Deep learning

3.2. Probabilistic view of a linear classifier

François Fleuret

https://fleuret.org/dlc/

Dec 20, 2020

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.

We have

$$
\begin{aligned}
P(Y=1 \mid X=x) &= \frac{\mu_{X \mid Y=1}(x) P(Y=1)}{\mu_X(x)} \\
&= \frac{\mu_{X \mid Y=1}(x) P(Y=1)}{\mu_{X \mid Y=0}(x) P(Y=0) + \mu_{X \mid Y=1}(x) P(Y=1)} \\
&= \frac{1}{1 + \frac{\mu_{X \mid Y=0}(x)}{\mu_{X \mid Y=1}(x)} \frac{P(Y=0)}{P(Y=1)}}.
\end{aligned}
$$

It follows that, with

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

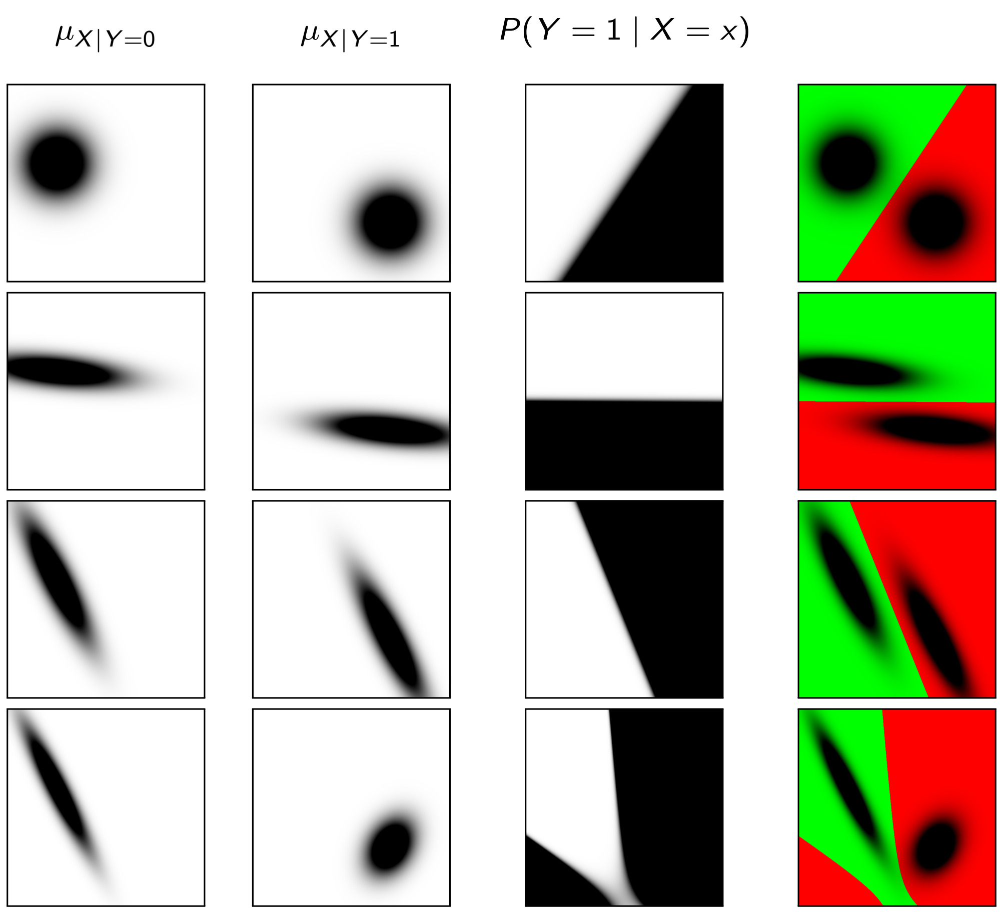
we get

$$
P(Y=1 \mid X=x) = \sigma\left(\log \frac{\mu_{X \mid Y=1}(x)}{\mu_{X \mid Y=0}(x)} + \log \frac{P(Y=1)}{P(Y=0)} \right).
$$

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

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

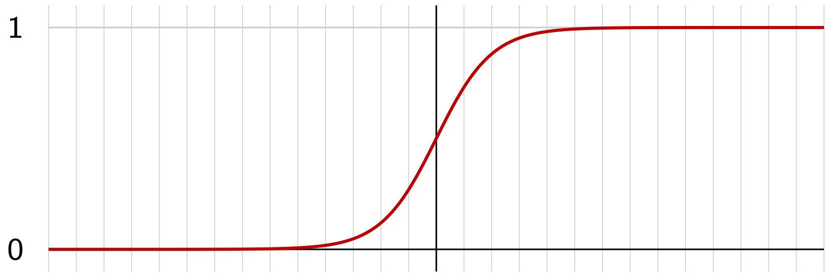
The homoscedasticity makes the second-order terms vanish.



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.

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

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

We have

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

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

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

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.