University of Washington
Department of Computer Science and Engineering / Department of Statistics

CSE 599 / Stat 592 Machine Learning (Statistics) for Big Data

Homework 3– Midterm

Winter 2013

Must be done individually, without any communication with other students. If you have questions, please contact the instructors for assistance.

Issued: Friday, February 15, 2013 **Due:** Thursday, February 28, 2013

Suggested Reading: Assigned Readings in Case Study III (see website).
Instructions: The homework consists of two parts: (i) Problems 3.1 and 3.2 cover theoretical and analytical questions and (ii) Problem 3.3 covers data analysis questions. Please submit each portion as *separate* sets of pages with your name and userid (UW student number) on each set. For Part II which involves coding, please print out your code and graphs and attach them to the written part of your homework.

Problem 3.1

Deriving sampling updates for Supervised latent Dirichlet allocation (sLDA) [40 points]

The LDA we covered during lecture is an unsupervised model. It is typically built on a discrete bag-of-words representation of input contents, which can be text documents, images or even network entities. However, in many practical applications, we can easily obtain useful side information such as ratings or labels associated with documents or images. Such side information often provides useful high-level or direct summarization of the content, but it is not directly used in the original LDA.

A supervised topic model can use side information to discover more predictive low dimensional topical representations of the data by capturing real-valued document ratings as a regression response. Presented below is the graphical illustration of unsupervised LDA and supervised LDA.

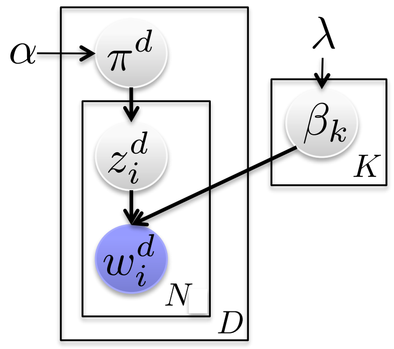
Recall from lecture that an LDA model with K topics, vocabulary size V, and N words per document (for notational simplicity, we assume that all documents have the same length) has the following generative process:

(i) Draw a topic-specific word distribution according to a V-dimensional Dirichlet prior: $\beta_k|\lambda \sim \text{Dirichlet}(\lambda_1, ..., \lambda_V), k = 1, ..., K$;

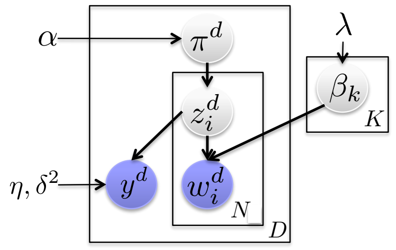
(ii) Draw a document-specific topic mixing proportion vector π^d according to a K-dimensional Dirichlet prior: $\pi^d|\alpha \sim \text{Dirichlet}(\alpha_1, ..., \alpha_K), d = 1, ..., D$;

(iii) For the i^{th} word in document d, where $1 \leq i \leq N$,

- draw a topic assignment z_i^d according to π^d: $z_i^d|\pi^d \sim \text{Multinomial}(\pi^d), i = 1, ..., N$;



(a)



(b)

Graphical illustration of (a) unsupervised LDA (Blei et al., 2003) and (b) supervised LDA (Blei and McAuliffe, 2007).

- draw the word instance w_i^d according to $z_i^d, \beta_{z_i^d}$: $w_i^d | z_i^d, \beta_{z_i^d} \sim \text{Multinomial}(\beta_{z_i^d})$

where z_i^d is a K-dimensional indicator vector (i.e., only one element is 1; all others are 0).

In order to consider side information for discovering more predictive representations, supervised topic models (sLDA) introduce a response variable y^d to LDA for each document d, as shown in Figure 1(b). The generative process of sLDA is similar to LDA, but with an additional step—draw a response variable: $y^d | z^d, \eta, \delta^2 \sim N(\eta^T \overline{z^d} + \eta_0, \delta^2)$ for each document d, where η is the regression weight vector, δ^2 is a noise variance parameter, and $\overline{z^d} = (1/N) \sum_{i=1}^{N} z_i^d$ are the empirical topic proportions for document d. Here, we consider Gaussian distributions for the additional labels $\mathbf{y}$, but the derivation can be easily extended to other models. We treat $\alpha, \lambda, \eta, \eta_0, \delta^2$ as unknown constants and restrict ourselves to simply inferring the latent variables z_i^d, π^d and β_k.

In the following exercise, we will derive an uncollapsed Gibbs sampler for the sLDA model presented above. Recall that in uncollapsed Gibbs sampling, we iterate over the variables in the model and sample each conditioned on the observations and our current set of samples for all other variables. We refer to this distribution as the *complete conditional*. For example, at iteration t we sample $\pi^{d,(t)}$ from $p(\pi^d \mid \{y^d\}, \{w_i^d\}, \{z_i^{d,(t)}\}, \{\beta_k\}, \alpha, \eta)$ where $\{z_i^{d,(t)}\}$ represents the currently sampled topic indicators at iteration t. Due to conditional independencies between random variables in the model, this full conditional often only depends on a subset of the other variables.

Please write all answers in terms of in terms of the distributions listed above describing the generative model. In particular, use: multinomial, Dirichlet, and normal distributions.

(a) [4 points] Write the joint distribution of the model, as factored according to the graphical model.

(b) [5 points] Derive the complete conditional for π^d. List the variables in the model that are *not* required in the conditioning set (i.e., those for which π^d is conditionally independent given the included variables).

(c) [5 points] Derive the complete conditional for β_k. List the variables in the model that are *not* required in the conditioning set.

(d) Assume $\eta = 0$.

  i. [5 points] Derive the complete conditional for z_i^d. List the variables in the model that are *not* required in the conditioning set.

  ii. [2 points] How does this model relate to LDA? Provide a brief explanation/justification.

(e) [10 points] Assume $\eta \neq 0$. Derive the complete conditional for z_i^d. List the variables in the model that are *not* required in the conditioning set.

(f) [5 points] Fill in the pseudocode in Algorithm 1 for the derived Gibbs sampler.

Algorithm 1: Uncollapsed Gibbs Sampler for sLDA

```
for $t = 1, \dots, Niter$ do
    for $k = 1, \dots, K$ do
        FILL IN SAMPLING STEPS
    end
    for $d = 1, \dots, D$ do
        FILL IN SAMPLING STEPS
        for $i = 1, \dots, N$ do
            FILL IN SAMPLING STEPS
        end
    end
end
```

(g) [4 points] Is it possible to derive a collapsed Gibbs sampler for sLDA where the parameters π^d and β_k are analytically marginalized? Briefly explain.

Problem 3.2

Deriving Variational Inference for Gaussian Mixture Models [EXTRA CREDIT 16 points]

In Gibbs sampling, there is no clear notion of convergence, or rules of thumb are used to determine the burn-in samples. Variational Inference is an alternative to Gibbs sampling, where the posterior distribution $P(\theta|x)$ is approximated by a distribution $q(\theta)$ that comes from a restricted family of distributions Q. Here, the main idea is to solve the following optimization problem:

$$\min_{q \in Q} \mathrm{KL}(q||p) = \int q(\theta) \ln \frac{q(\theta)}{p(\theta|X)} d\theta$$

We will work with a simple mixture of Gaussians model with univariate observations with known variance σ^2 for each cluster to illustrate the variational inference machinery. For a Bayesian treatment of this model, we place priors on the means and mixing weights:

$$\begin{aligned} \pi|\alpha &\sim \text{Dirichlet}(\pi_0) \\ P(z^i = c|\pi) &= \pi_c \\ \mu_c &\sim N(0, \rho^2) \\ P(x^i|z^i = c, \mu_c) &= N(x^i; \mu_c, \sigma^2) \end{aligned}$$

where $\pi_0 = \alpha e$ and e is the all-ones vector (i.e., the prior is a symmetric Dirichlet prior), π are the mixing proportions, z^i is the cluster assignment for data point i, x^i is the observable data, and μ_c is the mean for cluster c.

Joint probability distribution is $P(x, \pi, z, \mu) = P(x|\mu, z)P(\mu)P(z|\pi)P(\pi)$. In our variational approach, we will approximate this joint distribution by:

$$P(\pi, z, \mu|x) \approx q(\pi, z, \mu) = q(z)q(\pi)q(\mu)$$

In particular, the approximating distributions will be given by:

- $q(z) = \prod_i q_i(z^i)$, where each distribution is a multinomial: $q_i(z^i = c) = \phi^i_c$.
- $q(\pi)$ is Dirichlet with parameters $\beta_1, \ldots, \beta_K$.
- $q(\mu) = \prod_c q_c(\mu_c)$, where each distribution is a Gaussian $q_c(\mu_c) = N(\mu_c; \eta_c, \lambda_c^2)$.

In this question, you will derive iterative updates for this approximation. Specify your answers in terms of the observed data x, the known parameters σ^2, ρ^2, α, and the variational parameters $\phi, \beta, \eta, \lambda$.

(a) [4 points extra credit] Write down the "ELBO" for this model.

(b) [4 points extra credit] Provide the variational update for $q(z)$, when $q(\pi)$ and $q(\mu)$ are given.

(c) [4 points extra credit] Provide the variational update for $q(\pi)$, when $q(z)$ and $q(\mu)$ are given.

(d) [4 points extra credit] Provide the variational update for $q(\mu)$, when $q(z)$ and $q(\pi)$ are given.

Problem 3.3

READ THE MIND [60 points]

In this problem, you will implement Shotgun (parallel stochastic coordinate descent) for l_1 regularized regression. The goal is to learn a set of sparse linear models to predict word semantic features from fMRI signals. Using the predicted features, you will construct a binary classifier (using 1-NN) that takes 2 candidate words and predict which word the subject was thinking.

Matrix notation: We will use subscript(superscript) to denote the column(row) of a matrix, e.g. For any matrix X, X_j is the jth column, X^i is the ith row, and $X_{i,j}$ is the entry (i, j).
If you use the starter code we provide, please only print out the functions that you need to fill in. In this problem there are only two functions: "shoot()" and "scd()" in "Shooting.class".

Let $X \in R^{n\times p}$ and $Y \in R^n$. The LASSO solves the following optimization problem:

$$\min_{w_0,w} \frac{1}{2n}\sum_{i=1}^{n}(y_i - w_0 - (X^i)^T w)^2 + \lambda\sum_{j=1}^{p}|w_j| \tag{1}$$

If Y is centered (the mean is 0), we can usually ignore w_0 in the above equation. We will keep this assumption for the rest of this problem. In matrix form, we can rewrite Eq 1 as

$$\min_{w} L(w,\lambda) = \frac{1}{2n}\|Y - Xw\|^2 + \lambda\|w\|_1 \tag{2}$$

3.3.1 Deriving the Shooting Algorithm in Practice [15 points]

The general Stochastic Coordinate Descent algorithm is shown in Algorithm 2.

In order to find the minimum at coordinate j, the Shooting algorithm applies the following procedure: First, let's split each w_i into it's positive part w_i^+ and negative part w_i^-, such that $w_i = w_i^+ - w_i^-$. Note that both w_i^+ and w_i^- are non-negative. When w_i is positive $w_i^- = 0$ and vice versa. Then, we can rewrite Eq 2 as follows:

$$\min_{\tilde{w}} \tilde{L}(\tilde{w},\lambda) = \frac{1}{2n}\|Y - \tilde{X}\tilde{w}\|^2 + \lambda\sum_{j=1}^{2p}\tilde{w}_j \quad \tilde{w}_j >= 0, j = 1,\dots,2p \tag{3}$$

where $\tilde{X} = [X, -X] \in R^{n\times 2p}$, and $\tilde{w} = [w^+; w^-] \in R^{2p}$. Taking $w_i = \tilde{w}_i - \tilde{w}_{i+p}$ will recover the original w.

After getting rid of the l_1 norm, it becomes straight-forward to take the derivative $\nabla_j \tilde{L}(\tilde{w})$ at each coordinate $j = [1, 2, .., 2p]$. However, in the update step, we need to take extra

```
Algorithm 2: General Stochastic Coordinate Descent
input  : Objective function $L(w)$
output: Optimal $w$
$w^{(0)} = 0$;
$t = 0$;
while not converged do
    Choose a coordinate $j$ uniformly at random ;
    $w_j^{new} \leftarrow \min_{w_j} L(w_0^{(t)}, \ldots, w_{j-1}^{(t)}, w_j, w_{j+1}^{(t)}, \ldots, w_p^{(t)})$;
    $w^{(t+1)} \leftarrow w^{(t)}$;
    $w_j^{(t+1)} \leftarrow w_j^{new}$;
    $t \leftarrow t + 1$;
end
```

care to enforce the non-negativity constraint. In particular, the minimization step becomes simply:

$$\tilde{w}_j^{(t+1)} \leftarrow \tilde{w}_j^{(t)} + \max\{-\tilde{w}_j^{(t)}, -\nabla_j L(\tilde{w^{(t)}})\} \tag{4}$$

Algorithm 3 shows the pesudo-code for the Shooting algorithm.

```
Algorithm 3: Shooting: Sequential Stochastic Coordinate Descent for LASSO
input  : $X \in R^{n \times p}, Y \in R^n$, and $\lambda$
output: $\hat{w} = \arg\min_w L(w, \lambda)$
Set $\tilde{X} = [X, -X]$;
Initialize $\tilde{w}_j = 0$ for $j = 1, \ldots, 2p$;
while not converged do
    Choose $j$ uniformly at random ;
    $\tilde{w}_j \leftarrow \tilde{w}_j + \max\{-\tilde{w}_j, -\nabla_j \tilde{L}(\tilde{w}, \lambda)\}$
end
Set $w_j = \tilde{w}_j - \tilde{w}_{j+p}$ for $j = 1, \ldots, p$.
```

(a) [3 points] Derive $\nabla_j \tilde{L}(\tilde{w}, \lambda)$ for Eq 3 in terms of $\tilde{X}, \tilde{w}$, Y, and λ.

(b) [3 points] Show how to compute $\nabla_j \tilde{L}(\tilde{w}, \lambda)$ without explicitly replicating X. In other words, rewrite your solution in terms of X, w^+, w^-, Y and λ. (In practice, this step is important to avoid using extra memory.)

(c) [3 points] What is the complexity, in $\mathcal{O}$ notation, of computing the derivative?

(d) [3 points] Alternatively, if we cache $\tilde{X}\tilde{w}$, we can significantly decrease the complexity. If we knew the result of this product, what is the cost of the update now?

(e) [3 points] Suppose you updated $\tilde{w}_j = \tilde{w}_j + \delta$, show how to update $\tilde{X}\tilde{w}$ in terms of δ efficiently.

3.3.2 Shooting at Simulated Data [15 points]

- Download “lasso_synthetic.zip” from the course website.
- After unzipping the folder, there should be four files: trainX.mtx, trainY.mtx, testX.mtx, testY.mtx.
  - Xtrain.mtx: The training data X representing a $30 * 200$ matrix in dense matrix market format.
  - Ytrain.mtx: The training data Y representing a $30 * 1$ matrix in dense matrix market format.
  - Xtest.mtx: The test data X representing a $30*200$ matrix in dense matrix market format.
  - Ytest.mtx: The test data Y representing a $30 * 1$ matrix in dense matrix market format.

  If you are not using the starter code, here is how to interpret these files: In the dense matrix market format, the first line after the comment is ”np”, encoding the number of rows and columns of the matrix. The matrix is represented as an array which is the concatenation of all the columns. For example, for a matrix of dimension $n \times p$, the file looks like:

$$\begin{array}{l}\%\%\text{MatrixMarket matrix array real general} \\ \%SomeComments \\ np \\ A_{11} \\ A_{21} \\ \dots\end{array}$$

(a) [15 points] Implement the sequential Shooting algorithm for the synthetic data. Use $\lambda = \{0.02, \dots, 0.2\}$ spaced by 0.02. Please submit three plots for this question:
  - Plot the squared loss on training data against λ.
  - Plot the squared loss on test data against λ.
  - Plot the l_0 norm of the weight vector $\hat{w}$ against λ.

  (Hint: Complete the two core functions: “shoot()” and “scd()” in “Shooting.class” in the starter code. You may find pre-implemented linear matrix and vector operations in “MatUtil.class”.
  After implementing those two core functions, you should be able to run “LassoSimulation.class”, which will load the data, run “scd()” and outputs the result.)

3.3.3 Point Shotgun at Brain Images [30 points]

The fMRI records the brain activity of one subject while he was being shown a set of different words. The original data for this question is from `http://www.cs.cmu.edu/afs/cs/project/theo-73/www/science2008/data.html`.

The vocabulary set contains 60 nouns covering different categories. For each word, 218 semantic features are extracted. Each feature corresponds to a question about the semantic meaning of this word. The value is a score ranging from 1 to 5 provided by a human labeler as his response to the question. For example, feature 1 corresponds to the question: "IS IT AN ANIMAL". The word "ant" has value 4 for feature 1, whereas the word "airplane" has value 1.

At trial i, $word^i$ is shown to the subject. Then the fMRI is recorded as a 21764 dimension vector X^i. Each dimension corresponds to the activity in a voxel (a cube in the 3-d coordinates of the brain). There are totally 360 trials, for which the results are put into $X \in R^{360\times 21764}$ and $Y \in R^{360\times 218}$. $X_{i,j}$ is the signal at jth voxel at trial i, and $Y_{i,j}$ is the jth feature of the word displayed at trial i.

We further standardize X to have mean 0 and unit norm for each column and center Y to have mean 0 for each column. Finally, the data is split into training set (300 trials) and test set (60 trials).

The goal is to learn 218 sparse linear models, each predicting a semantic feature of the output space: $\mathbf{w}^i = LASSO(X, Y_i) \quad i = 1 \ldots 218$. Given a new input X^{n+1} the entire model will output a 218 dimension vector Y^{n+1}. Given two candidate words, we create a binary classifier by choosing the word whose semantic feature vector is closer to the predicted one in l_2 distance.

- Download "fmri.zip" from the course website.
- After unzipping the folder, there should be seven files:
  - subject1_fmri_std.train.mtx: The standardized fMRI signal for subject1. This will be the X of the training data where $X_{i,j}$ is the signal at jth voxel at trial i.
  - subject1_wordid_std.train.mtx: Each line corresponds to a word id. Line i is the id of the word shown to subject1 at trial i.
  - subject1_fmri_std.test.mtx: Same format as "subject1_fmri_std.train.mtx, used for testing.
  - subject1_wordid_std.test.mtx: Contains two columns. The first column is the ids of the ground truth words for test data. The second column contains ids choosing at random.
  - word_feature_centered.mtx: A 60×218 matrix, where row i is the sementic feature vector for word with $id = i$. The matrix is written in the dense Matrix Market

format.

- meta/dictionary.txt: Mapping from id to the word. Line i is the word whose $id = i$.
- meta/semantic_feature_name.txt: Contains the meta information of the 218 semantic features.

(Hint: For the following two questions, if you already implemented "shoot()" in the previous question, you should be able to directly run the functions in "FMRIWordPrediction.class" and see the results.
You do not need to write extra code for parallelism. Those functions in "FMRIWordPrediction.class" will call the pre-implemented "Shooting.shotgun()" which runs "shoot()" in parallel.)

(a) [15 points] For semantic feature 1, run Shotgun with $\lambda = [0.02, \ldots, 0.1]$ spaced by 0.01. Please submit three plots for this question:

- Plot the l2 loss on training data against λ.
- Plot the l2 loss on test data against λ.
- Plot the l_0 norm of the weight vector $\hat{w}$ against λ

(b) [15 points] Iterate over all 218 semantic features, run Shotgun with

$$\lambda = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4].$$

Choose the λ that has the smallest error on the test data.
At the end of each iteration, apply the current model to predict the word features on the test data. Given the two candidate words in the test data, choosing the one whose semantic feature vector is closest (in Euclidean distance) to the prediction and record the error rate of the classification. For example, at the end of iteration t, the distance between the candidate and our prediction for word j is

$$\sum_{i=1}^{t} \|Y_{j,i} - X^j w_i\|^2$$

- Plot the classification error (number of mistakes / total number) as a function of number of iterations (same as number of trained features).