LaTeX command declarations here.

EECS445 Machine Learning

Discussion 03: Linear Regression & Naive Bayes

Written by Zhao Fu; Edited by Chansoo Lee

Linear Regression: Notations

- Let vector $\mathbf{x}_n \in \mathbb{R}^D$ denote the $n\text{th}$ data. D denotes number of attributes in dataset.
- Let vector $\phi(\mathbf{x}_n) \in \mathbb{R}^M$ denote features for data $\mathbf{x}_n$. $\phi_j(\mathbf{x}_n)$ denotes the $j\text{th}$ feature for data x_n.
- Feature $\phi(\mathbf{x}_n)$ is the *artificial* features which represents the preprocessing step. $\phi(\mathbf{x}_n)$ is usually some combination of transformations of $\mathbf{x}_n$. For example, $\phi(\mathbf{x})$ could be vector constructed by $[\mathbf{x}_n^T, \cos(\mathbf{x}_n)^T, \exp(\mathbf{x}_n)^T]^T$. If we do nothing to $\mathbf{x}_n$, then $\phi(\mathbf{x}_n) = \mathbf{x}_n$.
- Continuous-valued label vector $t \in \mathbb{R}^N$ (target values). $t_n \in \mathbb{R}$ denotes the target value for $i\text{th}$ data.

Recall: Least Squares Error Function

- We will find the solution $\mathbf{w}$ to linear regression by minimizing a cost/objective function.
- When the objective function is sum of squared errors (sum differences between target t and prediction y over entire training data), this approach is also called **least squares**.
- The objective function is

$$
\begin{aligned}
E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}(y(\mathbf{x}_n, \mathbf{w}) - t_n)^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}\left(\sum_{j=0}^{M-1} w_j\phi_j(\mathbf{x}_n) - t_n\right)^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\right)^2
\end{aligned}
$$

Exercise 3.1

Consider a data set in which each data point t_n is associated with a weighting factor $r_n > 0$, so that the sum-of-squares error function becomes

$$E(\mathbf{w}; \mathbf{r}) = \frac{1}{2} \sum_{n=1}^{N} r_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) - t_n\right)^2$$

Find an expression for the solution $\mathbf{w}^*$ that minimizes this error function. Give two alternative interpretations of the weighted sum-of-squares error function in terms of (i) data dependent noise variance and (ii) replicated data points.

Solution (Matrix Calculus)

We can write the above error function as $E(\mathbf{w}; \mathbf{r}) = \frac{1}{2} \|S(\Phi \mathbf{w} - t)\|_2^2$ where S is an N-by-N diagonal matrix with entries $\sqrt{r_n}$. Note $S^\top S = SS^\top$ is a diagonal matrix (N-by-N) with entries r_n. We denote $R = S^\top S$. We also know that

Similarly to hands-on lecture 4, $\frac{1}{2} \|S(\Phi \mathbf{w} - t)\|_2^2 = \frac{1}{2}(\mathbf{w}^\top (S\Phi)^\top S^\top (S\Phi)\mathbf{w} - 2\mathbf{w}^\top \Phi^\top S^\top St - t^\top t)$, and thus

$$\frac{\partial E}{\partial \mathbf{w}} = \Phi^\top R \Phi \mathbf{w} - \Phi^\top Rt.$$

By setting it equal to 0, we have the closed form solution $\mathbf{w} = (\Phi^\top R \Phi)^{-1} \Phi^\top Rt$

Model Selection: Cross Validation

Suppose we are using the following linear regression model to fit a dataset

$$h_\theta(x) = \sum_{i=0}^{k} \theta_i \phi_i(x)$$

where $\phi_i(x) = x^i$, and wish to decide if k should be $0, 1, \ldots,$ or 10. How can we automatically select a model?

For the sake of concreteness, we assume we have some finite set of models $M = \{M_1, \ldots, M_d\}$ that we're trying to select among. For instance, in our first example above, the model M_i would be an ith order polynomial regression model.

Model Selection: Hold-out Cross Validation

Suppose we have a training dataset S.

In hold-out cross validation(also called simple cross validation), we do the following:

1. Randomly split S into S_{train} (say, 70% of the data) and S_{cv} (the remaining 30%). Here, S_{cv} is called the hold-out cross validation set.
2. Train each model M_i on S_{train} only, to get some hypothesis.
3. Select and output the hypothesis that had the smallest error $\epsilon_{S_{cv}}$ on the hold out cross validation set. (Recall, $\epsilon_{S_{cv}}$ denotes the empirical error on the set of examples in Scv.)

Model Selection: K-Fold Cross Validation

Here is a method, called k-fold cross validation, that holds out less data each time:

1. Randomly split S into k disjoint subsets of m/k training examples each. Let's call these subsets $S_1, \dots, S_k$.
2. For each model M_i, we evaluate it as follows: For $j = 1, \dots, k$ Train the model M_i on $S_1 \cup \cdots \cup S_{j-1} \cup S_{j+1} \cup \cdots S_k$ (i.e., train on all the data except S_j) to get some hypothesis h_{ij}. Test the hypothesis h_{ij} on S_j, to get $\epsilon_{S_j}(h_{ij})$. The estimated generalization error of model M_i is then calculated as the average of the $\epsilon_{S_j}(h_{ij})$, which is $\frac{1}{k}\sum_j \epsilon_{S_j}(h_{ij})$.
3. Pick the model M_i with the lowest estimated generalization error, and retrain that model on the entire training set S. The resulting hypothesis is then output as our final answer.

Bayesian Linear Regression

Recall **Maximum Likelihood Estimator(MLE)** for least square regression:

Given $\{\mathbf{x}_n, t_n\}_{n=1}^N$, we want to find $\mathbf{w}_{ML}$ that maximizes data likelihood function

$$\mathbf{w}_{ML} = \arg\max p(\mathbf{t}|\mathcal{X}, \mathbf{w}, \beta) = \arg\max \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1})$$

and by derivation we have shown in lecture $\mathbf{w}_{ML}$ is equivalent to the least squares solution $\hat{\mathbf{w}} = \Phi^{\dagger}\mathbf{t}$.

Bayesian Linear Regression

Recall **MAP Estimator** for least square regression:

$$
\begin{aligned}
\mathbf{w}_{MAP} &= \arg\max p(\mathbf{w}|\mathbf{t}, \mathcal{X}, \beta) && \text{(Posteriori Probability)} \\
&= \arg\max \frac{p(\mathbf{w}, \mathbf{t}, \mathcal{X}, \beta)}{p(\mathcal{X}, t, \beta)} \\
&= \arg\max \frac{p(\mathbf{t}|\mathbf{w}, \mathcal{X}, \beta)p(\mathbf{w}, \mathcal{X}, \beta)}{p(\mathcal{X}, t, \beta)} \\
&= \arg\max p(\mathbf{t}|\mathbf{w}, \mathcal{X}, \beta)p(\mathbf{w}, \mathcal{X}, \beta) && (p(X, t, \beta) \text{ is irrelevant to } \mathbf{w}) \\
&= \arg\max p(\mathbf{t}|\mathbf{w}, \mathcal{X}, \beta)p(\mathbf{w})p(\mathcal{X})p(\beta) && \text{(Independence)} \\
&= \arg\max p(\mathbf{t}|\mathbf{w}, \mathcal{X}, \beta)p(\mathbf{w}) && \text{(Likelihood} \times \text{Prior)}
\end{aligned}
$$

Here, we assume $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \alpha^{-1}I)$.

Exercise 3.2:

Suppose that we have already observed N data points, the posterior distribution over $\mathbf{w}$ can be regarded as the prior for the next observation. So we can predict the next data point $(\mathbf{x}_{N+1}, t_{N+1})$ by maximize $p(t_{N+1}|\mathbf{x}_{N+1}, \mathbf{X}, \alpha, \beta)$. Derive the full expression for the posterior of the new data point.

We can also predict the expectation value of t_{N+1} given $\mathbf{x}_{N+1}$ by $\mathbb{E}[t_{N+1}|\mathbf{x}_{N+1}]$ using the posterior probability.

Review: Naive Bayes

- The essence of Naive Bayes is the **conditionally independence assumption**

$$P(\mathbf{x}|y=c)=\prod_{d=1}^{D}P(x_d|y=c)$$

  i.e., given the label, all features are independent.
- The **full generative** model of Naive Bayes is:

$$y \sim \text{Categorical}(\pi)$$
$$x_d|y=c \sim \text{Categorical}(\theta_{cd}) \quad \forall\, d=1,\dots,D$$

  with parameters:
  - $P(y=c)=\pi_c, \forall c=1,\dots,C$
  - $\sum_{c=1}^{C}\pi_c=1$ and $\pi_c \geq 0, \forall c=1,\dots,C$
  - $P(x_d=m|y=c)=\theta_{cdm}$ for every $d=1,\dots,D, m=1,\dots,M, c=1,\dots,C$
  - $\sum_{m=1}^{M}\theta_{cdm}=1$
- Conditional indepence assumption
  - Conditional independence: for any $i \neq j$, $P(X_i|X_j,Y)=P(X_i|Y)$
  - The implication is: $P(X_1,\dots,X_n|Y)=P(X_1|X_2,\dots,X_n,Y)P(X_2,\dots,X_n|Y)$.
  - By the Bayes theorem,

$$P(Y|X_1,\dots,X_n)=\frac{P(Y)}{P(X_1,\dots,X_n)}P(X_1,\dots,X_n|Y)=\frac{P(Y)\prod_i P(X_i|Y)}{P(X_1,\dots,X_n)}$$

  - When computing the MAP estimate $P(Y|X_1,\dots,X_n)$, we can simply compare the numerator.
- Parameter π and θ are learned from training data.
  - $\hat{\pi}_c=\frac{N_c}{N}$
  - $\hat{\theta}_{cdm}=\frac{N_{cdm}}{N_c}$

Common problems

1. What if not all the words appear in a category, in which case we have $N_{cdm}=0$?
2. What if some attributes are continues variables, not discrete?