Plugin Classifiers

Matthieu R. Bloch

1 Plugin classifiers

Plugin classifier are a type of classifiers that attempt to mimic the Bayes classifier by learning estimates of the data distribution from the data, and plugging the result in the formula of the Bayes classifier. Recall from the expression of the Bayes classifier

$$h^{\mathrm{B}}(\mathbf{x}) \triangleq \underset{k\in[0;K-1]}{\operatorname{argmax}} \ \eta_k(\mathbf{x}) \tag{1}$$

that we only need to estimate $\eta_k(\mathbf{x}) \triangleq P_{y|\mathbf{x}}(k|\mathbf{x})$. However, some classifiers sometimes estimate the joint distribution $P_{\mathbf{x},y}$, which is more than we need. A *generative* classifier learns $P_{\mathbf{x},y}$; this allows one to compute the distribution that "generates" the data in every class $P_{\mathbf{x}|y}$. A *discriminatory* classifier only learns $P_{y|\mathbf{x}}$. The estimations can be of two types. The estimation is *parametric* if we impose an underlying model (Gaussian, multinomial, etc.) on the distributions and fit the parameters. The estimation is *non-parametric* if no model is assumed.

The K-NN classifiers are non-parametric discriminative plugin classifiers. They are non-parametric by definition, but their discriminative nature may not be obvious. We can see this with the following heuristic reasoning. Assume that there are L distinct classes, and consider a region $\mathcal{R}$ around a point $\mathbf{x}^*$. If $\mathcal{R}$ is small enough, we have

$$p \triangleq \mathbb{P}(\mathbf{x}\in\mathcal{R}) = \int_{\mathcal{R}} P_{\mathbf{x}}(\mathbf{x})d\mathbf{x} \approx P_{\mathbf{x}}(\mathbf{x}^*)\text{vol}(\mathcal{R})\,. \tag{2}$$

If the data set contains N points, the probability that ℓ points fall in $\mathcal{R}$ is $\binom{N}{\ell}p^\ell(1-p)^{N-\ell}$. As N gets really large, this converges to the mean $K \triangleq Np$, which together with (2) suggest that

$$P_{\mathbf{x}}(\mathbf{x}^*) \approx \frac{K}{N\text{vol}(\mathcal{R})}. \tag{3}$$

The same reasoning can be applied to the N_ℓ points of the data set belonging to class ℓ. The probability $P_{\mathbf{x}|y}(\mathbf{x}^*|\ell)$ of generating a class ℓ point $\mathbf{x}^*$ is

$$P_{\mathbf{x}|y}(\mathbf{x}^*|\ell) \approx \frac{K_\ell}{N_\ell\text{vol}(\mathcal{R})}, \tag{4}$$

where K_ℓ is the average number of points of class ℓ in $\mathcal{R}$. Finally, note that the probability of having a data point in class ℓ can be estimated as N_ℓ/N.[1] Consequently, we can approximate the probability of class ℓ given $\mathbf{x}^*$ by

$$P(\ell|\mathbf{x}*) = \frac{P(\mathbf{x}^*|\ell)P(\ell)}{P(\mathbf{x}*)} \approx \frac{K_\ell}{N_\ell\text{vol}(\mathcal{R})}\frac{N_\ell}{N}\frac{N\text{vol}(\mathcal{R})}{K} = \frac{K_\ell}{K}. \tag{5}$$

Hence viewing $\mathcal{R}$ as the region containing the K nearest neighbors of $\mathbf{x}^*$ (by definition of $\mathcal{R}$ and K), we see that the K-NN classifier computes an estimate of $P_{y|\mathbf{x}}$.

[1] More on this later, we will call that the maximum likelihood estimate.

2 Naive Bayes classifier

Although the idea behind plugin estimators is rather clear, one encounters many issues in practice. In particular, estimating densities in high dimension becomes really challenging because obtaining the large number of samples required is unrealistic or even sometimes unfeasible. The only solution to circumvent this is to impose some modeling assumptions, typically in the form of a parametric model. The number of parameters to estimate is typically small and the parameters can be estimated even with a small number of samples.

Consider a dataset $\mathcal{D} \triangleq \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with feature vectors $\mathbf{x}_j \in \mathbb{R}^d$ and labels $y_i \in [\![1, K]\!]$. The *naive bayes* assumption is to assume that the features are *conditionally independent* given the class label. Mathematically, we assume that the conditional distribution $P_{\mathbf{x}|y}$ factorizes as

$$P_{\mathbf{x}|y} \triangleq P_{x_1,\cdots,x_d|y} = \prod_{j=1}^{d} P_{x_j|y}. \tag{6}$$

This assumption is *unlikely* to be true in practice. For instance, income and zip codes are likely to be dependent. The main benefit of the naive Bayes assumption is to simplify the estimation of the densities. Instead of estimating a multivariate density $P_{\mathbf{x}|y}$, one only has to estimate d univariate densities $P_{x_j|y}$. In addition, one can use distinct models for the univariate densities, which is convenient if some features are categorical while others are numerical.

The general procedure can be summarized as follows.

1. Estimate the *a priori* class densities $\hat{\pi}_k \triangleq \mathbb{P}_y(k)$ for $k \in [\![1, K]\!]$.
2. Estimate the conditional class densities $P_{x_j|y}$ for $j \in [\![1, d]\!]$ and $k \in [\![1, K]\!]$.

We will revisit estimation techniques on several occasions, and for now we establish the following.

Lemma 2.1. *The Maximum Likelihood Estimate (MLE) of the prior class densities is*

$$\hat{\pi}_k = \frac{N_k}{N} \text{ where } N_k \triangleq |\{i \in [\![1, N]\!] : y_i = k\}| = \sum_{i=1}^{N} \mathbb{1}\{y_i = k\}. \tag{7}$$

Proof. Since the true class densities π_k are unknown, we can think of P_y as being dependent on deterministic unknown parameter $\theta \triangleq \{\pi_k\}_{k=1}^K$, denoted P_θ. Given the data $\{y_i\}_{i=1}^N$, the MLE is the parameter θ that maximizes the likelihood of the data $P_\theta(\{y_i\}_{i=1}^N)$:

$$\theta^{\text{MLE}} \triangleq \operatorname*{argmax}_\theta P_\theta(\{y_i\}_{i=1}^N). \tag{8}$$

In our setting, since we assume that the dataset is generated independent and identically distributed (i.i.d.), we have

$$P_\theta(\{y_i\}_{i=1}^N) = \prod_{i=1}^{N} P_\theta(y_i) = \prod_{i=1}^{N}\prod_{k=1}^{K} \pi_k^{\mathbb{1}\{y_i=k\}} = \prod_{k=1}^{K} \pi_k^{\sum_{i=1}^N \mathbb{1}\{y_i=k\}} = \prod_{k=1}^{K} \pi_k^{N_k}. \tag{9}$$

Since $x \to \log x$ is an increasing function, we can maximize the log-likelihood and write

$$\theta^{\text{MLE}} \triangleq \operatorname*{argmax}_\theta \log P_\theta(\{y_i\}_{i=1}^N) = \sum_{k=1}^{K} N_k \log \pi_k. \tag{10}$$

Since $\sum_{k=1}^{K}\pi_k = 1$ by definition, this is a *constrained* optimization problem, which we can nevertheless solve by using Lagrange multipliers. Specifically, for $\lambda \in \mathbb{R}$ we set

$$\mathcal{L} = \sum_{k=1}^{K} N_k \log \pi_k - \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right). \tag{11}$$

Then, for every $k \in [\![1, K]\!]$

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = 0 \Rightarrow \frac{N_k}{\pi_k} - \lambda = 0 \Rightarrow \pi_k = \frac{N_k}{\lambda}. \tag{12}$$

To find the value of λ, note that

$$\sum_{k=1}^{K}\pi_k = 1 = \frac{1}{\lambda}\sum_{k=1}^{K} N_k = \frac{N}{\lambda} \tag{13}$$

so that $\lambda = N$. Note that we have actually ignored the fact that every π_k was also constrained to be positive. We got lucky and our solution satisfies this without enforcing the constraints explicitly. ■

3 Review of Maximum Likelihood Estimate

Consider a parametric density $p_\theta(x)$ with unknown parameter $\theta \in \mathbb{R}^d$ and assume that we have i.i.d. generated data points $\{x_i\}_{i=1}^N$. The likelihood of the data points is defined as

$$\mathcal{L}(\theta) \triangleq \mathbb{P}_\theta\big(\{x_i\}_{i=1}^N\big) = \prod_{i=1}^{N} p_\theta(x_i) \tag{14}$$

Note that we view $\mathcal{L}(\theta)$ as a function of θ alone, treating the data as fixed. It is often convenient to work with the log-likelihood

$$\ell(\theta) \triangleq \log \mathcal{L}(\theta) \triangleq \log \mathbb{P}_\theta\big(\{x_i\}_{i=1}^N\big) = \sum_{i=1}^{N} \log p_\theta(x_i), \tag{15}$$

especially because any product becomes a sum.

Definition 3.1. *The Maximum Likelihood Estimate (MLE) is* $\theta_{\text{MLE}} = \textit{argmax}_\theta\, \mathcal{L}(\theta)$

Sometimes, there exists a closed-form solution for the MLE; however, more often than not, we need to resort to *numerical* solutions. The following example illustrates a situation in which we obtain an optimization problem when forming the MLE, which is representative of typical situations encountered in machine learning.

Example 3.2. *Assume that we have access to pairs of data* $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ *with* $\mathbf{y}_i \in \mathbb{R}$ *and* $\mathbf{X}_i \in \mathbb{R}^d$*. We also assume that all* y_i*'s are conditionally independent of each other given the* $\mathbf{x}_i$*'s, and satisfy* $y_i = \theta^\intercal \mathbf{x}_i + n_i$ *where* $n_i \sim \mathcal{N}(0, \sigma^2)$ *and all* n_i*'s are independent of each others. Then the MLE for* θ *is*

$$\theta_{\text{MLE}} = \underset{\theta}{\textit{argmin}} \sum_{i=1}^{N} \left|y_i - \theta^\intercal \mathbf{x}_i\right|^2. \tag{16}$$

Notice that we do not specify the distribution of the $\mathbf{x}_i$*'s, since all we are trying to do is estimate the* conditional *distribution of* y *given* x*. To obtain the result, we start by forming the likelihood*

$$\mathcal{L}(\theta) \triangleq \prod_{i=1}^{N} p_\theta(y_i|\mathbf{x}_i) \triangleq \prod_{i=1}^{N} \left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \theta^\intercal \mathbf{x}_i)^2}{2\sigma^2} \right) \right) \tag{17}$$

The log-likelihood is then

$$\ell(\theta) \triangleq -\sum_{i=1}^{N} \log \sqrt{2\pi}\sigma - \sum_{i=1}^{N} \frac{(y_i - \theta^\intercal \mathbf{x}_i)^2}{2\sigma^2}, \tag{18}$$

so that

$$\theta_{\text{MLE}} \triangleq \underset{\theta}{\textit{argmax}}\, \ell(\theta) \triangleq \underset{\theta}{\textit{argmin}} \sum_{i=1}^{N} |y_i - \theta^\intercal \mathbf{x}_i|^2 . \tag{19}$$

Once your reach that point, you typically run your favorite optimization algorithm. In this specific example, however, it turns out that there is a closed form solution. In fact, upon aggregating all the y_i*'s in a vector* $\mathbf{y} \in \mathbb{R}^N$ *and all the* $\mathbf{x}_i$*'s in a matrix* $\mathbf{X} \in \mathbb{R}^{N\times d}$*, which rows are* $\mathbf{x}_i^\intercal$*, note that the above minimization is can be rewritten*

$$\theta_{\text{MLE}} \triangleq \underset{\theta}{\textit{argmin}} \left\|\mathbf{y} - \mathbf{X}\theta\right\|_2^2 \tag{20}$$

Note that

$$R(\theta) \triangleq \left\|\mathbf{y} - \mathbf{X}\theta\right\|_2^2 = (\mathbf{y} - \mathbf{X}\theta)^\intercal(\mathbf{y} - \mathbf{X}\theta) = \left\|\mathbf{y}\right\|_2^2 - 2\mathbf{y}^\intercal\mathbf{X}\theta + \theta^\intercal\mathbf{X}^\intercal\mathbf{X}\theta, \tag{21}$$

so that

$$\frac{\partial R}{\partial \theta}(\theta) = -2\mathbf{X}^\intercal\mathbf{y} + 2\mathbf{X}^\intercal\mathbf{X}\theta \tag{22}$$

Assuming that $\mathbf{X}^\intercal\mathbf{X}$ *is invertible, we obtain*

$$\frac{\partial R}{\partial \theta}(\theta) = 0 \Leftrightarrow \theta = (\mathbf{X}^\intercal\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{y}. \tag{23}$$

4 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is an attempt to improve on of the shortcomings of Naive Bayes, namely the assumption that given a label, the features are independent. Instead, LDA models the features as jointly Gaussian, with a covariance matrix that is *class-independent*.

Specifically, let $\mathbf{x} = [x_1, \cdots, x_d]^\intercal \in \mathbb{R}^d$ be a random feature vector and let y be the label. LDA posits that given y the feature vector $\mathbf{x}$ has a Gaussian distribution $P_{\mathbf{x}|y} \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$. Note that the mean $\boldsymbol{\mu}_k$is class dependent but the covariance matrix $\boldsymbol{\Sigma}$ is class independent. It will be convenient to denote a Gaussian multivariate distribution with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ by

$$\phi(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \right). \tag{24}$$

Given this model, LDA then performs a parameter estimation of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}$, as well as of the prior π_k on the data.

Lemma 4.1. *Let N_k be the number of data points with label k. The MLEs for LDA are*

$$\forall k \quad \hat{\pi}_k = \frac{N_k}{N}, \tag{25}$$

$$\forall k \quad \hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_{i:y_i=k}\mathbf{x}_i \tag{26}$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{k=0}^{K-1}\sum_{i:y_i=k}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_k)^\intercal \tag{27}$$

Proof. The MLE for the prior class distributions was already derived in Lecture 4. What is perhaps a bit surprising is that the joint MLE for all the parameters $\boldsymbol{\theta} \triangleq (\{\pi_k\}_k, \{\boldsymbol{\mu}_k\}, \boldsymbol{\Sigma})$ takes the form given above. The likelihood of the parameters is

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N}\prod_{k=0}^{K-1}\pi_k^{\mathbb{1}\{y_i=k\}}\phi(\mathbf{x}_i;\boldsymbol{\mu}_k,\boldsymbol{\Sigma})^{\mathbb{1}\{y_i=k\}} \tag{28}$$

so that the log-likelihood takes the form

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N}\sum_{k=0}^{K-1}\mathbb{1}\{y_i=k\}\left(\ln\pi_k - \frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_k)^\intercal\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_k)\right) - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| \tag{29}$$

$$= \underbrace{\sum_{k=0}^{K-1}N_k\ln\pi_k}_{\ell_1(\boldsymbol{\theta})} + \underbrace{\sum_{k=0}^{K-1}\sum_{i=1}^{N}\frac{-\mathbb{1}\{y_i=k\}}{2}(\mathbf{x}_i-\boldsymbol{\mu}_k)^\intercal\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_k) - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}|}_{\ell_2(\boldsymbol{\theta})}. \tag{30}$$

Note that $\{\pi_k\}$ do not interact with $\{\boldsymbol{\mu}_k\}$ and $\boldsymbol{\Sigma}$. Consequently, the MLE of $\{\pi\}_k$ is the one we studied previously and $\pi_k = \frac{N_k}{N}$ where $N_k = \sum_{i=1}^{N}\mathbb{1}\{y_i=k\}$.

Let us focus on maximizing $\ell_2(\boldsymbol{\theta})$. Taking the gradient with respect to $\boldsymbol{\mu}_k$ and setting it to $\mathbf{0}$ yields

$$\frac{\partial\ell_2(\boldsymbol{\theta})}{\partial\boldsymbol{\mu}_k} = \sum_{k=0}^{K-1}\sum_{i=1}^{N}\frac{-\mathbb{1}\{y_i=k\}}{2}\left(-2\boldsymbol{\Sigma}^{-1}\mathbf{x}_i + 2\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k\right) \tag{31}$$

$$= \boldsymbol{\Sigma}^{-1}\left(\sum_{\mathbf{x}_i:y_i=k}\mathbf{x}_i - N_k\boldsymbol{\mu}_k\right) \tag{32}$$

$$= 0 \tag{33}$$

Conveniently, note that $\boldsymbol{\Sigma}^{-1}$ (assumed non-singular) does not enter the equation and we obtain $\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{\mathbf{x}_i:y_i=k}\mathbf{x}_i$.

Finally, to take the gradient with respect to $\boldsymbol{\Sigma}$, we rewrite $\ell_2(\boldsymbol{\theta})$ as

$$\ell_2(\boldsymbol{\theta}) = \sum_{k=0}^{K-1}\sum_{i=1}^{N} \frac{-\mathbb{1}\{y_i = k\}}{2} \text{tr}\left((\mathbf{x}_i - \boldsymbol{\mu}_k)^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right) - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| \tag{34}$$

$$= \frac{-1}{2}\text{tr}\left(\boldsymbol{\Sigma}^{-1}\underbrace{\sum_{k=0}^{K-1}\sum_{\mathbf{x}_i:y_i=k}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\intercal}_{\triangleq \mathbf{S}}\right) - \frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| \tag{35}$$

we obtain (check the matrix cookbook for the derivation rules)

$$\frac{\partial \ell_2(\boldsymbol{\theta})}{\partial \boldsymbol{\Sigma}} = -\frac{1}{2}\left(-\boldsymbol{\Sigma}^{-1}\mathbf{S}\boldsymbol{\Sigma}^{-1} - N\boldsymbol{\Sigma}^{-1}\right) = \frac{1}{2}\boldsymbol{\Sigma}^{-1}(\mathbf{S}\boldsymbol{\Sigma}^{-1} - N\mathbf{I}) = 0 \tag{36}$$

Again, for $\boldsymbol{\Sigma}^{-1}$ non singular, we obtain $\boldsymbol{\Sigma} = \frac{S}{N}$. Note that we have not been particularly careful in checking that we are indeed *maximizing* the likelihood. ■

You might notice that the covariance estimator is *biased*, but the bias vanishes as the number of points gets large. In practice, you could choose any other estimator of your liking, we will discuss this again in the context of bias-variance tradeoff.

Lemma 4.2. *The LDA classifier is*

$$h^{\text{LDA}}(\mathbf{x}) = \underset{k}{\textit{argmin}}\left(\frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)^\intercal \hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k) - \log\hat{\pi}_k\right) \tag{37}$$

For $K = 2$, the LDA classifier is a linear classifier.

Proof. The first part of the lemma follows by remembering that for a plug-in classifier, we have $h(\mathbf{x}) \triangleq \text{argmax}_k\, \eta_k(\mathbf{x})$. Here,

$$\underset{k}{\text{argmax}}\,\eta_k(\mathbf{x}) = \underset{k}{\text{argmax}}\,P_{y|\mathbf{x}}(k|\mathbf{x}) \tag{38}$$

$$\overset{(a)}{=} \underset{k}{\text{argmax}}\,P_{\mathbf{x}|y}(\mathbf{x}|k)\hat{\pi}_k \tag{39}$$

$$\overset{(b)}{=} \underset{k}{\text{argmax}}\left(\log P_{\mathbf{x}|y} + \log\hat{\pi}_k\right) \tag{40}$$

$$= \underset{k}{\text{argmax}}\left(-\log\left[(2\pi)^{\frac{d}{2}}\left|\hat{\boldsymbol{\Sigma}}\right|^{\frac{1}{2}}\right] - \frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)^\intercal\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k) + \log\hat{\pi}_k\right) \tag{41}$$

$$\overset{(c)}{=} \underset{k}{\text{argmin}}\left(\frac{1}{2}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k)^\intercal\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x} - \hat{\boldsymbol{\mu}}_k) - \log\hat{\pi}_k\right), \tag{42}$$

where (a) follows by Bayes' rule and the fact that $P_\mathbf{x}$ does not depend on k; (b) follows because $x \mapsto \log x$ is increasing; (c) follows by dropping all the terms that do not depend on k and the fact that $\text{argmax}_x\, f(x) = \text{argmin}_x\, -f(x)$.

For $K = 2$, notice that the classifier is effectively performing the test

$$\eta_0(\mathbf{x}) \lessgtr \eta_1(\mathbf{x}) \Leftrightarrow \frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_0)^\intercal\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_0) - \log\hat{\pi}_0 \gtrless \frac{1}{2}(\mathbf{x}-\hat{\boldsymbol{\mu}}_1)^\intercal\hat{\boldsymbol{\Sigma}}^{-1}(\mathbf{x}-\hat{\boldsymbol{\mu}}_1) - \log\hat{\pi}_1 \quad (43)$$

$$\Leftrightarrow -\hat{\boldsymbol{\mu}}_0^\intercal\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{x} + \frac{1}{2}\hat{\boldsymbol{\mu}}_0^\intercal\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_0 - \log\hat{\pi}_0 \gtrless -\hat{\boldsymbol{\mu}}_1^\intercal\hat{\boldsymbol{\Sigma}}^{-1}\mathbf{x} + \frac{1}{2}\hat{\boldsymbol{\mu}}_1^\intercal\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_1 - \log\hat{\pi}_1$$

$$\Leftrightarrow \underbrace{(\hat{\boldsymbol{\mu}}_1-\hat{\boldsymbol{\mu}}_0)^\intercal\hat{\boldsymbol{\Sigma}}^{-1}}_{\triangleq\mathbf{w}}\mathbf{x} + \underbrace{\frac{1}{2}\hat{\boldsymbol{\mu}}_0^\intercal\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_0 - \log\hat{\pi}_0 - \frac{1}{2}\hat{\boldsymbol{\mu}}_1^\intercal\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_1 + \log\hat{\pi}_1}_{\triangleq b} \gtrless 0 \quad (44)$$

$$\Leftrightarrow \mathbf{w}^\intercal\mathbf{x} + b \gtrless 0. \quad (45)$$

The set $\mathcal{H} \triangleq \{\mathbf{x}\in\mathbb{R}^d : \mathbf{w}^\intercal\mathbf{x}+b=0\}$ is a *hyperplane*, which is an affine subspace of $\mathbb{R}^d$ of dimension $d-1$. $\mathcal{H}$ acts as a linear boundary between the two classes that we are trying to distinguish, and the test in (45) is simply checking on what side of the hyperplane the point $\mathbf{x}$ lies. ■

To conclude on LDA, note that the generative model $P_{\mathbf{x}|y} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ is rarely accurate. In addition, there are quite a few parameters to estimate, including $K-1$ class priors, Kd means, $\frac{1}{2}d(d+1)$ elements of covariance matrix. This works well if $N \gg d$ but works poorly if $N \ll d$ without other tricks (dimensionality reduction, structured covariance) that we will discuss later.

An natural extension of LDA is Quadratic Discriminant Analysis (QDA), in which we allow the covariance matrix $\boldsymbol{\Sigma}_k$ to vary with each class k. This results in a quadratic decision boundary instead of the linear boundary established in Lemma 4.2. However, perhaps the biggest issue with LDA is, in Vapnik's words, that "*one should solve the [classification] problem directly and never solve a more general problem as an intermediate step [such as modeling $P(\mathbf{x}|y)$].*". With LDA, as should be clear from Lemma 4.1, we are actually modeling the entire joint distribution $P_{\mathbf{x},y}$, when we really only care about $\eta_k(\mathbf{x})$ for classification.

With Vapnik's word of caution in mind, let us revisit one last time the binary classifier with LDA. You should check for yourself that

$$\eta_1(\mathbf{x}) = \frac{\hat{\pi}_1\phi(\mathbf{x};\hat{\boldsymbol{\mu}}_1,\hat{\boldsymbol{\Sigma}})}{\hat{\pi}_1\phi(\mathbf{x};\hat{\boldsymbol{\mu}}_1,\hat{\boldsymbol{\Sigma}})+\hat{\pi}_0\phi(\mathbf{x};\hat{\boldsymbol{\mu}}_0,\hat{\boldsymbol{\Sigma}})} = \frac{1}{1+\exp(-(\mathbf{w}^\intercal\mathbf{x}+b))}, \quad (46)$$

where $\mathbf{w}$ and b are defined as per (45). In other words, we do *not* need to estimate the full joint distribution. All that seems to be required are the parameters $\mathbf{w}$ and b, and LDA makes a detour to compute these parameters as a function of the mean and covariance matrix of a Gaussian distribution. The *direct* estimation of these parameters leads to another linear classifier called the *logistic regression*.

5 Logistic regression

The key idea behind (binary) logistic regression is to *assume* that $\eta_1(\mathbf{x})$ is of the form

$$\frac{1}{1+\exp(-(\mathbf{w}^\intercal\mathbf{x}+b))} \triangleq 1-\eta_0(\mathbf{x}), \quad (47)$$

and to directly estimate $\hat{\mathbf{w}}$ and $\hat{b}$ from the data. One therefore obtains an estimate of the conditional distribution $P_{y|\mathbf{x}}(1|\mathbf{x})$ as

$$\eta_1(\mathbf{x}) = \frac{1}{1+\exp(-(\hat{\mathbf{w}}^\intercal\mathbf{x}+\hat{b}))}. \quad (48)$$

Since the function $x \mapsto \frac{1}{1+e^{-x}}$ is called the *logistic map*, the corresponding classifier inherited the name and is defined as

$$h^{\mathrm{LR}}(\mathbf{x}) = \mathbb{1}\left\{\eta_1(\mathbf{x}) \geqslant \frac{1}{2}\right\} = \mathbb{1}\left\{\hat{\mathbf{w}}^\intercal\mathbf{x} + \hat{b} \geqslant 0\right\}. \tag{49}$$

This is again a linear classifier. Note that LDA led to a similar classifier with the specific choice of parameters

$$\hat{\mathbf{w}} = \hat{\mathbf{\Sigma}}^{-1}(\hat{\boldsymbol{\mu}}_1 - \hat{\boldsymbol{\mu}}_0) \quad b = \frac{1}{2}\hat{\boldsymbol{\mu}}_0^\intercal\hat{\mathbf{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_0 - \frac{1}{2}\hat{\boldsymbol{\mu}}_1^\intercal\hat{\mathbf{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_1 + \log\frac{\hat{\pi}_1}{\hat{\pi}_0} \tag{50}$$

Note that this *not* what the MLE of $(\hat{\mathbf{w}}, b)$ would result in, and we will analyze this in more details.

6 MLE for logistic regression

We will start with a standard trick to simplify notation, which consists in defining $\tilde{\mathbf{x}} = [1,\ \mathbf{x}^\intercal]^\intercal$ and $\boldsymbol{\theta} = [b\,\mathbf{w}^\intercal]^\intercal$. This allows us to write the logistic model as

$$\eta(\mathbf{x}) \triangleq \eta_1(\mathbf{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^\intercal\tilde{\mathbf{x}})}. \tag{51}$$

To avoid carrying a tilde repeatedly in our notation, we will now simply write $\mathbf{x}$ in place of $\tilde{\mathbf{x}}$, but keep in mind that we operate under the assumption that the first component of $\mathbf{x}$ is set to one.

Given our dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ the likelihood is $\mathcal{L}(\boldsymbol{\theta}) \triangleq \prod_{i=1}^N \mathbb{P}_{\boldsymbol{\theta}}(y_i|\mathbf{x}_i)$, where we don't try to model the distribution of $\mathbf{x}_i$ as mentioned in Example 3.2. For $K = 2$ and $\mathcal{Y} = \{0, 1\}$, we obtain

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \prod_{i=1}^N \eta(\mathbf{x}_i)^{y_i}(1-\eta(\mathbf{x}_i))^{1-y_i} \tag{52}$$

In case you are not familiar with this way of writing the likelihood, note that

$$\eta(\mathbf{x}_i)^{y_i}(1-\eta(\mathbf{x}_i))^{1-y_i} = \begin{cases}\eta(\mathbf{x}_i) = \eta_1(\mathbf{x}_i) \text{ if } y_i = 1\\ (1-\eta(\mathbf{x}_i)) = \eta_0(\mathbf{x}_i) \text{ if } y_i = 0.\end{cases} \tag{53}$$

The log likelihood can therefore be written as

$$\ell(\boldsymbol{\theta}) \triangleq \log\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^N (y_i \log\eta(\mathbf{x}_i) + (1-y_i)\log(1-\eta(\mathbf{x}_i))) \tag{54}$$

$$= \sum_{i=1}^N \left(y_i \log\frac{1}{1+e^{-\boldsymbol{\theta}^\intercal\mathbf{x}}} + (1-y_i)\log\frac{e^{-\boldsymbol{\theta}^\intercal\mathbf{x}}}{1+e^{-\boldsymbol{\theta}^\intercal\mathbf{x}}}\right) \tag{55}$$

$$= \sum_{i=1}^N \left(y_i\boldsymbol{\theta}^\intercal\mathbf{x}_i - \log(1+e^{\boldsymbol{\theta}^\intercal\mathbf{x}_i})\right). \tag{56}$$

To find the minimum with respect to (w.r.t.) $\boldsymbol{\theta}$, a necessary condition for optimality is $\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta}) = \mathbf{0}$. Here, this means that

$$\nabla_{\boldsymbol{\theta}}\ell(\boldsymbol{\theta}) = \sum_{i=1}^N \left(y_i\mathbf{x}_i - \frac{e^{\boldsymbol{\theta}^\intercal\mathbf{x}_i}}{1+e^{\boldsymbol{\theta}^\intercal\mathbf{x}_i}}\mathbf{x}_i\right) = \sum_{i=1}^N \mathbf{x}_i\left(y_i - \frac{1}{1+e^{-\boldsymbol{\theta}^\intercal\mathbf{x}_i}}\right) = 0. \tag{57}$$

Solving this equation means solving a nonlinear system of $d+1$ equations, for which there exists no clear methodology. Hence, we must resort to a numerical algorithm to find the solution of $\text{argmin}_\theta -\ell(\theta)$.

You should check for yourself $-\ell(\boldsymbol{\theta})$ is *convex* in $\boldsymbol{\theta}$, and there exists algorithms with *provable* convergence guarantees. We will mention a few specific techniques, such as gradient descent, Newton's method, but there are many more that especially useful in high dimension.

7 Conclusion regarding plug-in methods

Naive Bayes, LDA, and logistic classification are all plugin methods that result in *linear* classifiers, i.e., classifiers for which decision boundaries are hyperplanes in $\mathbb{R}^d$. All have advantages and drawbacks:

- Naive Bayes is plugin method based on a seldom valid assumption (independence of features given the class), but which scales well to high-dimensions and naturally handles mixture of discrete and continuous features;
- LDA tends to work well if the assumption regarding the Gaussian distribution of the feature vectors in a class is valid;
- Logistic classification models only the distribution of $P_{y|\mathbf{x}}$, not $P_{y,\mathbf{x}}$, which is valid for a larger class of distributions and results in fewer parameters to estimate.

Plugin methods can be useful in practice, but ultimately they are very limited. There are always distributions for which assumptions are violated and if our assumptions are wrong, the output is totally unpredictable. It can be hard to verify whether our assumptions are right and plugin methods often require solving a more difficult problem as an intermediate step, see for instance the detour made by LDA to obtain a linear model.