The Support Vector Machine and regularization

We proposed a simple relaxed optimization problem for finding the maximum margin separator when some of the examples may be misclassified:

$$\text{minimize } \frac{1}{2}\|\theta\|^2 + C\sum_{t=1}^{n}\xi_t \tag{1}$$

$$\text{subject to } \; y_t(\theta^T\mathbf{x}_t + \theta_0) \geq 1 - \xi_t \; \text{ and } \xi_t \geq 0 \; \text{ for all } t = 1,\ldots,n \tag{2}$$

where the remaining parameter C could be set by cross-validation, i.e., by minimizing the leave-one-out cross-validation error.

The goal here is to briefly understand the relaxed optimization problem from the point of view of *regularization*. Regularization problems are typically formulated as optimization problems involving the desired objective (classification loss in our case) and a regularization penalty. The regularization penalty is used to help stabilize the minimization of the objective or infuse prior knowledge we might have about desirable solutions. Many machine learning methods can be viewed as regularization methods in this manner. For later utility we will cast SVM optimization problem as a regularization problem.

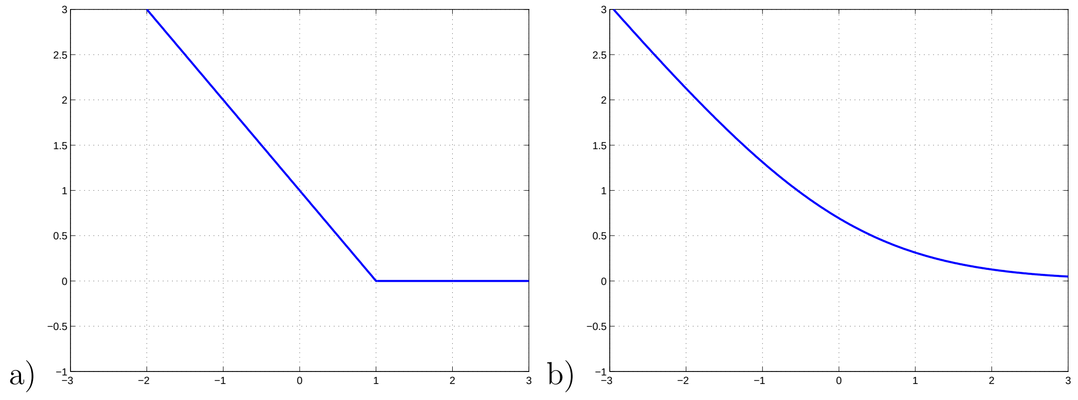

Figure 1: a) The hinge loss $(1-z)^+$ as a function of z. b) The logistic loss $\log[1+\exp(-z)]$ as a function of z.

To turn the relaxed optimization problem into a regularization problem 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 ξ values for a fixed θ and θ_0. This will lead to an expression of $C\sum_t \xi_t$ as a function of θ and θ_0.

The loss function we need for this purpose is based on the *hinge loss* $\text{Loss}_h(z)$ defined as the positive part of $1 - z$, written as $(1 - z)^+$ (see Figure 1a). The relaxed optimization problem can be written using the hinge loss as

$$\text{minimize } \frac{1}{2}\|\theta\|^2 + C\sum_{t=1}^{n} \overbrace{\left(1 - y_t(\theta^T\mathbf{x}_t + \theta_0)\right)^+}^{=\hat{\xi}_t} \tag{3}$$

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

$$C\sum_{t=1}^{n}\left(1 - y_t(\theta^T\mathbf{x}_t + \theta_0)\right)^+ \tag{4}$$

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

Logistic regression, maximum likelihood estimation

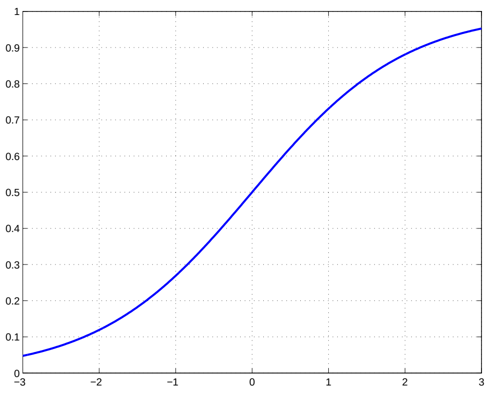

Figure 2: The logistic function $g(z) = (1 + \exp(-z))^{-1}$.

Another way of dealing with noisy labels in linear classification is to model how the noisy labels are generated. For example, human assigned labels tend to be very good for "typical examples" but exhibit some variation in more difficult cases. One simple model of noisy labels in linear classification is a logistic regression model. In this model we assign a probability distribution over the two labels in such a way that the labels for examples further away from the decision boundary are more likely to be correct. More precisely, we say that

$$P(y = 1|\mathbf{x}, \theta, \theta_0) = g\left(\, \theta^T \mathbf{x} + \theta_0 \,\right) \tag{5}$$

where $g(z) = (1 + \exp(-z))^{-1}$ is known as the logistic function (Figure 2). 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:

$$\log \frac{P(y = 1|\mathbf{x}, \theta, \theta_0)}{P(y = -1|\mathbf{x}, \theta, \theta_0)} = \theta^T \mathbf{x} + \theta_0 \tag{6}$$

So for example, when we predict the same probability (1/2) for both classes, the log-odds term is zero and we recover the decision boundary $\theta^T \mathbf{x} + \theta_0 = 0$. The precise functional form of the logistic function, or, equivalently, the fact that we chose to model log-odds with the linear prediction, may seem a little arbitrary (but perhaps not more so than the hinge loss used with the SVM classifier). We will derive the form of the logistic function later on in the course based on certain assumptions about *class-conditional* distributions $P(\mathbf{x}|y = 1)$ and $P(\mathbf{x}|y = -1)$.

In order to better compare the logistic regression model with the SVM we will write the conditional probability $P(y|\mathbf{x}, \theta, \theta_0)$ a bit more succinctly. Specifically, since $1 - g(z) = g(-z)$ we get

$$P(y = -1|\mathbf{x}, \theta, \theta_0) = 1 - P(y = 1|\mathbf{x}, \theta, \theta_0) = 1 - g(\, \theta^T \mathbf{x} + \theta_0 \,) = g\left(\, -(\theta^T \mathbf{x} + \theta_0) \,\right) \tag{7}$$

and therefore

$$P(y|\mathbf{x}, \theta, \theta_0) = g\left(\, y(\theta^T \mathbf{x} + \theta_0) \,\right) \tag{8}$$

So now we have a linear classifier that makes probabilistic predictions about the labels. How should we train such models? A sensible criterion would seem to be to maximize the probability that we predict the correct label in response to each example. Assuming each example is labeled independently from others, this probability of assigning correct labels to examples is given by the product

$$L(\theta, \theta_0) = \prod_{t=1}^{n} P(y_t|\mathbf{x}_t, \theta, \theta_0) \tag{9}$$

$L(\theta, \theta_0)$ is known as the (conditional) likelihood function and is interpreted as a function of the parameters for a fixed data (labels and examples). By maximizing this conditional likelihood with respect to θ and θ_0 we obtain *maximum likelihood estimates* of the parameters. Maximum likelihood *estimators*[1] 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), and b) *efficient* (no other estimator will converge to the correct parameter values faster in the mean squared sense). But what if we do not have the right model class? Neither property may hold as a result. More robust estimators can be found in a larger class of estimators called *M-estimators* that includes maximum likelihood. We will nevertheless use the maximum likelihood principle to set the parameter values.

The product form of the conditional likelihood function is a bit difficult to work with directly so we will maximize its logarithm instead:

$$l(\theta, \theta_0) = \sum_{t=1}^{n} \log P(y_t|\mathbf{x}_t, \theta, \theta_0) \tag{10}$$

Alternatively, we can *minimize* the negative logarithm

$$\begin{aligned} -l(\theta, \theta_0) &= \sum_{t=1}^{n} \overbrace{-\log P(y_t|\mathbf{x}_t, \theta, \theta_0)}^{\text{log-loss}} && (11) \\ &= \sum_{t=1}^{n} -\log g\left(y_t(\theta^T \mathbf{x}_t + \theta_0) \right) && (12) \\ &= \sum_{t=1}^{n} \log\left[1 + \exp\left(-y_t(\theta^T \mathbf{x}_t + \theta_0) \right)\right] && (13) \end{aligned}$$

We can interpret this similarly to the sum of the hinge losses in the SVM approach. As before, we have a base loss function, here $\log[1+\exp(-z)]$ (Figure 1b), similar to the hinge loss (Figure 1a), and this loss depends only on the value of the "margin" $y_t(\theta^T \mathbf{x}_t + \theta_0)$ for each example. The difference here is that we have a clear probabilistic interpretation of the "strength" of the prediction, i.e., how high $P(y_t|\mathbf{x}_t, \theta, \theta_0)$ is for any particular example. Having a probabilistic interpretation does not, however, mean that the probability values are in any way sensible or *calibrated*. Predicted probabilities are *calibrated* when they

[1] An *estimator* is a function that maps data to parameter values. An *estimate* is the value obtained in response to specific data.

correspond to observed frequencies. So, for example, if we group together all the examples for which we predict positive label with probability 0.5, then roughly half of them should be labeled +1. Probability estimates are rarely well-calibrated but can nevertheless be useful.

The minimization problem we have defined above is convex and there are a number of optimization methods available for finding the minimizing $\hat{\theta}$ and $\hat{\theta}_0$ including simple gradient descent. In a simple (stochastic) gradient descent, we would modify the parameters in response to each term in the sum (based on each training example). To specify the updates we need the following derivatives

$$\begin{aligned}
\frac{d}{d\theta_0} \log\left[1 + \exp\big(-y_t(\theta^T \mathbf{x}_t + \theta_0)\big)\right] &= -y_t \frac{\exp\big(-y_t(\theta^T \mathbf{x}_t + \theta_0)\big)}{1 + \exp\big(-y_t(\theta^T \mathbf{x}_t + \theta_0)\big)} && (14)\\
&= -y_t[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)] && (15)
\end{aligned}$$

and

$$\frac{d}{d\theta} \log\left[1 + \exp\big(-y_t(\theta^T \mathbf{x}_t + \theta_0)\big)\right] = -y_t \mathbf{x}_t[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)] \qquad (16)$$

The parameters are then updated by selecting training examples at random and moving the parameters in the opposite direction of the derivatives:

$$\begin{aligned}
\theta_0 &\leftarrow \theta_0 + \eta \cdot y_t[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)] && (17)\\
\theta &\leftarrow \theta + \eta \cdot y_t \mathbf{x}_t[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)] && (18)
\end{aligned}$$

where η is a small (positive) learning rate. Note that $P(y_t|\mathbf{x}_t, \theta, \theta_0)$ is the probability that we predict the training label correctly and $[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)]$ 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, 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 of optimality:

$$\begin{aligned}
\frac{d}{d\theta_0}(-l(\theta, \theta_0) &= -\sum_{t=1}^{n} y_t[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)] = 0 && (19)\\
\frac{d}{d\theta}(-l(\theta, \theta_0)) &= -\sum_{t=1}^{n} y_t \mathbf{x}_t[1 - P(y_t|\mathbf{x}_t, \theta, \theta_0)] = 0 && (20)
\end{aligned}$$

The sum in Eq.(19) 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 θ is characterized by mistake probabilities that are orthogonal to all rows of the label-example matrix $\tilde{\mathbf{X}} = [y_1\mathbf{x}_1, \ldots, y_n\mathbf{x}_n]$. In other words, for each dimension j of the example vectors, $[y_1 x_{1j}, \ldots, y_n x_{nj}]$ is orthogonal to the mistake probabilities. Taken together, these orthogonality conditions ensure that there's no further linearly available information in the examples to improve the predicted probabilities (or mistake probabilities). This is perhaps a bit easier to see if we first map ± 1 labels into 0/1 labels: $\tilde{y}_t = (1 + y_t)/2$ so that $\tilde{y}_t \in \{0, 1\}$. Then the above optimality conditions can be rewritten in terms of prediction errors $[\tilde{y}_t - P(y = 1|\mathbf{x}_t, \theta, \theta_0)]$ rather than mistake probabilities as

$$\sum_{t=1}^{n} [\tilde{y}_t - P(y = 1|\mathbf{x}_t, \theta, \theta_0)] = 0 \tag{21}$$

$$\sum_{t=1}^{n} \mathbf{x}_t [\tilde{y}_t - P(y = 1|\mathbf{x}_t, \theta, \theta_0)] = 0 \tag{22}$$

and

$$\theta'_0 \sum_{t=1}^{n} [\tilde{y}_t - P(y = 1|\mathbf{x}_t, \theta, \theta_0)] + \theta'^T \sum_{t=1}^{n} \mathbf{x}_t [\tilde{y}_t - P(y = 1|\mathbf{x}_t, \theta, \theta_0)] \tag{23}$$

$$= \sum_{t=1}^{n} (\theta'^T \mathbf{x}_t + \theta_0)[\tilde{y}_t - P(y = 1|\mathbf{x}_t, \theta, \theta_0)] = 0 \tag{24}$$

meaning that the prediction errors are orthogonal to any linear function of the inputs.

Let's try to briefly understand the type of predictions we could obtain via maximum likelihood estimation of the logistic regression model. Suppose the training examples are linearly separable. In this case we can find parameter values such that $y_t(\theta^T \mathbf{x}_t + \theta_0)$ are positive for all training examples. 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 $y_t(\theta^T \mathbf{x}_t + \theta_0)$ (the loss $\log\left[1 + \exp\left(-y_t(\theta^T \mathbf{x}_t + \theta_0)\right)\right]$ is strictly decreasing). 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. So, when the number of training examples is small we would need to add the regularizer $\|\theta\|^2/2$ just as in the SVM model. The regularizer helps select reasonable parameters when the available training data fails to sufficiently constrain the linear classifier.

To estimate the parameters of the logistic regression model with regularization we would minimize instead

$$\frac{1}{2}\|\theta\|^2 + C\sum_{t=1}^{n} \log\left[1 + \exp\left(-y_t(\theta^T\mathbf{x}_t + \theta_0)\right)\right] \tag{25}$$

where the constant C again specifies the trade-off between correct classification (the objective) and the regularization penalty. The regularization problem is typically written (equivalently) as

$$\frac{\lambda}{2}\|\theta\|^2 + \sum_{t=1}^{n} \log\left[1 + \exp\left(-y_t(\theta^T\mathbf{x}_t + \theta_0)\right)\right] \tag{26}$$

since it seems more natural to vary the strength of regularization with λ while keeping the objective the same.