Week 4: Logistic Regression

Instructor: Sergey Levine

1 Linear Classification via Logistic Regression

So far, we learned about two kinds of classifiers: naïve Bayes, which models and optimizes $p(\mathbf{x}, y)$, and decision trees, which can model $p(y|\mathbf{x})$, but do not actually optimize $p(y|\mathbf{x})$ (instead, we use a heuristic method that does not actually optimize any well-defined objective, but tends to work well in practice). Although naïve Bayes is flexible and optimizes a well-defined objective, its main weakness is the unreasonable independence assumption on the attributes, which can cause the method to fail in cases where these assumptions are strongly violated. For example, if we have two binary attributes and a binary label, and $y = 1$ whenever $x_1 = x_2$, and $y = 0$ whenever $x_1 \neq x_2$, naïve Bayes would fail completely: assuming the dataset is split evenly between the two labels, we would get $p(y = 1) = 0.5$, $p(x_1 = 1|y = 0) = 0.5$, $p(x_1 = 1|y = 1) = 0.5$, $p(x_2 = 1|y = 0) = 0.5$, and $p(x_2 = 1|y = 1) = 0.5$: nothing would be learned, and the classifier would simply guess each time.

Can we devise a classification method that both models and optimizes $p(y|\mathbf{x})$, without needing to model the relationship between different attributes or make unreasonable independence assumptions? We can take inspiration for linear regression, and devise a linear *classification* method. First, let's look at the intuition, and then we'll build a probabilistic interpretation.

In linear regression, we fit a function that is linear in some features $h(\mathbf{x})$,[1] given by $f(\mathbf{x}) = \mathbf{w} \cdot h(\mathbf{x})$. If we want to solve a binary classification problem, can we devise a classifier $g(\mathbf{x})$ such that $g(\mathbf{x}) = 1$ if $f(\mathbf{x}) > 0$, and $g(\mathbf{x}) = 0$ if $f(\mathbf{x}) \leq 0$? Such a linear classifier corresponds to drawing a line through the feature space $h(\mathbf{x})$ that separates the two classes.

If we want to build a probabilistic model $p(y|\mathbf{x})$, this kind of hard linear classifier is not very desirable. If we simply say $p(y = 1|\mathbf{x}) = 1$ if $f(\mathbf{x}) > 0$ and zero otherwise, then even a single mistake would give our entire dataset a probability of zero (and a log-likelihood of $-\infty$). The trouble with this approach is that it does not model *noise*: it assumes that our classifier can be perfect. Recall that in linear regression, we modeled noise by assuming that $p(y|\mathbf{x})$ was Gaussian. Can we construct a noise model for the classification case?

We would like this noise model to give equal probability to both labels when $f(\mathbf{x}) = 0$, for the probability of $y = 1$ to increase rapidly for $f(\mathbf{x}) > 0$, and

[1] Remember to include the bias feature 1!

decrease rapidly for $f(\mathbf{x}) < 0$. One good choice for a function that increases rapidly for positive values is the exponential. So we can try the following:

$$p(y = 1|\mathbf{x}) \propto \exp(\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w}).$$

We would like the classifier to be symmetric, so we'll choose

$$p(y = 0|\mathbf{x}) \propto \exp(-\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w}).$$

The normalizing constant is then given by $\exp(\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w}) + \exp(-\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w})$. We can simplify this a bit by noting that

$$p(y = 1|\mathbf{x}) = \frac{\exp(\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w})}{\exp(\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w}) + \exp(-\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w})} = \frac{\exp(\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w})^2}{\exp(\frac{1}{2}h(\mathbf{x}) \cdot \mathbf{w})^2 + 1} = \frac{\exp(h(\mathbf{x}) \cdot \mathbf{w})}{\exp(h(\mathbf{x}) \cdot \mathbf{w}) + 1}.$$

It then follows that

$$p(y = 0|\mathbf{x}) = 1 - \frac{\exp(h(\mathbf{x}) \cdot \mathbf{w})}{\exp(h(\mathbf{x}) \cdot \mathbf{w}) + 1} = \frac{\exp(h(\mathbf{x}) \cdot \mathbf{w}) + 1 - \exp(h(\mathbf{x}) \cdot \mathbf{w})}{\exp(h(\mathbf{x}) \cdot \mathbf{w}) + 1} = \frac{1}{\exp(h(\mathbf{x}) \cdot \mathbf{w}) + 1}.$$

Now we've recovered the model for logistic regression. The name comes from the fact that

$$\frac{\exp(h(\mathbf{x}) \cdot \mathbf{w})}{\exp(h(\mathbf{x}) \cdot \mathbf{w}) + 1} = \frac{1}{\exp(-h(\mathbf{x}) \cdot \mathbf{w}) + 1}$$

is the equation for the logistic function.

Let's recap:

Question. What is the data?

Answer. The dataset $\mathcal{D} = \{(\mathbf{x}^1, y^1), \ldots, (\mathbf{x}^N, y^N)\}$, where $y \in \{0, \ldots, L_y\}$ and $\mathbf{x}$ is a vector of K features. Note that each x_k may be either categorical or continuous: the logistic regression model doesn't care about this. Note, however, that if x_k is categorical, the model assumes a certain semantic relationship between sequential values of x_k. In practice, if we have a categorical value, we might choose to use a "one hot" encoding: if there are M possible values, we might introduce M attributes, such that only one of them is 1 for any record, and the rest are all 0.

Question. What is the hypothesis space?

Answer. The hypothesis space is defined by the weights vector $\mathbf{x} \in \mathbb{R}^{|h(\mathbf{x})|}$, which models

$$p(y = 1|\mathbf{x}) = \frac{1}{\exp(-h(\mathbf{x}) \cdot \mathbf{w}) + 1}$$

Question. What is the objective?

Answer. The objective is the conditional log likelihood:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} \log p(y^i | \mathbf{x}^i, \mathbf{w})$$

Question. What is the algorithm?

Answer. Let's try to figure it out. First, write down the log-likelihood:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) &= \sum_{i=1}^{N} \log p(y^i | \mathbf{x}^i, \mathbf{w}) \\
&= \sum_{i=1}^{N} \begin{cases} \text{if } y^i = 0\text{: } \log \frac{1}{\exp(h(\mathbf{x}^i)\cdot\mathbf{w})+1} \\ \text{if } y^i = 1\text{: } \log \frac{\exp(h(\mathbf{x}^i)\cdot\mathbf{w})}{\exp(h(\mathbf{x}^i)\cdot\mathbf{w})+1} \end{cases} \\
&= \sum_{i=1}^{N} \begin{cases} \text{if } y^i = 0\text{: } \log 1 - \log[\exp(h(\mathbf{x}^i)\cdot\mathbf{w}) + 1] \\ \text{if } y^i = 1\text{: } \log \exp(h(\mathbf{x}^i)\cdot\mathbf{w}) - \log[\exp(h(\mathbf{x}^i)\cdot\mathbf{w}) + 1] \end{cases} \\
&= \sum_{i=1}^{N} -\log[\exp(h(\mathbf{x}^i)\cdot\mathbf{w}) + 1] + \underbrace{\begin{cases} \text{if } y^i = 0\text{: } 0 \\ \text{if } y^i = 1\text{: } h(\mathbf{x}^i)\cdot\mathbf{w} \end{cases}}_{y^i h(\mathbf{x}^i)\cdot\mathbf{w}} \\
&= \sum_{i=1}^{N} \left(y^i h(\mathbf{x}^i)\cdot\mathbf{w} - \log[\exp(h(\mathbf{x}^i)\cdot\mathbf{w}) + 1] \right).
\end{aligned}$$

Now compute the gradient:

$$\begin{aligned}
\frac{d\mathcal{L}}{d\mathbf{w}} &= \sum_{i=1}^{N} \left(y^i h(\mathbf{x}^i) - \frac{h(\mathbf{x}^i)\exp(h(\mathbf{x}^i)\cdot\mathbf{w})}{\exp(h(\mathbf{x}^i)\cdot\mathbf{w}) + 1} \right) \\
&= \sum_{i=1}^{N} \left(h(\mathbf{x}^i) \left[y^i - \frac{\exp(h(\mathbf{x}^i)\cdot\mathbf{w})}{\exp(h(\mathbf{x}^i)\cdot\mathbf{w}) + 1} \right] \right).
\end{aligned}$$

Unfortunately, we can't simply set the gradient equal to zero and solve for $\mathbf{w}$ in closed form: there is actually no analytic formula for $\mathbf{w}$ in this case. Instead, we can use a procedure called gradient ascent (or gradient descent, if we instead minimize the negative log-likelihood).