Machine Learning 1 - Homework assignment 3

Available: Tuesday, November 13th, 2018
Deadline: Thursday 23.59, November 29th, 2018

General instructions
Unless stated otherwise, write down a derivation of your solutions. Solutions presented without a derivation that shows how the solution was obtained will not be awarded with points.

1 Naive Bayes Text Classification

Naive Bayes (NB) is a particular form of classification that makes strong independence assumptions regarding the features of the data, conditional on the classes (see Bishop section 4.2.3). Specifically, NB assumes each feature is independent given the class label. In contrast, when we looked at probabilistic generative models for classification in the lecture, we used a full-covariance Gaussian to model data from each class, which incorporates correlation between all the input features (i.e. they are not conditionally independent).

If correlated features are treated independently, the evidence for a class will be overcounted. However, Naive Bayes is very simple to construct, because by ignoring correlations the *class-conditional likelihood*, $p(\mathbf{x}|\mathcal{C}_k)$, is a product of D univariate distributions, each of which is simple to learn:

$$p(\mathbf{x}|\mathcal{C}_k) = \prod_{d=1}^{D} p(x_d|\mathcal{C}_k) \tag{1}$$

Consider a document classification task, that classifies your documents into K classes $\mathcal{C}_k$. To do this you first make a *bag-of-words* (BoW) representation of your entire training set. A BoW is a vector x_n of dimension D for each document indicating whether each word in the vocabulary appears in the document (i.e. the words go into a bag and are shaken, losing their order so only their presence matters). This means that $x_{ni} = 1$ if word i is present in document n, $x_{ni} = 0$ otherwise. You can think of D as the vocabulary size of the training set, but it may also contain tokens or special features. Your training set therefore consists of an N by D matrix of word counts $\mathbf{X}$, and the target matrix $\mathbf{T}$, who's rows consist of the row vectors $\mathbf{t}_n^T = (t_{n1}, \dots, t_{nk})$, one-hot-encoded such that $\mathbf{t}_n^T = (0, \dots, 1, \dots, 0)$ with the scalar 1 at postion i if $n \in \mathcal{C}_i$. Assume we know $p(\mathcal{C}_i) = \pi_i$ (with the constraint $\sum_{i=1}^K \pi_i = 1$). We can model the word counts using different distributions (in the practice homework we modeled it with a Poisson distribution), for this question we will use a Bernoulli distribution model, hence each word is distributed according to a Bernoulli distribution with parameter θ_{dk}, when conditioned on class $\mathcal{C}_k$:

$$p(\mathbf{x}|\mathcal{C}_k, \theta_{1k}, \dots, \theta_{Dk}) = \prod_{d=1}^{D} \theta_{dk}^{x_d}(1 - \theta_{dk})^{1-x_d}$$

with distribution parameters $\theta_{dk} = P(x_d = 1|\mathcal{C}_k)$.

With this information answer the following questions:

1. (2 points) Write down the data likelihood, $p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta})$ without independence assumptions. Now derive the data likelihood for the *general k* classes naive Bayes classifier, stating where you make use of the product rule and the naive Bayes assumption. You should write the likelihood in terms of $p(x_d|\mathcal{C}_k)$, meaning you should not assume the explicit Bernoulli distribution.

2. (0.5 points) How does the number of parameters change if you make use of the naive Bayes assumption? Why is the assumption called *naive* and can you think of an example in which this assumption does not hold?

3. (0.5 points) Write down the data log-likelihood $\ln p(\mathbf{T}, \mathbf{X}|\boldsymbol{\theta})$ for the Bernoulli model.

4. (4 points) Solve for the MLE estimators for θ_{dk}. Express in your own words how the result can be interpreted.

5. (1 point) Write $p(\mathcal{C}_1|\mathbf{x})$ for the *general* k classes naive Bayes classifier.

6. (1 point) Write $p(\mathcal{C}_1|\mathbf{x})$ for the Bernoulli model.

7. (2 points) For the Bernoulli model, express the conditions (inequalities) of the region where $\mathbf{x}$ is predicted to be in $\mathcal{C}_1$. Provide linear inequalities in the form $\mathbf{x}^T\mathbf{a} > c$.

8. (0.5 points) (Bonus) So far we only considered feature vectors with discrete values, is it possible to have continuous features? Argue why yes/no and if yes, what would be an example?

2 Multi-class Logistic Regression and Multilayer Perceptrons

In class we saw the binary classification version of logistic regression. Here you will derive the gradients for the general case $K > 2$. Much of the preliminaries are in Bishop 4.3.4. This will be useful for Lab 2.

For $K > 2$ the posterior probabilities take a generalized form of the sigmoid called the softmax:

$$y_k(\boldsymbol{\phi}) = p(\mathcal{C}_k|\boldsymbol{\phi}) = \frac{\exp(a_k)}{\sum_i \exp(a_i)}$$

where $a_k = \mathbf{w}_k^T\boldsymbol{\phi}$ and $\boldsymbol{\phi}$ is short for $\boldsymbol{\phi}(\mathbf{x}) = (\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), ..., \phi_{M-1}(\mathbf{x}))^T$ with $\phi_0(\mathbf{x}) = 1$. Note that the posterior for class k depends on all the other classes i; keep this in mind when working out the derivatives for $\mathbf{w}_k$. The training set is a pair of matrices $\mathbf{\Phi}$ and $\mathbf{T}$. Each row of $\mathbf{T}$ uses a one-hot encoding of the class labeling for that training example, meaning that the n-th row contains a row vector $\mathbf{t}_n^T$ with all entries zero except for the k-th entry which is equal to 1 if datapoint n belongs to class $\mathcal{C}_k$.

Answer the following questions:

1. (3 points) In this question we will be deriving the matrix form for the gradient of the log-likelihood $\nabla_{\mathbf{w}_j} \ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K)$. The steps to follow in order to compute the gradient are:

   - Write down the likelihood $p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K)$. Use the entries of $\mathbf{T}$ as selectors of the correct class.
   - Write down the log-likelihood $\ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K)$.
   - Compute the derivative $\frac{\partial y_k}{\partial \mathbf{w}_j}$.
   - Use the chain rule and the derivatives $\frac{\partial y_k}{\partial \mathbf{w}_j}$ to compute $\nabla_{\mathbf{w}_j} \ln p(\mathbf{T}|\mathbf{\Phi}, \mathbf{w}_1, ..., \mathbf{w}_K)$
   - Your gradient is now a sum, can you rewrite it as a matrix multiplication? Why is this useful?

   Note that we are following the convention that both derivatives and gradients with respect to a column vector are row vectors.

2. (1 point) Write down the negative log-likelihood, this will be our objective function. Can you recognize the function you obtain? What is the relationship between the two (the function you obtain and the original log-likelihood)? What changes in terms of weight updates during optimization?

3. (2 points) Write a mini-batch gradient algorithm for logistic regression using this objective function. Let B be the batch size, n_B the number of batches and $E_n = -\sum_{k=1}^{K} t_{nk} \ln y_k(\boldsymbol{\phi}_n)$ the objective evaluated on data point n. Make sure to include indices for time, to define the learning rate and to check what dimension the weights you are updating have, transposing the gradient if necessary. Name one advantage of mini-batch gradient descent over stochastic gradient descent with single data points. How does it compare to full batch gradient descent?

4. (1 point) Consider now the Multilayer Perceptron in Fig. 1, with input features of dimension 2, 2 hidden units and 3 output classes. For which weight values and type of activation functions can we fall back to the multiclass logistic regression case?

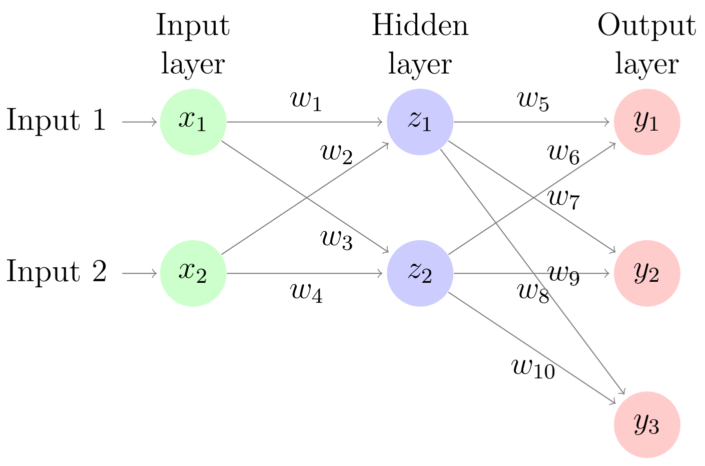

Figure 1: Neural Network

5. (3 points) Consider again the network in Fig. 1. We now consider the case where the activation function on the hidden layer is a ReLU, the activation for the output is a Softmax and we consider again the cross-entropy loss E. The weights of the network are initialized as follows:

$$W_1 = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} = \begin{bmatrix} 0.4 & 0.87 \\ 0.58 & 0.34 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} w_5 & w_6 \\ w_7 & w_8 \\ w_9 & w_{10} \end{bmatrix} = \begin{bmatrix} 0.12 & 0.87 \\ 0.82 & 0.31 \\ 0.77 & 0.9 \end{bmatrix}$$

Perform the following steps. Round round off to three decimal digits after each operation (e.g. after you apply the first matrix multiplication, after you apply the ReLU...).

- Compute the forward pass for the data point $x = (0.3, 0.7)^T$ with label $y = (0, 0, 1)^T$. Evaluate the loss.
- Compute the derivative $\frac{\partial E}{\partial w_5}$. Write down the formula first and then compute the numerical result.
- The other derivatives are given by:

$$\begin{bmatrix} \frac{\partial E}{\partial w_1} & \frac{\partial E}{\partial w_2} \\ \frac{\partial E}{\partial w_3} & \frac{\partial E}{\partial w_4} \end{bmatrix} = \begin{bmatrix} -0.044 & -0.103 \\ -0.062 & -0.144 \end{bmatrix}$$

$$\begin{bmatrix} & \frac{\partial E}{\partial w_6} \\ \frac{\partial E}{\partial w_7} & \frac{\partial E}{\partial w_8} \\ \frac{\partial E}{\partial w_9} & \frac{\partial E}{\partial w_{10}} \end{bmatrix} = \begin{bmatrix} & 0.104 \\ 0.244 & 0.138 \\ -0.429 & -0.242 \end{bmatrix}$$

  Perform a weight update with learning rate 0.05.
- Compute the loss again, what happens?