Risk Minimization and Optimization Abstractions

Piyush Patil

March 10, 2017

In all the lectures thus far, we've applied a pre-selected algorithm or model to a dataset $\{(x_i, y_i), 1 \leq i \leq n\}$, tuning the parameters w of the model to better fit the data. Specifically, we've tried to solve

$$\min_w \left(\frac{1}{n} \sum_{i=1}^{n} L(w; (x_i, y_i)) + \lambda R(w) \right)$$

where L is our loss function and R is some regularization term.
The loss function is some quantitative way to measure the quality of our model, while the regularization term is akin to a loss function which prefers simpler models to more complex ones. The main algorithm we looked at for solving the minimization problem above was gradient descent, where we essentially used calculus to compute $\nabla L = 0$.
Of course, in practice we don't want our model to just fit the data well; we want to find a model which will, at least in expectation, do well on all possible data. That is, the problem we actually want to solve, and for which solutions to the above problem will provide approximations, is

$$\min_w \left(\mathbb{E}(L(w; (x, y))) \right) \text{ over all possible } x, y$$

If we model x and y as drawn from some joint probability distribution ρ over the complete input and output spaces X and Y, then we know that the above is equal to

$$\text{risk} := \min_w \int_{\mathbb{R}^2} L(w; (x, y)) \rho(x, y) \, dx \, dy$$

Of course, we can't compute the above integral as we don't know ρ, and computing ρ by sampling is computationally exponential with the number of dimensions of x. A fast way to approximate the expected value is to simply take the average of many samples of the loss function on sample data:

$$\text{risk} \approx \text{empirical risk} := \frac{1}{n} \sum_{i=1}^{n} L(w; (x_i, y_i))$$

Essentially, we can approximate the **risk** with the **empirical risk**. Moreover, as $n \to \infty$, the approximation becomes exact. This is where the idea of having a training set and a validation set comes in. As touched on above, although we may fit the data we have very well, in solving the first minimization problem (this can happen even with regularization), our model may not generalize. Thus, if we train our model on a training set, thereby minimizing the empirical training risk, and periodically check to make sure we are also minimizing the empirical validation risk, then because the model was never trained on the validation set, we can keep track of our model's ability to generalize, by ensuring that even on data it's never seen before, it performs well. Typically we partition our data into a training set that's much larger than the validation set. It's also important to randomly partition the data, and periodically re-partition it to prevent meta-overfitting.

Notice that thus far we've only talked about binary linear classifiers. Binary classifiers are, in some sense, quadratically universal in the sense that multiclass classification can be reduced to binary classification by training a binary classifier for each class that's able to recognize if an element is or isn't in a class. However, in practice much of the time the data is not linearly separable, and so linear classifiers will not fare well. One technique to extend linear classifiers to linearly inseparable data is **lifting**. This technique involves extending d-dimensional data to higher (more than d) dimensions by adding new features built out of existing features, with the hope that although the data isn't linearly separable in d-dimensional space, it might be in higher dimensional space, if we choose our features correctly. The intuition for this is that if data is not linearly separable in d-dimensional space, then throwing the data into higher dimensional space will, if we choose the relationship between the higher dimensions and the original d dimensions well, "lift" the data points of one class through the higher dimensions while leaving the data points of the other class, allowing the data points to be separated by a higher dimensional hyperplane.

One example of this is data which can be separated in two dimensions by a circle (or more generally an ellipse) is clearly not linearly separable in two dimensions. But if we add to the data vectors $x = [x_1, x_2]^\intercal$ the feature $x_1^2 + x_2^2$, so the data points become $x = [x_1, x_2, x_1^2 + x_2^2]^\intercal$, then in three dimensions the data becomes linearly separable by a plane, since the resulting 3-dimensional space lifts points outside the separating circle higher onto the parabola $x_3 = x_1^2 + x_2^2$ while the points inside the circle remain untouched, allowing a plane to pass through (parallel to the x_1x_2-plane) the data and separate it.

1 Decision Theory

In this section, we study a type of classifier based in Bayesian probability, a different approach from linear separators. This approach is motivated not only by the fact that we'd prefer a probabilistic confidence level to accompany the classification of a feature vector but also because many times the same feature vectors will have contradictory labels, reflecting an incompleteness in the featurization - probabilistic classification can help with this. The main idea behind Bayesian decision theory is that given a feature vector x, we compute the probability of classification with

$$\Pr(y = 1|x) = \frac{\Pr(x|y = 1)\Pr(y = 1)}{\Pr(x)},$$
$$\Pr(y = -1|x) = \frac{\Pr(x|y = -1)\Pr(y = -1)}{\Pr(x)} = 1 - \Pr(y = 1|x)$$

We can define a loss function which quantifies the loss of mis-classification. This might seem unnecessary, as we might just try to maximize the probability of a correct prediction, but in many real world applications false negatives and false positives don't carry equal weight (e.g. when diagnosing cancer we'd much rather have a false positive than a false negative). In such cases we use asymmetric loss functions (i.e. with $l_1 \neq l_2$ below), which are of the general form

$$L(\hat{y}, y) = \begin{cases} 0, \text{ if } \hat{y} = y \\ l_1, \text{ if } \hat{y} = 1, y = -1 \\ l_2, \text{ if } \hat{y} = -1, y = 1 \end{cases}$$

If we can compute all the probabilities above, then let's define a **decision rule**, or classifier, $r : \mathbb{R}^d \to \{-1, 1\}$ to be our computable function for classifying feature vectors. The **Bayes risk** $\mathcal{R}$ associated with r is, as usual, the expected loss over all possible input-output pairs:

$$\begin{aligned} \mathcal{R}(r) &= \mathbb{E}(L(r(x), y)) \text{ for random variables } x, y \\ &= \sum_{x \in \mathbb{R}^d} \left(\Pr(x) \sum_{y \in \{-1,1\}} \Pr(y|x) L(r(x), y) \right) \\ &= \Pr(y = 1) \sum_{x \in \mathbb{R}^d} (\Pr(x|y = 1) L(r(x), 1)) + \Pr(y = -1) \sum_{x \in \mathbb{R}^d} (\Pr(x|y = -1) L(r(x), -1)) \end{aligned}$$

The sums above should technically be integrals if x is a continuous random variable. We wish to find the optimal classifier r^* which minimizes the Bayes risk above, where $L(r(x), y) = 0 \iff r(x) = y$. This function is given by

$$r^*(x) = \begin{cases} 1, \text{ if } \Pr(y = -1|x) L(1, -1) < \Pr(y = 1|x) L(-1, 1) \\ -1, \text{ otherwise} \end{cases}$$

Intuitively, the above function classifies x as 1 if and only if the expected loss of mis-classifying x as 1 (when in fact it should have been -1) is less than the expected loss of mis-classifying as -1 (when it should have been 1). In other words, r^* minimizes, at least in expectation, the risk of mis-classification for any given feature vector. Of course, in practice we rarely know the true Bayes probabilities, but given a dataset $\{(x_i, y_i), 1 \leq i \leq n\}$ we can approximate the probabilities. For example,

$$\Pr(y = 1) \approx \frac{\text{number of } y_i \text{ where } y_i = 1}{n}$$

Visually, we can think of the optimal Bayes decision boundary as the point where $\Pr(y = -1|x) L(1, -1) = \Pr(y = 1|x) L(-1, 1)$; for a symmetric loss function (where $l_1 = l_2$), this is the intersection of the probability distributions of x given $y = 1$ and given $y = -1$.

2 Generative and Discriminatory Models

Decision theory is a very idealized theory which pre-supposes knowledge of all the distributions above, namely $\Pr(x|y = k), \Pr(y = k)$, or $\Pr(x)$, for $k = -1, 1$. In reality, all we have is data, and we must compute approximations to these distributions ourselves. Let's now discuss some strategies for building these distributions.

There are generally two ways of modeling (often probabilistic) classification problems: generative and discriminative. The fundamental difference between the two is that **discriminative models** learn a boundary between classes, whereas **generative models** model the distribution of each of the individual classes. Thus, discriminative models typically look at $\Pr(y|x)$ and choose the class y which maximizes the probability given a fixed x, based on a learned underlying probability distribution. In contrast, generative models look at $\{\Pr(y, x),\ \text{for all classes } y\}$ and try to model the overall joint distribution of each of the classes and the fixed input, and use that information to choose a class y.

In short, generative models model how data is actually generated, i.e. from the underlying distribution, whereas discriminative models simply provide useful classification boundaries. Discriminative models can be either probabilistic, in which case they try to learn $\Pr(y|x)$ as we've seen, or non-probabilistic, such as support vector machines and linear other classifiers.

3 Random Variables in Several Dimensions

In our study of probability we began with random variables and then extended the theory to real random vectors. In this section we cover the foundational basics of random vectors and specifically the multivariate Gaussian distribution. A **random vector** x is simply a vector in $\mathbb{R}^d$ over d joint probability distributions, with each entry x_i sampled from a probability distribution. The expected value of x is a natural extension of the single-variable definition - $\mathbb{E}(x) = \left[\mathbb{E}(x_i), 1 \leq i \leq d\right]^\intercal$. Extending the definition of variance is tricker; we define the **covariance matrix** Σ of x as the matrix given by

$$\Sigma_{i,j} = \text{Cov}(x_i, x_j) = \mathbb{E}(x - \mathbb{E}(x))\mathbb{E}(y - \mathbb{E}(y))$$

Notice that Σ is guaranteed to be symmetric, and is a diagonal matrix (with standard deviations on the diagonal) if the joint probability distributions over which x was sampled are independent. The matrix is defined thusly so as to obey the equation

$$\text{Cov}(x) = \Sigma = \mathbb{E}((x - \mathbb{E}(x))^\intercal(x - \mathbb{E}(x)))$$

which is analogous to the definition of variance in one variable. The definition of the multivariate Gaussian distribution is an almost straightforward extension of the definition over a single variable.

$$\text{For } x \sim \mathcal{N}(\mu, \Sigma) : \Pr(x) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\intercal\Sigma^{-1}(x-\mu)\right)$$

where $|\Sigma|$ is the determinant of Σ. As in the single dimensional case, we can view the definition above as a composition of a quadratic form into an exponential, with a normalizing coefficient to ensure that the distribution integrates to 1. In the quadratic form (which composed inside the exponential) Σ^{-1} serves as a metric tensor, which controls the curvature of the distribution and, intuitively, gives more or less weight to particular dimensions. An isotropic Guassian distribution is, as the name suggests, the same in every direction, and thus gives equal "weight" to every dimension, and so occurs when Σ is a diagonal matrix, which is to say, x is sampled from *independent* joint distributions. This forces the level curves of the Gaussian to be circles; otherwise, in an anisotrpoic Gaussian, some dimensions are "stretched" while others are "compressed", giving elliptical level curves.
We conclude this section with an important theorem from linear algebra that is of interest to us, given the intuitive ramifications on how symmetric matrices affect n-dimensional space in terms of the n dimensions.

(Theorem) Spectral theorem: *Symmetric matrices in $\mathbb{R}^{n\times n}$ have n pairwise orthogonal eigenvectors.*

- An immediate consequence of this statement is that any symmetric matrix A can be factored as

$$A = U\Lambda U^\intercal$$

where U contains the eigenvectors of A as columns and Λ is a diagonal matrix with the eigenvalues of A on the diagonal.

- This is important because it means that the eigenvectors of symmetric matrices can be used as a basis for $\mathbb{R}^n$. More specifically, the theorem implies that if we work in the basis given by the eigenvectors of a symmetric matrix A, then the linear transformation induced by A reduces to a element-wise multiplication by the eigenvalues of A. More formally, for symmetric matrix A, let $v_1, \cdots, v_n$ be the unit eigenvectors of A, corresponding to the eigenvalues $\lambda_1, \cdots, \lambda_n$. By the spectral theorem we have

$$v_i^\intercal v_j = 0 \text{ for } i \neq j \rightarrow \forall x \in \mathbb{R}^n : \exists c_1, \cdots, c_n \in \mathbb{R} \text{ s.t. } x = \sum_{i=1}^{n} c_i v_i$$

$$\rightarrow Ax = A\sum_{i=1}^{n} c_i v_i = \sum_{i=1}^{n} c_i \cdot (Av_i) = \sum_{i=1}^{n} c_1\lambda_i v_i$$

Thus, letting $\beta = \{v_1, \cdots, v_n\}$,

$$x = [c_i]_\beta \to Ax = [\lambda_i c_i]_\beta \text{ for } 1 \leq i \leq n$$

We can use the spectral theorem to find a geometric interpretation of the covariance matrix of a Gaussian distribution - whereas the mean determines the center of the distribution, the covariance matrix controls the spread and curvature of the distribution about the center, just as the variance in one dimension determines the "width" of the one dimensional Gaussian distribution. More precisely, the isocontours of a general multivariate Gaussian distribution are ellipses; the eigenvalues of the covariance matrix determine the relative lengths of the isocontours' axes. This makes sense intuitively because the axes of the isocontours capture varition in the data - the closer a distribution is to standard, where all variables are completely independent, the closer the isocontours are to circles. The spectral theorem helps us see this because it implies that since the covariance matrix is symmetric, it can be decomposed as above and has orthonormal eigenvectors. This means that the eigenvectors form an orthonormal basis; in other words, they form a basis which is a simple rotation of the standard basis. Because of the above eigendecomposition, the covariance matrix is diagonal in this basis, and has the eigenvalues of the original covariance matrix along its diagonal. Then the quadratic form in the equation of the Gaussian distribution, which involves multiplication by the covariance matrix, becomes a simple elementwise multiplication which determines the lengths of the axes of the quadratic form, and hence the lengths of the axes of the elliptical isocontours of the Gaussian distribution after passing through the exponential.
The reason diagonal covariance matrices are so important is that they allow the multivariate Gaussian distribution to decompose into the product of single-variate Gaussian distributions, one per variable. Even when the covariance matrix isn't diagonal, however, multivariate Gaussians in $\mathbb{R}^n$ are, in a sense, built out of n independent standard univariate Gaussians, one per variable. The key to this insight is the following theorem.

(Theorem) *Let $X \sim \mathcal{N}(\mu, \Sigma)$ be a random variable in $\mathbb{R}^n$ with Gaussian distribution given by mean μ and covariance Σ. Then*

$$\exists B \in \mathbb{R}^{n \times n} \text{ s.t. } Z \sim \mathcal{N}(0, 1) \text{ where } Z = B^{-1} \cdot (X - \mu)$$

In other words, if we center the variable, then it's simply a linear transformation away from $\mathcal{N}(0, 1)$. Another way to look at is is from the implied relation

$$X = BZ + \mu$$

which indicates that any multivariate Gaussian is a shifted linear transformation of independent univariate Gaussians.

4 Maximum Likelihood

The idea behind maximum likelihood estimation, or MLE, is simple. Suppose we have data $\{x_i, y_i, 1 \leq i \leq n\}$ drawn from a known class of probability distributions which depend on parameters θ (a vector of parameters which determines the distribution). Although we know the underlying *family* of probability distributions the data comes from, we want to find the specific distribution which the data conforms to, which is equivalent to finding parameters which parameterize the model. According to MLE, we should choose θ so as to maximize the probability of observing the data $\{x_i, y_i\}$ given the parameters. In other words, we define the likelihood

$$\mathcal{L}(\theta, \{x_i, y_i\}) := \Pr(\{x_i, y_i\}|\theta) = \prod_{i=1}^{n} \Pr((x_i, y_i)|\theta)$$

and maximize $\mathcal{L}$ over fixed data points x_i and y_i. Thus, MLE requires us to shift our perspective and consider the observed data as fixed "parameters" of a new distribution of the possible parameters of the underlying distribution of the data, and choose the most probable θ from this new distribution.
We can maximize $\mathcal{L}$ with simple calculus; to make differentiation easier, we typically first put the expression through a logarithm so as to convert the product to a sum. This doesn't affect extrema since the logarithm is a monotonically increasing function. Thus, we seek

$$\theta \text{ s.t. } \frac{\partial \log(\mathcal{L})}{\partial \theta}(\theta) = 0$$

5 Gaussian Discriminant Analysis

In accordance with the decision theoretic framework we've been working with so far, Gaussian discriminant analysis is a generative approach to classification in which we look at the distribution of data within a given, fixed class. Specifically, we assume that the data in each class, given by $\Pr(x|y)$, follows a Gaussian distribution. Under this framework we can classify a feature vector x by using Bayes theorem to compute $\Pr(y|x)$ for every possible class y and select the class with the maximum probability. That is, we classify x into class C given by

$$C = \operatorname*{argmax}_y \Pr(y|x) = \operatorname*{argmax}_y \frac{\Pr(x|y)\Pr(y)}{\Pr(x)} = \operatorname*{argmax}_y (\Pr(x|y)\Pr(y))$$

where we assume $\Pr(x|y)$ follows a Gaussian distribution; we can approximate $\Pr(y)$ with our dataset $\{(x_i, y_i), 1 \leq i \leq n\}$ with

$$\Pr(y) = \frac{\text{number of } y_i \text{ equal to } 1}{n}$$

We can use maximum likelihood estimation to approximate $\Pr(x|y)$, since all we're looking for to be able to compute the above arg max is the parameters μ and Σ. After applying MLE to the Gaussian distribution, given the data $\{(x_i, y_i), 1 \leq i \leq n\}$, we obtain the intuitive results

$$\Sigma_C = \frac{1}{n_C} \sum_{i \text{ s.t. } y_i = C} (x_i - \mu_C)(x_i - \mu_C)^\intercal$$

$$\mu_C = \frac{1}{n_C} \sum_{i \text{ s.t. } y_i = C} x_i$$

where n_C is the number of data points in class C. Thus, we can classify points by computing the parameters above, using the parameters to compute the probabilities $\Pr(x|y)$ and $\Pr(y)$, and choose the class for which the product of the two probabilities is maximized.

To further analyze the posterior probability, let's explicitly compute the posterior probability for a class C. First, since composing through a logarithm doesn't affect maxima, let's define the quadratic form

$$\begin{aligned} Q_C(x) &= \log(\Pr(x|y = C)\pi_C) = \log\left(\frac{\pi_C}{\sqrt{(2\pi)^d |\Sigma_C|}} \exp\left(-\frac{1}{2}(x - \mu_C)^\intercal \Sigma_C^{-1}(x - \mu_C)\right)\right) \\ &= \log(\pi_C) - \frac{1}{2}\log((2\pi)^d |\Sigma_C|) - \frac{1}{2}(x - \mu_C)^\intercal \Sigma_C^{-1}(x - \mu_C) \end{aligned}$$

where $\pi_C = \Pr(y = C)$. Passing through the logarithm eliminates the exponential and simplifies the expression, which will make the computation of the posterior probability easier. We can now maximize $Q_C(x)$ over all possible classes C. In the special case for binary classification, where $C \in \{-1, 1\}$, the Bayes decision boundary occurs at x for which $Q_1(x) = Q_{-1}(x)$. To compute the posterior probability, which we were after in the first place, we have, for $C \in \{-1, 1\}$,

$$\begin{aligned} \Pr(y = C|x) &= \frac{\Pr(x|y = C)\pi_C}{\Pr(x)} = \frac{\Pr(x|y = C)\pi_C}{\Pr(x|y = 1)\pi_1 + \Pr(x|y = -1)\pi_{-1}} = \frac{e^{Q_C(x)}}{e^{Q_1(x)} + e^{Q_{-1}(x)}} \\ &= \frac{1}{1 + e^{-(Q_1(x) - Q_{-1}(x))}} = \sigma(Q_1(x) - Q_{-1}(x)) \end{aligned}$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is the logistic function. Thus, imposing Gaussian class distributes yields a logistic posterior probability composed with a quadratic form. Logistic posterior functions are very common with exponential based class distributions, and the general pattern holds for many different class distributions, such as exponential, Poisson, and others, with different functions composed within the logistic function (e.g. for the exponential distribution, instead of a quadratic form inside the logistic function, we have a linear function). To see why, consider that

$$\Pr(y = 1|x) = \frac{\Pr(x|y = 1)\Pr(y = 1)}{\Pr(x)} = \frac{\Pr(x|y = 1)\Pr(y = 1)}{\Pr(x|y = 1)\Pr(y = 1) + \Pr(x|y = 0)\Pr(y = 0)} = \frac{1}{1 + \frac{\Pr(y=0)}{\Pr(y=1)}\frac{\Pr(x|y=0)}{\Pr(x|y=1)}}$$

We often estimate $\Pr(y = 0)$ and $\Pr(y = 1)$ to be constants from the sample distribution, which means that if the class distribution is exponential in nature, i.e. $\Pr(x|y) = e^{f(x)}$, then

$$\Pr(y = 1|x) = \frac{1}{1 + \exp(f(x) - \log(\frac{\pi_0}{\pi_1}))} = \sigma\left(-f(x) + \log\left(\frac{\pi_0}{\pi_1}\right)\right)$$

The above general formulation is known as *quadratic discrimination analysis*, or QDA, as opposed to *linear discrimination analysis*, or LDA, in which we add a simplifying assumption to the above; namely, that the distributions of data within a given class all have the same covariance, so that Σ_C is the same for all C. In this case the MLE approximation for the mutual covariance Σ becomes

$$\Sigma = \frac{1}{n} \sum_C \sum_{i \text{ s.t. } y_i = C} (x_i - \mu_C)(x_i - \mu_C)^\intercal$$

QDA is so named because the decision boundary $Q_1 - Q_{-1}$ is a quadratic function; in LDA, however, the decision boundary becomes a linear function:

$$Q_1(x) - Q_{-1}(x) = w^\intercal x + \beta \text{ where } w = (\mu_1 - \mu_{-1})^\intercal \Sigma^{-1}, \beta = -\frac{1}{2}\left((\mu_1^\intercal \Sigma^{-1}\mu_1 - \mu_{-1}^\intercal \Sigma^{-1}\mu_1 - 1\right) + (\log(\pi_1) - \log(\pi_{-1}))$$

Having defined Q_C as above, we can now classify points, in binary classification at least, by taking $\hat{y}_i = \text{sgn}(Q_1(x_i) - Q_{-1}(x_i))$, assuming we have a good reason to believe that the classes are normally distributed. QDA is a more complex model than LDA and can attain higher accuracy, but is also more prone to overfitting.