Last Lecture: The Simplex Method

- General method for optimization
  - Makes few assumptions about function

- Crawls towards minimum

- Some recommendations
  - Multiple starting points
  - Restart maximization at proposed solution
Summary:
The Simplex Method

- Original Simplex
- High
- Low
- Reflection
- Contraction
- Reflection and Expansion
- Multiple Contraction
Other Strategies for Multidimensional Optimization

Most strategies will define a series of vectors or lines through parameter space.

- Estimate of minimum improved by adding an optimal multiple of each vector.

Some intuitive choices might be:
  - The function gradient
  - Unit vectors along one dimension
The key is to right angle turns!

Most methods that use derivatives don’t simply optimize function along current gradient or the unit vectors …
Today ...

- The E-M algorithm
  - General algorithm for missing data problems
  - Requires "specialization" to the problem at hand
  - Frequently applied to mixture distributions
The E-M Algorithm

- Original Citation
  - Dempster, Laird and Rubin (1977)
    *J Royal Statistical Society (B)* 39:1-38
  - Cited in over 6,082 research articles

- For comparison
  - Nelder and Mead (1965)
    *Computer Journal* 7: 308-313
  - Cited in over 6,762 research articles
The Basic E-M Strategy

- \( X = (Y, Z) \)
  - Complete data \( X \)  \textit{(eg. what we’d like to have!)}
  - Observed data \( Y \) \textit{(eg. individual observations)}
  - Missing data \( Z \) \textit{(eg. class assignments)}

- The algorithm
  - Use estimated parameters to infer \( Z \)
  - Update estimated parameters using \( Y \) and \( Z \)
  - Repeat until convergence
The E-M Algorithm

- Consider a set of starting parameters
- Use these to “estimate” the missing data
- Use “complete” data to update parameters
- Repeat as necessary
Setting for the E-M Algorithm...

- Problem is simpler to solve for complete data
  - Maximum likelihood estimates can be calculated using standard methods

- Estimates of mixture parameters could be obtained in straightforward manner if the origin of each observation is known…
Filling In Missing Data

- The missing data is the group assignment for each observation

- Complete data generated by assigning observations to groups
  - Probabilistically
  - We will use “fractional” assignments
Classification Probabilities

\[
\Pr(Z_i = j \mid x_i, \pi, \phi, \eta) = \frac{\pi_j f(x_i \mid \phi_j, \eta)}{\sum_l \pi_l f(x_i \mid \phi_l, \eta)}
\]

- Results from the application of Bayes' theorem
- Implemented in `classprob()` function
  - `classprob(int j, double x, int k, double *prob, double *mean, double *sd)`
void update_class_prob(int n, double * data,
    int k, double * prob, double * mean, double * sd,
    double ** class_prob)
{
    int i, j;

    for (i = 0; i < n; i++)
        for (j = 0; j < k; j++)
            class_prob[i][j] =
                classprob(j, data[i],
                        k, prob, mean, sd);
}
Updating Mixture Proportions

\[ \pi_i = \frac{\Pr(Z_i = j \mid x_i, \pi, \varphi, \eta)}{n} \]

- "Count" the observations assigned to each group
void update_prob(int n, double * data,
                int k, double * prob,
                double ** class_prob)
{
    int i, j;

    for (int j = 0; j < k; j++)
    {
        prob[j] = 0.0;

        for (int i = 0; i < n; i++)
            prob[j] += class_prob[i][j];

        prob[j] /= n;
    }
}
Updating Component Means

\[ \hat{\mu}_j = \frac{\sum x_i \Pr(Z_i = j \mid x_i, \pi, \varphi, \eta)}{\sum \Pr(Z_i = j \mid x_i, \pi, \varphi, \eta)} \]

- Calculate weighted mean for group
- Weights are probabilities of group membership
### C Code:
**Update Component Means**

```c
void update_mean(int n, double * data,
                 int k, double * prob, double * mean,
                 double ** class_prob)
{
    int i, j;

    for (int j = 0; j < k; j++)
    {
        mean[j] = 0.0;

        for (int i = 0; i < n; i++)
            mean[j] += data[i] * class_prob[i][j];

        mean[j] /= n * prob[j] + TINY;
    }
}
```
Updating Component Variances

\[ \hat{\sigma}_i^2 = \frac{\sum_i (x_i - \mu_i)^2 \Pr(Z_i = j \mid x_i, \pi, \varphi, \eta)}{n \pi_j} \]

- Calculate weighted sum of squared differences
- Weights are probabilities of group membership
```c
void update_sd(int n, double * data,
    int k, double * prob, double * mean, double * sd,
    double ** class_prob)
{
    int i, j;

    for (int j = 0; j < k; j++)
    {
        sd[j] = 0.0;

        for (int i = 0; i < n; i++)
        {
            sd[j] += square(data[i] - mean[j]) * class_prob[i][j];
        }

        sd[j] /= (n * prob[j] + TINY);
        sd[j] = sqrt(sd[j]);
    }
}
```
C Code: Update Mixture

```c
void update_parameters
   (int n, double * data,
    int k, double * prob, double * mean, double * sd,
    double ** class_prob)
{
   // First, we update the mixture proportions
   update_prob(n, data, k, prob, class_prob);

   // Next, update the mean for each component
   update_mean(n, data, k, prob, mean, class_prob);

   // Finally, update the standard deviation
   update_sd(n, data, k, prob, mean, sd, class_prob);
}
```
E-M Algorithm For Mixtures

1. “Guesstimate” starting parameters

2. Use Bayes' theorem to calculate group assignment probabilities

3. Update parameters using estimated assignments

4. Repeat steps 2 and 3 until likelihood is stable
double em(int n, double * data, 
  int k, double * prob, double * mean, double * sd, 
  double eps)
{
  double llk = 0, prev_llk = 0;
  double ** class_prob = alloc_matrix(n, k);

  start_em(n, data, k, prob, mean, sd);
  do {
    prev_llk = llk;
    update_class_prob(n, data, k, prob, mean, sd, class_prob);
    update_parameters(n, data, k, prob, mean, sd, class_prob);
    llk = mixLLK(n, data, k, prob, mean, sd);
  } while ( !check_tol(llk, prev_llk, eps) );

  return llk;
}
## Picking Starting Parameters

- **Mixing proportions**
  - Assumed equal

- **Means for each group**
  - Pick one observation as the group mean

- **Variances for each group**
  - Use overall variance
void start_em(int n, double * data,
               int k, double * prob, double * mean, double * sd)
{
  int i, j; double mean1 = 0.0, sd1 = 0.0;

  for (i = 0; i < n; i++)
    mean1 += data[i];
  mean1 /= n;
  for (i = 0; i < n; i++)
    sd1 += square(data[i] - mean1);
  sd1 = sqrt(sd1 / n);

  for (j = 0; j < k; j++)
  {
    prob[j] = 1.0 / k;
    mean[j] = data[rand() % n];
    sd[j] = sd1;
  }
}
Example Application
Old Faithful Eruptions (n = 272)
Using Simplex Method
A Mixture of Two Normals

- Fit 5 parameters
  - Proportion in the first component
  - Two means
  - Two variances

- Required about ~700 evaluations
  - First component contributes 0.34841 of mixture
  - Means are 2.0186 and 4.2734
  - Variances are 0.055517 and 0.19102

  - Maximum log-likelihood = -276.36
Using Simplex Method
A Mixture of Two Normals

- Fit 5 parameters
  - Proportion in 1st component, 2 means, 2 variances

- 44/50 runs found minimum

- Required about ~700 evaluations
  - First component contributes 0.348 of mixture
  - Means are 2.018 and 4.273
  - Variances are 0.055 and 0.191

  - Maximum log-likelihood = -276.36
Using E-M Algorithm
A Mixture of Two Normals

- Fit 5 parameters

- 50/50 runs found maximum

- Required about ~25 evaluations
  - First component contributes 0.348 of mixture
  - Means are 2.018 and 4.273
  - Variances are 0.055 and 0.191

- Maximum log-likelihood = -276.36
Two Components

Old Faithful Eruptions

Fitted Distribution

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Duration (mins)

Duration (mins)
Simplex Method:
A Mixture of Three Normals

- Fit 8 parameters
  - 2 proportions, 3 means, 3 variances

- Required about ~1400 evaluations
  - Found best solution in 7/50 runs
  - Other solutions effectively included only 2 components

- The best solutions …
  - Components contributing 0.339, 0.512 and 0.149
  - Component means are 2.002, 4.401 and 3.727
  - Variances are 0.0455, 0.106, 0.2959
  - Maximum log-likelihood = -267.89
Three Components

Old Faithful Eruptions

Duration (mins)

Frequency

1 2 3 4 5 6

0 5 10 15 20

Fitted Distribution

Duration (mins)

Density

0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
E-M Algorithm: A Mixture of Three Normals

- Fit 8 parameters
  - 2 proportions, 3 means, 3 variances

- Required about ~150 evaluations
  - Found log-likelihood of ~267.89 in 42/50 runs
  - Found log-likelihood of ~263.91 in 7/50 runs

- The best solutions …
  - Components contributing .160, 0.195 and 0.644
  - Component means are 1.856, 2.182 and 4.289
  - Variances are 0.00766, 0.0709 and 0.172
  - Maximum log-likelihood = -263.91
Three Components

Old Faithful Eruptions

Fitted Density
Convergence for E-M Algorithm

LogLikelihood

Iteration

Likelihood

LogLikelihood

Iteration
Convergence for E-M Algorithm

Mixture Means

Iteration

Mean

0 50 100 150 200

0 1 2 3 4 5
E-M Algorithm: A Mixture of Four Normals

- Fit 11 parameters
  - 3 proportions, 4 means, 4 variances

- Required about \( \sim 300 \) evaluations
  - Found log-likelihood of \( \sim 267.89 \) in 1/50 runs
  - Found log-likelihood of \( \sim 263.91 \) in 2/50 runs
  - Found log-likelihood of \( \sim 257.46 \) in 47/50 runs

- "Appears" more reliable than with 3 components
Four Components

Old Faithful Eruptions

<table>
<thead>
<tr>
<th>Duration (mins)</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Duration (mins)</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>0.6</td>
</tr>
<tr>
<td>5</td>
<td>0.8</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Today ...

- The E-M algorithm
- Missing data formulation
- Application to mixture distributions
  - Consider multiple starting points
Further Reading

- There is a nice discussion of the E-M algorithm, with application to mixtures at: