STAT 542: Homework 4

Ahmadreza Eslaminia (ae15)

Please make sure that your solutions are readable and the file size is reasonable. Typing the answers is highly encouraged.

Problem 1.

[2pts] Derive the forward and backward propagation equations for the cross-entropy loss function.

Hint: The gradient update equations for the sum of squares loss can be found in $p396$ of Elements of Statistical Learning, which is used for regression tasks. Here we consider the classification task and replace the sum of squares loss by the cross-entropy loss. You should highlight in your solution what parts have changed.

Solution 1.

Building on the solution provided on page 396 for sum of squares loss, we now derive the forward and backward propagation equations for the neural network classification task using the cross-entropy loss function.

Cross-Entropy Loss Function

The cross-entropy loss for a multi-class classification problem is defined as:

$$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} \log f_k(x_i) \tag{1}$$

where y_{ik} is a binary indicator of whether class k is the correct classification for observation i.

Forward Propagation

The forward propagation computes the predicted probabilities for each class based on the input data and the current model weights.

Given an input vector x_i, the logits for class k are computed as:

$$z_{ik} = \beta_{0k} + \sum_{m=1}^{M} \beta_{km}\sigma\left(\alpha_{0m} + \sum_{l=1}^{P} \alpha_{ml} x_{il}\right) \tag{2}$$

where σ is the activation function for the hidden layer, and α and β are the weights connecting the input to the hidden layer and the hidden layer to the output layer, respectively.

The softmax function then provides the predicted probabilities for each class:

$$f_k(x_i) = \frac{\exp(z_{ik})}{\sum_{j=1}^{K} \exp(z_{ij})} \tag{3}$$

Backward Propagation

In the backward propagation step, we compute the gradients of the cross-entropy loss with respect to the network weights.

Gradient with Respect to Output Layer Weights β_{km}

For the weights β_{km} connecting the hidden layer to the output layer, the gradient under cross-entropy loss is:

$$\frac{\partial R}{\partial \beta_{km}} = -\sum_{i=1}^{N} (y_{ik} - f_k(x_i)) \sigma \left(\alpha_{0m} + \sum_{l=1}^{P} \alpha_{ml} x_{il} \right) \tag{4}$$

Gradient with Respect to Hidden Layer Weights α_{ml}

For the weights α_{ml} connecting the input layer to the hidden layer, the gradient is:

$$\frac{\partial R}{\partial \alpha_{ml}} = -\sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) \beta_{km} \sigma' \left(\alpha_{0m} + \sum_{l=1}^{P} \alpha_{ml} x_{il} \right) x_{il} \tag{5}$$

where σ' denotes the derivative of the activation function σ.

Gradient Descent Update Rules

With the computed gradients, we update the weights according to the gradient descent update rules:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma \frac{\partial R}{\partial \beta_{km}} \tag{6}$$

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} - \gamma \frac{\partial R}{\partial \alpha_{ml}} \tag{7}$$

where γ is the learning rate.

The complete expansions for the backpropagation updates with the cross-entropy loss function are as follows. For the output layer weights β_{km}, the update rule is:

$$\beta_{km}^{(r+1)} = \beta_{km}^{(r)} + \gamma \sum_{i=1}^{N} (y_{ik} - f_k(x_i)) \sigma(z_{mi}) \tag{8}$$

And for the hidden layer weights α_{ml}, the update rule is:

$$\alpha_{ml}^{(r+1)} = \alpha_{ml}^{(r)} + \gamma \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i)) \beta_{km} \sigma'(z_{mi}) x_{il} \tag{9}$$

These update rules are derived from the gradients of the cross-entropy loss, where $\sigma(z_{mi})$ is the activation of hidden unit m for input i, $\sigma'(z_{mi})$ is the derivative of the activation function, and γ is the learning rate. The positive sign before the learning rate indicates that the negative of the gradient has been taken into account.

This completes the derivation of the forward and backward propagation equations for the cross-entropy loss function.

Problem 2.

[2pts] In a K-class classification problem, suppose that for given neural network parameter (weights) θ and input x, the output of the network are functions $f_k(x, \theta), k = 1, \ldots,$ which are nonnegative numbers that sum to 1 . Show that minimizing the cross-entropy loss is equivalent to maximum likelihood estimation, if $f_k(x, \theta)$ is interpreted as the likelihood $P(Y = k \mid x, \theta)$.

Solution 2.

Cross-Entropy Loss

The cross-entropy loss for a multi-class classification problem is defined as follows:

$$\mathcal{L}(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(f_k(x_i, \theta)) \tag{10}$$

where y_{ik} is a one-hot encoded vector with a 1 indicating the correct class for observation i, and $f_k(x_i, \theta)$ is the predicted probability for class k given the parameters θ.

Likelihood Interpretation

We interpret $f_k(x_i, \theta)$ as the likelihood $P(Y = k|x_i, \theta)$, which denotes the probability of the true class being k given the observation x_i and parameters θ. The likelihood for all observations is then:

$$L(\theta) = \prod_{i=1}^{N} \prod_{k=1}^{K} f_k(x_i, \theta)^{y_{ik}} \tag{11}$$

Log-Likelihood

The log-likelihood can be derived by taking the natural logarithm of the likelihood:

$$\log L(\theta) = \log \left(\prod_{i=1}^{N} \prod_{k=1}^{K} f_k(x_i, \theta)^{y_{ik}} \right) = \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(f_k(x_i, \theta)) \tag{12}$$

Maximizing the Log-Likelihood

Maximizing the log-likelihood is equivalent to minimizing the negative of it. Therefore, the objective function for maximum likelihood estimation becomes:

$$-\log L(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log(f_k(x_i, \theta)) \tag{13}$$

This is identical to the cross-entropy loss function $\mathcal{L}(\theta)$, implying that minimizing $\mathcal{L}(\theta)$ is equivalent to maximizing $L(\theta)$.

Problem 3.

[3pts] In the "comparison of test errors" slide we saw that the test error does not decrease further as the number of bagged trees exceeds a certain point. Give an explicit calculation to verify this in the example of linear statistics.

Hint: see exercise 15.4 in Elements of Statistical Learning.

Solution 3.

We aim to verify why the test error does not further decrease as the number of bagged trees in a model exceeds a certain point. The verification will be in the context of linear statistics, as guided by Exercise 15.4 in *The Elements of Statistical Learning*.

Given the random variables x_i, $i = 1, \dots, N$, which are independent and identically distributed with mean μ and variance σ^2, and the bootstrap samples $\tilde{x}_i^*$ and $\hat{x}_i^*$, we denote their sample means by $\tilde{x}^*$ and $\hat{x}^*$, respectively.

For the bootstrap means, we have the following properties:

$$E[\tilde{x}^*] = E[\hat{x}^*] = \mu,$$

$$\mathrm{Var}(\tilde{x}^*) = \mathrm{Var}(\hat{x}^*) = \frac{(2n-1)}{n^2}\sigma^2,$$

$$\mathrm{Cov}(\tilde{x}^*, \hat{x}^*) = \frac{\sigma^2}{n}.$$

The covariance of the sample means from the bootstrap samples is:

$$\mathrm{Cov}(\tilde{x}^*, \hat{x}^*) = \frac{1}{n^2} \sum_{i,j} \mathrm{Cov}(\tilde{x}_i, \hat{x}_j) = \frac{\sigma^2}{n}. \tag{14}$$

For the variance of a single bootstrap sample mean, we have:

$$\begin{aligned}
\mathrm{Var}(\tilde{x}^*) &= \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n}\tilde{x}_i\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(\tilde{x}_i) + \frac{1}{n^2}\sum_{i\neq j}\mathrm{Cov}(\tilde{x}_i, \tilde{x}_j) \\
&= \frac{1}{n^2}\left(n\cdot\sigma^2 + n(n-1)\frac{\sigma^2}{n}\right) \\
&= \frac{(2n-1)\sigma^2}{n^2}.
\end{aligned}$$

Thus, the correlation between two bootstrap sample means is:

$$\mathrm{Corr}(\tilde{x}^*, \hat{x}^*) = \frac{\mathrm{Cov}(\tilde{x}^*, \hat{x}^*)}{\sqrt{\mathrm{Var}(\tilde{x}^*)\mathrm{Var}(\hat{x}^*)}} = \frac{n}{2n-1}. \tag{15}$$

Moving on to the bagged estimator $\bar{x}_{\mathrm{bag}}$ which is the average of B bootstrap sample means, the variance is:

$$\begin{aligned}
\mathrm{Var}(\bar{x}_{\mathrm{bag}}) &= \mathrm{Var}\left(\frac{1}{B}\sum_{i=1}^{B}\tilde{x}_i^*\right) \\
&= \frac{1}{B^2}\sum_{i=1}^{B}\mathrm{Var}(\tilde{x}_i^*) + \frac{1}{B^2}\sum_{i\neq j}\mathrm{Cov}(\tilde{x}_i^*, \tilde{x}_j^*) \\
&= \frac{1}{B}\frac{(2n-1)\sigma^2}{n^2} + \frac{B-1}{B}\frac{\sigma^2}{n}.
\end{aligned}$$

For large B, the first term $\frac{2n-1}{Bn^2}\sigma^2$ becomes negligible, and the variance asymptotically approaches $\frac{\sigma^2}{n}$. This indicates that after a certain number of bagged trees, additional trees do not contribute to a reduction in variance. Since the test error is a function of variance for linear statistics, the test error will also not decrease significantly with more bagged trees beyond this point.

This result demonstrates the property of bagging in the context of linear statistics: the reduction in variance, and consequently the test error, has diminishing returns as the number of bagged trees increases. It provides an analytical explanation for the empirical observations made in the "comparison of test errors" slide regarding bagged trees.