Machine Learning *CS 4364/6364, Fall 2017*

HW2

Instructions

Collaboration: Collaboration is allowed, to discuss the problems with other students. However you must write up your own solutions for the math questions, and implement your own solutions for the programming problems. Please list all collaborators and sources consulted, at the top of your homework.

Submission: Please submit homework pdf's via blackboard. Electronic submissions preferred. (Math can be formatted in Latex (some easy editors for Latex are Lyx, TexShop, Texmaker), Word, or other editors). Written homework can be scanned or submitted in class. All code should be submitted electronically. You may code in your language of choice. All homeworks are due at 11:59 pm on Monday 11/6. 20% penalty for late homeworks. No homeworks accepted after one week after the due date.

Questions

1. **Logistic Function**
   [From Bishop, Exercise 4.7] Recall the logistic function that was used for logistic regression (this function will also be important for neural networks):
   $$g(z) = \frac{1}{1+e^{-z}}$$
   Prove the following properties of the logistic function:
   (a) $g(-z) = 1 - g(z)$
   (b) The inverse $g^{-1}(y) = \ln \frac{y}{1-y}$
2. **Learning mixtures of Gaussians with EM algorithm**
   In this problem, you'll implement Expectation Maximization(EM) algorithm you learned in the lecture. Follow the guidelines below to build your program
   (a) Your function should be in the form $[params, loglikes] = EM(X, k)$, where `params` contains the estimated parameters: **mixing proportions, the means, and covariances of each of the component Gaussian**. `loglikes` stores the expected log likelihoods of the mixture models you produce in the course of the EM iterations (so it'll be a vector that keeps track of the expected loglikelihood after each iteration).

(b) The algorithm is provided here below. Let ω_j,μ_j denote the mixing proportions, means of each Gaussian component $y = j$. We assume that the Gaussians are spherical, i.e. it can be represented as $\sigma_j I$, where σ_j is a scalar and I is the identity matrix. Let x_i denote a data point, let k denote the number of Gaussian components, let N denote the number of data points, and let n_j denote the effective number of data points currently assigned to component j.

Algorithm 1 EM algorithm

```
initialize $\omega_j$,$\mu_j$,$\sigma_j^2$    // various ways to do this; see (e)(i).
while number of iterations $<$ max do
                                                                        E Step
    For each data point $x_i$, for each component $j$: update $P(y = j|x_i)$, using Bayes rule,
    where: $P(x_i|y = j) = \mathcal{N}(x_i; \mu_j, \sigma_j^2 I)$, $P(y = j) = \omega_j$, and $P(x_i) = \Sigma_{j=1}^{k} P(y = j)P(x_i|y = j)$
    Let $n_j = \Sigma_{i=1}^{N} P(y = j|x_i)$
    for $y = j$ do
        $\mu_j \leftarrow \Sigma_{i=1}^{N} \frac{P(y=j|x_i)x_i}{n_j}$                                  M Step
        $\sigma_j^2 \leftarrow \Sigma_{i=1}^{N} \frac{P(y=j|x_i)||x_i-\mu_j||^2}{dn_j}$
        $\omega_j \leftarrow \frac{n_j}{N}$
    end for
    loglikes[iteration] = log-likelihood of whole training data set, with respect to the
    current model
end while
return params, loglikes
```

(c) To compute $P(\boldsymbol{x}|y = j)$, you should use the spherical Gaussian model

$$P(\boldsymbol{x}|y = j) = N(\boldsymbol{x}; \boldsymbol{\mu_j}, \sigma_j^2 I) = \frac{1}{(2\pi\sigma_y^2)^{\frac{d}{2}}} e^{-\frac{1}{2\sigma_y^2}||\boldsymbol{x}-\boldsymbol{\mu_j}||^2}$$

where d is the dimension of each data point.

(d) Let $D = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$ denote the data set, θ denote all the current model parameters.The expected log-likelihood of the mixture model is:

$$L(D; \theta) = \sum_{i=1}^{N} \sum_{j=1}^{k} P(y = j|x_i; \theta) \log P(x_i, j; \theta) =$$

$$\sum_{i=1}^{N} \sum_{j=1}^{k} P(y = j|x_i; \theta) \left[\log P(x_i|j; \theta) + \log P(j; \theta)\right]$$

The first term inside the summations was computed at the beginning of the iteration. To compute the first term inside the square brackets, use the Gaussian formula above, with the current values of μ_j, σ_j^2. The second term inside the square brackets is simply $\log(\omega_j)$.

(e) Initialization of the EM algorithm

EM is sensitive to the initialization of the parameters. Now you are going to explore the performance of your EM algorithm with different initializations using the `IRIS` data.

(i) Initialize your parameters μ_j,σ_j^2, and ω_j: randomly select different k data points from your training data to be your k means; let all of your covariance matrices to be identity matrices (one's on the diagonal, zeros elsewhere); initialize the mixing proportion to be $\frac{1}{k}$ for each of the Gaussian.

(ii) Use the provided `IRIS` data in irisData.csv, which has 150 examples (it's already randomized for you). Run your EM algorithm with k = 2 and 3 on the trainData for number of iterations = 10,100, and 1000, respectively. Plot the expected loglikelihood graph for each of these settings. What's the trend of the expected loglikelihood in each of these settings? (Is it increasing, decreasing, or varying?) What are the maximum expected loglikelihood you get in each of those settings? Does the maximum expected loglikelihood improve significantly when you increase the number of iterations?

(iii) What would happen if you initialize your parameters in the following way:

- set all the means to vector $\mathbf{1}$
- In the case where $k = 3$, set two of the means to $\mathbf{1}$ and the other to be a random datapoint. Compare the three Gaussians with the two you have where $k = 2$.

(Briefly describe what happened in each case)

(iv) **[Grad Students Only]** The `IRIS` data you used is already randomized for you, i.e., the data points come in a random order. What would happen if the data come in their group order? Sort the data according to their labels (found in irisLabels.csv) and feed the sorted feature to your EM algorithm. Describe what happened comparing to your previous result (for example, do the means and covariances change? Do the loglikehoods change?). Explain this result.

3. **Online learning**

In this problem you will implement and plot a learning curve for a simple online learning algorithm that uses multiplicative updates. The algorithm is Static-Expert, which is also similar to the Weighted Majority Algorithm. The pseudocode is given. In this regression framework, we will use $\texttt{loss}(\hat{y}, y) = (\hat{y} - y)^2$, which can be applied to any prediction $\hat{y}$. The prediction of the algorithm is given in the first line inside the first for-loop. As experts, we will simply use the components of x, so the loss of expert i will therefore be $\texttt{loss}(\hat{y}_i, y) = (x_i - y)^2$.

Use the Cloud data provided in cloud.csv (from the UCI Machine Learning Repository). Filter it to use the 7th feature (the 7th column) as the label, not as a feature (be sure to **remove** this column from the input features!). You should have 9 remaining features. For this online learning problem, do **not** permute the data set first, since the order matters. You will use this batch of data to simulate a stream of data by going through the whole data set once.

1. Let the learning rate $b = 1$. Make a plot of the learning curve. The x-axis is iteration t. The y axis is the current value of the average loss of the algorithm. The average loss at

Algorithm 2 Static-Expert Algorithm [Herbster and Warmuth '98]

```
Input: $d$ the number of features (columns of the data), and learning rate $b$
Initialize: $t = 1$, $p_t(i) = 1/d$ for all $i \in \{1, \dots, d\}$
For each example $(x, y)$ in the stream
    Output prediction: $\hat{y} = \sum_{i=1}^{d} p_t(i) x_i$
    For $i \in \{1, \dots, d\}$
        loss $= (x_i - y)^2$
        $p_{t+1}(i) = p_t(i) \exp\{-b \cdot \texttt{loss}\}$
    Normalize the distribution $p_{t+1}$.        // so be sure to keep track of sum inside for-loop.
    $t = t + 1$
```

time t is given as,

$$L_t = \frac{1}{t} \sum_{i=1}^{t} Loss(i)$$

where $Loss(i)$ is the loss at iteration i.

Turn in your code, and this plot.

2. Experiment with 3 other values of b, and plot the results super-imposed on the plot above. How does the performance change with b?