) + K''_h(sx,t). \tag{41}$$

Note that (41) is identical to (39). Now observe that $K''_h(sx, tu) = \lambda^{|u|} K''_h(sx, t)$ provided x does not occur in u, while

$$K''_h(sx, tx) = \lambda \left(K''_h(sx, t) + \lambda K'_{h-1}(s, t) \right). \tag{42}$$

These observations together give an $O(|s||t|)$ recursion for computing $K''_h(s, t)$. Hence, we can evaluate the overall kernel in $O(n|s||t|)$ time. This is much better than the naïve implementation in (33) which requires $O(|\Sigma|^n)$ time and space.

4 Diffusion Kernels (Optional)

Let $G = (S, E)$ be a graph. The vertices are the data points. Let B be a symmetric base similarity matrix of size $|S| \times |S|$ whose entries are the weights of the edges of the graph G. For example, let us consider a biological application. S is a set of proteins, and B is a matrix of 1's and 0's which represent protein-protein interaction. Each location in B with a 1 indicates that the corresponding proteins interact, while a 0 stands for no interaction.

In general, B is not positive semi-definite. Therefore, it cannot be used directly as a kernel. Diffusion kernels convert the similarity rule into a kernel. Please see [KL02] for more details of the construction mentioned below.

Consider $B^2 = BB^T$. If the graph G is unweighted then the (i, j)-th entry of B^2 is the number of common friends between the the i-th and j-th data points (or the number of paths of length 2 between i and j) and it can be thought of as a measure of their similarity. Clearly B^2 is positive semi-definite. Higher powers of B measure higher order similarities. In general, only the even powers are guaranteed to be positive semi-definite. It is natural to consider a weighted sum of the powers of B in which the higher orders are given lower weights. Let us consider

$$\exp(\lambda B) = \sum_{k=0}^{\infty} \frac{1}{k!} \lambda^k B^k \tag{43}$$

for $\lambda < 1$. If $B = U \Lambda U^T$ is the spectral decompsotion of B, then we know that

$$B^k = U \Lambda^k U^T \tag{44}$$

therefore

$$\exp(\lambda B) = U \exp(\lambda \Lambda) U^T \tag{45}$$

Since $\exp(\lambda \Lambda)$ is a kernel, we get that $\exp(\lambda B)$ is a kernel as well.

This is an example of a diffusion kernel. The term diffusion derives from the connection to random walks and the heat equation in physics.

References

[KL02] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *International Conference on Machine Learning (ICML)*, 2002.

[KW70] G. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation of stochastic processes and smoothing by splines. *Ann. Math. Statist.*, 41:495–502, 1970.

[LSST+12] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, pages 419–444, 20012.

[SHS01] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, pages 416–426. Lecture Notes in Computer Science, 2001.

MA4270 : SVM Duality and the XOR Problem

September 9, 2021

These are notes on SVM duality (with offsets and with slack). You need to know everything here. To understand this, you need to know the KKT conditions which is in another document ConvexOpt.pdf.

1 SVM Duality

In the MIT lecture notes (Lecture 8), the derivation of the dual for the SVM problem was done in detail for the case without slack variables, i.e., $\xi_t = 0$ for all t. Here, I provide the detailed derivations for the general case with offset, with slack, and in kernel form. Let the kernel $K(\mathbf{x}_t, \mathbf{x}_s)$ be induced by the feature mapping $\boldsymbol{\phi}(\mathbf{x}_t)$, i.e., $K(\mathbf{x}_t, \mathbf{x}_s) = \langle \boldsymbol{\phi}(\mathbf{x}_t), \boldsymbol{\phi}(\mathbf{x}_s) \rangle$ where the inner product is defined in an appropriate reproducing-kernel Hilbert space (RKHS). The primal problem is

$$\min_{\boldsymbol{\theta}, \theta_0, \xi} \frac{1}{2}\|\theta\|^2 + C\sum_{t=1}^{n} \xi_t, \quad \text{s.t.} \quad y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t) \rangle + \theta_0) \geq 1 - \xi_t, \quad \xi_t \geq 0, \quad \forall\, t. \tag{1}$$

The Lagrangian is

$$J(\boldsymbol{\theta}, \theta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\lambda}) = \frac{1}{2}\|\boldsymbol{\theta}\|^2 + C\sum_{t=1}^{n} \xi_t + \sum_{t=1}^{n} \alpha_t \big(1 - \xi_t - y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t) \rangle + \theta_0)\big) + \sum_{t=1}^{n} \lambda_t(-\xi_t). \tag{2}$$

Now, we derive the 4 KKT conditions to assert that $(\boldsymbol{\theta}, \theta_0, \xi)$ and $(\boldsymbol{\alpha}, \boldsymbol{\lambda})$ is *primal-dual optimal*, i.e., they achieve

$$\min_{\boldsymbol{\theta}, \theta_0, \boldsymbol{\xi}} \max_{\boldsymbol{\alpha}, \boldsymbol{\lambda}} J(\boldsymbol{\theta}, \theta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\lambda}) = \max_{\boldsymbol{\alpha}, \boldsymbol{\lambda}} \min_{\boldsymbol{\theta}, \theta_0, \boldsymbol{\xi}} J(\boldsymbol{\theta}, \theta_0, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\lambda}). \tag{3}$$

The equality of the two sides (swapping of min and max) is justified by appealing to Slater's condition which ensures strong duality. In fact, one can also invoke the so-called Sion's minimax theorem. You can read up on this. Differentiating J with respect to the primal variables and setting to zero, we obtain the *stationarity conditions*:

$$\frac{\partial J}{\partial \boldsymbol{\theta}} = 0 \qquad \Rightarrow \qquad \sum_{t=1}^{n} \alpha_t y_t \boldsymbol{\phi}(\mathbf{x}_t) = \boldsymbol{\theta}, \tag{4}$$

$$\frac{\partial J}{\partial \theta_0} = 0 \qquad \Rightarrow \qquad \sum_{t=1}^{n} \alpha_t y_t = 0, \tag{5}$$

$$\frac{\partial J}{\partial \xi_t} = 0 \qquad \Rightarrow \qquad C - \alpha_t - \lambda_t = 0. \tag{6}$$

The *primal feasibility conditions* are

$$y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t) \rangle + \theta_0) \geq 1 - \xi_t, \quad \forall\, t \tag{7}$$

$$\xi_t \geq 0, \quad \forall\, t. \tag{8}$$

The *dual feasibility conditions* are

$$\alpha_t \geq 0, \quad \forall t \tag{9}$$
$$\lambda_t \geq 0, \quad \forall t. \tag{10}$$

The *complementary slackness conditions* are

$$\alpha_t\big(1 - \xi_t - y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0)\big) = 0, \quad \forall t \tag{11}$$
$$\lambda_t \xi_t = 0, \quad \forall t. \tag{12}$$

To obtain these, we simply multiply the terms in the primal and dual optimality conditions and set the product to zero. Combining (6), (9) and (10), we obtain that

$$\alpha_t \in [0, C]. \tag{13}$$

Now, we partition the training points $\{(\mathbf{x}_t, y_t)\}_{t=1}^n$ into three disjoint subsets.

1. Non-margin support vectors (SVs), i.e., those t such that $\alpha_t = C > 0$.

   For these, we immediately have from (11) that $1 - \xi_t = y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0)$. Also, from (6), we have that $\lambda_t = 0$ (hence, we can have $\xi_t > 0$).

2. Margin SVs, i.e., those t such that $\alpha_t \in (0, C)$.

   For these, we again have from (11) that $1 - \xi_t = y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0)$. This time, from (6), we have that $\lambda_t > 0$ so $\xi_t = 0$. Zero hinge loss for these data samples as they do not violate the margin constraints.

3. Non SVs, i.e., those t such that $\alpha_t = 0$.

   In this case, we can have $1 - \xi_t < y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0)$. From (6), we have $\lambda_t > 0$ and hence $\xi_t = 0$. Consequently, $1 < y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0)$ and there is zero hinge loss for these points.

Consequently, we can define the set of support vectors as

$$\text{SV} := \big\{(\mathbf{x}_t, y_t) : \alpha_t \in (0, C]\big\}. \tag{14}$$

Please see Fig. 1 to make sure you understand this. We observe the following:

1. The solution is sparse: points which are not on the margin, or "margin errors", have $\alpha_t = 0$. These are the non SVs.

2. The support vectors: those points on the decision boundary, or which are margin errors, contribute to the decision of a new unlabelled sample, i.e., the label of a new sample x' is the sign of

$$\hat{y}(\mathbf{x}') = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}')\rangle + \theta_0 \tag{15}$$
$$= \left\langle \sum_{t=1}^n \alpha_t y_t \boldsymbol{\phi}(\mathbf{x}_t), \boldsymbol{\phi}(\mathbf{x}') \right\rangle + \theta_0 \tag{16}$$
$$= \sum_{t=1}^n \alpha_t y_t K(\mathbf{x}_t, \mathbf{x}') + \theta_0 \tag{17}$$
$$= \sum_{t:(\mathbf{x}_t, y_t)\in \text{SV}} \alpha_t y_t K(\mathbf{x}_t, \mathbf{x}') + \theta_0 \tag{18}$$

3. Influence of the non-margin SVs is bounded, since their weight cannot exceed C.

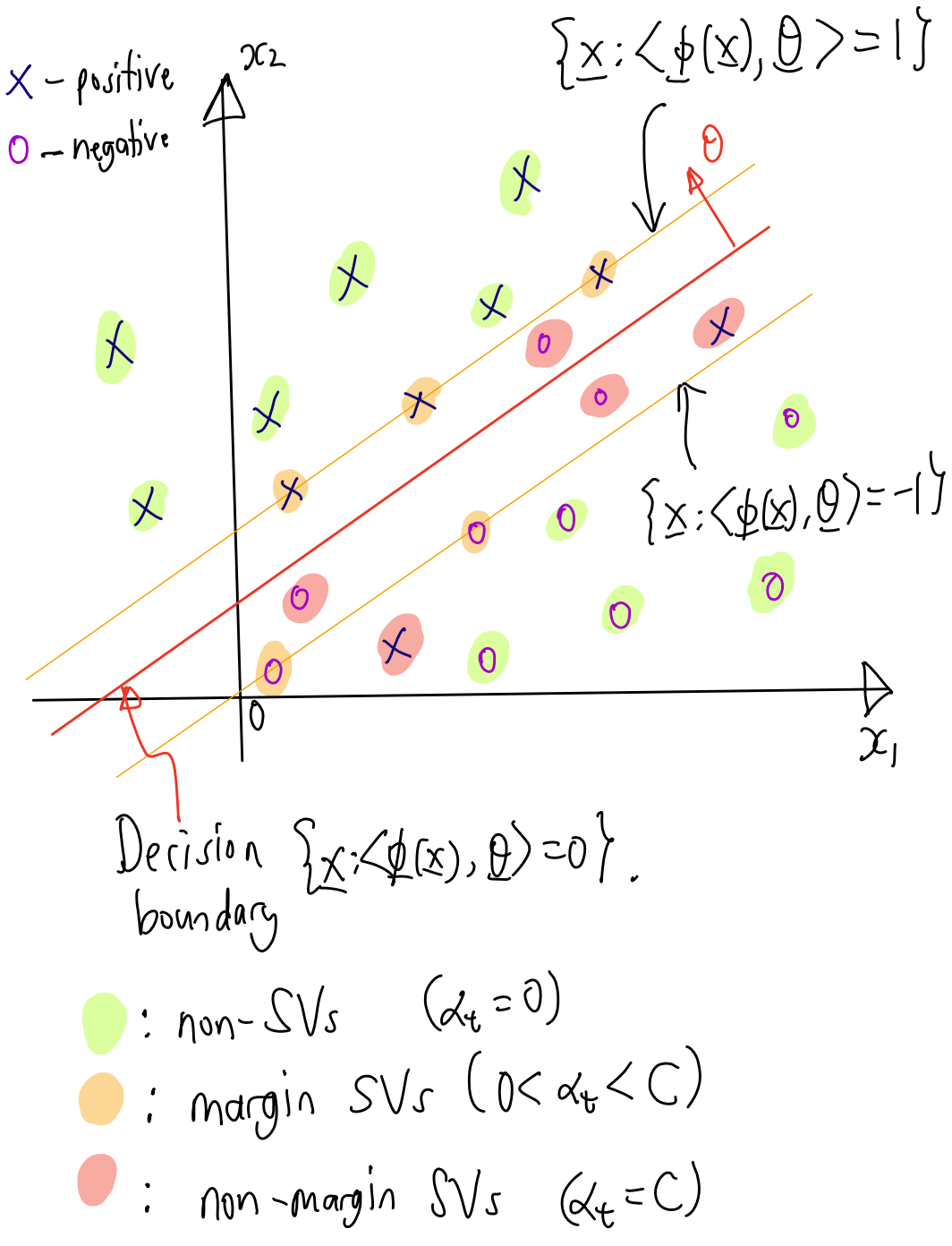


Figure 1: Illustration of different types of points

How can we estimate the offset θ_0? Pick a margin support vector with index t (from the second group of points as mentioned above). In this case

$$y_t(\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0) = 1. \tag{19}$$

Once we have solved for α, we can use the form of θ in (4) to obtain that

$$y_t\left(\sum_{s=1}^{n} \alpha_s y_s \langle \boldsymbol{\phi}(\mathbf{x}_s), \boldsymbol{\phi}(\mathbf{x}_t)\rangle + \theta_0\right) = y_t\left(\sum_{s=1}^{n} \alpha_s y_s K(\mathbf{x}_s, \mathbf{x}_t) + \theta_0\right) = 1. \tag{20}$$

Everything here except θ_0 is known so it can be solved easily.

2 The XOR Problem

Define the order-2 kernel

$$K(\mathbf{x}, \mathbf{z}) = (1 + \langle \mathbf{x}, \mathbf{z}\rangle)^2, \tag{21}$$

where $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ and $\mathbf{z} = (z_1, z_2) \in \mathbb{R}^2$. We may thus express the inner-product kernel $K(\mathbf{x}, \mathbf{z}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{z})\rangle$ in terms of various orders (monomials) as follows

$$K(\mathbf{x}, \mathbf{z}) = 1 + x_1^2 z_1^2 + 2x_1 x_2 z_1 z_2 + x_2^2 z_2^2 + 2x_1 z_1 + 2x_2 z_2. \tag{22}$$

The image of the input vector $\mathbf{x}$ induced in the feature space is

$$\boldsymbol{\phi}(\mathbf{x}) = [1, x_1^2, \sqrt{2}x_1 x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2]^\top. \tag{23}$$

Let the dataset be

$$\begin{aligned} \mathbf{x}_1 &= (-1, -1)^\top, \qquad y_1 = -1 \\ \mathbf{x}_2 &= (-1, +1)^\top, \qquad y_2 = +1 \\ \mathbf{x}_3 &= (+1, -1)^\top, \qquad y_3 = +1 \\ \mathbf{x}_4 &= (+1, +1)^\top, \qquad y_4 = -1 \end{aligned} \tag{24}$$

We find that the kernel matrix $\mathbf{K}$ with elements $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$ is

$$\mathbf{K} = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix} \tag{25}$$

The objective function in the dual form is

$$\begin{aligned} g(\alpha) = \sum_{t=1}^{4} \alpha_t - \frac{1}{2}\Big(&9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 \\ &+ 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2\Big) \end{aligned} \tag{26}$$

Optimizing $g(\alpha)$ w.r.t. the vector α, we obtain

$$\alpha_t = \frac{1}{8}, \qquad \forall\, t \in \{1, 2, 3, 4\}. \tag{27}$$

Since $\alpha_t > 0$ for all t, all training examples are support vectors. Note that the solution satisfies

$$\alpha_t \geq 0, \quad \forall\, t \qquad \text{and} \qquad \sum_t \alpha_t y_t = 0 \tag{28}$$

as required.

The optimum weight vector $\boldsymbol{\theta}$ can be found from $\boldsymbol{\theta} = \sum_{t=1}^{4} \alpha_t y_t \boldsymbol{\phi}(\mathbf{x}_t)$:

$$\boldsymbol{\theta} = \frac{1}{8}\left\{ -\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} \right\} = \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{29}$$

The first element indicates that the bias is 0. The optimal hyperplane (decision boundary) is

$$\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}) = 0 \qquad \Rightarrow \qquad \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}^\top \begin{bmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0 \qquad \Rightarrow \qquad -x_1x_2 = 0. \tag{30}$$

One can now check that the XOR problem is solved because now the feature under consideration is the product of the two components of $\mathbf{x}_t$, which then agrees with the label y_t.

MA4270 : Model Selection, the Vapnik-Chervonenkis Bound, and Bayesian Hypothesis Testing

Vincent Y. F. Tan

September 27, 2021

You should know all of Sections 1 and 3. Section 2 is optional, but Theorem 3 therein is good to know. We will talk about the definition of the VC dimension in great detail later on. For a finite set $\mathcal{F}$, we use $|\mathcal{F}|$ to denote its cardinality. So if $\mathcal{F} = \{a, b, c\}$, $|\mathcal{F}| = 3$.

These notes are heavily adapted from Jaakkola's MIT notes.

Everything I say here is very nicely summarized at this link, which also contains some very pretty pictures – https://mostafa-samir.github.io/ml-theory-pt2/.

1 Model Selection

We have seen different types of kernels. Which kernel should we use in practice? By choosing a kernel we specify the feature vectors on the basis of which linear predictions are made. Each model (class) refers to a set of linear functions (classifiers) based on the chosen feature representation. In many cases the models are nested in the sense that the more "complex" model contains the "simpler" one. Consider, for example, solving a classification problem with either

$$K_1(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle) \quad \text{or} \tag{1}$$

$$K_2(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^2. \tag{2}$$

Classifiers making use of the quadratic polynomial kernel can, in principle, reproduce the classifiers based on the linear kernel. Why? Recall that for the one-dimensional case, the feature map is $\boldsymbol{\phi}(x) = [1, \sqrt{2}x, x^2]^\top$. Hence, the quadratic kernel produces discriminant functions of the form $x \mapsto \theta_0 + \theta_1\sqrt{2}x + \theta_2 x^2$. If we specialize this to the case in which $\theta_2 = 0$, then we recover *affine* functions which is what the linear kernel $\boldsymbol{\phi}(x) = [1, x]^\top$ produces. As a model, i.e., as a set of linear classifiers based on the quadratic kernel, it therefore contains the simpler linear one. We can state this a bit more formally in terms of discriminant functions. For example, based on the linear kernel K_1, our discriminant functions are of the form

$$f_1(\mathbf{x}; \boldsymbol{\theta}, \theta_0) = \boldsymbol{\theta}^\top \boldsymbol{\phi}^{(1)}(\mathbf{x}) + \theta_0 \tag{3}$$

where $\boldsymbol{\phi}^{(1)}(\mathbf{x})$ is the feature representation corresponding to K_1 such that $K_1(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\phi}^{(1)}(\mathbf{x}), \boldsymbol{\phi}^{(1)}(\mathbf{x}') \rangle$. By varying the parameters $\boldsymbol{\theta}$ and θ_0 we can generate the set of possible discriminant functions corresponding to this kernel:

$$\mathcal{F}_1 = \left\{ f_1(\cdot; \boldsymbol{\theta}, \theta_0) : (\boldsymbol{\theta}, \theta_0) \in \mathbb{R}^{d_1} \times \mathbb{R} \right\}. \tag{4}$$

The class of functions $\mathcal{F}_2$ is defined similarly for the quadratic feature map $\boldsymbol{\phi}^{(2)}$. The fact that the two models are nested means that $\mathcal{F}_1 \subset \mathcal{F}_2$.

The formal problem for us to solve is then to select a kernel K_i from a set of possible kernels $K_1, K_2, \ldots$ where the models associated with the kernels are nested $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \ldots$. This is a *model selection problem* in a standard nested form. From here on we will be referring to discriminant functions rather than kernels so as to emphasize the point that the discussion applies to other types of classifiers as well.

1.1 Model Selection Preliminaries

Recall that in the learning problem, we are given a dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}$ and we are tasked to learn a good discriminant function $f_{\boldsymbol{\theta},\theta_0}(\mathbf{x})$ such that it generalizes well on data samples that are not in the training set $\mathcal{D}$. These are called *test samples*. Model selection is intended to facilitate this process. In other words, we switch from one model to another so as to generalize better. The model we select will define how we will respond to any training data, i.e., which classifier we choose to make predictions on new examples.

So given the dataset and a certain model class $\mathcal{F}_i$ (a set of functions, e.g., linear), we have seen that we should choose the best fitting function $\hat{f}_i \in \mathcal{F}_i$ by minimizing

$$J(\boldsymbol{\theta}, \theta_0) = \sum_{t=1}^{n} \text{Loss}(y_t, f_{\boldsymbol{\theta},\theta_0}(\mathbf{x}_t)) + \lambda \|\boldsymbol{\theta}\|^2. \tag{5}$$

The loss above could be the hinge loss $\text{Loss}_{\text{h}}(z) = (1-z)^+$, logistic loss $\text{Loss}_{\text{logistic}}(z) = \log[1+\exp(-z)]$ where $z = y_t f_{\boldsymbol{\theta},\theta_0}(\mathbf{x}_t)$. The regularization parameter $\lambda = \lambda_n$ could, in general, depend on the number of training samples and number of parameters to estimate. We are interested in how the classifier $\hat{f}_i(\mathbf{x}; \boldsymbol{\theta}, \theta_0)$ generalizes to new unseen samples.

Each parameter setting $(\boldsymbol{\theta}, \theta_0)$, i.e., each discriminant function in our set, has an associated *expected loss* or *risk*

$$R(\boldsymbol{\theta}, \theta_0) = \mathbb{E}_{(\mathbf{x},y)\sim P}\left[\text{Loss}_{0,1}\Big(y, f(\mathbf{x}; \boldsymbol{\theta}, \theta_0)\Big)\right] \tag{6}$$

where the new/test sample is generated from an underlying unknown joint distribution P and $\text{Loss}_{0,1}(y, \hat{y}) = \mathbb{1}\{y \neq \hat{y}\}$ is the zero-one loss.

The quantity of interest to us is the *best expected loss* or *smallest risk* $R(\hat{\boldsymbol{\theta}}, \hat{\theta}_0)$ or $R(\hat{f}_i)$ for short, corresponding to the classifier or discriminant function we would choose from $\mathcal{F}_i$ in response to the training data $\mathcal{D}_n$. Thus, $\hat{f}_i \in \mathcal{F}_i$ is learned from the training data. Ideally, we would then select the model $\mathcal{F}_i$ that leads to the smallest expected loss

$$R(\hat{f}_i) = \mathbb{E}_{(\mathbf{x},y)\sim P}\left[\text{Loss}_{0,1}\Big(y, \hat{f}_i(\mathbf{x})\Big)\right]. \tag{7}$$

Note that this risk $R(\hat{f}_i)$ is still a random variable as $\hat{f}_i$ depends on the training data that we assume was also sampled from the same underlying distribution P.

Now, we clearly do not have access to the underlying distribution and therefore cannot evaluate $R(\hat{f}_i)$ since it's an average over P. In fact, the whole model selection problem would go away if had access to the underlying distribution $P(\mathbf{x}, y) = P_{\mathbf{X},Y}(\mathbf{x}, y)$. To classify new instances, we would simply forget about the training set and use the minimum probability of error classifier

$$\hat{y}(\mathbf{x}) = \underset{y\in\{-1,+1\}}{\arg\max}\ P_{Y|\mathbf{X}}(y \mid \mathbf{x}) \tag{8}$$

To show that this is the minimum probability of error rule, see Section 3. No classifier could lead to a lower probability of error. Our task is much more difficult since we have to select $\hat{f}_i \in \mathcal{F}_i$ as well as the model class $\mathcal{F}_i$ on the basis of the training data alone, without access to P.

Let's try to understand first intuitively what the model selection criterion has to be able to do. To make this a bit more concrete, consider just choosing between $\mathcal{F}_1$ and $\mathcal{F}_2$ corresponding to linear or quadratic feature vectors. Since the models are nested, $\mathcal{F}_1 \subset \mathcal{F}_2$ we can always achieve lower classification error on the training set by adopting $\mathcal{F}_2$. This is regardless of whether the true underlying model is linear. So, by choosing $\mathcal{F}_2$, we may be over-fitting. If the true relationship between the labels and examples were linear (the minimum probability of error classifier is linear), then the quadratic nature of the resulting decision boundary would simply be due to noise and couldn't generalize very well. So we should be able to see an increasing gap between the training and test errors as a function of the model complexity as in Figure 1 below. Clearly, all things being equal, we should select $\mathcal{F}_1$ as it is a simpler model. The real question is how to balance the "complexity" of the model, some measure of *size* or *power* of $\mathcal{F}_i$, against their fit to the training data.

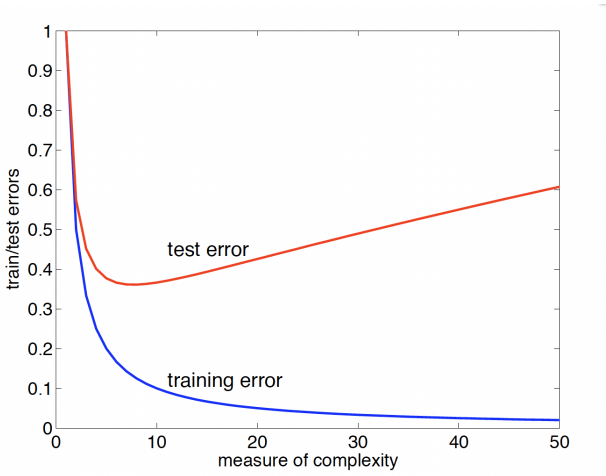


Figure 1: Training and test errors as a function of model order (e.g., degree of polynomial kernel).

1.2 Structural Risk Minimization

One approach to model selection is to try to directly relate the smallest expected loss in (7) to the *empirical risk*

$$\hat{R}_n(\hat{f}_i) = \frac{1}{n}\sum_{t=1}^{n} \text{Loss}_{0,1}\Big(y_t, \hat{f}_i(\mathbf{x}_t)\Big). \tag{9}$$

In contrast to the expected loss in (7), this quantity is something that we can calculate from the training dataset $\mathcal{D}$. For our purposes here, $\hat{f}_i \in \mathcal{F}_i$ could be any estimate derived from the training set that approximately tries to minimizing the empirical risk in (9).

We are interested in quantifying how much $R(\hat{f}_i)$ can deviate from $R(\hat{f}_i)$. The larger the deviation the less representative the training error is about the generalization error. This happens with more complex models $\mathcal{F}_i$. Indeed, we aim to show that with high probability

$$R(\hat{f}_i) \leq \hat{R}_n(\hat{f}_i) + C(n, \mathcal{F}_i, \delta) \tag{10}$$

where the complexity parameter $C(n, \mathcal{F}_i, \delta)$ only depends on the model $\mathcal{F}_i$, the number of training instances, and a parameter δ. The penalty does not depend on the actual training data. We will discuss the parameter δ below in more detail. For now, it suffices to say that $1-\delta$ specifies the probability that the bound holds. We can only give a probabilistic guarantee in this sense since the empirical risk (training error) is a random quantity that depends on the specific instantiation of the data.

For nested models, $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \ldots$, the penalty is necessarily an increasing function of i, the model order (e.g., the degree of polynomial kernel). Moreover, the penalty should go down as a function n. In other words, the more data we have, the more complex models we expect to be able to fit and still have the training error close to the generalization error. The type of result in (10) gives us an upper bound guarantee of generalization error. We can then select the model with the best guarantee, i.e., the one with the lowest bound. Figure 1 shows how we would expect the upper bound to behave as a function of increasingly complex models in our nested "hierarchy" of models.

Theorem 1. *Consider function classes $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3 \subset \ldots \subset \mathcal{F}_k$ such that $|\mathcal{F}_i| < \infty$ (function classes that are finite). With probability at least $1-\delta$ over the training set*

$$R(f) \leq \hat{R}_n(f) + \sqrt{\frac{\log|\mathcal{F}_i| + \log(2/\delta)}{2n}} \quad \textit{uniformly for all } f \in \mathcal{F}_i. \tag{11}$$

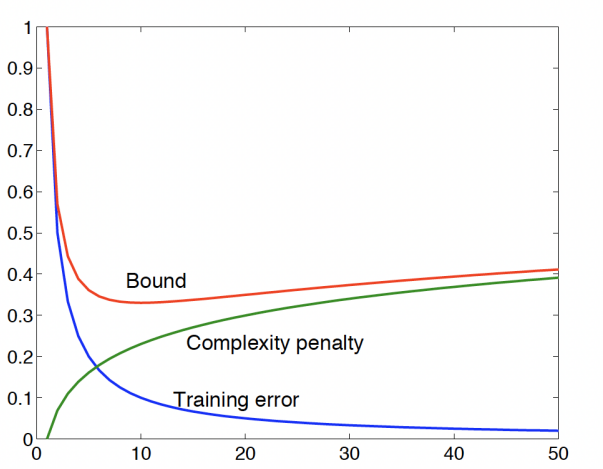


Figure 2: Bound on the expected risk as a function of model order (e.g., degree of polynomial kernel).

The implication here is that for model selection using structural risk minimization, we would then estimate $\hat{f}_i \in \mathcal{F}_i$ for each model class, plug the resulting $\hat{f}_i$ and $\mathcal{F}_i$ on the right hand side of (11), and choose the model with the lowest bound, i.e.,

$$\hat{f}_{\text{srm}} = \underset{i=1,2,\ldots,k}{\arg\min} \left\{ \hat{R}_n(\hat{f}_i) + \sqrt{\frac{\log|\mathcal{F}_i| + \log(2/\delta)}{2n}} \right\}. \tag{12}$$

Note that n and δ would be the same for all models under consideration.

The proof of Theorem 1 using two ingredients which we summarize here.

- (Union bound) Let $\mathcal{E}_i, i \in \mathbb{N}$ be events. Then

$$\Pr\left(\bigcup_{i\in\mathbb{N}} \mathcal{E}_i\right) \leq \sum_{i\in\mathbb{N}} \Pr(\mathcal{E}_i). \tag{13}$$

  Exercise 1. *Prove this. Note that the union bound works for countably infinitely many sets.*

- (Hoeffding's inequality) Let $S_i, i = 1, \ldots, n$ be i.i.d. random variables such that $S_i \in [0, 1]$ almost surely. Let $\mu = \mathbb{E}[S_1]$. Then for all $n \in \mathbb{N}$,

$$\Pr\left(\left|\left(\frac{1}{n}\sum_{i=1}^{n} S_i\right) - \mu\right| > \epsilon\right) \leq 2e^{-2n\epsilon^2}. \tag{14}$$

  Please see https://en.wikipedia.org/wiki/Hoeffding%27s_inequality for the proof.

Proof of Theorem 1. Our goal is to find a tight upper bound on $\Pr(\max_{f\in\mathcal{F}_i} |R(f) - \hat{R}_n(f)| > \epsilon)$ for any ϵ. The probability here is taken over the choice of the training data that is randomly sampled from P. So if we used $\epsilon > 0$ to claim that

$$R(f) \leq \hat{R}_n(f) + \epsilon \quad \forall\, f \in \mathcal{F}_i \tag{15}$$

then this expression would fail with probability

$$\delta = \Pr\left(\max_{f\in\mathcal{F}_i} |R(f) - \hat{R}_n(f)| > \epsilon\right), \tag{16}$$

or, put another way, it would hold with probability $1-\delta$ over the choice of the training data. If we fix δ, then the smallest $\epsilon = \epsilon(n, \mathcal{F}_i, \delta)$ that satisfies (16) is the complexity penalty we are after. Note that since the expression holds for all $f \in \mathcal{F}_i$ it necessarily also holds for $\hat{f}_i$.

In most cases we cannot compute δ exactly from (16) but we can derive an upper bound. This upper bound will lead to a larger than necessary complexity penalty but at least we will get a closed form expression (the utility of the model selection criterion will indeed depend on how tight a bound we can obtain). We will proceed as follows

$$\Pr\left(\max_{f\in\mathcal{F}_i} |R(f) - \hat{R}_n(f)| > \epsilon\right) = \Pr\left(\bigcup_{f\in\mathcal{F}_i} \left\{|R(f) - \hat{R}_n(f)| > \epsilon\right\}\right) \tag{17}$$

$$\leq \sum_{f\in\mathcal{F}_i} \Pr\left(|R(f) - \hat{R}_n(f)| > \epsilon\right), \tag{18}$$

where we applied the union bound. Now, we focus on the inner probabilities. We associate to each training sample $(\mathbf{x}_t, y_t)$ an indicator/binary/Bernoulli random variable $S_t \in \{0, 1\}$ of whether the sample disagrees with f. Namely, $S_t = 1$ iff $y_t f(\mathbf{x}) \leq 0$ and $S_t = 0$ otherwise. The empirical error is therefore just an average of independent random variables (indicators) S_t:

$$\hat{R}_n(f) = \frac{1}{n}\sum_{t=1}^{n} S_t. \tag{19}$$

What is the expected value of each S_t when the expectation is taken over the choice of the training data? It's just $R(f)$, the expected risk of f. So, we can rewrite

$$\Pr\left(|R(f) - \hat{R}_n(f)| > \epsilon\right) = \Pr\left(\left|\mu - \frac{1}{n}\sum_{t=1}^{n} S_t\right| > \epsilon\right), \tag{20}$$

where $\mu = R(f)$ and the probability is now over n independent binary random variables $S_1, \ldots, S_n$ for which $\Pr(S_t = 1) = \mu$. By Hoeffding's inequality,

$$\Pr\left(|R(f) - \hat{R}_n(f)| > \epsilon\right) \leq 2e^{-2n\epsilon^2}. \tag{21}$$

Combining this with (18) yields

$$\Pr\left(\max_{f\in\mathcal{F}_i} |R(f) - \hat{R}_n(f)| > \epsilon\right) \leq \sum_{f\in\mathcal{F}_i} 2e^{-2n\epsilon^2} = 2|\mathcal{F}_i|e^{-2n\epsilon^2}. \tag{22}$$

Setting the rightmost term to be δ and solving for ϵ yields

$$\epsilon = \epsilon(n, \mathcal{F}_i, \delta) = \sqrt{\frac{\log|\mathcal{F}_i| + \log(2/\delta)}{2n}}. \tag{23}$$

This is the complexity penalty we were after in this simple case with only a finite number of classifiers in our set. □

Of course, this result is a huge simplification of real life because most function classes are uncountable, e.g., the set of linear classifiers $\{f : \mathbb{R}^d \to \{-1, +1\} : f(\mathbf{x}) = \text{sgn}(\langle \mathbf{x}, \boldsymbol{\theta}\rangle)$ for some $\boldsymbol{\theta} \in \mathbb{R}^d\}$. So we must do more and this is the subject of the next (optional) section on the analogue of Theorem 1 for function classes that contain uncountably many functions.

2 The Vapnik-Chervonenkis Bound (Optional)

2.1 Basic Definitions

Later on, you are expected to know the definition of VC dimension and how to calculate it for various function classes. You should also know the statement of Theorem 3 but, of course, not the detailed proof.

Let $\mathcal{X}$ denote the feature space (think of $\mathcal{X}$ as $\mathbb{R}$), and $\mathcal{Y} = \{0, 1\}$ denote the label space. Let $f : \mathcal{X} \to \mathcal{Y}$ be a classifier (also called predictor or model). Let $(\mathbf{X}, Y) \sim P_{\mathbf{X}Y}$ where the joint probability distribution $P_{\mathbf{X}Y}$ is unknown to us. We measure the classification/prediction error using the 0-1 loss function $\ell(f(\mathbf{X}), Y) = \mathbb{1}\{f(\mathbf{X}) \neq Y\}$, which gives rise to the *expected risk*

$$R(f) = \mathbb{E}[\ell(f(\mathbf{X}), Y)] = \Pr(f(\mathbf{X}) \neq Y) \tag{24}$$

Let $\mathcal{F}$ be a class of candidate models (classification rules), i.e., $\mathcal{F}$ is simply a collection of functions $f : \mathcal{X} \to \mathcal{Y}$. We have access to a dataset

$$\mathcal{D}_n = \{(\mathbf{X}_i, Y_i)\}_{i=1}^n \tag{25}$$

where $(\mathbf{X}_i, Y_i)$ is assumed to be drawn i.i.d. from $P_{\mathbf{X}Y}$. The *empirical risk* is defined as

$$\hat{R}_n(f) := \frac{1}{n}\sum_{i=1}^n \ell(f(\mathbf{X}_i), Y_i) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\}. \tag{26}$$

This is a random variable because the dataset $\mathcal{D}_n$ is random.

Each function $f \in \mathcal{F}$ gives rise to a labeling sequence

$$(f(\mathbf{X}_1), \ldots, f(\mathbf{X}_n)) \in \{0, 1\}^n. \tag{27}$$

For a given training set $\mathcal{D}_n$ there are at most 2^n such distinct sequences, but often much less. Let $S(\mathcal{F}, n)$ be the maximum number of labeling sequences the class $\mathcal{F}$ induces over n training points in $\mathcal{X}$. Formally let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and define

$$N_{\mathcal{F}}(\mathbf{x}_1, \ldots, \mathbf{x}_n) := \{(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)) : f \in \mathcal{F}\}. \tag{28}$$

Definition 1. *The* shatter coefficient *of the class $\mathcal{F}$ is defined as*

$$S(\mathcal{F}, n) := \sup_{\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}} |N_{\mathcal{F}}(\mathbf{x}_1, \ldots, \mathbf{x}_n)|. \tag{29}$$

Clearly $S(\mathcal{F}, n) \leq 2^n$ but it is often much smaller.

Let's consider an example of a function class. Let the feature space $\mathcal{X} = \mathbb{R}$. Consider the following class of functions

$$\mathcal{F} := \{f : \mathcal{X} \to \mathcal{Y} : f(x) = \mathbb{1}\{x > t\}, t \in \mathbb{R}\} \cup \{f : \mathcal{X} \to \mathcal{Y} : f(x) = \mathbb{1}\{x < t\}, t \in \mathbb{R}\} \tag{30}$$

This is simply a decision stump in one dimension. Now let us determine $S(\mathcal{F}, n)$ for this class. Suppose we have n scalars $x_1, \ldots, x_n \in \mathbb{R}$. Assume there are no identical points (otherwise the number of possible labelings is even smaller). Any classifier in $\mathcal{F}$ labels all points to the left of a number $t \in [0, 1]$ as "1" or "0", and points to the right as "0" and "1", respectively. For $t \in [0, x_1)$, all points are either labelled "0" and "1". For $t \in (x_1, x_2)$, x_1 is labelled "0" and "1" and $x_2, \ldots, x_n$ are labeled "1" and "0" and so on. We see that there are exactly $2n$ different possible labelings, therefore $S(\mathcal{F}, n) = 2n$. This is far less than the very conservative bound $S(\mathcal{F}, n) \leq 2^n$.

Definition 2. *The* Vapnik-Chervonenkis (VC) dimension *is defined as the largest integer k such that $S(\mathcal{F}, n) = 2^k$. The VC dimension of a class $\mathcal{F}$ is denoted by* $\mathrm{VC}(\mathcal{F})$.

Note that the VC dimension is not a function of the number of training data.

Lemma 2 (Sauer's Lemma)**.** *Let* $d := \mathrm{VC}(\mathcal{F})$.[1] *We have*

$$S(\mathcal{F}, n) \leq \Phi_d(n) := \sum_{k=1}^{d} \binom{n}{k} \tag{31}$$

In particular, if $n > d$, we have

$$S(\mathcal{F}, n) \leq \left(\frac{en}{d}\right)^d. \tag{32}$$

To show this, we simply note that if we let $d := \mathrm{VC}(\mathcal{F})$ and $n > d$, then $0 \leq \frac{d}{n} < 1$, we have

$$\left(\frac{d}{n}\right)^d \sum_{k=1}^{d} \binom{n}{k} \leq \sum_{k=1}^{d} \left(\frac{d}{n}\right)^k \binom{n}{k} \leq \sum_{k=1}^{n} \left(\frac{d}{n}\right)^k \binom{n}{k} = \left(1 + \frac{d}{n}\right)^n \leq e^d \tag{33}$$

as desired.

Proof of Lemma 2. The proof proceeds by induction on both $d = \mathrm{VC}(\mathcal{F})$ and n. We have two base cases: when $n = 0$ and d is arbitrary, and when $d = 0$ and n is arbitrary. When $n = 0$, there can only be one subset, hence $S(\mathcal{F}, n) \leq 1 = \Phi_d(0)$. When $d = \mathrm{VC}(\mathcal{F}) = 0$, no set of points can be shattered, hence all points can be labeled only one way. From this we conclude that $\mathcal{S}(\mathcal{F}, n) = 1 \leq \Phi_0(n)$. So the lemma holds for the base case.

We assume for induction that for all n', d' such that $n' \leq n$ and $d' \leq d$ and at least one of these inequalities is strict, we have $\mathcal{S}(\mathcal{F}, n') \leq \Phi_{d'}(n')$. Now suppose we have a set $S = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ of cardinality n. Let H be a class of functions defined only over $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ such that $C(S) = H(S) = H$ where

$$C(S) = \{(c(\mathbf{x}_1), c(\mathbf{x}_2), \ldots, c(\mathbf{x}_n)) \in \{0,1\}^n : c \in C\}. \tag{34}$$

Since any $\tilde{S} \subset S$ that is shattered by H is also shattered by C, we have $\mathrm{VC}(H) \leq \mathrm{VC}(C)$.

We now construct H_1 and H_2 on which we apply our induction hypothesis as follows: for each possible labelling of $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}\}$ induced by a function in H, we add a representative function from H to H_1; we let $H_2 = H \setminus H_1$. So for each $h \in H_2$, $\exists \tilde{h} \in H_1$ such that $h(\mathbf{x}_i) = \tilde{h}(\mathbf{x}_i)$ for $i \in \{1, \ldots, n-1\}$ and $h(\mathbf{x}_n) \neq \tilde{h}(\mathbf{x}_n)$. For convenience, let's choose the representatives such that $h(\mathbf{x}_n) = 1$ and $\tilde{h}(\mathbf{x}_n) = 0$, so all $h \in H_2$ label $\mathbf{x}_n$ as positive.

By construction, we have

$$|C(S)| = |H(S)| = |H_1(S)| + |H_2(S)|. \tag{35}$$

Since $H_1 \subset H$ we have $\mathrm{VC}(H_1) \leq \mathrm{VC}(H) \leq d$. Moreover, we can show that

$$|H_1(S)| = |H_1(S \setminus \{\mathbf{x}_n\})|. \tag{36}$$

In one direction it is clear that $|H_1(S)| \geq |H_1(S \setminus \{\mathbf{x}_n\})|$. In the other direction, we also have $|H_1(S)| \leq |H_1(S \setminus \{\mathbf{x}_n\})|$ since there is no labelling of h of $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}\}$ such that both $(h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_{n-1}), 0)$ and $(h(\mathbf{x}_1), h(\mathbf{x}_2), \ldots, h(\mathbf{x}_{n-1}), 1)$ are in H_1.

By induction, we have

$$|H_1(S)| \leq \Phi_d(n-1). \tag{37}$$

Now note that if T is shattered by H_2, then $T \cup \{\mathbf{x}_n\}$ is shattered by H. If T is shattered by H_2 then (a) $\mathbf{x}_n \notin T$ (because all $h \in H_2$ label $\mathbf{x}_n$ as positive), and (b) $T \cup \{\mathbf{x}_n\}$ is shattered by H (because each $h \in H_2$ has a twin $\tilde{h} \in H_1$ that is identical except on $\mathbf{x}_n$). So

$$\mathrm{VC}(H_2) \leq \mathrm{VC}(H) - 1 \leq d - 1. \tag{38}$$

[1] Note that d is not the dimension of the data in this lemma.

We can also show that

$$|H_2(S)| = |H_2(\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n-1}\})| \tag{39}$$

and by induction, we get

$$|H_2(S)| \leq \Phi_{d-1}(n-1). \tag{40}$$

Combining all these, we get

$$|C(S)| \leq \Phi_d(n-1) + \Phi_{d-1}(n-1). \tag{41}$$

Since

$$\sum_{k=1}^{d} \binom{n-1}{k} + \sum_{k=1}^{d-1} \binom{n-1}{k} = \binom{n}{0} + \sum_{k=1}^{d} \binom{n-1}{k} + \sum_{k=1}^{d} \binom{n-1}{k-1} = \sum_{k=1}^{d} \binom{n}{k} \tag{42}$$

we get $|C(S)| \leq \Phi_d(n)$ as desired. □

2.2 Main Result for the Vapnik-Chervonenkis Bound

The main result in this section is

Theorem 3 (VC Inequality)**.** *The expected risk can be bounded in terms of the empirical risk as follows. Fix $\epsilon > 0$ and for all n such that $n\epsilon^2 \geq 2$,*

$$\Pr\left(\sup_{f \in \mathcal{F}} |\hat{R}_n(f) - R(f)| > \epsilon\right) \leq 8S(\mathcal{F}, n)e^{-n\epsilon^2/32}. \tag{43}$$

Invoking Sauer's lemma, specifically (32), we see that, with probability at least $1 - \delta$,

$$R(f) \leq \hat{R}_n(f) + O\left(\sqrt{\frac{\mathrm{VC}(\mathcal{F})}{n} \log \frac{n}{\delta \cdot \mathrm{VC}(\mathcal{F})}}\right) \tag{44}$$

uniformly in $f \in \mathcal{F}$.

This extends a similar bound for finite function classes $\mathcal{F}$ that we showed in Theorem 1. Note that now the VC dimension $\mathrm{VC}(\mathcal{F})$ plays the role of the number of functions in the functions class $|\mathcal{F}|$.

2.3 Proof of the Vapnik-Chervonenkis Bound

The proof of Theorem 3 is not trivial but it contains several interesting elements worth highlighting. The approach presented here is in the book by Devroye, Györfi and Lugosi. We simplify the exposition in the notes by Prof. Robert Nowak.

Proof. Step 1: Symmetrization by a "ghost" sample. Let $\mathcal{D}'_n = \{(\mathbf{X}'_i, Y'_i)\}_{i=1}^n$ where $(\mathbf{X}'_i, Y'_i) \sim P_{\mathbf{X}Y}$ be an dataset which is independent of $\mathcal{D}_n$. Denote its empirical risk by $\hat{R}'_n(f) = \frac{1}{n}\sum_{i=1}^n \ell(f(\mathbf{X}'_i), Y'_i)$. We will show that

$$\Pr\left(\sup_{f \in \mathcal{F}} |\hat{R}_n(f) - R(f)| > \epsilon\right) \leq 2\Pr\left(\sup_{f \in \mathcal{F}} |\hat{R}_n(f) - \hat{R}'_n(f)| > \frac{\epsilon}{2}\right). \tag{45}$$

Observe that the expected risk has disappeared on the right-hand-side (RHS) and subsequently we only have to understand the stochastic behavior of the difference of the empirical risks $\hat{R}_n(f) - \hat{R}'_n(f)$.

To prove (45), define $\tilde{f} = \tilde{f}(\mathcal{D}_n)$ to be any function in $\mathcal{F}$ such that $|\hat{R}_n(f) - R(f)| > \epsilon$ if such a function exists, otherwise $\tilde{f}$ is an arbitrary element of $\mathcal{F}$. We now start with the RHS of (45):

$$\Pr\left(\sup_{f\in\mathcal{F}} |\hat{R}_n(f) - \hat{R}'_n(f)| > \frac{\epsilon}{2}\right) \geq \Pr\left(|\hat{R}_n(\tilde{f}) - \hat{R}'_n(\tilde{f})| > \frac{\epsilon}{2}\right) \tag{46}$$

$$\geq \Pr\left(|\hat{R}_n(\tilde{f}) - R(\tilde{f})| > \epsilon, |\hat{R}'_n(\tilde{f}) - R(\tilde{f})| < \frac{\epsilon}{2}\right) \tag{47}$$

$$= \mathbb{E}\left[\mathbb{1}\left\{|\hat{R}_n(\tilde{f}) - R(\tilde{f})| > \epsilon\right\} \mathbb{1}\left\{|\hat{R}'_n(\tilde{f}) - R(\tilde{f})| < \frac{\epsilon}{2}\right\}\right] \tag{48}$$

$$= \mathbb{E}\left[\mathbb{1}\left\{|\hat{R}_n(\tilde{f}) - R(\tilde{f})| > \epsilon\right\} \mathbb{E}\left[\mathbb{1}\left\{|\hat{R}'_n(\tilde{f}) - R(\tilde{f})| < \frac{\epsilon}{2}\right\} \middle| \mathcal{D}_n\right]\right] \tag{49}$$

$$= \mathbb{E}\left[\mathbb{1}\left\{|\hat{R}_n(\tilde{f}) - R(\tilde{f})| > \epsilon\right\} \Pr\left(|\hat{R}'_n(\tilde{f}) - R(\tilde{f})| < \frac{\epsilon}{2} \middle| \mathcal{D}_n\right)\right] \tag{50}$$

where (47) follows from the fact that $|a-b| > \epsilon$ and $|c-b| < \epsilon$ means that $|a-c| > \epsilon/2$ and (49) follows by the tower property of conditional expectation. We now bound the probability within the expectation. To do so, note that conditioned on the dataset $\mathcal{D}_n$, we have that

$$\hat{R}'_n(\tilde{f}) - R(\tilde{f}) = \frac{1}{n}\sum_{i=1}^{n} U_i \tag{51}$$

where $U_i = \mathbb{1}\{\tilde{f}(\mathbf{X}'_i) \neq Y'_i\} - \Pr(\tilde{f}(\mathbf{X}'_i) \neq Y'_i|\mathcal{D}_n)$ are zero-mean i.i.d. random variables that satisfy $|U_i| \leq 2$ almost surely. Note that $\tilde{f}$ is now deterministic (due the the conditioning on $\mathcal{D}_n$) and the randomness arises due to $\mathcal{D}'_n$. Thus, by Chebyshev's inequality

$$\Pr\left(|\hat{R}'_n(\tilde{f}) - R(\tilde{f})| < \frac{\epsilon}{2} \middle| \mathcal{D}_n\right) \geq 1 - \Pr\left(\left|\frac{1}{n}\sum_{i=1}^{n} U_i\right| \geq \frac{\epsilon}{2} \middle| \mathcal{D}_n\right) \tag{52}$$

$$\geq 1 - \frac{4}{n^2\epsilon^2} n \,\mathsf{Var}(U_1|\mathcal{D}_n) \tag{53}$$

$$= 1 - \frac{1}{n\epsilon^2} \tag{54}$$

$$\geq \frac{1}{2}. \tag{55}$$

We used the fact that for a random variable supported on $[-1, 1]$ the variance is maximized by the discrete distribution with point masses at -1 and $+1$ which is $1/4$ and in the final step, we used the fact that $n\epsilon^2 \geq 2$. Thus, putting everything together, we see that

$$\Pr\left(\sup_{f\in\mathcal{F}} |\hat{R}_n(f) - \hat{R}'_n(f)| > \frac{\epsilon}{2}\right) \geq \frac{1}{2}\mathbb{E}\left[\mathbb{1}\left\{|\hat{R}_n(\tilde{f}) - R(\tilde{f})| > \epsilon\right\}\right] \tag{56}$$

$$= \frac{1}{2}\Pr\left(|\hat{R}_n(\tilde{f}) - R(\tilde{f})| > \epsilon\right) \tag{57}$$

$$\geq \frac{1}{2}\Pr\left(\sup_{f\in\mathcal{F}} |\hat{R}_n(f) - R(f)| > \epsilon\right). \tag{58}$$

This proves (45).

Step 2: Symmetrization by random signs. Now observe that the random variable of interest is

$$\hat{R}_n(f) - \hat{R}'_n(f) = \frac{1}{n}\sum_{i=1}^{n} (\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} - \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\}). \tag{59}$$

Also note that the random variables $\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\}$ and $\mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\}$ have the same distribution so their difference $\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} - \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\}$ is a zero-mean, symmetric random variable (a symmetric random variable Z is one in which its distribution is the same as $-Z$.). Now let us introduce new random variables $\sigma_i, i = 1, \ldots, n$ such that $\Pr(\sigma_i = +1) = \Pr(\sigma_i = -1) = 1/2$. These random variables are mutually independent and also independent of $\mathcal{D}_n, \mathcal{D}'_n$. In light of the fact that $\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} - \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\}$ is symmetric, we note that

$$\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} - \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\} \stackrel{d}{=} \sigma_i(\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} - \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\}) \tag{60}$$

where $\stackrel{d}{=}$ means equality in distribution. Thus, we have

$$\begin{aligned}
&\Pr\left(\sup_{f\in\mathcal{F}} |\hat{R}_n(f) - \hat{R}'_n(f)| > \frac{\epsilon}{2}\right) \\
&= \Pr\left(\sup_{f\in\mathcal{F}} \frac{1}{n}\Big|\sum_{i=1}^n \sigma_i(\mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} - \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\})\Big| > \frac{\epsilon}{2}\right) &(61)\\
&\leq \Pr\left(\sup_{f\in\mathcal{F}} \frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\}\Big| > \frac{\epsilon}{4}\right) + \Pr\left(\sup_{f\in\mathcal{F}} \frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{X}'_i) \neq Y'_i\}\Big| > \frac{\epsilon}{4}\right) &(62)\\
&= 2\Pr\left(\sup_{f\in\mathcal{F}} \frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\}\Big| > \frac{\epsilon}{4}\right) &(63)
\end{aligned}$$

Step 3: Conditioning on $\mathcal{D}_n$. This step is conceptually the same we used in all the generalization bounds for finite function classes. We are going to perform a union bound over all the models under consideration. The difference here is that this set in no longer the entire class $\mathcal{F}$ but instead just a finite subset of it.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$ and $y_1, \ldots, y_n \in \mathcal{Y}$ be arbitrary sequences. Let examine the quantity

$$\frac{1}{n}\left|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\}\right|. \tag{64}$$

where the randomness is solely on the random signs σ_i. The sequence $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$ can take at most $S(\mathcal{F}, n)$ different values, therefore

$$(\mathbb{1}\{f(\mathbf{x}_1) \neq y_1\}, \ldots, \mathbb{1}\{f(\mathbf{x}_n) \neq y_n\}) \tag{65}$$

can take at most $S(\mathcal{F}, n)$ different values. Let $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \subset \mathcal{F}$ be the smallest subset of $\mathcal{F}$ such that

$$N_{\mathcal{F}}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = N_{\mathcal{F}(\mathbf{x}_1,\ldots,\mathbf{x}_n)}(\mathbf{x}_1, \ldots, \mathbf{x}_n). \tag{66}$$

In words $\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ is the smallest subset of $\mathcal{F}$ that gives rise to all the different prediction rules for the data $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, therefore $|\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)| \leq S(\mathcal{F}, n)$. We can now apply the union bound as follows

$$\begin{aligned}
&\Pr\left(\sup_{f\in\mathcal{F}} \frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\}\Big| > \frac{\epsilon}{4}\right) \\
&= \Pr\left(\max_{f\in\mathcal{F}(\mathbf{x}_1,\ldots,\mathbf{x}_n)} \frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\}\Big| > \frac{\epsilon}{4}\right) &(67)\\
&\leq \sum_{f\in\mathcal{F}(\mathbf{x}_1,\ldots,\mathbf{x}_n)} \Pr\left(\frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\}\Big| > \frac{\epsilon}{4}\right) &(68)\\
&\leq |\mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)| \max_{f\in\mathcal{F}(\mathbf{x}_1,\ldots,\mathbf{x}_n)} \Pr\left(\frac{1}{n}\Big|\sum_{i=1}^n \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\}\Big| > \frac{\epsilon}{4}\right) &(69)
\end{aligned}$$

$$\leq S(\mathcal{F}, n) \max_{f \in \mathcal{F}(\mathbf{x}_1, \ldots, \mathbf{x}_n)} \Pr\left(\frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\} \Big| > \frac{\epsilon}{4} \right) \tag{70}$$

$$\leq S(\mathcal{F}, n) \sup_{f \in \mathcal{F}} \Pr\left(\frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\} \Big| > \frac{\epsilon}{4} \right). \tag{71}$$

Hence we have effectively brought the $\sup_f$ outside the probability, which makes controlling the probability much easier.

Step 4: Application of Hoeffding. This last step is also easy. Note that given the dataset $\mathcal{D}_n$, the random variables $\sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\}$ for $i = 1, \ldots, n$ are zero-mean, independent (but not necessarily identically distributed) and almost surely in $[-1, 1]$. Hence an application of Hoeffding yields

$$\Pr\left(\frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{x}_i) \neq y_i\} \Big| > \frac{\epsilon}{4} \right) \leq 2e^{-n\epsilon^2/32}. \tag{72}$$

Putting everything together, we see that

$$\Pr\left(\sup_{f \in \mathcal{F}} \frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} \Big| > \frac{\epsilon}{4} \right)$$

$$= \mathbb{E}\left[\mathbb{1}\left\{ \sup_{f \in \mathcal{F}} \frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} \Big| > \frac{\epsilon}{4} \right\} \right] \tag{73}$$

$$= \mathbb{E}\left[\mathbb{E}\left[\mathbb{1}\left\{ \sup_{f \in \mathcal{F}} \frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} \Big| > \frac{\epsilon}{4} \right\} \Big| \mathcal{D}_n \right] \right] \tag{74}$$

$$= \mathbb{E}\left[\Pr\left(\sup_{f \in \mathcal{F}} \frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} \Big| > \frac{\epsilon}{4} \Big| \mathcal{D}_n \right) \right] \tag{75}$$

$$\leq S(\mathcal{F}, n) \mathbb{E}\left[\sup_{f \in \mathcal{F}} \Pr\left(\frac{1}{n} \Big| \sum_{i=1}^{n} \sigma_i \mathbb{1}\{f(\mathbf{X}_i) \neq Y_i\} \Big| > \frac{\epsilon}{4} \Big| \mathcal{D}_n \right) \right] \tag{76}$$

$$\leq 2S(\mathcal{F}, n) \mathbb{E}\left[\sup_{f \in \mathcal{F}} e^{-n\epsilon^2/32} \right] \tag{77}$$

$$= 2S(\mathcal{F}, n) e^{-n\epsilon^2/32}. \tag{78}$$

Combining this calculation with Steps 1 and 2 yields the desired bound. □

3 Bayesian Hypothesis Testing

Let $P_{\mathbf{X}Y}$ be a joint distribution where $\mathbf{X} \in \mathcal{X}$ and $Y \in \mathcal{Y} = \{1, \ldots, m\}$. The set $\mathcal{X}$ is arbitrary. In a Bayesian hypothesis testing problem, the complete model therefore consists of the *a priori* probabilities

$$P_Y(y), \qquad y \in \mathcal{Y} \tag{79}$$

together with a characterization of the observed data under each hypothesis/class y, which takes the form of the conditional probability distributions

$$P_{\mathbf{X}|Y}(\cdot|y), \qquad y \in \mathcal{Y}. \tag{80}$$

Of course, a complete characterization of our knowledge of the correct hypothesis based on our observations is the set of *a posteriori* probabilities

$$P_{Y|\mathbf{X}}(y|\mathbf{x}), \qquad y \in \mathcal{Y}. \tag{81}$$

The distribution of possible values of Y is often referred to as our belief about the hypothesis. From this perspective, we can view the a priori probabilities as our prior belief, and view (81) as the revision of our belief based on having observed the data $\mathbf{x}$. The belief update is, of course, computed from the particular data $\mathbf{x}$ based on the model via Bayes' Rule:

$$P_{Y|\mathbf{X}}(y|\mathbf{x}) = \frac{P_{\mathbf{X}|Y}(\mathbf{x}|y)P_Y(y)}{P_{\mathbf{X}}(\mathbf{x})} = \frac{P_{\mathbf{X}|Y}(\mathbf{x}|y)P_Y(y)}{\sum_{y' \in \mathcal{Y}} P_{\mathbf{X}|Y}(\mathbf{x}|y')P_Y(y')}. \tag{82}$$

Now given a data sample $\mathbf{x} \in \mathcal{X}$, we would like to estimate what y is. Our decision rule can be encoded in a function $g : \mathcal{X} \to \mathcal{Y}$. This is equivalent to partitioning the space $\mathcal{X}$ into the following disjoint subsets.

$$\mathcal{X}_y := \{\mathbf{x} \in \mathcal{X} : g(\mathbf{x}) = y\}, \qquad y \in \mathcal{Y}. \tag{83}$$

Given g, we can consider the *probability of error*

$$\begin{aligned} \Pr(\text{error}) &= 1 - \Pr(\text{no error}) && (84)\\ &= 1 - \sum_{y \in \mathcal{Y}} P_Y(y) \int_{\mathcal{X}_y} P_{\mathbf{X}|Y}(\mathbf{x}|\mathbf{y})\, \mathrm{d}\mathbf{x}. && (85) \end{aligned}$$

Let's understand what this means. For Pr(no error), are averaging over Y using the prior distribution P_Y and conditioned on $\{Y = y\}$, we make the right decision if, based on $\mathbf{x}$, we declare that y is true, i.e., $\mathbf{x} \in \mathcal{X}_y$. To minimize the probability of error, it suffices to maximize Pr(no error). For this note that

$$\begin{aligned} \Pr(\text{no error}) &= \sum_{y \in \mathcal{Y}} P_Y(y) \int_{\mathcal{X}_y} P_{\mathbf{X}|Y}(\mathbf{x}|y)\, \mathrm{d}x && (86)\\ &= \int_{\mathcal{X}} P_{\mathbf{X}}(\mathbf{x}) P_{Y|X}(g(\mathbf{x})|\mathbf{x}) \mathrm{d}\mathbf{x}. && (87) \end{aligned}$$

Now since $P_X(\mathbf{x})$ is non-negative, to maximize Pr(no error), it is clear that we maximize $P_{Y|\mathbf{X}}(g(\mathbf{x})|\mathbf{x})$ (over g) for each value of $\mathbf{x}$. Hence, we can determine the optimal decision rule g on a point-by-point basis, i.e., $g(\mathbf{x})$ for each $\mathbf{x}$. It is also clear that we should choose $g(\mathbf{x})$ such that

$$g(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} P_{Y|\mathbf{X}}(y|\mathbf{x}). \tag{88}$$

This is exactly the *maximum a posteriori (MAP) rule*. Thus the MAP rule minimizes the probability of error.

MA4270 : Model Selection using Bayesian Score and Bayesian Information Criteria

Vincent Y. F. Tan

September 16, 2021

In this set of notes, we describe model selection via the Bayesian score and the Bayesian Information Criteria (BIC). We also give a proof sketch of the BIC. You should know Section 1.

We also provide a proof sketch of the Bayesian information criteria (BIC) [1]. You need to what the BIC is but its proof here in Section 2 and in the problem set are completely optional.

These notes are heavily adapted from Jaakkola's MIT notes.

1 The Bayesian Score and the Bayesian Information Criterion

So, suppose our model $\mathcal{F}$ takes a d-dimensional input $\mathbf{x}$ and maps it to a real valued output y (a distribution over y) according to:

$$P(y|\mathbf{x}, \boldsymbol{\theta}, \sigma^2) = N(y; \langle \boldsymbol{\theta}, \mathbf{x} \rangle, \sigma^2). \tag{1}$$

We omit the offset θ_0 and assume that the noise variance σ^2 is known. As usual, given a dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$, we can form the likelihood function

$$L(\mathcal{D}; \boldsymbol{\theta}) = \prod_{t=1}^{n} N(y_t; \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle, \sigma^2) = \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_t - \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle)^2}{2\sigma^2} \right). \tag{2}$$

We have previously used only the maximizing parameters

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \tag{3}$$

as estimates of the underlying parameter value (if $\mathbf{X}$ is full column rank). In Bayesian analysis, we are no longer satisfied with selecting a *single* linear regression function but would like to keep all of them, just weighted by their ability to explain the data, i.e., weighted by the corresponding likelihood $L(\mathcal{D}; \boldsymbol{\theta})$. From this perspective, our knowledge about the parameter $\boldsymbol{\theta}$ after seeing the data is defined by the posterior distribution $P(\boldsymbol{\theta}; \mathcal{D})$ proportional to the *likelihood*

$$P(\boldsymbol{\theta}; \mathcal{D}) \propto L(\mathcal{D}; \boldsymbol{\theta}). \tag{4}$$

We impose a prior belief on the parameters of the model $\boldsymbol{\theta}$ via a *prior distribution*. This distribution captures what we believe about the parameter values before seeing any data. Similarly to the regularization penalty, we will typically choose the prior to prefer small parameter values, e.g.,

$$P(\boldsymbol{\theta}) = N(\boldsymbol{\theta}; \mathbf{0}, \sigma_{\mathrm{p}}^2 \mathbf{I}) \tag{5}$$

which is a zero mean spherical Gaussian (same variance in all directions). The smaller σ_{p}^2 is, the smaller values of $\boldsymbol{\theta}$ we prefer prior to seeing the data. The *posterior* distribution is proportional to the *prior* distribution times the *likelihood*:

$$P(\boldsymbol{\theta}; \mathcal{D}) \propto L(\mathcal{D}; \boldsymbol{\theta}) P(\boldsymbol{\theta}), \tag{6}$$

The name *prior* means *before*; it is the belief we have about the parameters $\boldsymbol{\theta}$ before seeing the data. The term *likelihood* denotes how likely we are to see the data $\mathcal{D}$ given the current set of parameters $\boldsymbol{\theta}$. Finally, the term posterior means *after*; it is the belief we have about the parameters $\boldsymbol{\theta}$ after seeing the data, merging both our prior belief and the effect of the data. The normalizing constant of (6) is known as the *model evidence*, *marginal likelihood*, or *Bayesian score*

$$P(\mathcal{D}|\mathcal{F}) = \int L(\mathcal{D};\boldsymbol{\theta}')P(\boldsymbol{\theta}')\,\mathrm{d}\boldsymbol{\theta}'. \tag{7}$$

This depends on the model $\mathcal{F}$ and the data $\mathcal{D}$ but not specific parameter values.

Putting everything together, we get the *fundamental equation of Bayesian inference*

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{L(\mathcal{D};\boldsymbol{\theta})P(\boldsymbol{\theta})}{\int L(\mathcal{D};\boldsymbol{\theta}')P(\boldsymbol{\theta}')\,\mathrm{d}\boldsymbol{\theta}'}, \qquad \text{or} \qquad \text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{model evidence}}. \tag{8}$$

In our regression context, we can actually evaluate the marginal likelihood in closed form:

$$\begin{aligned}\log P(\mathcal{D}|\mathcal{F}) = &-\frac{n}{2}\log(\sigma^2) + \frac{d}{2}\log\lambda - \frac{1}{2}\log|\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}| \\ &- \frac{1}{2\sigma^2}\left(\|\mathbf{y}\|^2 - \mathbf{y}\mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}\right),\end{aligned} \tag{9}$$

where $\lambda = \sigma^2/\sigma_\mathrm{p}^2$ (ratio of noise to prior variance), the design matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ and $\mathbf{y} = [y_1, \ldots, y_n]^\top$.

Exercise 1. *Prove* (9). *You might want to use the fact that the Gaussian distribution is a conjugate prior to the Gaussian with unknown mean.*

In our context the posterior is also Gaussian $P(\boldsymbol{\theta};\mathcal{D}) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ given by

$$\boldsymbol{\mu} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \tag{10}$$

$$\boldsymbol{\Sigma} = \sigma^2(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}. \tag{11}$$

Exercise 2. *Prove* (10) *and* (11) *by combining* (2) *and* (5) *and completing the square in the exponent.*

Note that (10) is exactly the same as the point estimate of $\boldsymbol{\theta}$ when we consider the penalized (regularized) log-likelihood; see Lecture notes 5. However, in this approach, we have the entire distribution of $\boldsymbol{\theta}$ including its covariance $\boldsymbol{\Sigma}$. Indeed, the posterior mean of the parameters is exactly the parameter estimate we derived earlier using penalized log-likelihood with the same prior. This is not an accident when all the distributions involved are indeed Gaussians. It is also worth pointing out that $P(\boldsymbol{\theta}|\mathcal{D})$ is very different from the normal distribution over $\hat{\boldsymbol{\theta}}$ we derived earlier when assuming that the responses $\mathbf{y}$ came from a linear model of the same type. We have made no such assumption here and the distribution $P(\boldsymbol{\theta}|\mathcal{D})$ is defined on the basis of the single observed $\mathbf{y}$.

In Bayesian analysis the prediction of $\mathbf{y}$ in response to a new $\mathbf{x}$ would be given by weighting predictions based on individual $\boldsymbol{\theta}$'s by the posterior distribution:

$$P(y|\mathbf{x},\mathcal{D}) = \int P(y|\mathbf{x},\boldsymbol{\theta})P(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta}. \tag{12}$$

So what is the model selection problem in this context? We will try to select different regression models, specified by different feature mappings $\mathbf{x} \mapsto \boldsymbol{\phi}(\mathbf{x})$. Let's consider then two regression models specified by linear $\boldsymbol{\phi}^{(1)}(\mathbf{x})$ and quadratic $\boldsymbol{\phi}^{(2)}(\mathbf{x})$ feature mappings. The models we compare are therefore

$$\mathcal{F}_1: \qquad P(y|\mathbf{x},\boldsymbol{\theta},\sigma^2) = N(y;\langle\boldsymbol{\theta},\boldsymbol{\phi}^{(1)}(\mathbf{x})\rangle,\sigma^2) \qquad \boldsymbol{\theta}\in\mathbb{R}^{d_1}, \qquad P(\boldsymbol{\theta}|\mathcal{F}_1) \tag{13}$$

$$\mathcal{F}_2: \qquad P(y|\mathbf{x},\boldsymbol{\theta},\sigma^2) = N(y;\langle\boldsymbol{\theta},\boldsymbol{\phi}^{(2)}(\mathbf{x})\rangle,\sigma^2) \qquad \boldsymbol{\theta}\in\mathbb{R}^{d_2}, \qquad P(\boldsymbol{\theta}|\mathcal{F}_2) \tag{14}$$

Note that $\boldsymbol{\theta}$ is of different dimension in the two models and thus the prior distributions over the parameters, $P(\boldsymbol{\theta}|\mathcal{F}_1)$ and $P(\boldsymbol{\theta}|\mathcal{F}_2)$, will have to be different.

So, how do we select between the two competing models? We simply select the one whose marginal likelihood is larger. In other words, after seeing data $\mathcal{D}$ we favor model $\mathcal{F}_1$ if

$$P(\mathcal{D}|\mathcal{F}_1) > P(\mathcal{D}|\mathcal{F}_2) \tag{15}$$

and we favor $\mathcal{F}_2$ if the inequality above is reversed. For non-conjugate models, the evaluation of the marginal likelihood is intractable and we typically resort to asymptotic approximations. One of them such approximations is the famous *Bayesian information criterion* or BIC, often used due to its simplicity. The criterion is to maximize

$$\text{BIC}(\mathcal{F}) = \ell(\mathcal{D}; \hat{\boldsymbol{\theta}}, \mathcal{F}) - \frac{d_{\mathcal{F}}}{2} \log n \tag{16}$$

where

1. $\ell(\mathcal{D}; \hat{\boldsymbol{\theta}}, \mathcal{F})$ is the maximized log-likelihood under $\mathcal{F}$

$$\ell(\mathcal{D}; \hat{\boldsymbol{\theta}}, \mathcal{F}) = \max_{\boldsymbol{\theta}} \log L(\mathcal{D}; \boldsymbol{\theta}, \mathcal{F}), \tag{17}$$

2. $d_{\mathcal{F}}$ is the number of independent parameters in the model; and

3. n is the number of data samples.

Under certain regularity conditions, the difference between the BIC and the Bayesian score will converge to 0 in as $n \to \infty$. The Bayesian score is typically difficult to evaluate in practice and BIC serves as a simple tractable alternative. Similarly to the Bayesian score (marginal likelihood), we would select the model with the largest BIC score. Notice that if $d_{\mathcal{F}}$, a measure of the complexity of a model, penalizes the data-dependent term $\ell(\mathcal{D}; \hat{\boldsymbol{\theta}}, \mathcal{F})$. This says that we have a preference for simpler models.

Exercise 3. *The maximized log-likelihoods for a simple (e.g., kernel with linear features) and more complex (e.g., kernel with quadratic features) are* $\ell_1 = -100$ *and* $\ell_2 = -50$ *respectively. Say the input data* $\mathbf{x}$ *has* $d = 2$ *dimensions. Then the linear model has* $d_1 = d + 1$ *parameters and the quadratic model has* $\binom{d+2}{2} = \frac{1}{2}(d+2)(d+1)$ *parameters.*

1. *Why is* $\ell_1 \leq \ell_2$ *for nested models* $\mathcal{F}_1 \subset \mathcal{F}_2$*?*

2. *Based on the BIC, find the range of sample sizes* n *such that we would prefer the more complex model over the simpler one.*

If correctly done, you will see that for small sample sizes, we prefer the complex model over the simpler one (and vice versa).

2 Proof Sketch of the Bayesian Information Criterion (Optional)

Suppose we have k model classes $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$ where each model is a set of densities

$$\mathcal{F}_j := \{p(\mathbf{z}; \boldsymbol{\theta}_j) : \boldsymbol{\theta}_j \in \boldsymbol{\Theta}_j\}. \tag{18}$$

We obtain data $Z_1, Z_2, \ldots, Z_n$ from some density p^*. We do not necessarily have to assume that $p^* \in \mathcal{F}_j$ for any $j = 1, \ldots, k$.

We recall that the BIC [1] is an approximation of the model evidence $P(\mathcal{D}_n|\mathcal{F}_i)$, where $\mathcal{D}_n = \{\mathbf{z}_t = (\mathbf{x}_t, y_t)\}_{t=1}^n$ is the dataset and $\mathcal{F}_i$ is a class of functions. We will show how to approximate this in the calculations below. First, we recall the definition of the model evidence as

$$P(\mathcal{D}_n|\mathcal{F}_i) = \int L(\mathcal{D}_n; \boldsymbol{\theta}_i) P(\boldsymbol{\theta}_i) \, \mathrm{d}\boldsymbol{\theta}_i = \int \exp\Big(\log\big(L(\mathcal{D}_n; \boldsymbol{\theta}_i) P(\boldsymbol{\theta}_i)\big)\Big) \, \mathrm{d}\boldsymbol{\theta}_i \tag{19}$$

where $L(\boldsymbol{\theta}_i; \mathcal{D}_n)$ is the likelihood and $P(\boldsymbol{\theta}_i)$ is the prior under the function class $\mathcal{F}_i$. We drop the dependence of these terms on $\mathcal{F}_i$ for the sake of brevity.

Define the MAP estimate as

$$\tilde{\boldsymbol{\theta}}_i = \arg\max_{\boldsymbol{\theta}_i} \log\big(L(\mathcal{D}_n; \boldsymbol{\theta}_i) P(\boldsymbol{\theta}_i)\big). \tag{20}$$

Then we can use a Taylor series expansion to write

$$\underbrace{\log\big(L(\mathcal{D}_n; \boldsymbol{\theta}_i) P(\boldsymbol{\theta}_i)\big)}_{Q} \approx \log\big(L(\mathcal{D}_n; \tilde{\boldsymbol{\theta}}_i) P(\tilde{\boldsymbol{\theta}}_i)\big) + (\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i)^T \nabla_{\boldsymbol{\theta}_i} Q\big|_{\tilde{\boldsymbol{\theta}}_i} + \frac{1}{2}(\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i)^T H_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i) \tag{21}$$

where $\mathbf{H}_{\boldsymbol{\theta}_i}$ is a $\dim(\boldsymbol{\theta}_i) \times \dim(\boldsymbol{\theta}_i)$ matrix consisting of the second-order partial derivatives of Q. Since Q attains its maximum at $\tilde{\boldsymbol{\theta}}_i$, we know that $\nabla_{\boldsymbol{\theta}_i} Q\big|_{\tilde{\boldsymbol{\theta}}_i} = \mathbf{0}$ and $\mathbf{H}_{\boldsymbol{\theta}_i}$ is negative definite. Let us denote $\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i} = -\mathbf{H}_{\boldsymbol{\theta}_i}$, which is positive definite. Then we can write

$$P(\mathcal{D}_n|\mathcal{F}_i) \approx \int \exp\left\{Q\big|_{\tilde{\boldsymbol{\theta}}_i} - \frac{1}{2}(\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i)^T \tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i)\right\} \mathrm{d}\boldsymbol{\theta}_i \tag{22}$$

$$= \exp\left\{Q\big|_{\tilde{\boldsymbol{\theta}}_i}\right\} \int \exp\left\{-\frac{1}{2}(\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i)^T \tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i - \tilde{\boldsymbol{\theta}}_i)\right\} \mathrm{d}\boldsymbol{\theta}_i \tag{23}$$

$$= L(\mathcal{D}_n; \tilde{\boldsymbol{\theta}}_i) P(\tilde{\boldsymbol{\theta}}_i) \frac{(2\pi)^{\dim(\boldsymbol{\theta}_i)/2}}{|\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}|^{1/2}}, \tag{24}$$

where $|\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}|$ is the determinant of the matrix $\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}$. The last step follows from the fact that

$$\int_{\mathbb{R}^m} \frac{1}{\sqrt{(2\pi)^m |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \mathrm{d}\mathbf{x} = 1. \tag{25}$$

Taking the log of (24), we obtain

$$\log P(\mathcal{D}_n|\mathcal{F}_i) = \log\big(L(\mathcal{D}_n; \tilde{\boldsymbol{\theta}}_i) P(\tilde{\boldsymbol{\theta}}_i)\big) + \frac{\dim(\boldsymbol{\theta}_i)}{2} + \log(2\pi) - \frac{1}{2}\log|\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}|. \tag{26}$$

The likelihood $L(\boldsymbol{\theta}_i; \mathcal{D}_n)$ attains its maximum at the maximum likelihood estimate $\boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i$. If we further set $P(\boldsymbol{\theta}_i) = \text{const}$, the so-called *flat prior*, then each element of the matrix $\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}$ can be expressed as

$$[\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}]_{kl} = -\frac{\partial^2 \log L(\boldsymbol{\theta}_i; \mathcal{D}_n)}{\partial\theta_{ik}\partial\theta_{il}}\Big|_{\boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i} \tag{27}$$

$$= -\frac{\partial^2 \left(\sum_{t=1}^n \log L(\boldsymbol{\theta}_i; Z_t)\right)}{\partial\theta_{ik}\partial\theta_{il}}\Big|_{\boldsymbol{\theta}_i = \hat{\boldsymbol{\theta}}_i}. \tag{28}$$

Now by the law of large numbers,

$$\frac{1}{n}[\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}]_{kl} \to [\mathbf{I}_i]_{kl} = -\frac{\partial^2 \mathbb{E}[\log L(\boldsymbol{\theta}_i; Z_1)]}{\partial\theta_{ik}\partial\theta_{il}}. \tag{29}$$

Hence, by a standard property of determinants ($|a\mathbf{A}| = a^m|\mathbf{A}|$ where m is the dimension of the square matrix $\mathbf{A}$),

$$|\tilde{\mathbf{H}}_{\boldsymbol{\theta}_i}| = n^{\dim(\boldsymbol{\theta}_i)|}|\mathbf{I}_{\boldsymbol{\theta}_i}| \tag{30}$$

where $\mathbf{I}_{\boldsymbol{\theta}_i}$ is the Fisher information matrix for a single data point Z_1. Plugging the result back into (26), we obtain

$$\log P(\mathcal{D}_n|\mathcal{F}_i) = \log\big(L(\mathcal{D}_n; \hat{\boldsymbol{\theta}}_i) P(\hat{\boldsymbol{\theta}}_i)\big) + \frac{\dim(\boldsymbol{\theta}_i)}{2} + \log(2\pi) - \frac{\dim(\boldsymbol{\theta}_i)}{2}\log n - \frac{1}{2}\log|\mathbf{I}_{\boldsymbol{\theta}_i}|. \tag{31}$$

For large n, keeping the terms involving n and ignoring the rest, we find that

$$\log P(\mathcal{D}_n|\mathcal{F}_i) = \log L(\hat{\boldsymbol{\theta}}_i; \mathcal{D}_n) - \frac{\dim(\boldsymbol{\theta}_i)}{2} \log n \tag{32}$$

which is precisely the BIC.

Remark 1. *The above way of approximating the integral in* (19) *via the second-order Taylor expansion is known as* Laplace's method.

References

[1] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

MA4270 : Basics of Convex Optimization

Vincent Y. F. Tan

September 6, 2021

Very soon, we will need to use basic ideas from (duality in) convex optimization so it's good to review the basics. For more details, you may wish to refer to [1] or [2]. Throughout, for two integers a, b, we use $[a:b]$ to denote the set of integers $\{a, a+1, \ldots, b\}$. You should know everything here particularly, Examples 5 and 7.

These notes are adapted in part from El Gamal and Kim [3].

1 Basics of Convex Optimization

An optimization problem

$$\min_{\mathbf{x}} \; g_0(\mathbf{x}), \quad \text{s.t.} \quad g_j(\mathbf{x}) \leq 0, \; \forall\, j \in [1:k], \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

with variables $\mathbf{x} \in \mathbb{R}^n$ is *convex* if the functions $g_j, j \in [0:k]$ are convex. We denote by

$$\mathscr{D} := \{\mathbf{x} \in \mathbb{R}^n : g_j(\mathbf{x}) \leq 0, j \in [1:k], \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{2}$$

the set of feasible points (the domain of the optimization problem). The convex optimization problem is said to be feasible if $\mathscr{D} \neq \emptyset$. The optimal value of the problem is denoted by $p^* = \inf\{g_0(\mathbf{x}) : \mathbf{x} \in \mathscr{D}\}$ (or $-\infty$ if the problem is infeasible). Any $\mathbf{x}$ that attains the infimum is said to be optimal and is denoted by $\mathbf{x}^*$.

Example 1. *Linear Program:*

$$\min_{\mathbf{x}} \; \mathbf{c}^\top \mathbf{x}, \quad s.t. \quad \mathbf{x} \geq \mathbf{0}, \quad \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{3}$$

Example 2. *Differential entropy maximization under correlation constraint*

$$\max \; \log\det(\mathbf{K}) \quad s.t. \quad K_{j,j+k} = a_k, \; \forall\, k \in [1:l]. \tag{4}$$

Note that this problem is a special case of matrix determinant maximization (max-det) with linear matrix inequalities

We define the *Lagrangian* associated with a feasible optimization problem in (1)

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = g_0(\mathbf{x}) + \sum_{j=1}^{k} \lambda_j g_j(x) + \boldsymbol{\nu}^\top (\mathbf{A}\mathbf{x} - \mathbf{b}). \tag{5}$$

We further define the *Lagrangian dual function* (or *dual function* in short) as

$$\phi(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}). \tag{6}$$

It can be easily seen that for any $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ with $\lambda_j \geq 0$ for all j and any feasible $\mathbf{x}$,

$$\phi(\boldsymbol{\lambda}, \boldsymbol{\nu}) \leq g_o(\mathbf{x}). \tag{7}$$

This leads to the (Lagrange) *dual problem*

$$\max\ \phi(\boldsymbol{\lambda},\boldsymbol{\nu}) \quad \text{s.t.} \quad \lambda_j \geq 0\ \forall\, j \in [1:k] \tag{8}$$

Note that $\phi(\boldsymbol{\lambda},\boldsymbol{\nu})$ is concave and hence the dual problem is convex (regardless of whether the primal problem is convex or not). The optimal value of the dual problem is denoted by d^* and the dual optimal point is denoted by $(\boldsymbol{\lambda}^*,\boldsymbol{\nu}^*)$.

The original optimization problem, its feasible set, and optimal value are sometimes referred to as the primal problem, primal feasible set, and primal optimal value, respectively.

Example 3. *Consider the LP discussed above with variables* $\mathbf{x}$*. The Lagrangian is*

$$L(\mathbf{x},\boldsymbol{\lambda},\boldsymbol{\nu}) = \mathbf{c}^\top\mathbf{x} - \sum_{j=1}^{n}\lambda_j x_j + \boldsymbol{\nu}^\top(\mathbf{A}\mathbf{x}-\mathbf{b}) = -b^\top\boldsymbol{\nu} + (\mathbf{c}+\mathbf{A}^\top\boldsymbol{\nu}-\boldsymbol{\lambda})^\top\mathbf{x} \tag{9}$$

and the dual function is

$$\phi(\boldsymbol{\lambda},\boldsymbol{\nu}) = -\mathbf{b}^\top\boldsymbol{\nu} + \inf_{\mathbf{x}}(\mathbf{c}+\mathbf{A}^\top\boldsymbol{\nu}-\boldsymbol{\lambda})^\top x \tag{10}$$

$$= \begin{cases} -\mathbf{b}^\top\boldsymbol{\nu} & \text{if } \mathbf{A}^\top\boldsymbol{\nu}-\boldsymbol{\lambda}+\mathbf{c}=\mathbf{0} \\ -\infty & \text{else} \end{cases} \tag{11}$$

Hence, the dual problem is

$$\max\ -\mathbf{b}^\top\boldsymbol{\nu} \quad \text{s.t.} \quad \mathbf{A}^\top\boldsymbol{\nu}-\boldsymbol{\lambda}+\mathbf{c}=0, \quad \lambda_j \geq 0, \quad \forall\, j \tag{12}$$

which is another LP.

Example 4. *Consider the following quadratic program with* $\mathbf{H}$ *being a positive definite symmetric matrix and* $\mathbf{x}$ *being the decision variable.*

$$\min_{\mathbf{x}}\ \frac{1}{2}\mathbf{x}^\top\mathbf{H}\mathbf{x} \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b}. \tag{13}$$

The Lagrangian is

$$L(\mathbf{x},\boldsymbol{\lambda}) = \frac{1}{2}\mathbf{x}^\top\mathbf{H}\mathbf{x} + \boldsymbol{\lambda}^\top(\mathbf{b}-\mathbf{A}\mathbf{x}). \tag{14}$$

The dual function is

$$\phi(\boldsymbol{\lambda}) = \inf_{\mathbf{x}} L(\mathbf{x},\boldsymbol{\lambda}) = -\frac{1}{2}\boldsymbol{\lambda}^\top\mathbf{A}\mathbf{H}^{-1}\mathbf{A}^\top\boldsymbol{\lambda} + \boldsymbol{\lambda}^\top\mathbf{b} \tag{15}$$

Thus, the dual problem (in variables $\boldsymbol{\lambda},\mathbf{m}$*) is*

$$\max\ \boldsymbol{\lambda}^\top\mathbf{b} - \frac{1}{2}\mathbf{m}^\top\mathbf{H}\mathbf{m} \quad \text{s.t.} \quad \boldsymbol{\lambda} \geq \mathbf{0}, \mathbf{A}^\top\boldsymbol{\lambda} = \mathbf{m}, \tag{16}$$

which is another quadratic program.

Example 5. *This is the SVM example without slack variables but with offset. The primal with variables* $(\boldsymbol{\theta},\boldsymbol{\theta}_0)$ *is*

$$\min\ \frac{1}{2}\|\boldsymbol{\theta}\|^2 \quad \text{s.t.} \quad y_t(\langle\mathbf{x}_t,\boldsymbol{\theta}\rangle+\theta_0) \geq 1 \quad \forall\, t \tag{17}$$

Please show that the dual problem in variables $\alpha \in \mathbb{R}^n$ *is*

$$\max_{\boldsymbol{\alpha}\geq\mathbf{0}}\ \sum_t \alpha_t - \frac{1}{2}\sum_{t,s}\alpha_t\alpha_s y_t y_s x_t x_s \quad \text{s.t.} \quad \sum_t \alpha_t y_t = 0. \tag{18}$$

Note that the primal SVM problem is a quadratic program and so is the dual. However, the dimension of the primal program is d *and the dual is* n*. In the kernelized case,* d *can be much larger than* n*;* d *could potentially be infinite for the radial basis kernel. In this scenario, we would prefer to solve the dual program.*

By the definition of the dual function and dual problem, we already know that the lower bound on the primal optimal value is $d^* \le p^*$; this is known as *weak duality*. When this bound is tight (i.e., $d^* = p^*$), we say that *strong duality* holds. One simple sufficient condition for strong duality to hold is the following.

Theorem 1 (Slater's condition)**.** *If the primal problem is convex and there exists a feasible* $\mathbf{x}$ *in the relative interior*[1] *of* $\mathscr{D}$*, i.e.,* $g_j(\mathbf{x}) < 0$ *for all* $j \in [1:k]$*, and* $\mathbf{A}\mathbf{x} = \mathbf{b}$*, then strong duality holds.*

If strong duality holds (say, Slater's condition holds), then

$$g_0(\mathbf{x}^*) = \phi(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*) \tag{19}$$

$$\le \inf_{\mathbf{x}} \left\{ g_0(\mathbf{x}) + \sum_{j=1}^{k} \lambda_j^* g_j(\mathbf{x}) + (\boldsymbol{\nu}^*)^\top (\mathbf{A}\mathbf{x} - \mathbf{b}) \right\} \tag{20}$$

$$\le g_0(\mathbf{x}^*) + \sum_{j=1}^{k} \boldsymbol{\lambda}_j^* g_j(\mathbf{x}^*) + (\boldsymbol{\nu}^*)^\top (\mathbf{A}\mathbf{x}^* - \mathbf{b}) \tag{21}$$

$$\le g_0(\mathbf{x}^*) \tag{22}$$

Following the equality conditions, we obtain the following sufficient and necessary condition, commonly referred to as the Karush-Kuhn-Tucker (KKT) condition, for the primal optimal point $\mathbf{x}^*$ and dual optimal point $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$.

1. $\mathbf{x}^*$ minimizes $L(\mathbf{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$. This condition can be easily checked if $L(\mathbf{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ is differentiable.
2. Complementary slackness: $\lambda_j^* g_j(\mathbf{x}^*) = 0$ for all $j \in [1:k]$.

Example 6. *Consider the determinant maximization problem*

$$\max_{\mathbf{X}} \ \log\det(\mathbf{X} + \mathbf{K}) \quad s.t. \quad \mathbf{X} \succeq 0,\ \mathrm{tr}(\mathbf{X}) \le P \tag{23}$$

where K *is a given positive definite matrix.*

Noting that $\mathbf{X} \succeq 0$ *iff* $\mathrm{tr}(\mathbf{Y}\mathbf{X}) \ge 0$ *for every* $\mathbf{Y} \succeq 0$*, we form the Lagrangian*

$$L(\mathbf{X}, \mathbf{Y}, \lambda) = \log\det(\mathbf{X} + \mathbf{K}) + \mathrm{tr}(\mathbf{Y}\mathbf{X}) - \lambda(\mathrm{tr}(\mathbf{X}) - P). \tag{24}$$

Since the domain $\mathscr{D}$ *has a nonempty interior (for example,* $\mathbf{X} = (P/2)\mathbf{I}$ *is feasible), Slater's condition is satisfied and strong duality holds. Hence the KKT condition characterizes the optimal* $\mathbf{X}^*$ *and* $(\mathbf{Y}^*, \lambda^*)$*:*

1. $\mathbf{X}^*$ *maximizes the Lagrangian. Since the derivative of* $\log\det(\mathbf{Y})$ *is* $\mathbf{Y}^{-1}$ *for* $\mathbf{Y} \succ 0$ *and the derivative of* $\mathrm{tr}(\mathbf{A}\mathbf{Y})$ *is* $\mathbf{A}$*,*

$$\frac{\partial}{\partial \mathbf{X}} L(\mathbf{X}, \mathbf{Y}^*, \lambda^*) = (\mathbf{X} + \mathbf{K})^{-1} + \mathbf{Y}^* - \lambda \mathbf{I} = \mathbf{0}. \tag{25}$$

2. *Complementary Slackness*

$$\mathrm{tr}(\mathbf{Y}^*\mathbf{X}^*) = 0 \tag{26}$$

$$\lambda^*(\mathrm{tr}(\mathbf{X}^*) - P) = 0. \tag{27}$$

Example 7. *For the SVM in Example 5, we note that strong duality holds iff there exists a vector* $(\boldsymbol{\theta}^*, \theta_0^*)$ *satisfying*

$$y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}^* \rangle + \theta_0^*) > 1, \quad \forall\, t \in [1:n]. \tag{28}$$

This means that the dataset is linearly separable with positive margin. If we modify the SVM problem in Example 5 such that we allow for a slack variables $\xi_t \ge 0, t \in [1:n]$*, then Slater's condition holds trivially because for any dataset (linearly separable or not) we can always find* $(\boldsymbol{\theta}^*, \theta_0^*, \boldsymbol{\xi}^*) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^n$ *satisfying*

$$y_t(\langle \mathbf{x}_t, \boldsymbol{\theta}^* \rangle + \theta_0^*) > 1 - \xi_t^*, \quad \forall\, t \in [1:n]. \tag{29}$$

We just have to set the slack variables large enough if some point is not classified correctly given $(\boldsymbol{\theta}^*, \theta_0^*)$*.*

[1] The *relative interior* of a set S, denoted as $\mathbf{ri}(S)$, is defined as its interior within the affine hull of S. In other words, $\mathbf{ri}(S) = \{\mathbf{x} \in S : \exists\, \epsilon > 0, N_\epsilon(\mathbf{x}) \cap \mathrm{aff}(S) \subseteq S\}$ where $\mathrm{aff}(S)$ is the affine hull of S and $N_\epsilon(\mathbf{x})$ is a ball of radius ϵ centered at $\mathbf{x}$.

References

[1] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] A. El Gamal and Y.-H. Kim. *Network Information Theory*. Cambridge University Press, Cambridge, U.K., 2012.