CS540 Spring 2010: homework 7

1 High dimensional classification

The goal of this exercise to reproduce table 18.1 of [HTF09, p656]. (Recall that this book is available online for free at `http://www-stat.stanford.edu/~tibs/ElemStatLearn/download.html`.) This is a classification problem with $N = 144$ samples, $D = 16,603$ features and $C = 14$ classes (types of cancer). The data is available from `http://www-stat.stanford.edu/~tibs/ElemStatLearn/data.html`. The methods are all sparse classifiers of various kinds, and are listed below. Code for many of these already exists in pmtk, but you may have to make some minor changes.

- Nearest shrunken centroids (this is naive Bayes where σ_{jc}^2 is tied across c and we use a sparsity-promoting prior for μ_{jc}.) Use `naiveBayesGaussFitShrunkenCentroids` and `naiveBayesGaussPredict`. Choose λ by CV.
- Standard naive Bayes with MLE and untied μ_{jc} and σ_{jc}^2 (`naiveBayesGaussFit`).
- Regularized linear discriminant analysis. This means using the following expression (p656 of Hastie) for the tied class conditional covariance matrix

$$\hat{\mathbf{\Sigma}}(\gamma) = \gamma\hat{\mathbf{\Sigma}} + (1-\gamma)\text{diag}(\hat{\mathbf{\Sigma}}) \tag{1}$$

  where $0 \leq \gamma \leq 1$ is chosen by CV and $\hat{\mathbf{\Sigma}}$ is the MLE. If $\gamma = 0$, this becomes diagonal LDA, which is equivalent to nearest shrunken centroids with $\lambda = 0$. You'll need to modify `discrimAnalysisFit`, but can use `discrimAnalysisPredict` as is.

- Regularized quadratic discriminant analysis. This means using the following untied class conditional covariance matrix (p112 of Hastie)

$$\hat{\mathbf{\Sigma}}_k(\alpha,\gamma) = \alpha\hat{\mathbf{\Sigma}}_k + (1-\alpha)\hat{\mathbf{\Sigma}}(\gamma) \tag{2}$$

  You'll need to modify `discrimAnalysisFit`, but can use `discrimAnalysisPredict` as is.

- SVM classifier. You'll need to implement a one-versus-all method on top of the existing binary SVM code (`svmlightFitCV` and `svmlightPredict`). You'll also need to install SVMlight first, from `http://svmlight.joachims.org/`.
- KNN classifier (`knnClassify`).
- ℓ_2 penalized multinomial logistic regression (`logregFit` with `regType='L2'`).
- ℓ_1 penalized multinomial logistic regresion (`logregFit` with `regType='L1'`).

For brevity, you can omit multinomial logistic regression with the elastic net (ℓ_1+ℓ_2) regularizer, and lasso regression (which is not appropriate for classification problems).

2 Imputation for gene microarrays

A gene microarray dataset is an $N \times D$ matrix, where X_{ij} is the expression level (relative to some baseline) of gene i in experiment j. The experiments often correspond to measuring genes over time (so each row represents a time series), or measuring genes under different environmental conditions. For example, the yeast cell cycle data [ESBB98] measures 6221 genes over 80 time steps, and the yeast environmental stress data [GSK+00] measures 6361 genes over 156 conditions.

Most microarray data has missing values due to measurement problems. For example in the yeast cell cycle data, about half of the rows are missing one or more values. It is common to impute the missing values before doing further downstream analysis (such as clustering). Many methods have been proposed for this. In this exercise, we will compare a few of them on the above two datasets.

We will follow the protocol of [OWG04]. The yeast cell cycle data is available from `http://rana.lbl.gov/EisenData.htm`. Use the 3222 rows which have no missing values. The yeast stress data is available at `http://genome-www.stanford.edu/yeast_stress/data.shtml`. Use the 5068 rows that have no missing values. In addition, use the 15 columns defined in footnote 2 of [OWG04]; this ensures the features are not too highly correlated, in order to make the data more different to the cell cycle data. Note: various methods for reading in data in Matlab are described at `http://www.mathworks.com/support/tech-notes/1400/1403.html`.

For each dataset, set a random fraction p of the entries to NaN (use $p \in \{0.001, 0.01, 0.1\}$) impute the missing entries, and then measure the accuracy using the normalized root mean squared error

$$NRMSE \stackrel{\text{def}}{=} \frac{\frac{1}{|\mathcal{M}|}\sum_{ij\in\mathcal{M}}(\hat{X}_{ij} - X_{ij})^2}{\frac{1}{|\mathcal{M}|}\sum_{ij\in\mathcal{M}}(X_{ij})^2} \tag{3}$$

where $\mathcal{M}$ is the set of indices of the missing entries and $\hat{X}_{ij}$ is the imputed entry. Repeat this 10 times and compute the mean and standard error of the NRMSE.

Many of the methods below have a free parameter K which controls the complexity of the model used for imputation. For each method, you should plot the mean NRMSE (with error bars) vs K. Think of a way to make the x-axis comparable across methods, so you can superimpose all your plots on the same graph to make comparison easy. Also, for each method, pick the best K by eye (if required). Then make a boxplot (one per dataset) comparing all the methods evaluated at their best K.

The methods you should implement are as follows.

- The simplest method is to use row imputation, where we replace X_{ij} (if it is missing) with $\overline{x}_{i,:}$, computed by averaging over the columns (in row i) which are not missing. Also try column imputation, where we use $\overline{x}_{:,j}$.
- Another simple but popular approach is **K-nearest neighbor imputation**, first proposed in [TCS+01]. To explain the idea, let us partition $\mathbf{X}$ into a set of rows which are fully observed, V, and the remaining rows, with missing values. Consider a row with missing values, call it i. Suppose that the first M entries are missing, and the remaining $D - M$ are observed. We compute the Euclidean distance of this row to each row in V using the last $D - M$ columns; let d_k be the distance to the k'th row in V, and let $\text{nbr}(i, K)$ be the indices of the K rows in V that are closest to row i. Then we fill in the missing entries using

$$\hat{X}_{ij} = \frac{\sum_{k\in\text{nbr}(i,K)} X_{kj}/d_k}{\sum_{k\in\text{nbr}(i,K)} 1/d_k} \tag{4}$$

  This approach is an example of a more general class of methods called **hot-deck imputation**, which fills in missing values on incomplete records using values from similar, but complete records of the same dataset.[1] Write a function `knnImputation` that implements the above method. (Note that `knnimpute` already exists in Matlab's bioinformatics toolbox. Also, there is an R package called `impute` (available at `http:`

[1] According to Wikipedia, "the term 'hot deck' dates back to the storage of data on punch cards, and indicates that the information donors come from the same dataset as the recipients; the stack of cards was 'hot' because it was currently being processed. Cold-deck imputation, by contrast, selects donors from another dataset."

//cran.r-project.org/web/packages/impute/impute.pdf) that implements this functionality. However, we need our own version for later modification.) Try using raw Euclidean distance, and Mahalanobis distance using a diagonal covariance matrix. Note that a naive implementation of KNN-impute takes $O(DN^2)$ time, but you can use approximate nearest neighbor methods to reduce this to $O(DN \log N)$ time if you wish.

- Use `mixGaussMissingFitEm` and `mixGaussImpute` from pmtk. (This method is described in [OWG04], except they got many details wrong!). You may need to perform MAP estimation for numerical stability. This takes $O(DNKT)$ time, where T is the number of iterations needed by EM. This is better than $O(DN \log N)$ but can still be quite slow. This method is similar in some ways to KNN imputation, since it fills in missing entries by averaging over K prototypes, but these prototypes are inferred using all the data in a statistically sound way.
- Modify the `mixGaussMissingFitEm` code so it uses mixtures of diagonal Gaussians.

3 Imputation of mixed data types

Consider the problem of imputing missing entries in a data matrix where some features are binary, some are categorical, and some are real-valued. (We could also imagine including ordinal data, count data, positive real-valued data, etc.) As a concrete example, consider the famous "adult" dataset from the UC Irvine repository, which was derived from the US Census in 1994. There are $N = 48,842$ records, and $D = 14$ attributes, of which 6 are continuous (all positive) and 8 are categorical (of which 1 is binary). See Table 1 for details.
3620 records containing one or missing entries. We will use the rows which have no missing entries. In addition, we will skip features 11 (capgain), 12 (caploss), and 14 (country), which are relatively constant across rows. Use the function `data = adultDataPreprocess()` on the class web page to load the data (in `adultCensus.zip`). Extract the first 1,000 rows and artificially create missing values by setting a random fraction p of the entries to NaN (use $p \in \{0.001, 0.01, 0.1\}$). (Use `setSeed(0)` first.) Apply one of the methods below to impute and then meaure accuracy as follows. For each row, define the loss as follows, where $\mathcal{C}$ is the set of continuous features and $\mathcal{D}$ is the set of discrete features:

$$L(\mathbf{x}_i, \hat{\mathbf{x}}_i) = \frac{1}{D}\left[\sum_{j\in\mathcal{C}}(x_{ij} - \hat{x}_{ij})^2 + \sum_{j\in\mathcal{D}}\mathbb{I}(x_{ij} \neq \hat{x}_{ij})\right] \tag{5}$$

Define $\overline{L} = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ as the loss for an entire imputed matrix.
For each method below, and for each percentage missing, compute the loss over 10 random trials. Plot the mean loss and its standard error vs K. To get a more detailed understanding of how well each method is doing, make scatter plots of x_{ij} vs $\hat{x}_{ij}$ for $j \in \mathcal{C}$, and print the R^2 value in the title (as in hw6). For the discrete features, compute confusion matrices where $C_j(c, c')$ is the number of times the true feature j has value c but the predicted feature j is c'. A perfect reconstruction would have all the entries on the diagonal. Thus a simple error measure computes the fraction of off-diagonal entries

$$E_j = \frac{1}{N}\sum_{c\neq c'} C_j(c, c') \tag{6}$$

Visualize the confusion matrices using `hintonDiagram` (or `imagesc` with `colorbar`). Print E_j in the title. Turn in your code and plots.
Implement and evaluate the following methods

1. Modify the KNNimputation function so it handles mixed data, by using Hamming distance on the discrete features.
2. A standard hack people use is to treat discrete data as if it was continuous. For ordinal data (as in the netflix competition, where people rated movies on a scale of 1 to 5), this is reasonable. But for categorical data, it does not make sense. However, you can convert the categorical features to binary using a 1-of-K encoding (see

Num	Name	Type	Values
1	age	Cts	-
2	workclass	Cat(8)	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
3	fnlwgt	Cts	-
4	education	Cat(16)	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
5	education-num	Cts	-
6	marital-status	Cat(7)	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
7	occupation	Cat(14)	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
8	relationship	Cat(6)	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
9	race	Cat(5)	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
10	sex	Cat(2)	Female, Male
11	capital-gain	Cts	-
12	capital-loss	Cts	-
13	hours-per-week	Cts	-
14	native-country	Cat(41)	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands
15	Income	Cat(2)	$> 50K, <= 50K$

Table 1: Description of the features in the "adult" dataset from the UC Irvine repository (`http://archive.ics.uci.edu/ml/datasets/Adult`). The feature `fnlwgt` stands for final weight, and is related to the sampling design used to collect the survey data. This was extracted by Barry Becker from the US 1994 Census database. The standard task is to predict the binary income label, but we can also use this data for unsupervised learning.

`dummyEncoding`). You can then replace 0s with -1s and treat all the data as continuous. Use this transform and apply your mixtures of Gaussians code. Try both full and diagonal covariances. Try $K \in \{1, 5, 10, 50\}$ mixture components. You will need to post-process your results to convert imputed continuous 1-of-K vectors back to a discrete value.

3. A more suitable model is to use a mixture model where the class-conditional density is a product of different scalar distributions, depending on the type of the feature, i.e.,

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \left[\prod_{j \in \mathcal{C}} \mathcal{N}(x_j|\mu_{jk}, \sigma_{jk}^2) \prod_{j \in \mathcal{D}} \text{Discrete}(x_j|\boldsymbol{\beta}_{jk}) \right] \quad (7)$$

Implement a function called `mixFactoredFitEm` which fits the model to data, where some of the entries may be missing (NaN). The interface should be as follows `model = mixFactoredFitEm(X, K, types)`, where K is the number of mixture components, and `types` is a string encoding the type of distribution to use for each feature, e.g., 'DGD' means use a Discrete for features 1 and 3 and a Gaussian for feature 2. Also, implement a function called `mixFactoredImpute` which imputes missing entries of a data matrix. For simplicity, use the posterior mean for continuous quantities and the posterior mode for discrete quantities. (This ignores any uncertainty in our predictions, but keeps things simple.) The interface should be as follows `Xhat = mixFactoredImpute(model, X)`, where the NaNs in `X` get replaced by point predictions in `Xhat`.

References

[ESBB98] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. of the National Academy of Science, USA*, 1998.

[GSK+00] A. Gasch, P. Spellman, C. Kao, O. Carmel-Harel, M. Eisen, G. Storz, D. Botstein, and P. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol. Biol. Cel*, 11:4241–4257, 2000.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009. 2nd edition.

[OWG04] M. Ouyang, W. Welsh, and P. Georgopolous. Gaussian mixture clustering and imputation of microarray data. *Bioinformatics*, 20:917–923, 2004.

[TCS+01] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.