Contents

0.1 Naïve Bayes: Two Classes

- Naïve Bayes classifier gives a method for predicting the most likely class rather than an explicit class.
- In the case of two classes, $y \in \{0, 1\}$ we predict that $y = 1$ iff

$$\ldots$$

Take logarithm;

$$\log \frac{P(y_j = 1)}{P(y_j = 0)} + \sum_i \log \frac{1 - p_i}{1 - q_i} + \sum_i \left(\log \frac{p_i}{1 - p_i} - \log \frac{q_i}{1 - q_i} \right) x_i > 0$$

- We get that Naïve bayes is a linear separator with –

$$w_i = \log \frac{p_i}{1 - p_i} - \log \frac{q_i}{1 - q_i} = \log \frac{p_i(1 - q_i)}{q_i(1 - p_i)}$$

- In the case of two classes, we can say:

- but since $P(y_j = 1|x) = 1 - P(y_j = 0|x)$, we get:
$$P(y_j = 1|x) = \frac{1}{1 + e^{-\left(\sum_i w_i x_i + b\right)}}$$
- This is logistic regression

0.2 Generative Classifiers

If we are distinguishing cat from dog images using a Generative Classifier, we build a model of what is in a cat image.

- Knows about whiskers, ears, eyes.
- Assigns a probability to any image to determine how cat-like is that image?

Similarly, build a model of what is in a dog image. Now given a new image, run both models and see which one fits better.

0.3 Discriminative Classifiers

If we are distinguishing cat from dog images using a Discriminative Classifier.

- Just try to distinguish dogs from cats.
  - Oh look, dogs have collars.
  - Ignore everything else.

0.4 Generative vs Discriminative Classifiers

Generative Classifiers (Naïve Bayes) –

- Assume some functional form for conditional independence.
- Estimate parameters of $P(D|h)$, $P(h)$ directly from training data.
- Use Bayes rule to calculate $P(h|D)$.

Why not learn

0.5 Learning a Logistic Regression Classifier

Given n input-output pairs –

1. A feature representation of the input. For each input observation x_i, a vector of features $[x_1, x_2, \ldots, x_d]$.
2. A classification function that computes y, the estimated class, via $P(y|x)$, using the sigmoid of softmax functions.
3. An objective function for learning, like cross-entropy loss.
4. An algorithm for optimizing the objective function, like stochastic gradient ascent/descent.

0.6 Logistic Regression

Logistic Regression assumes the following function form for $P(y|x)$:

$$P(y=1|x) = \frac{1}{1+e^{-\left(\sum_i w_i x_i + b\right)}}$$

$$\begin{aligned}
P(y=1|x) &= \frac{1}{1+e^{-\left(\sum_i w_i x_i + b\right)}} \\
&= \frac{e^{\left(\sum_i w_i x_i + b\right)}}{e^{\left(\sum_i w_i x_i + b\right)}+1} \\
P(y=0|x) &= 1 - \frac{1}{1+e^{\left(\sum_i w_i x_i + b\right)}} \\
&= \frac{1}{e^{\left(\sum_i w_i x_i + b\right)}+1} \\
\frac{P(y=1|x)}{P(y=0|x)} &= e^{\left(\sum_i w_i x_i + b\right)} > 1 \\
&\Rightarrow \sum_i w_i x_i + b > 0
\end{aligned}$$

Logistic Regression is a linear classifier. Turning a probability into a classifier using the logistic function:

$$y_{LR} \begin{cases} 1 & \text{if } P(y=1|x) \geq 0.5 & \leftarrow w_i x_i + b \geq 0 \\ 0 & \text{otherwise} & \leftarrow w_i x_i + b < 0 \end{cases}$$

0.7 LR Example

0.8 Sentiment Classification

Let's assume for the moment that we've already learned a real-valued weight for each of these features, and that the 6 weights corresponding to the 6 features are $[2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$, while $b = 0.1$.

0.9 Training Logistic Regression

We'll focus on binary classification. We parameterize w_i, b as θ:

$$P(y_i = 0|x_i, \theta) = \frac{1}{e^{\sum_i w_i x_i + b} + 1}, P(y_i = 1|x_i, \theta) = \frac{e^{\sum_i w_i x_i + b}}{e^{\sum_i w_i x_i + b} + 1}, P(y_i|x_i, \theta) = \frac{e^{y_i \sum_i w_i x_i + b}}{e^{\sum_i w_i x_i + b} + 1}$$

How do we learn parameters θ?

0.10 Cross-Entropy Loss

- We want to know how far is the classifier output $\hat{y}$ from the true output y. Let's call this difference $L(\hat{y}, y)$.
- Since there are only 2 discrete outcomes (0 or 1), we can express the probability $P(y|x)$ from our classifiers as:
$$P(y|x) = \hat{y}^y \cdot (1 - \hat{y})^{1-y}$$
- Goal: maximize the probability of the correct label $P(y|x)$.
- Maximize:
$$\begin{aligned} P(y|x) &= \hat{y}^y \cdot (1 - \hat{y})^{1-y} \\ \log(P(y|x)) &= \log\left(\hat{y}^y \cdot (1 - \hat{y})^{1-y}\right) \\ &= y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}) \end{aligned}$$
- We want to minimize the cross-entropy loss:

0.11 Minimizing Cross-Entropy Loss

$$\min_{\theta} L_{CE}(\hat{y}, y)$$

- Minimizing loss function $L_{CE}(\hat{y}, y)$ is a convex optimization problem.

0.12 Optimizing a Convex/Concave Function

- Maximum of a concave function is equivalent to the minimum of a convex function.
- Gradient Ascent is used for finding the maximum of a concave function.
- Gradient Descent is used for finding the minimum of a convex function.

0.13 Gradients

- The gradient of a function is a vector pointing in the direction of the greatest increase in a function.

Gradient Ascent

Gradient Descent

0.14 Gradient Descent for Logistic Regression

- Let us represent $\hat{y} = f(x, \theta)$
- Gradient:

$$\nabla_\theta L(f(x,\theta),y) = \left[\frac{\partial L(f(x,\theta),y)}{\partial b}, \frac{\partial L(f(x,\theta),y)}{\partial w_1}, \frac{\partial L(f(x,\theta),y)}{\partial w_2}, \dots, \frac{\partial L(f(x,\theta),y)}{\partial w_d}\right] \tag{1}$$

- Update Rule:

$$\begin{aligned} \Delta\theta &= \eta \cdot \nabla_\theta L(f(x,\theta),y) \\ \theta_{t+1} &= \theta_t - \eta \cdot \frac{\partial}{\partial(w,b)} L(f(x,\theta),y) \end{aligned} \tag{2}$$

Gradient descent algorithm will iterate until $\Delta\theta < \epsilon$.

$$\begin{aligned} L_{CE}(f(x,\theta),y) &= \log\left(1 + e^{\sum_i w_i x_i + b}\right) - y\left(\sum_i w_i x_i + b\right) \\ \theta_{t+1} &= \theta_t - \eta \cdot \frac{\partial}{\partial(w,b)} L(f(x,\theta),y) \\ &= \theta_t - \eta \cdot x_i \left[\frac{e^{\sum_i w_i x_i + b}}{1 + e^{\sum_i w_i x_i + b}} - y\right] \\ &= \theta_t - \eta \cdot x_i \left[\hat{P}(y = 1 | x, \theta_t) - y\right] \end{aligned}$$

0.15 Learning Rate

- η is a hyperparameter.
- Large $\eta \Rightarrow$ Fast convergence but larger residual error. Also, possible oscillations.
- Small $\eta \Rightarrow$ Slow convergence but small residual error.