Deep learning

3.2. Probabilistic view of a linear classifier

François Fleuret

https://fleuret.org/dlc/

The Linear Discriminant Analysis (LDA) algorithm provides a nice bridge between these linear classifiers and probabilistic modeling.

Consider the following class populations

$$\forall y \in \{0,1\}, x \in \mathbb{R}^D,$$

$$\mu_{X|Y=y}(x) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} \exp\left(-\frac{1}{2}(x - m_y)\Sigma^{-1}(x - m_y)^T\right).$$

That is, they are Gaussian with **the same covariance matrix Σ.** This is the **homoscedasticity** assumption.

Intuitively we can map data linearly to make all the covariance matrices identity, there the Bayesian separation is a plan, so it is also in the original space.

Notes

In other words, the populations have the same ellipsoid shape, but are located at different places in the space.

We have

$$
\begin{aligned}
P(Y=1 \mid X=x) &= \frac{\mu_{X|Y=1}(x)P(Y=1)}{\mu_X(x)} \\
&= \frac{\mu_{X|Y=1}(x)P(Y=1)}{\mu_{X|Y=0}(x)P(Y=0)+\mu_{X|Y=1}(x)P(Y=1)} \\
&= \frac{1}{1+\frac{\mu_{X|Y=0}(x)}{\mu_{X|Y=1}(x)}\frac{P(Y=0)}{P(Y=1)}} \\
&= \sigma\left(\log\frac{\mu_{X|Y=1}(x)}{\mu_{X|Y=0}(x)}+\log\frac{P(Y=1)}{P(Y=0)}\right),
\end{aligned}
$$

with

$$
\sigma(x) = \frac{1}{1+e^{-x}}.
$$

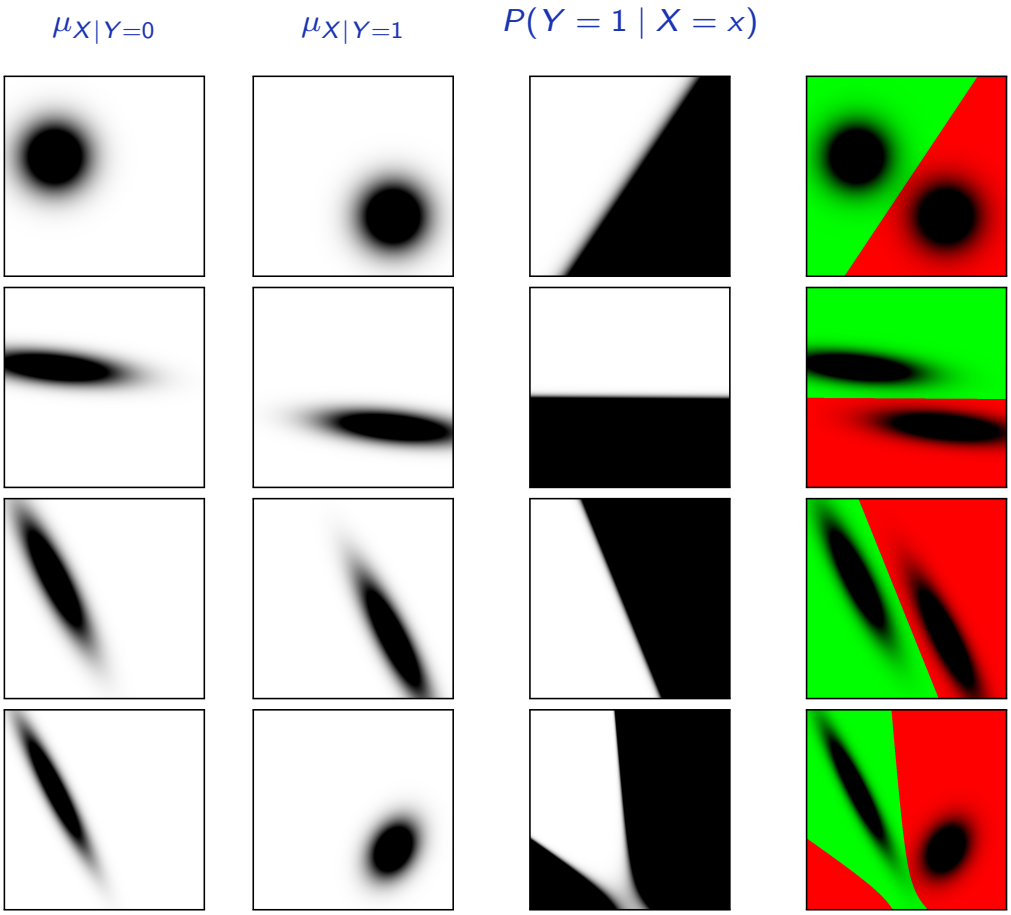
So with our Gaussians $\mu_{X|Y=y}$ of same Σ, we get

$$
\begin{aligned}
&P(Y=1 \mid X=x) \\
&\qquad = \sigma\left(\log \frac{\mu_{X|Y=1}(x)}{\mu_{X|Y=0}(x)} + \underbrace{\log \frac{P(Y=1)}{P(Y=0)}}_{Z} \right) \\
&\qquad = \sigma\left(\log \mu_{X|Y=1}(x) - \log \mu_{X|Y=0}(x) + Z \right) \\
&\qquad = \sigma\left(-\frac{1}{2}(x-m_1)\Sigma^{-1}(x-m_1)^T + \frac{1}{2}(x-m_0)\Sigma^{-1}(x-m_0)^T + Z \right) \\
&\qquad = \sigma\Bigg(-\frac{1}{2}x\Sigma^{-1}x^T + m_1\Sigma^{-1}x^T - \frac{1}{2}m_1\Sigma^{-1}m_1^T \\
&\qquad\qquad + \frac{1}{2}x\Sigma^{-1}x^T - m_0\Sigma^{-1}x^T + \frac{1}{2}m_0\Sigma^{-1}m_0^T + Z \Bigg) \\
&\qquad = \sigma\left(\underbrace{(m_1-m_0)\Sigma^{-1}}_{w} x^T + \underbrace{\frac{1}{2}\left(m_0\Sigma^{-1}m_0^T - m_1\Sigma^{-1}m_1^T \right) + Z}_{b} \right) \\
&\qquad = \sigma(w \cdot x + b).
\end{aligned}
$$

The homoscedasticity makes the second-order terms vanish.

Notes

The homoscedasticity assumption is key here to make the quadratic term vanish.
Finally, $P(Y=1 \mid X=x)$ looks a lot like the linear predictor previously seen, the perceptron.



Notes

Each row is an illustration of the decision boundary for different values of Σ, the means m_0 and m_1 remaining the same.

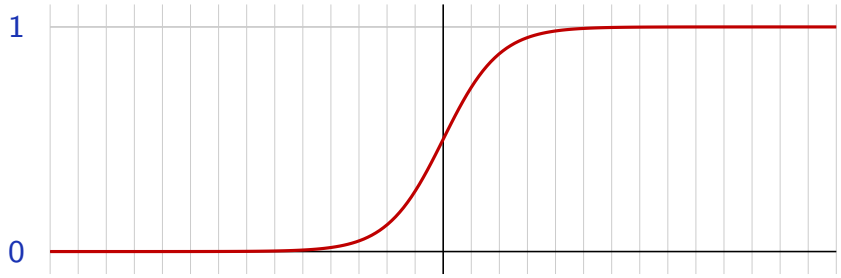
- White stands for values close to 0, and black for values close to 1.
- Column 1 depicts the distribution of class 0, that is $\mu_{X|Y=0}$.
- Column 2 depicts the distribution of class 0, that is $\mu_{X|Y=1}$.
- Column 3 depicts the posterior for a point of being of class 1, $P(Y = 1 \mid X = x)$; the black area is the set of points which are more likely to be of class 1, and by symmetry, the white area is the set of points which are more likely to be of class 0.
- Column 4 shows the set of points assigned to class 1 (red area) and to class 0 (green area). The threshold used is 0.5.

We see that the covariance drives the orientation of the separation: $(m_1 - m_0)\Sigma^{-1}$, m_0 and m_1 being fixed here.
When the two populations have a different covariance (bottom row), the decision boundary is no longer linear.

Note that the (logistic) sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}},$$

looks like a "soft heavyside"



So the overall model

$$f(x; w, b) = \sigma(w \cdot x + b)$$

looks very similar to the perceptron.

Notes

In some sense, this model is a smooth version of the hard threshold seen before with the perceptron.
We have a nice bridge between linear classifiers and probabilistic models: LDA shows that in a probabilistic context with a reasonable assumption on the distribution of the data, we get a decision function which is very close to the one of the perceptron. It makes sense to have a signed-like function applied to a linear expression to make a decision.

We can use the model from LDA

$$f(x; w, b) = \sigma(w \cdot x + b)$$

but instead of modeling the densities and derive the values of w and b, directly compute them by maximizing their probability given the training data.

First, to simplify the next slide, note that we have

$$1 - \sigma(x) = 1 - \frac{1}{1 + e^{-x}} = \sigma(-x),$$

hence if Y takes value in $\{-1, 1\}$ then

$$\forall y \in \{-1, 1\}, \quad P(Y = y \mid X = x) = \sigma(y(w \cdot x + b)).$$

Notes

Here, we assume we no longer have the formal expression for the normal density, and we will directly attack the problem with maximum likelihood.

We have

$$\begin{aligned}
&\log \mu_{W,B}(w, b \mid \mathscr{D} = \mathbf{d}) \\
&\qquad = \log \frac{\mu_{\mathscr{D}}(\mathbf{d} \mid W = w, B = b)\, \mu_{W,B}(w, b)}{\mu_{\mathscr{D}}(\mathbf{d})} \\
&\qquad = \log \mu_{\mathscr{D}}(\mathbf{d} \mid W = w, B = b) + \log \mu_{W,B}(w, b) - \log Z \\
&\qquad = \sum_n \log \sigma(y_n(w \cdot x_n + b)) + \log \mu_{W,B}(w, b) - \log Z'
\end{aligned}$$

This is the **logistic regression**, whose loss aims at minimizing

$$-\log \sigma(y_n f(x_n)).$$

Notes

We want to compute the likelihood of the parameters w and b given the training data.
As we did previously, $\log \mu_{\mathscr{D}}(\mathbf{d})$ is constant w.r.t. w and b, so we can simplify it with $\log Z$.
We also have the assumption that the data points are independent and identically distributed, so

$$\begin{aligned}
&\log \mu_{\mathscr{D}}(\mathbf{d} \mid W = w, B = b) \\
&= \log \prod_n \mu(x_n, y_n \mid W = w, B = b) \\
&= \log \prod_n P(Y = y_n \mid X = x_n, W = w, B = b) \\
&\quad + \log \prod_n \mu_X(x_n \mid W = w, B = b) \\
&= \sum_n \log \sigma(y_n(w \cdot x_n + b)) + cst
\end{aligned}$$

So in the end, given training points, we cast the estimation of the parameters of the linear model as a Bayesian inference problem. Maximizing $\log \mu_{W,B}(w, b \mid \mathscr{D} = \mathbf{d})$ boils down to minimizing $-\log \sigma(y_n f(x_n))$.

- When the sample is well classified (the response of a sample is on the right), its loss is 0 and the sample is not penalized.
- When the sample is misclassified (the response of a sample is on the left), its loss increases linearly with how wrong it is.

Although the probabilistic and Bayesian formulations may be helpful in certain contexts, the bulk of deep learning is disconnected from such modeling.

We will come back sometime to a probabilistic interpretation, but most of the methods will be envisioned from the signal-processing and optimization angles.