10-715: Adv. Intro to Machine Learning **Fall 2015**

Recitation 1: September 15

Eric Wong

1.1 Biased Variance

Suppose we have $x_i \sim N(\mu, \sigma^2)$, and recall that $\hat{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i$ was an unbiased estimator for μ. We noticed in class that

Lemma 1.1 *The following MLE estimator for the variance of a Gaussian is biased:*

$$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

We can verify this as follows:

Proof:

$$\begin{aligned}
E[\hat{\sigma}^2_{MLE}] &= E\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2\right] \\
&= \frac{1}{n}E\left[\sum_{i=1}^{n} x_i^2 - n\hat{\mu}^2\right] \\
&= \frac{1}{n}\left(\sum_{i=1}^{n}(\sigma^2 + \mu^2) - n\left(\frac{\sigma^2}{n} + \mu^2\right)\right) \\
&= \frac{n-1}{n}\sigma^2
\end{aligned}$$

■

1.2 Naive Bayes

1.2.1 Restatement of the classifier

Now suppose we have observations $(y^{(i)}, x^{(i)})$ with $x^{(i)} \in \mathbb{R}^k$ and we wish to predict a value for unobserved x^*. Recall the naive bayes classifier:

$$y^\star(x^\star) = \arg\max_y P(y)\prod_{i=1}^{k} P(x_i^*|y)$$

- Incomplete data: What if data (i.e. some feature) is missing? Simply ignore them in the counts.

- Underflow: How to deal with very small probabilities? Use the log probabilities instead.
- Feature selection: How do we deal with overfitting of large amounts of data? Can filter by probability, even better use mutual information $I(X;Y) = \sum_{x,y} P_{XY}(x,y) \log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)}$
- Correlation: What if features are highly correlated (a.k.a. the assumptions are violated)? Features get voted multiple times and skew the predictions.

1.3 Smoothing the Naive Bayes Classifier

How do we estimate P? Using the relative frequencies and counting the number of examples:

$$\hat{P}(y) = \frac{\{\#j : y^{(j)} = y\}}{n}, \quad \hat{P}(x_i, y) = \frac{\{\#j : x_i^{(j)} = x_i, y^{(j)} = y\}}{n}$$

$$\hat{P}(x_i|y) = \frac{\hat{P}(x_i, y)}{\hat{P}(y)} = \frac{\{\#j : x_i^{(j)} = x_i, y^{(j)} = y\}}{\{\#j : y^{(j)} = y\}}$$

1.3.1 Laplace smoothing

If feature i don't appear for label y in the training dataset, then the estimate for $P(x_i^*|y) = 0$. Recall that we can deal with this by simply adding a prior that adds 1 to every count. If feature x_i can attain m_i distinct values, then:

$$\hat{P}(x_i|y) = \frac{\{\#j : x_i^{(j)} = x_i, y^{(j)} = y\} + 1}{\{\#j : y^{(j)} = y\} + m_i}$$

1.3.2 Dirichlet prior smoothing

We can generalize the above. What if the prior adds more than 1 to every count, say α? Then we get Dirichlet prior smoothing:

$$\hat{P}(x_i|y) = \frac{\{\#j : x_i^{(j)} = x_i, y^{(j)} = y\} + \alpha}{\{\#j : y^{(j)} = y\} + \alpha m_i}$$

1.4 Continuous features

If the features x_i are instead continuous, then we can model the conditional probability as a Gaussian, and estimate the mean and variance (i.e. using MLE estimation) using the corresponding subset of feature data:

$$\hat{P}(x_i|y) = \mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$$

where $\hat{\mu}, \hat{\sigma}$ are estimated from feature set $\{x_i^{(j)} : y^{(j)} = y,\ \forall j\}$

1.5 Perceptrons

1.5.1 Update Rule is Stochastic Gradient descent

In lecture we measured error as

$$y^{(i)}[\langle w, x^{(i)} \rangle + b] < 0$$

Many learning algorithms can be seen as minimizing some loss function. For example, recall the hinge loss function, for intended output $t \in \{-1, 1\}$:

$$l(y) = \max(0, 1 - t \cdot y)$$

In the context of the perceptron algorithm, this is

$$L(x^{(i)}, y^{(i)}, w, b) = \sum_i \max(0, 1 - y^{(i)}[\langle w, x^{(i)} \rangle + b])$$

In stochastic gradient descent, we perform a gradient update on some random index i. Derive the gradient to see that the update rule for the perceptron algorithm is exactly the same as the stochastic gradient descent rule, which is 0 if the ith data point is classified correctly, and otherwise $b \leftarrow b + y^{(i)}$ and $w \leftarrow w + y^{(i)}x^{(i)}$.

1.5.2 Acceleration of Perceptron algorithm

The perceptron algorithm could take many iterations to converge. One way to speed it up is to adjust the weight of update to guarantee that the ith value will be classified correctly [1].

If $x^{(i)}$ is incorrectly classified, then recall that

$$y^{(i)} \langle w, x^{(i)} \rangle < 0$$

Consider the case when $y^{(i)} = 1$, and so $\langle w, x^{(i)} \rangle < 0$. Then, define an error δ as

$$\delta = -\langle w, x^{(i)} \rangle$$

Using this, define a new update rule as

$$w^* \leftarrow w + \frac{\delta + \epsilon}{||x^{(i)}||^2} x^{(i)}$$

Why does w^* classify $x^{(i)}$ correctly?

$$\begin{aligned} \langle w^*, x^{(i)} \rangle &= \langle w + \frac{\delta + \epsilon}{||x^{(i)}||^2} x^{(i)}, x^{(i)} \rangle \\ &= \langle w, x^{(i)} \rangle + \delta + \epsilon \\ &= \epsilon > 0 \end{aligned}$$

References

[1] Rojas, Ral. Neural networks: a systematic introduction. Springer Science & Business Media, 2013.