Chapter 4

Logistic Regression

Contents

4.1 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.

4.2 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.

4.3 Generative vs Discriminative Classifiers

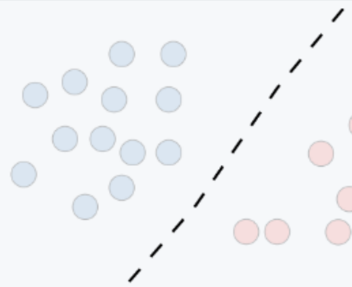
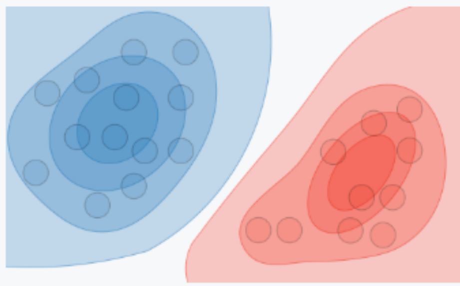
	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Figure 4.1: Differences between Generative and 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 $P(h|D)$ or the decision boundary directly? Discriminative Classifiers (Logistic Regression) –

- Assume some functional form for $P(h|D)$ or for the decision boundary.
- Estimate parameters of $P(h|D)$ directly from training data.

4.4 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.

4.5 Logistic Regression

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

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

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

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

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

4.6 LR Example

Suppose we are doing binary sentiment classification on movie review text, and we would like to know whether to assign the sentiment class position $= 1$ or negative $= 0$ to the following review:

It's hokey. There are virtually no surprises, and the writing is second-rate. So why was is so enjoyable? For one thing, the case is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you.

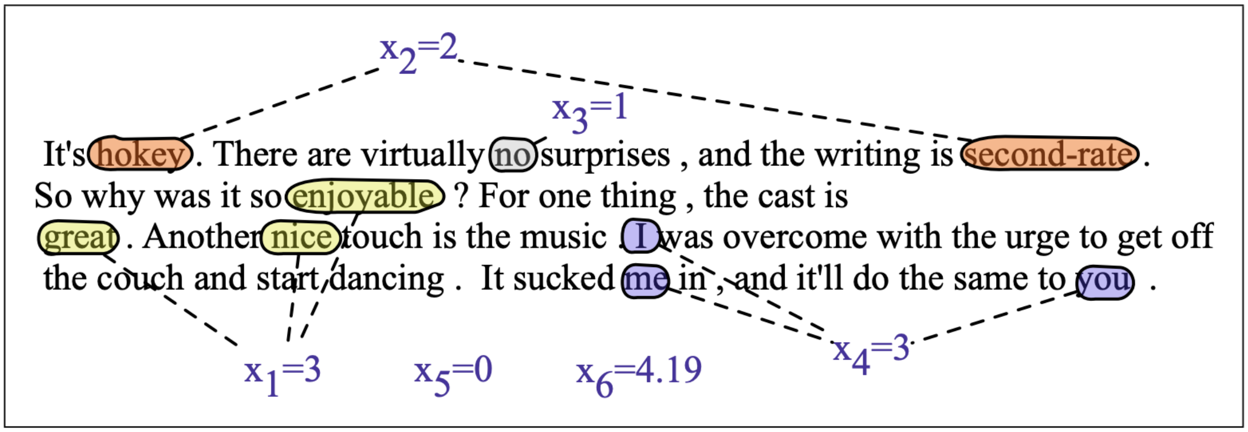


Figure 4.2: LR Example.

x_1	count(positive lexicon words $\in$ doc)	3
x_2	count(negative lexicon words $\in$ doc)	2
x_3	$\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	1
x_4	count(1st and 2nd pronouns $\in$ doc)	3
x_5	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	0
x_6	ln(word count of doc)	$\ln(66) = 4.19$

Figure 4.3: Feature vector for the LR Example.

4.7 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 $b = 0.1$.

$$
\begin{aligned}
P(+ve|x) &= P(y=1|x) \\
&= \frac{1}{1+e^{(\sum_i w_i x_i + b)}} \\
&= \frac{1}{1+e^{-(2.5(3)+(-5)(2)+(-1.2)(1)+0.5(3)+2.0(0)+0.7(4.19)+0.1)}} \\
&= 0.30 \\
P(-ve|x) &= P(y=0|x) \\
&= 1 - P(y=1|x) \\
&= 1 - 0.70 \\
&= 0.30
\end{aligned}
$$

Since $P(+ve|x) > P(-ve|x)$, the output sentiment class is positive.

4.8 Training Logistic Regression

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

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

How do we learn parameters θ?

4.9 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:

$$
\begin{aligned}
\textbf{Minimize}: L_{CE}(\hat{y}, y) &= -\log P(y|x) \\
&= -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})] \\
\min_\theta L_{CE}(\hat{y}, y) &= -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})] \\
&= -\left[y\log\left(\frac{e^{\sum_i w_i x_i + b}}{1+e^{\sum_i w_i x_i + b}}\right) + (1-y)\log\left(1 - \frac{e^{\sum_i w_i x_i + b}}{1+e^{\sum_i w_i x_i + b}}\right)\right] \\
&= -\left[y\left(\sum_i w_i x_i + b - \log\left(1+e^{\sum_i w_i x_i + b}\right)\right) + (1-y)\left(-\log\left(1+e^{\sum_i w_i x_i + b}\right)\right)\right] \\
&= -\left[y\left(\sum_i w_i x_i + b\right) - \log\left(1+e^{\sum_i w_i x_i + b}\right)\right] \\
&= \log\left(1+e^{\sum_i w_i x_i + b}\right) + y\left(\sum_i w_i x_i + b\right)
\end{aligned}
$$

4.10 Minimizing Cross-Entropy Loss

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

- Minimizing loss function $L_{CE}(\hat{y}, y)$ is a convex optimization problem.
- Convex function have a global minimum.
- Concave function have a global maxima.

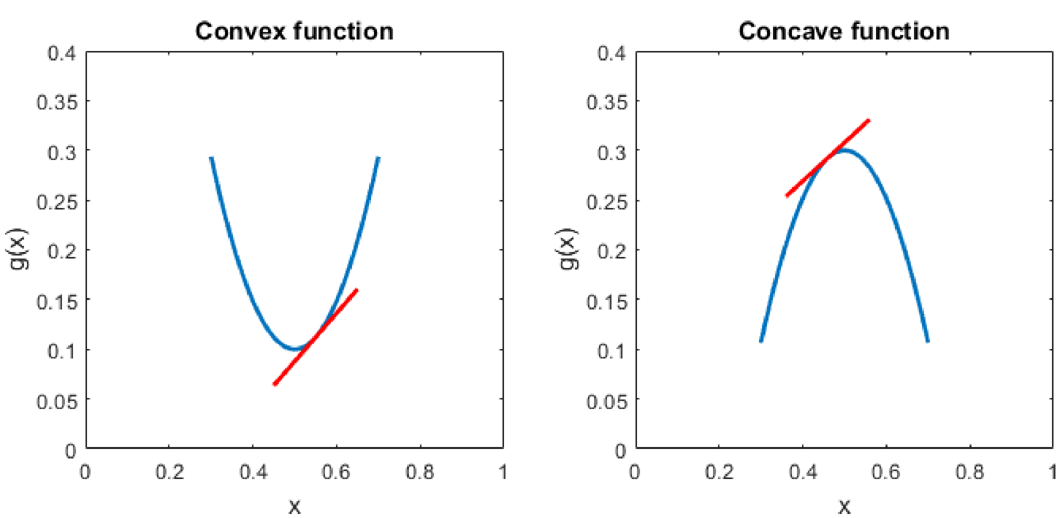


Figure 4.4: An example of a convec and concave function.

4.11 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.

4.12 Gradients

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

Gradient Ascent: Find the gradient of the function at the current point and move in the same direction.

Gradient Descent: Find the gradient of the function at the current point and move in the opposite direction.

4.13 Gradient Descent for Logistic Regression

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

$$\nabla_\theta L(f(x,\theta),y) = \left[\frac{\partial L(f(x,\theta),y)}{\partial b}, \frac{\partial L(f(x,\theta),y)}{\partial w_1}, \frac{\partial L(f(x,\theta),y)}{\partial w_2}, \dots, \frac{\partial L(f(x,\theta),y)}{\partial w_d}\right] \tag{4.1}$$

- Update Rule:

$$\begin{aligned} \Delta\theta &= \eta \cdot \nabla_\theta L(f(x,\theta),y) \\ \theta_{t+1} &= \theta_t - \eta \cdot \frac{\partial}{\partial(w,b)} L(f(x,\theta),y) \end{aligned} \tag{4.2}$$

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

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

4.14 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.

4.15 Batch Training

- Stochastic gradient descent is called stochastic because it chooses a single random example at a time, moving the weights to improve performance on that single example.
- This results in very choppy movements, so it's common to compute the gradient over batches of training instances rather than a single instance.
- Training data: $\{x_i, y_i\}_{i=i...n}$ where $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$, n is the total instances in a batch and d is the dimension of an instance.

$$\theta_{t+1} = \theta_t - \frac{\eta}{n} \times \sum_{i=1}^{n} xij \left[\frac{1}{1 + e^{-\theta^T \mathbf{x}}} - y_i \right] \tag{4.3}$$

4.16 Understanding the Sigmoid

- Large weights lead to overfitting.
- Penalizing larger weights can reduce overfitting.