Chapter 3: Linear Models

Motivation

Recall that a model is defined both by its hypothesis class $\mathcal{H}$ and the training algorithm used to select a hypothesis from $\mathcal{H}$ on a particular dataset. A reasonable idea for a training algorithm is to select a hypothesis that minimizes the in-sample error:

$$g = \arg\min_{h \in \mathcal{H}} E_{\text{in}}(g). \tag{1}$$

For finite hypothesis classes, we saw that the *try everything algorithm* (TEA) can exactly compute the minimization in (1) in finite time. Unfortunately, the TEA algorithm cannot be used on infinite hypothesis classes.

In this set of notes, we will study how to compute the minimization above for infinite classes.

Notation

Definition 1. A *loss function* $\ell(\hat{y}, y)$ is a function that measures "how bad" the output of a hypothesis is. The *in-sample error* with respect to ℓ is defined to be

$$E_{\text{in}}^{\ell}(h) = \frac{1}{N} \sum_{i=1}^{N} \ell\big(h(\mathbf{x}_i), y_i\big) \tag{2}$$

and the *out-of-sample error* with respect to ℓ is

$$E_{\text{out}}^{\ell}(h) = \mathbb{P}\, \ell\big(h(\mathbf{x}), y\big). \tag{3}$$

Example 1. The *0-1 loss* is defined to be

$$\ell_{0-1}(\hat{y}, y) = [\![\hat{y} \neq y]\!]. \tag{4}$$

Substituting (4) into (3) and (2) above yield the familiar definitions for E_{in} and E_{out} from Chapter 1.

The Squared Loss (Section 3.2)

Example 2. In a regression problem, the goal is to predict a real number instead of a binary class label. That is, the output space is $\mathcal{Y} = \mathbb{R}$ instead of $\mathcal{Y} = \{+1, -1\}$. A popular model for regression is *ordinary least squares* (OLS), which uses the linear hypothesis class

$$\mathcal{H} = \left\{ \mathbf{x} \mapsto \mathbf{w}^T \mathbf{x} : \mathbf{w} \in \mathbb{R}^d \right\}. \tag{5}$$

It is common to use the *squared loss* when evaluating regression problems. It is defined to be

$$\ell(\hat{y}, y) = (\hat{y} - y)^2. \tag{6}$$

Substituting (6) into (2) gives the equation for the in-sample error

$$E_{\text{in}}(h) = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^T \mathbf{w} - y_i)^2 = \|X\mathbf{w} - Y\|_2^2, \tag{7}$$

where X is a $N \times d$ matrix with ith row equal to $\mathbf{x}_i$ and Y is the d dimensional vector with ith position equal to y_i.

Problem 1. The OLS model is convenient because the hypothesis g which minimizes E_{in} can be computed in closed form. Show this computation.

HINT: WolframAlpha can't take matrix/vector derivatives. But `https://www.matrixcalculus.org` can take these higher-order derivatives.

Problem 2. What is the runtime of computing the ERM for OLS?

Problem 3. The VC dimension is undefined for the OLS problem. Why?

Note 1. The textbook bounds the generalization error of the squared loss in this section and Chapter 2.3. These bounds are simpler than the VC dimension bounds to understand and prove. We will not cover them in this course, however, because they require knowledge of probability that is (unfortunately IMNSHO) not a pre-requisite for this course. These bounds are often expected knowledge in technical machine learning interviews.

Fact 1. The squared loss is essentially unique in having a closed form minimizer. Most other loss functions do not have closed form minimizers and require iterative procedures to solve.

The 0-1 Loss (Section 3.1)

What you need to know:

1. The 0-1 loss used by the perceptron has no closed form solution.
2. In chapter 1, we introduced the PLA algorithm for computing a hypothesis g that minimizes E_{in} only when the data is linearly separable. When the data is not linearly separable, the PLA will never terminate. (You should review the PLA algorithm and ensure you understand why.)
3. In Section 3.1, the book introduces the *pocket algorithm* for computing a hypothesis g when the data is not linearly separable. Unfortunately, this algorithm is not guaranteed to converge to a minimizer of E_{in}.
4. In general, minimizing the 0-1 loss is *NP-hard*.

   Fact 2. The best algorithms that we have for solving NP-hard problems currently take exponential time. In the case of minimizing the 0-1 loss, that means that the best known algorithms take time $\Omega((1+\epsilon)^d)$ where ϵ is a small constant greater than 0. Most people believe that minimizing the 0-1 loss cannot be done in sub-exponential time, but we do not currently have a proof that this is the case. Therefore, finding the optimal runtime for directly minimizing the 0-1 loss is an **open problem**. There are no meaningful known lower bounds on the optimal runtime.

   If you haven't studied NP-hard problems before, I strongly recommend you watch the video `https://www.youtube.com/watch?v=YX40hbAHx3s`.

The Logistic Loss (Section 3.3)

Since the 0-1 loss cannot be efficiently optimized, our goal is to introduce new loss functions that are "similar" to the 0-1 loss but can be efficiently optimized.

Recall that the 0-1 loss was previously defined to be

$$\ell_{0-1}(\hat{y}, y) = [\![\hat{y} \neq y]\!]. \tag{8}$$

where $\hat{y}$ and y are restricted to be in the set $\{-1, 1\}$. We can generalize this definition to allow the $\hat{y}$ variable to be any real number as follows

$$\ell_{0-1}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y}y > 0 \\ 1 & \text{otherwise} \end{cases} \tag{9}$$

and these two definitions agree whenever they are both defined.

The quantity $z = \hat{y}y$ appears commonly in classification losses, and so classification losses are commonly written as functions of a single variable. The 0-1 loss can be equivalently defined in this way as

$$Q_{0-1}(z) = \begin{cases} 0 & \text{if } z > 0 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

Definition 2. A *surrogate loss function* ℓ is any loss function that upper bounds the 0-1 loss. That is,

$$Q(z) \geq Q_{0-1}(z) \tag{11}$$

for all values of z.

Definition 3. A loss function is *convex* if the line segment between any two points on the graph of the function lies above the graph between the two points.

Definition 4 (informal)**.** The closer a loss function is to the 0-1 loss, the more *robust to outliers* that loss function is. The more robust a loss function is, the closer its minimizer will be to the hard-to-compute minimizer of the 0-1 loss.

Example 3. Complete the table below, and draw each loss function.

name	$Q(z)$	surrogate?	convex?	robust?
exponential loss	$\exp(-z)$			
squared loss	$(1-z)^2$			
logistic loss	$\log(1+\exp(-z))$			
hinge loss	$\max\{0, 1-z\}$			
trimmed hinge loss	$\min\left\{2, \max\{0, 1-z\}\right\}$			
smooth 0-1 loss	$\frac{1}{2} + \arctan(-\alpha \cdot z)/\pi$			

Fact 3. Every surrogate loss function ℓ has the property that

$$E_{\text{in}}^{\ell_{0-1}} \leq E_{\text{in}}^{\ell} \tag{12}$$

and so we can use VC theory to get a generalization bound on the classification error for any model trained with a surrogate loss. In particular, substituting (12) into the VC theorem gives

$$E_{\text{out}}^{\ell_{0-1}} \leq E_{\text{in}}^{\ell} + O\left(\sqrt{\frac{d_{\text{VC}} \log N}{N}}\right) \tag{13}$$

with high probability. In particular, (13) holds for the logistic regression model (using the logistic loss) and the SVM model (using the hinge loss).

Note 2. The textbook also discusses a probabilistic interpretation of the logistic loss. Again, due to the course prereqs, we will not cover this interpretation, but this is sometimes material that is asked about in technical interviews.

Optimizing the Surrogate Loss

Section 3.3.2 of the textbook has an "intuitive" overview of the gradient descent and stochastic gradient descent algorithms. This is not detailed enough, and so we will supplement with Léon Bottou's paper "Large-Scale Machine Learning with Stochastic Gradient Descent."