CS229 Supplemental Lecture notes

John Duchi

1 Binary classification

In **binary classification** problems, the target y can take on at only two values. In this set of notes, we show how to model this problem by letting $y \in \{-1, +1\}$, where we say that y is a 1 if the example is a member of the positive class and $y = -1$ if the example is a member of the negative class. We assume, as usual, that we have input features $x \in \mathbb{R}^n$.

As in our standard approach to supervised learning problems, we first pick a representation for our hypothesis class (what we are trying to learn), and after that we pick a loss function that we will minimize. In binary classification problems, it is often convenient to use a hypothesis class of the form $h_\theta(x) = \theta^T x$, and, when presented with a new example x, we classify it as positive or negative depending on the sign of $\theta^T x$, that is, our predicted label is

$$\text{sign}(h_\theta(x)) = \text{sign}(\theta^T x) \quad \text{where} \quad \text{sign}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \\ -1 & \text{if } t < 0. \end{cases}$$

In a binary classification problem, then, the hypothesis h_θ with parameter vector θ classifies a particular example (x, y) correctly if

$$\text{sign}(\theta^T x) = y \quad \text{or equivalently} \quad y\theta^T x > 0. \tag{1}$$

The quantity $y\theta^T x$ in expression (1) is a very important quantity in binary classification, important enough that we call the value

$$yx^T\theta$$

the *margin* for the example (x, y). Often, though not always, one interprets the value $h_\theta(x) = x^T\theta$ as a measure of the confidence that the parameter

vector θ assigns to labels for the point x: if $x^T\theta$ is very negative (or very positive), then we more strongly believe the label y is negative (or positive).

Now that we have chosen a representation for our data, we must choose a loss function. Intuitively, we would like to choose some loss function so that for our training data $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$, the θ chosen makes the margin $y^{(i)}\theta^T x^{(i)}$ very large for each training example. Let us fix a hypothetical example (x, y), let $z = yx^T\theta$ denote the margin, and let $\varphi : \mathbb{R} \to \mathbb{R}$ be the loss function—that is, the loss for the example (x, y) with margin $z = yx^T\theta$ is $\varphi(z) = \varphi(yx^T\theta)$. For any particular loss function, the empirical risk that we minimize is then

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \varphi(y^{(i)}\theta^T x^{(i)}). \tag{2}$$

Consider our desired behavior: we wish to have $y^{(i)}\theta^T x^{(i)}$ positive for each training example $i = 1, \ldots, m$, and we should penalize those θ for which $y^{(i)}\theta^T x^{(i)} < 0$ frequently in the training data. Thus, an intuitive choice for our loss would be one with $\varphi(z)$ small if $z > 0$ (the margin is positive), while $\varphi(z)$ is large if $z < 0$ (the margin is negative). Perhaps the most natural such loss is the *zero-one* loss, given by

$$\varphi_{\rm zo}(z) = \begin{cases} 1 & \text{if } z \le 0 \\ 0 & \text{if } z > 0. \end{cases}$$

In this case, the risk $J(\theta)$ is simply the average number of mistakes—misclassifications—the parameter θ makes on the training data. Unfortunately, the loss $\varphi_{\rm zo}$ is discontinuous, non-convex (why this matters is a bit beyond the scope of the course), and perhaps even more vexingly, NP-hard to minimize. So we prefer to choose losses that have the shape given in Figure 1. That is, we will essentially always use losses that satisfy

$$\varphi(z) \to 0 \;\; \text{as } z \to \infty, \quad \text{while} \quad \varphi(z) \to \infty \;\; \text{as } z \to -\infty.$$

As a few different examples, here are three loss functions that we will see either now or later in the class, all of which are commonly used in machine learning.

(i) The *logistic loss* uses

$$\varphi_{\rm logistic}(z) = \log(1 + e^{-z})$$

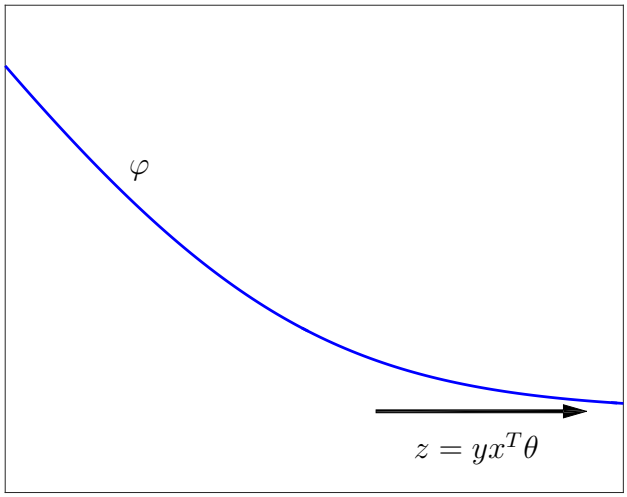

Figure 1: The rough shape of loss we desire: the loss is convex and continuous, and tends to zero as the margin $z = yx^T\theta \to \infty$.

(ii) The *hinge loss* uses

$$\varphi_{\text{hinge}}(z) = [1 - z]_+ = \max\{1 - z, 0\}$$

(iii) The *exponential loss* uses

$$\varphi_{\exp}(z) = e^{-z}.$$

In Figure 2, we plot each of these losses against the margin $z = yx^T\theta$, noting that each goes to zero as the margin grows, and each tends to $+\infty$ as the margin becomes negative. The different loss functions lead to different machine learning procedures; in particular, the logistic loss $\varphi_{\text{logistic}}$ is logistic regression, the hinge loss φ_{hinge} gives rise to so-called *support vector machines*, and the exponential loss gives rise to the classical version of *boosting*, both of which we will explore in more depth later in the class.

2 Logistic regression

With this general background in place, we now we give a complementary view of logistic regression to that in Andrew Ng's lecture notes. When we

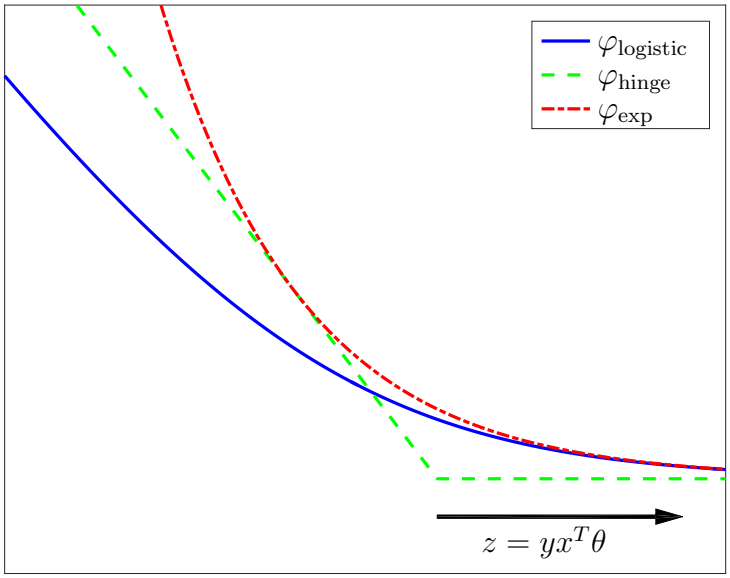

Figure 2: The three margin-based loss functions logistic loss, hinge loss, and exponential loss.

use binary labels $y \in \{-1, 1\}$, it is possible to write logistic regression more compactly. In particular, we use the logistic loss

$$\varphi_{\text{logistic}}(yx^T\theta) = \log\left(1 + \exp(-yx^T\theta)\right),$$

and the *logistic regression* algorithm corresponds to choosing θ that minimizes

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \varphi_{\text{logistic}}(y^{(i)}\theta^T x^{(i)}) = \frac{1}{m}\sum_{i=1}^{m} \log\left(1 + \exp(-y^{(i)}\theta^T x^{(i)})\right). \quad (3)$$

Roughly, we hope that choosing θ to minimize the average logistic loss will yield a θ for which $y^{(i)}\theta^T x^{(i)} > 0$ for most (or even all!) of the training examples.

2.1 Probabilistic intrepretation

Similar to the linear regression (least-squares) case, it is possible to give a probabilistic interpretation of logistic regression. To do this, we define the

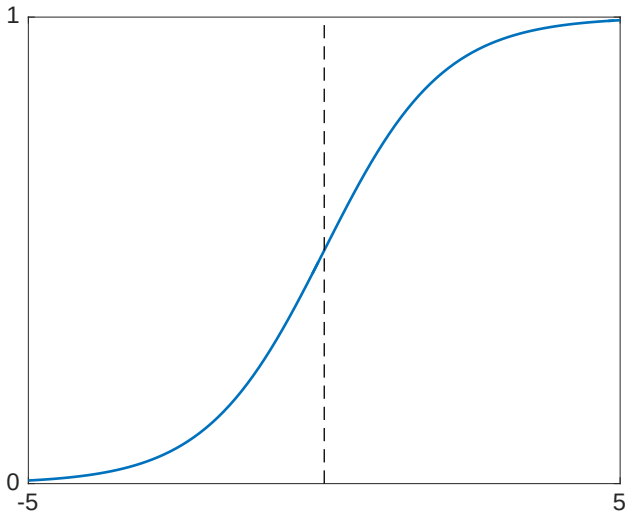

Figure 3: Sigmoid function

sigmoid function (also often called the *logistic function*)

$$g(z) = \frac{1}{1+e^{-z}},$$

which is plotted in Fig. 3. In particular, the sigmoid function satisfies

$$g(z) + g(-z) = \frac{1}{1+e^{-z}} + \frac{1}{1+e^{z}} = \frac{e^{z}}{1+e^{z}} + \frac{1}{1+e^{z}} = 1,$$

so we can use it to define a probability model for binary classification. In particular, for $y \in \{-1, 1\}$, we define the *logistic model* for classification as

$$p(Y = y \mid x; \theta) = g(yx^T\theta) = \frac{1}{1+e^{-yx^T\theta}}. \tag{4}$$

For intepretation, we see that if the margin $yx^T\theta$ is large—bigger than, say, 5 or so—then $p(Y = y \mid x; \theta) = g(yx^T\theta) \approx 1$, that is, we assign nearly probability 1 to the event that the label is y. Conversely, if $yx^T\theta$ is quite negative, then $p(Y = y \mid x; \theta) \approx 0$.

By redefining our hypothesis class as

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}},$$

then we see that the likelihood of the training data is

$$L(\theta) = \prod_{i=1}^{m} p(Y = y^{(i)} \mid x^{(i)}; \theta) = \prod_{i=1}^{m} h_\theta(y^{(i)} x^{(i)}),$$

and the log-likelihood is precisely

$$\ell(\theta) = \sum_{i=1}^{m} \log h_\theta(y^{(i)} x^{(i)}) = -\sum_{i=1}^{m} \log \left(1 + e^{-y^{(i)} \theta^T x^{(i)}}\right) = -mJ(\theta),$$

where $J(\theta)$ is exactly the logistic regression risk from Eq. (3). That is, maximum likelihood in the logistic model (4) is the same as minimizing the average logistic loss, and we arrive at logistic regression again.

2.2 Gradient descent methods

The final part of logistic regression is to actually fit the model. As is usually the case, we consider gradient-descent-based procedures for performing this minimization. With that in mind, we now show how to take derivatives of the logistic loss. For $\varphi_{\text{logistic}}(z) = \log(1 + e^{-z})$, we have the one-dimensional derivative

$$\frac{d}{dz}\varphi_{\text{logistic}}(z) = \varphi'_{\text{logistic}}(z) = \frac{1}{1 + e^{-z}} \cdot \frac{d}{dz} e^{-z} = -\frac{e^{-z}}{1 + e^{-z}} = -\frac{1}{1 + e^{z}} = -g(-z),$$

where g is the sigmoid function. Then we apply the chain rule to find that for a single training example (x, y), we have

$$\frac{\partial}{\partial \theta_k} \varphi_{\text{logistic}}(y x^T \theta) = -g(-y x^T \theta) \frac{\partial}{\partial \theta_k} (y x^T \theta) = -g(-y x^T \theta) y x_k.$$

Thus, a stochastic gradient procedure for minimization of $J(\theta)$ iteratively performs the following for iterations $t = 1, 2, \ldots$, where α_t is a stepsize at time t:

1. Choose an example $i \in \{1, \ldots, m\}$ uniformly at random

2. Perform the gradient update

$$\begin{aligned} \theta^{(t+1)} &= \theta^{(t)} - \alpha_t \cdot \nabla_\theta \varphi_{\text{logistic}}(y^{(i)} x^{(i)^T} \theta^{(t)}) \\ &= \theta^{(t)} + \alpha_t g(-y^{(i)} x^{(i)^T} \theta^{(t)}) y^{(i)} x^{(i)} = \theta^{(t)} + \alpha_t h_{\theta^{(t)}}(-y^{(i)} x^{(i)}) y^{(i)} x^{(i)}. \end{aligned}$$

This update is intuitive: if our current hypothesis $h_{\theta^{(t)}}$ assigns probability close to 1 for the *incorrect* label $-y^{(i)}$, then we try to reduce the loss by moving θ in the direction of $y^{(i)}x^{(i)}$. Conversely, if our current hypothesis $h_{\theta^{(t)}}$ assigns probability close to 0 for the incorrect label $-y^{(i)}$, the update essentially does nothing.