Probabilistic Methods

Motivations

- Specify a model structure (BYOModel)
  - Parametric versus nonparametric
- Sometimes you need a simulation
  - No closed form solution
  - We must use a simulation to discover optimal
- Can we even find a solution and how will we know when we have it?
  - MLE
  - Convergence

DataScience@SMU

Maximum Likelihood Estimation

Maximum Likelihood Estimation

Fundamental question: What parameters maximize the expectation of seeing the observed data?

- Always: find best model for data
- In parametric models: What parameters for the model are best?
- Best = highest chance of seeing the observed data → MLE

Fundamental solution: Maximize the likelihood of seeing your observed data from considered parameter space given a model.

- Try different parameter values (models)
- Evaluate how likely the observed data is using these models
- Pick best one

A (Toy) Example

- Suppose you need to model a possibly biased coin (~ binary classification), and observe the data **HHTH** (H/T for heads/tails). Assuming: (i) all flips come from the same coin; (ii) each flip is independent; (iii) the coin has a fixed probability β of coming up H (hence (1- β) of coming up T) → the parameter of the model is simply the scalar β. ***MLE:*** **select β that maximizes the probability of the data under β.**
- Probability of the data (D) is is given by:

$$
\begin{aligned}
p_\beta(D) &= p_\beta(\text{HHTH}) && \textbf{definition of } \boldsymbol{D} && (9.2)\\
&= p_\beta(\text{H})\, p_\beta(\text{H})\, p_\beta(\text{T})\, p_\beta(\text{H}) && \textbf{data is independent} && (9.3)\\
&= \beta\beta\,(1-\beta)\,\beta && && (9.4)\\
&= \beta^3\,(1-\beta) && && (9.5)\\
&= \beta^3 - \beta^4 && && (9.6)
\end{aligned}
$$

Source: *A Course in Machine Learning*, Ha Daume III (http://ciml.info/)

A (Toy) Example

- To find β that maximizes $p_\beta(D)$, take the derivative of $p_\beta(D)$, set it to 0, solve for β:

$$\frac{d}{d\beta}[\beta^3 + \beta^4] = 3\beta^2 + 4\beta^3 = 0 \rightarrow \beta = \frac{3}{4}$$

- Afterthought, β = 0.75 is just the empirical average. However, as the parameter space grows, the maximization problem becomes nontrivial → EM algorithm.

MLE Version .01

- **Problem: find best parameters for modeling data**
  - Brute force/exhaustive search
    - Try everything
  - Sampling
    - Try random ones
- Many solutions use some combination of these

MLE Version 1

- **Problem: find best parameters for modeling data**
  - Find parameters that have highest probability of seeing data
  - Find parameters that **maximize** something
- **Gradient Descent**
  - Minimization problem + differentiation gives slope
  - → We know which way to go!
- ***Three basic approaches:***
  1. **Finite differences:** local calculations
  2. **Symbolic differentiation:** manipulate formulas
  3. **Automatic differentiation:** preform calculations

DataScience@SMU

Gaussian Mixture Models

Gaussian Mixture Models

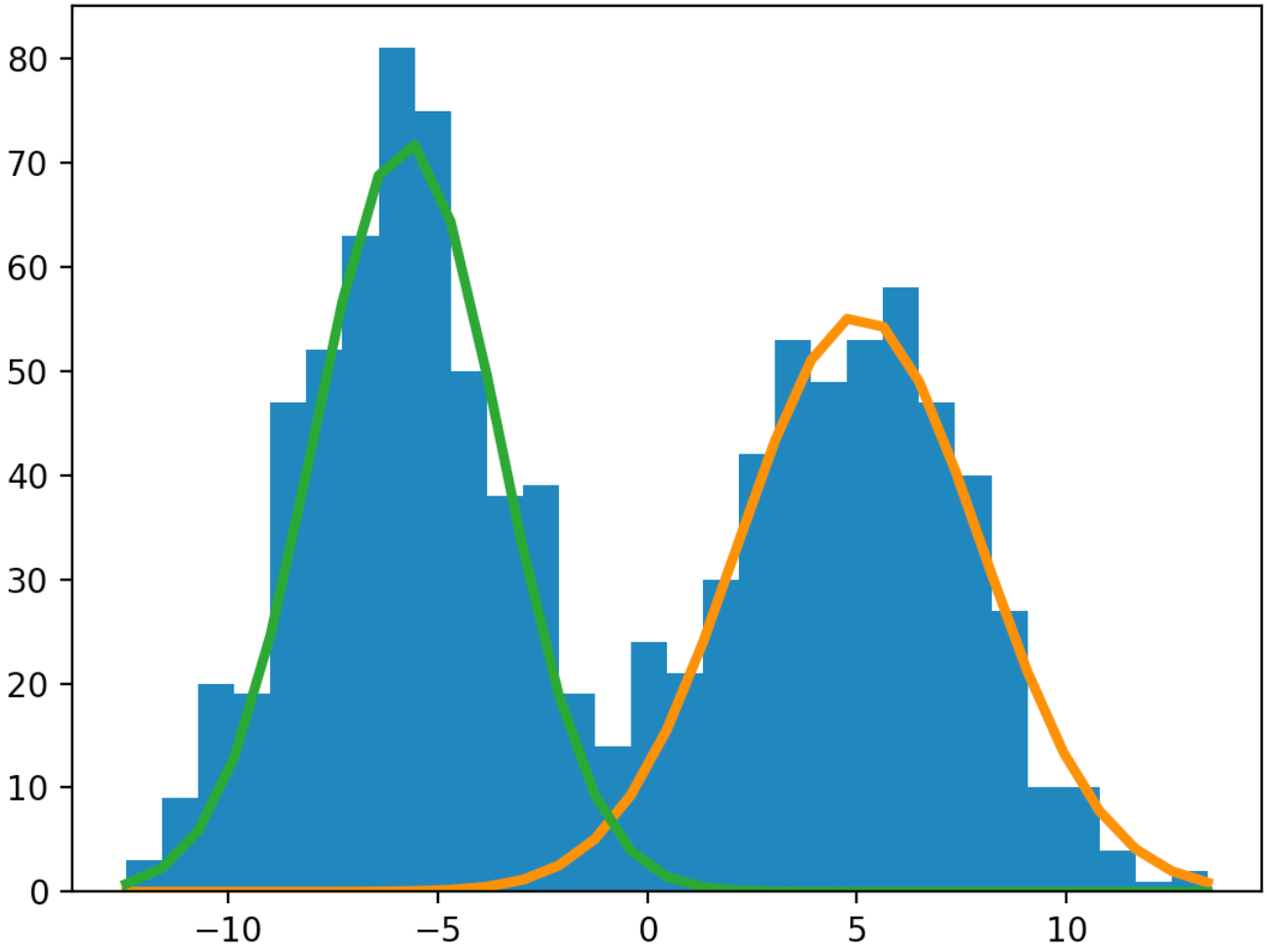
The Problem:

- You have a graph that you believe is a composite of other hidden graphs. You assume all the underlying graphs are Gaussian. Can you construct a plausible graph (i.e., very close to the empirical graph) by using a collection of Gaussian graphs?

Alternatively:

- I worked on a crowd sourcing problem at CMU years ago. We were interested in evaluating crowd sourced data. We plotted a histogram of the average accuracy for each individual as the x axis, while the y value was the count. The histogram was bimodal. By using GMM we could see that data was well represented by two communities of users, those with an very higher accuracy, and those with an accuracy approximately equal to chance. This provided justification that would allow us to evaluate each new user and determine if we should treat that user as well intentioned, or just randomly choosing, which would remove them from the study.

Gaussian Mixture Models



DataScience@SMU

Hidden Markov Models

Hidden Markov Models

The Problem:

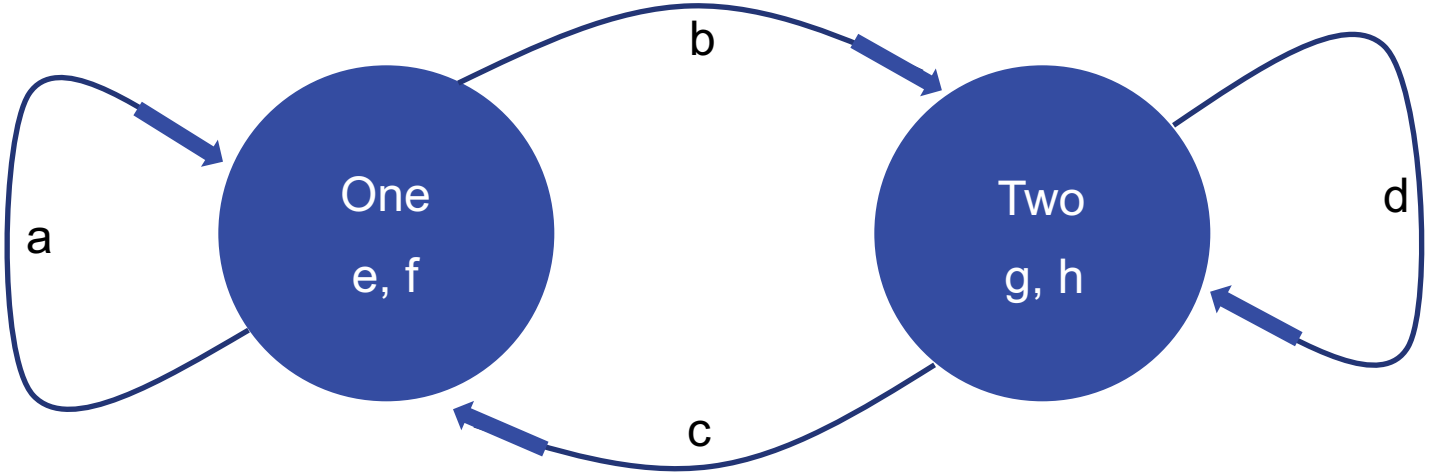
- You believe you have a system of two hidden states. You can't be sure if you are in state One or state Two at any given point in time. Your system switches between the states with some probability (transition probabilities). Each state gives you a set of outputs with different probabilities (emission probabilities). Your job is to calculate all the probabilities (transition and emission).

Alternatively:

- You bosses have good days and bad days. Sometimes they yell at you over how you do your job, sometimes they give you praise. Sometimes, even if its their good day, they yell at you and sometimes, even on a bad day, they give you praise. Can you determine how they transition from good days to bad and with what probability in either state they will yell at you or give you praise?

Hidden Markov Models

A particular type of graphical model:



Transition probabilities {a, b, c, d}

Emission probabilities {e, f, g, h}

DataScience@SMU

Multi-armed Bandits

Multi-armed Bandits

The Problem:

- You have a collection of slot machines (one-armed bandits) and they each pay you a certain amount of money. Your job is to discover as fast as possible what the right machine is to maximize your profits.

Alternatively:

- You can choose to show to people on a website one of a collection of possible ads. What strategy of ad display should you choose to optimize the number of conversions?

Multi-armed Bandits

```
import numpy as np
import csv, math, random

probs = [1703,1295,728,1196,2695,926,1112,2091,952,489]
rows = 10000
cols = 10
dataset = np.zeros((rows,cols))
for c in range(cols):
  for r in random.sample(list(range(rows)), probs[c]):
    dataset[r,c]=1

random_reward = 0
for r in range(rows):
  random_reward+= dataset[r, np.random.randint(cols)]

print(random_reward) # our baseline: playing randomly the slots

# Result 1339.0
```

Multi-armed Bandits

```
times_tried = np.zeros(cols)
reward = np.zeros(cols)
for row in range(rows):
  row_values = list(np.zeros(cols))
  for col in range(cols):
    if times_tried[col]==0:
      row_values[col] = 1000
    else:
      row_values[col] = ((reward[col]/times_tried[col]) +
                         (math.sqrt(2*math.log(row+1)/times_tried[col])))
  max_idx = row_values.index(np.max(row_values))
  times_tried[max_idx] += 1
  reward[max_idx] += dataset[row, max_idx]

print(random_reward)
print(sum(reward))
```

DataScience@SMU