Homework 1
Estimation, Naive Bayes, Convexity, Deep Learning

CMU 10-715: Machine Learning (Fall 2015)
http://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/
OUT: Sep 21, 2015
DUE: Oct 5, 2015, 10:20 AM

Guidelines

- The homework is due at 10:20 am on Monday October 5, 2015. Each student will given two late days that can be spent on any homeworks, but at most one late day per homework. Once you have used up your late days for the term, late homework submissions will receive no credit.
- Submit both a paper copy and an electronic copy through the submission website: https://autolab.cs.cmu.edu/courses/10715-f15. You can sign in using your Andrew credentials. You should make sure to edit your account information and choose a nickname/handle. This handle will be used to display your results for any competition style questions on the class leaderboard.
- Some questions will be *autograded*. Please make sure to carefully follow the submission instructions for these questions.
- We recommend that you typeset your solutions using software such as LaTeX. If you choose handwriting, ensure your handwriting is clear and legible. The TAs will not invest undue effort to decrypt bad handwriting.
- Programming guidelines:
  - **Octave:** You must write submitted code in Octave. Octave is a free scientific programming language, with syntax identical to that of MATLAB. Installation instructions can be found on the Octave website. (You can develop your code in MATLAB if you prefer, but you *must* test it in Octave before submitting, or it may fail in the autograder.)
  - **Autograding:** This problem is autograded using the CMU Autolab system. The code which you write will be executed remotely against a suite of tests, and the results used to automatically assign you a grade. To make sure your code executes correctly on our servers, you should avoid using libraries which are not present in the *basic* Octave install.
  - **Submission Instructions:** For each programming question you will be given a function signature. You will be asked to write a single Octave function which satisfies the signature. In the code handout linked above, we have provided you with a single folder containing stubs for each of the functions you need to complete. *Do not modify the structure of this directory or rename these files.* Complete each of these functions, then compress this directory *as a tar file* and submit to Autolab online. You may submit code as many times as you like.

    When you download the files, you should confirm that the autograder is functioning correctly by compressing and submitting the directory of stubs provided. This should result in a grade of zero for all questions.
  - **SUBMISSION CHECKLIST**
    - Submission executes on our machines in less than 10 minutes.
    - Submission is smaller than 2000K.
    - Submission is a `.tar` file.
    - Submission returns matrices of the *exact* dimension specified.
  - **Data:** All questions will use the following datastructures:
    - $XTrain \in \mathbb{R}^{n \times k}$ is a matrix of training data, where each row is a training point, and each column is a feature.

* $XTest \in \mathbb{R}^{m \times k}$ is a matrix of test data, where each row is a test point, and each column is a feature.
* $yTrain \in \{1, ..., c\}^{n \times 1}$ is a vector of training labels
* $yTest \in \{1, ..., c\}^{m \times 1}$ is a (hidden) vector of test labels.

1 Estimating Parameters [Eric; 30 pts]

1.1 Closed Form Estimation

1. (3pts) An exponential distribution with parameter λ follows a distribution $p(x) = \lambda e^{-\lambda x}$. Given some i.i.d. data $\{x_i\}_{i=1}^n \sim \text{Exp}(\lambda)$, derive the maximum likelihood estimate (MLE) $\hat{\lambda}_{MLE}$. Is this estimator biased?

2. (5pts) A gamma distribution with parameters α, β has density function $p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$ where $\Gamma(t)$ is the gamma function (see https://en.wikipedia.org/wiki/Gamma_distribution).

   If the posterior distribution is in the same family as the prior distribution, then we say that the prior distribution is the conjugate prior for the likelihood function. Show that the Gamma distribution (that is $\lambda \sim \text{Gamma}(\alpha, \beta)$) is a conjugate prior of the $\text{Exp}(\lambda)$ distribution. In other words, show that if $x \sim Exp(\lambda)$ and $\lambda \sim \text{Gamma}(\alpha, \beta)$, then $P(\lambda|x) \sim \text{Gamma}(\alpha^*, \beta^*)$ for some values α^*, β^*.

   Derive the maximum a posteriori estimator (MAP) $\hat{\lambda}_{MAP}$ as a function of α, β. What happens as n gets large?

3. (4pt) Let's perform an experiment in the above setting. Generate $n = 20$ random variables drawn from $\text{Exp}(\lambda = 0.2)$. Fix $\beta = 100$ and vary α over the range $(1, 30)$ using a stepsize of 1. Compute the corresponding MLE and MAP estimates for λ. For each α, repeat this process 50 times and compute the mean squared error of both estimates compared against the true value. Plot the mean squared error as a function of α. (Note: Octave parameterizes the Exponential distribution with $\Theta = 1/\lambda$. https://www.gnu.org/software/octave/doc/interpreter/Random-Number-Generation.html may be helpful)

4. (2pt) Now, fix $(\alpha, \beta) = (30, 100)$ and vary n up to 1000. Plot the MSE for each n of the corresponding estimates.

5. (4pts) Under what conditions is the MLE estimator better? Under what conditions is the MAP estimator better? Explain the behavior in the two above plots.

1.2 Non-closed Form Estimation

For this question, please make use of the digamma and trigamma functions. You can find them in any scientific computing package (e.g. Octave, Matlab, Python...). This question requires some coding but is not being submitted to Autolab.

1. (3pts) Sometimes we don't have closed forms for the estimators. Given some data $\{x_i\}_{i=1}^n \sim \text{Gamma}(\alpha, \beta)$, use gradient descent to maximize the likelihood and derive the steps to calculate the MLE estimators $\hat{\alpha}_{MLE}, \hat{\beta}_{MLE}$.

2. (3pts) We can also use Newton's method to calculate the same estimate. Provide the Newton's method updates to calculate the above MLE estimators for the Gamma distribution.

3. (6pts) Inside the handout, estimators.mat contains a vector drawn from a Gamma distribution. Run your implementation of gradient descent and Newton's method to obtain the MLE estimators for this distribution. Create a plot showing the convergence of the two above methods. How do they compare? Which took more iterations? Lastly, provide the actual estimated values obtained.

2 Naive Bayes [Eric; 30 pts]

In this question you will implement a variation on the Naive Bayes classification algorithm. You will be asked to fill in function stubs in the code handout. To submit, please tar the folder named code (e.g. with tar -cvf code.tar code) and upload it to the Autolab submission site. Your datasets will have class labels $\{1 \dots k\}$ and have real valued features. Assume the features are normally distributed.

Your code will be run on various datasets following the above description. Provided are two datasets: the iris dataset and the forests dataset (as mentioned in recitation). You can find the function stubs in the **code/avg** folder and the datasets in the **data** folder. As a reference, our solution code runs on Autolab in less than 2 minutes.

2.1 Simple Naive Bayes

1. (6pts) First, implement [model] = NaiveBayes(XTrain, yTrain) and [predictions] = NaiveBayesClassify(model, XTest) with an implementation of the naive Bayes classifier for warmup. *model* should contain any precomputed values necessary for classification, which is passed directly to the classifier function. *Predictions* is a $m \times 1$ vector of predicted labels for the datapoints in *XTest*.

2.2 Bayes Model Averaging

Recall the naive Bayes assumption. A naive Bayes classifier may not perform as well on datasets with redundant or excessively large numbers of features. To deal with this, one option is to reduce the number of features and choose a smaller subset based on some criterion (e.g. mutual information, [1]). A different way of dealing with this is to not remove any features, but to take an average over many possible feature subsets. This procedure of averaging over many models instead of explicitly selecting one is more commonly known as Bayes model averaging [2].

In this scenario, let F be the set of all possible feature subsets, where $f = (f_1, \dots, f_k) \in F$ represents a possible subset with $f_i \in \{0, 1\}$. $f_i = 1$ denotes that feature i is used. Assume the following prior on $P(f_i)$:

$$P(f_i) \propto \begin{cases} \frac{1}{\beta} & \text{if } f_i = 1 \\ 1 & \text{if } f_i = 0 \end{cases}$$

As usual, let $D = \{x^{(i)}, y^{(i)}\}$ be the set of training data. Lastly, we define what it means for a model to use feature k as follows:

$$P(x_j^{(i)} | f_j, y_i) = \begin{cases} P(x_j^{(i)} | y_i) & \text{if } f_j = 1 \\ P(x_j^{(i)}) & \text{if } f_j = 0 \end{cases}$$

In other words, if the feature is used, the the probability depends on the output y_i. If a feature is not used, it is reduced to a constant that does not depend on y_i.

For a new observation x^*, the Bayes classifier relies on choosing the label that maximizes the conditional probability of the label given the data, $P(x^*|y, D)P(y|D)$. Now, we want to choose a label that maximizes the same quantity marginalized over all possible models:

$$\operatorname*{argmax}_y \sum_f P(x^*, f|y, D)P(y|D)$$

At first glance, this sum looks terrible: there are exponentially many feature subsets! However, in the following questions, you will derive an equivalent classifier that runs in time linear to the number of features.

1. (3pts) Using Bayes rule, rewrite the classifier in terms of $P(x|f, y, D)$, $P(y|D)$, and $P(f|D)$.

2. (3pts) Write an expression for $P(f|D)$ using Bayes rule, and substitute into the classifier. Assume that each feature is selected independently from one another, and relabel the new observation (x, y) as $(x^{(N+1)}, y^{(N+1)})$ to simplify.

3. (6pts) Finally, derive the following form of the classification rule that runs in time linear to the number of features. You will need to exploit the properties of sums and products.

$$\underset{y}{\text{argmax}} \left[\prod_{i=1}^{N+1} P(y^{(i)}) \right] \prod_{k=1}^{K} \left[\prod_{i=1}^{N+1} P(x_k^{(i)}) + \frac{1}{\beta} \prod_{i=1}^{N+1} P(x_i^{(k)}|y) \right]$$

4. (12pts) Implement [model] = AvgNaiveBayes(XTrain, yTrain) and [predictions] = AvgNaiveBayesClassify(model,XTest). *model* should contain any precomputed values necessary for classification, which is passed directly to the classifier function. *Predictions* is a $m \times 1$ vector of predicted labels for the datapoints in *XTest*.

3 What breaks the convexity of deep learning? [Fish; 40 pts]

In gradient descent algorithm, convexity of the target function plays an important role in determining whether or not the algorithm will converge, the convergence rate and so on. We didn't cover too much about convex function in the class. So here we will go through some useful techniques for examining convexity of a function.

3.1 Basic definition of convex function(28pts)

- Definition of convex set:
  A set $C \subseteq \mathbb{R}^n$ is called convex set, if $\forall x, y \in C$, $tx + (1-t)y \in C$ for $0 \leq t \leq 1$.
  That means for every pairs of points in the convex set C, every point falls on the straight line segment that joins the pair of points will also be in the set. A set that has such properties is super powerful, because you can get to any point in the set from a given point through a straight line without hitting the boundary. Hitting the boundary often means you might get stuck at a local minimum and never have the chance to find the global optimal solution.

- Definition of convex function:
  A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function, if $\text{dom}(f)$ is a convex set, and $f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$ for $0 \leq t \leq 1$.
  The inequalities say that the line segment between two points on the function will always lies above the function. Function that has such property is very powerful when you are trying to find the minimum value on the function. Because for any randomly chosen two points, x, y, you can always find a point z that lies between x and y such that $f(z) = \min(f(x), f(y))$. So you will surely be able find the optimal value.

Sometimes it is not easy to examine the convexity though the basic definition of convex. Here are some properties of convex function. You need to prove them in both directions. Half of the points in each questions are given if you prove the argument based on the definition, half of the points are given for proof from argument to definition. (For simplicity, let's assume $dom(f) = \mathbb{R}$)

1. (4pts) If f is continuous, then $f(\frac{x+y}{2}) \leq \frac{f(x)+f(y)}{2}$, $\forall x, y \in \mathbb{R}$
2. (8pts) If f is continuously differentiable, $f(y) \geq f(x) + f'(x)(y-x)$, $\forall x, y \in \mathbb{R}$.
3. (8pts) If f is twice differentiable, $f''(x) \geq 0$, $\forall x \in \mathbb{R}$.

Also, use the definition to prove that

1. (2pts) If f, g are convex functions, show that $h(x) = max\{f(x), g(x)\}$ is also a convex function.
2. (2pts) If f, g are convex functions, show that $h(x) = f(x) + g(x)$ is also a convex function.
3. (4pts) If function f and g are convex functions, is $f(g(x))$ necessarily a convex function? If not, what kind of conditions do we need to add to make it convex? (Please provide reasons. I know you can always find answer on Wiki.)

3.2 Functions used in deep learning(12pts)

Here are several commonly used objective functions in deep learning. Determine whether they are convex or not and provide formal proof. (Of course you can use the properties we have proved so far) Provide only yes/no answer will get only partial scores. We define some commonly used notations as follows: $w \in \mathbb{R}^d$ is the weight; $x_i \in \mathbb{R}^d$ is the feature vector for the i-th sample. (note that the first element in x_i is set to 1, so we can get rid of the bias term in the lecture note.); $r_i = < w, x_i > \in \mathbb{R}$; $y \in \mathbb{R}$ is the label for the i-th sample.

1. (2pts) Hinge Loss: $H([r_1, r_2, ...r_N]) = \sum_{i=1}^{N} \max(0, 1 - yr_i)$
2. (2pts) Relu: $R(r_i) = \max(0, r_i)$
3. (2pts) Logistic Function: $L(r_i) = -\log(1 + e^{-r_i})$
4. (3pts) Fully connected layer with Soft-max loss: Define $\mathbf{W} \in \mathbb{R}^{d \times k}$ and w_i is the i-th column of $\mathbf{W}$. $\mathbf{W}^T x_i = [< w_1, x_i >, \cdots, < w_k, x_i >] := [r_i(1), r_i(2), \cdots, r_i(k)]$. $S_s(\mathbf{W}) = \log\left(\sum_{j=1}^{k} e^{r_i(j)}\right) - r_i(s)$.
5. (3pts) From the above questions, do you know what breaks the convexity of deep learning? Briefly describe why most of the deep learning functions are non-convex.

References

[1] Peng, H.; Fulmi Long; Ding, C., "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," in Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.27, no.8, pp.1226-1238, Aug. 2005 doi: 10.1109/TPAMI.2005.159

[2] Hoeting, Jennifer A., et al. "Bayesian model averaging." In Proceedings of the AAAI Workshop on Integrating Multiple Learned Models. 1998.