You are given data associated with a speech classification task, where there are 8 classes. You have four sets of data: a general training set from 25 speakers, a small set of matched data for each of the test speakers, an independent test set with known labels for development, and a second unlabeled test set. Pick one of the following two options, and use the data to design and test three different classifiers: i) train on the general set only, ii) train on the matched set only, and iii) use the matched set together with the general set in an adaptation mode. The two options are:

1. Gaussian class-conditional distributions: Choose between full and diagonal covariance and justify your choice. For condition (iii), use MAP estimation of the mean, where the prior is the model that you found from general training, and use either the general variance or Bayesian learning to find the adapted model variance.

2. Nearest-neighbor: For condition (i), use a computation reduction technique that you explored in HW4. (No code sharing with others from the class.) For condition (ii), there is much less data, so you may not need to reduce computation. For condition (iii), you can get an adapted model by first using the matched data to edit the general data (i.e. find the nearest neighbors in the general set to the matched samples, and throw out all general samples that misclassify the matched data). Combine the remaining data and do further computation reduction as needed.

Use the three models to predict classes on the unlabeled test. Create a separate ascii file for each of the 3 conditions, with the predicted class label on a separate line for each test sample. Use the following naming convention for the files:

\texttt{<yourname>_<Gauss/KNN>_C<condition number>.txt}

The classifications and code should be emailed to the grader. The written discussion of your approach is due in hard copy with a signed version of this cover page. The discussion should include details of your approach, results on the development set, and discussion, written in the style of a technical report.

To constrain the scope of the problem and facilitate comparison of results, please do not experiment with any type of feature transformation, and only implement one of the two approaches (Gaussian vs. KNN). You will be graded on your description of the method you chose and how well you do for that particular method, but grades will not be based on performance relative to those who chose a different method.

\textbf{Grading}

This exam was graded on a scale of 30 points, allocated as follows:

- Performance relative to others for option chosen: 8 points
- Soundness of technical approach: 10 points
  Includes appropriate use of data for the different conditions, justification of assumptions, and solution for condition 3 (adaptation).
- Completeness of report: 12 points
  Includes details of approach, results on dev set, and discussion of differences. Up to 2 points may be taken off for typos and grammatical errors.
Background on the Problem

We did not give you much background on the problem, since we did not want those students with experience in speech recognition to apply their domain knowledge to the problem. For those who are interested, this is a vowel classification task, where the vowels are: ae, a, aa, ow, uw, ix, iy, and ey. The data is a small subset of the data collected in the Vocal Joystick project, directed by Prof. Bilmes (Thanks for Jon Malkin for pre-preprocessing the data). In this project, speakers use the different vowels to control movement of a mouse. For more information, see:


Option 1: Gaussian Class-Conditional Distributions

The choices that you needed to make in this case were related to the covariance matrix: full vs. diagonal for all parts, and in part (3) whether you use the general or matched covariance or a weighted combination. Typical results that people achieved for the different choices are given below for both the development and evaluation test sets. For all parts, classification should be based on the MAP rule using the log probability form:

\[ \omega^* = \arg\max_i \log p(x_i|\omega_i)p(\omega_i) = \arg\max_i \log p(x_i|\omega_i) \]

\[ = \arg\min_i \left[ \log |\Sigma_i| + (x - \mu_i)'\Sigma_i^{-1}(x - \mu_i) \right] \]

where the class prior is dropped since all classes are equally probable. If you didn’t use the log form, you may have had numerical problems.

1. General training: In this case, where you have a large amount of data relative to the dimensionality of the vector, the full covariance assumption makes sense. However, you might decide based on comparing performance of the full and diagonal covariance on the dev set, in which case you would conclude that the diagonal covariance is the right choice. Either solution is acceptable. The equations for the full covariance using the maximum likelihood estimate are:

\[ \hat{\mu}_i^G = \frac{1}{N_i} \sum_{t:l(t)=i} x_t \]

\[ \hat{\Sigma}_i^G = \left( \frac{1}{N_i} \sum_{t:l(t)=i} x_t'x_t \right) - (\hat{\mu}_i^G)'\hat{\mu}_i^G \] (1)

(You could also use the unbiased version of \( \hat{\Sigma} \) which normalizes by \( N - 1 \) rather than \( N \).) For the diagonal covariance case, it is best to work with scalars, rather than estimate a full covariance and zero out the off-diagonal terms, both for numerical reasons and efficiency.

\[ \hat{\sigma}_{i,j}^2 = \frac{1}{N_i} \sum_{t:l(t)=i} x_{i,j}^2 - \hat{\mu}_{i,j}^2 \] (2)

In this case,

\[ |\Sigma_i| = \prod_{j=1}^{d} \sigma_{i,j}^2 \]

\[ (x - \mu_i)'\Sigma_i^{-1}(x - \mu_i) = \sum_{j=1}^{d} (x_j - \mu_{i,j})^2 / \sigma_{i,j}^2 \] (3)

As it turned out, the dev and eval test sets were fairly different, and the full covariance was the best choice. The accuracy (percent correct) on the two test sets for the two assumptions is given below. Either alternative was acceptable.

2
2. **Matched training:** In this case, where you have a much smaller amount of data, the diagonal covariance assumption makes sense. Again, you might decide based on comparing performance of the two alternatives on the dev set, in which case you would conclude that the diagonal covariance is the right choice. The diagonal covariance also gave the best performance on the eval set. The accuracy (percent correct) on the two test sets for the two assumptions is given below.

<table>
<thead>
<tr>
<th>Covariance</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev</td>
</tr>
<tr>
<td>Full</td>
<td>86.7</td>
</tr>
<tr>
<td>Diagonal</td>
<td>92.3</td>
</tr>
</tbody>
</table>

3. **Adaptation:** In this problem, you were told to use a MAP estimate for the mean, assuming that the prior for the mean is the same as the general distributions, so \( \mu_i \sim N(\mu_i^G, \Sigma_i^G) \). In the MAP estimation example that you had in class, the covariance was given, i.e. \( \Sigma_i \) for \( p(x|c_i) \sim N(\hat{\mu}_i, \Sigma_i) \).

In this case, the MAP estimate is:

\[
\hat{\mu}_i = \Sigma_i^G[\Sigma_i^G + \frac{1}{n}\Sigma_i]^{-1}\mu_i^M + \frac{1}{n}\Sigma_i[\Sigma_i^G + \frac{1}{n}\Sigma_i]^{-1}\mu_i^G
\]

where \( n \) is the number of samples in the adaptation (matched) data. To get the solution, you need to assume something about \( \Sigma_i \). In addition, you had choices on what covariance to use in classification, since Bayesian learning argues for a broader covariance than \( \Sigma_i \) due to estimation error. There were several possible solutions:

- **Assume** \( \Sigma_i = \Sigma_i^G \) for both the MAP mean update and the classification. One could argue that \( \Sigma_i = \Sigma_i^G \) is not a good choice, since \( \Sigma_i^G \) describes the variation of many speakers, and you will have only one. However, the bigger spread may be good for accounting for estimation error in classification.

- **Assume** \( \Sigma_i = \Sigma_i^M \) in the MAP mean estimate and \( \Sigma_i = \Sigma_i^G \) for the classification. One could argue \( \Sigma_i = \Sigma_i^M \) (the covariance for the matched training data) in the MAP equation is closer to the true case because of only one speaker, but it is less reliable. Again, the bigger spread of \( \Sigma_i^G \) may be good for accounting for estimation error in classification.

- **Assume** \( \Sigma_i = \Sigma_i^M \) in the mean estimation formula above, and then use Bayesian learning (BL) to get the covariance used in classification:

\[
\Sigma_i = \Sigma_i^M + \frac{1}{n}\Sigma_i^G[\Sigma_i^G + \frac{1}{n}\Sigma_i]^{-1}\Sigma_i^M
\]

Note that this is effectively \( \Sigma_i = \Sigma_i^M + \alpha\Sigma_i^G \), and you could instead find \( \alpha \) from the dev set.

- **Assume** \( \Sigma_i = \Sigma_i^M \) in the mean estimation formula above, and then use regularization (i.e. interpolation of the matched and general covariances) to get the covariance used in classification:

\[
\Sigma_i = \alpha\Sigma_i^M + (1-\alpha)\Sigma_i^G
\]

- Use regularization as above to get \( \Sigma_i \) to be used in both the mean formula and the covariance for classification.

3
For both of the last two (or three) options, find $\alpha$ on the dev set. All three options effectively give you a linear combination of $\Sigma^M_i$ and $\Sigma^G_i$. Most (if not all) of the best results were achieved with full covariance, but the diagonal assumption was not explored in all cases. The best results were obtained using the second approach, with the weight on $\Sigma^G_i$ less than 0.35 (and typically smaller for people who searched more). The accuracy range on the two test sets for different approaches is given below.

<table>
<thead>
<tr>
<th>$\Sigma_i$ in $\hat{\mu}_i$ in classifying</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Sigma^G_i$</td>
<td>$\Sigma^G_i$</td>
</tr>
<tr>
<td>$\Sigma^M_i$</td>
<td>$\Sigma^G_i$</td>
</tr>
<tr>
<td>$\Sigma^M_i$</td>
<td>$\Sigma^G_i$ Regularize</td>
</tr>
<tr>
<td>$\Sigma^M_i$</td>
<td>$\Sigma^G_i$ BL/reg</td>
</tr>
<tr>
<td>Regularize</td>
<td>Regularize</td>
</tr>
</tbody>
</table>

Results for cases where the mean estimate was incorrect are not included. In many of these cases, good performance was achieved on the dev set (92-96% accuracy), but poor performance on the eval set, suggesting the possibility of overtraining on the dev set.

**Option 2: Nearest-Neighbor Classifier**

While the original intention of this problem was for you to do 1-nearest neighbor, the instructions were ambiguous enough that many people did KNN, so both solutions were acceptable.

1. **General training:** The implementation of kNN is straightforward, so most people got similar results for both the General training and the Matched training cases. The variations involve vanilla neighborhood-based kNN vs. distance-based kNN (i.e. weighing samples by its inverse distance to the test point), different heuristics to deal with ties, and different computational speed-up techniques. The different varieties of kNN gave roughly similar performance, as long as $k$ was tuned on the dev set.

   Interestingly, the distribution of dev accuracy vs. $k$ is bimodal, with optimal $k$ occurring at low values ($k < 10$) as well as at a very high values e.g. $700 < k < 1000$. The hypothesis was that the general training data was from multiple speakers, and thus the classes contain multiple modes. This seems like a reasonable explanation, but has not been verified.

<table>
<thead>
<tr>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev Eval</td>
</tr>
<tr>
<td>82 61</td>
</tr>
</tbody>
</table>

2. **Matched training:** For the same implementation of kNN (but $k$ tuned on the dev set once again), matched training results were significantly better than the general training condition. Good ranges of $k$ ranged from 1 to 6. Importantly, whereas there was a significant degradation in Eval results for General condition, the results for the Matching condition was quite good.

<table>
<thead>
<tr>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev Eval</td>
</tr>
<tr>
<td>90 96</td>
</tr>
</tbody>
</table>
3. **Adaptation:** The main idea for kNN adaptation is to pick a good subset of the General training data and augment it to the Matched data. There were a variety of different heuristics used, with a varying degree of success. The basic algorithm is:

(a) Use kNN from General dataset to classify Matched dataset
(b) Delete samples from the General dataset that resulted in poor classification
(c) Use kNN on the Augmented dataset to classify Dev/Eval data.

There are two $k$ parameters to tune now, one for Step 1 and one for Step 3. Some people used the same $k$ from Condition 1 for Step 1, while others opted to tune both $k$’s jointly on the dev set.

Step 2 was critical for the success of the algorithm: one needs to retain as many general samples as possible, which effectively increases the coverage of the adapted kNN classifier in feature space. However, if too many general samples are retained such that some are noisy, then the adapted classifier will suffer.

One heuristic was to delete a general sample whenever it was part of a group that misclassified a matched sample. Some people felt that this was too harsh and instead tallied how many times a given general sample was responsible for correct vs. incorrect classification. For example, every time a general sample is part of a group that classified correctly, it gets a positive count; every time it misclassifies, its count is decremented. Finally, one needs to choose a threshold on the counts to determine the proportion of general samples to retain.

Regardless of the heuristics used in Step 2, there is an additional issue that needs to be considered: some general samples are simply not close enough to any matched sample and therefore no decision can be reached about how it classifies a test sample. Some people chose to delete all these points, whereas others chose to retain all these points. An alternative solution to avoid this problem is to flip the editing problem around: i.e. use the matched sample to classify each general sample, and retain the general samples that were correctly classified.

It is difficult to conclude what heuristic gave the best performance, since there were many parameters to tune and minute implementation differences can result in significant result deviation. In general, it appears that people who performed a more comprehensive experimentation and data analysis could eventually get the adaptation condition to outperform all previous conditions.

<table>
<thead>
<tr>
<th>% Correct</th>
<th>Dev</th>
<th>Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>88-92</td>
<td>94-96</td>
<td></td>
</tr>
</tbody>
</table>