Item Response Theory
CS398
Goals

- Can use AutoGrad in Torch
- Can use Item Response Theory to simulate students
- Can use Item Response Theory to estimate student abilities given data
There are many open problems in education?
Open Research Problems

Feedback to teachers?
Open Research Problems

Autonomous Tutor Gym
Let's dive in
What is Item Response Theory

A model which gives a probability that a particular student will answer a particular question correctly
Warmup
import torch

def main():
    data = load_data()
    model = MyModel()
    optimize(model, data)
import torch

# The machine learning paradigm.
def main():
    data = loadData()
    model = MyModel()
    optimize(model, data)
```python
import torch

# The machine learning paradigm.
def main():
    data = loadData()
    model = MyModel()
    optimize(model, data)
```
```python
import torch

# The machine learning paradigm.

def main():
    data = loadData()
    model = MyModel()
    optimize(model, data)
```
Gradient Ascent / Descent

Choose parameters that **minimize loss** (or maximize likelihood)

Walk uphill and you will find a local maxima (if your step size is small enough)

Note: Logistic regression LL function is convex
Example
Logistic Regression

\[
P(y^{(i)} = 1) = \sigma(\sum_{j} \theta_j \cdot x_j^{(i)})
\]
Logistic Regression

\[ P(y^{(i)} = 1) = \sigma(\sum_{j} \theta_j \cdot x_j^{(i)}) \]
Inputs $x = [0, 1, 1]$ 

$P(y^{(i)} = 1) = \sigma(\sum_j \theta_j \cdot x^{(i)}_j)$
\[ P(y^{(i)} = 1) = \sigma\left( \sum_j \theta_j \cdot x_j^{(i)} \right) \]
\[ P(y^{(i)} = 1) = \sigma(\sum_{j} \theta_j \cdot x_j^{(i)}) \]
\[
P(y^{(i)} = 1) = \sigma\left(\sum_j \theta_j \cdot x_j^{(i)}\right)
\]

Inputs

\[x_0\]

\[\theta_0\]

\[x_1\]

\[\theta_1\]

\[x_2\]

\[\theta_2\]

\[x_3\]

\[\theta_3\]
Weights

\[ P(y^{(i)} = 1) = \sigma\left(\sum_j \theta_j \cdot x_j^{(i)}\right) \]
Weighed Sum

\[ P(y^{(i)} = 1) = \sigma(\sum_{j} \theta_j \cdot x_j^{(i)}) \]
Squashing Function

\[ P(y^{(i)} = 1) = \sigma(\sum_j \theta_j \cdot x_j^{(i)}) \]
Prediction

\[ P(y^{(i)} = 1) = \sigma(\sum_{j} \theta_j \cdot x_j^{(i)}) \]
Parameters Affect Prediction

\[ P(y^{(i)} = 1) = \sigma \left( \sum_{j} \theta_j \cdot x_j^{(i)} \right) \]
Parameters Affect Prediction

\[ P(y^{(i)} = 1) = \sigma(\sum_{j} \theta_j \cdot x_j^{(i)}) \]

\[ z = -1.5 \quad \sigma(z) = 0.4 \]
Parameters Affect Prediction

\[ P(y^{(i)} = 1) = \sigma\left(\sum_{j} \theta_j \cdot x_j^{(i)}\right) \]
Different Predictions for Different Inputs

\[ P(y^{(i)} = 1) = \sigma \left( \sum_{j} \theta_j \cdot x_j^{(i)} \right) \]

\[ z = 2.1 \quad \sigma(z) = 0.7 \]
Different Predictions for Different Inputs

\[ P(y^{(i)} = 1) = \sigma \left( \sum_j \theta_j \cdot x_j^{(i)} \right) \]

\[ z = 2.1 \]

\[ \sigma(z) = 0.7 \]

\[ P(y^{(i)} = 1) \]
Different Predictions for Different Inputs

$x_0$  $\theta_0$

$x_1$  $\theta_1$

$x_2$  $\theta_2$

$x_3$  $\theta_3$

$z = -1.9$

$\sigma(z) = 0.3$

$P(y^{(i)} = 1) = \sigma \left( \sum_j \theta_j \cdot x_j^{(i)} \right)$
Logistic Regression - Lite

\[ P(y^{(i)} = 1) = \sigma(\theta_0 + \theta_1 \cdot x^{(i)}) \]
Gradient Ascent / Descent

Walk uphill and you will find a local maxima (if your step size is small enough)

Note: Logistic regression LL function is convex
Logistic Regression

1. Make a prediction

\[ P(y^{(i)} = 1) = \sigma(\theta_0 + \theta_1 \cdot x^{(i)}) \]
Logistic Regression

1. Make a prediction

\[ P(y^{(i)} = 1) = \sigma(\theta_0 + \theta_1 \cdot x^{(i)}) \]

2. Calculate the prediction loss (- log likelihood)

\[
\text{Loss}(\theta) = \sum_{i=0}^{n} y^{(i)} \log \sigma(\theta^T x^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T x^{(i)})]
\]

3. Get derivative of log likelihood with respect to thetas

\[
\frac{\partial \text{Loss}}{\partial \theta_j} = \sum_{i=0}^{n} \left[ y^{(i)} - \sigma(\theta^T x^{(i)}) \right] x_j^{(i)}
\]
Logistic Regression

1. Make a prediction

2. Calculate the prediction loss (log likelihood)

$$LL(✓) = \sum_{i=0}^{n} y(i) \log(✓^T x(i)) + (1 - y(i)) \log[1 - ✓^T x(i)]$$

3. Get derivative of log likelihood with respect to thetas

$$\partial \theta_j \sum_{i=0}^{n} [y(i) x(i)_j - (1 - y(i)) x(i)_j]$$
Logistic Regression - Modern

1. Make a prediction

\[ P(y^{(i)} = 1) = \sigma(\theta_0 + \theta_1 \cdot x^{(i)}) \]

2. Calculate the prediction loss (- log likelihood)

\[ \text{Loss}(\theta) = \sum_i \text{binaryCrossEntropy}( P(y^{(i)} = 1), y^{(i)} ) \]
Binary Cross Entropy

- Consider I.I.D. random variables $X_1, X_2, \ldots, X_n$
  - $X_i \sim \text{Ber}(p)$
  - Probability mass function, $f(X_i \mid p)$:

$$f(X_i \mid p) = p^{x_i} (1 - p)^{1-x_i}$$

- Likelihood of Bernoulli

$$f(x) = 0.2^x (1 - 0.2)^{1-x}$$
Logistic Regression

1. Make a prediction

\[ P(y^{(i)} = 1) = \sigma(\theta_0 + \theta_1 \cdot x^{(i)}) \]

2. Calculate the prediction loss (- log likelihood)

\[ \text{Loss}(\theta) = \sum_i \text{binaryCrossEntropy}( P(y^{(i)} = 1), y^{(i)} ) \]

3. Get derivative of log likelihood with respect to thetas

\[ \frac{\partial \text{Loss}}{\partial \theta_j} = \text{loss.backward()} \]

Auto grad
def main():
    data = loadData()
    model = MyModel()
    optimize(model, data)

# 1) define the model you are optimizing over...
# This is a component of a neural net or any
# parameterized function you are optimizing over.
class LogRegressionModel(nn.Module):
    # the initialize method
    def __init__(self):
        super().__init__()  # necessary

        # define any parameters are part of the model
        initial1 = torch.ones(1) * 0
        initial2 = torch.ones(1) * 0
        self.theta1 = nn.Parameter(initial1)
        self.theta0 = nn.Parameter(initial2)

    # this what happens when you apply the function
    # to input. In this case I have implemented
    # y = sigma(self.theta1 * x + self.theta0)
    def forward(self, x):
        a = self.theta1 * x + self.theta0
        return torch.sigmoid(a)

1. Make a prediction
   \[ P(y^{(i)} = 1) = \sigma(\theta_0 + \theta_1 \cdot x^{(i)}) \]

2. Calculate the prediction loss (- log likelihood)
   \[ \text{Loss}(\theta) = \text{binaryCrossEntropy} \left( P(y^{(i)} = 1) , y^{(i)} \right) \]

3. Get derivative of log likelihood with respect to thetas
   \[ \frac{\partial \text{Loss}}{\partial \theta_j} = \text{loss.backward()} \]

   Auto grad
How would you solve a problem like this?

Human responses to different questions

Tell me what the students know

What if this is Khan Academy data? What if you get to chose the next item?
What is Item Response Theory

A **model** which gives a probability that a particular student will answer a particular question correctly
What is Item Response Theory

A model which gives a probability that a particular student will answer a particular question correctly.

“This student has a 70% chance of answering problem 5 correctly”

Used for:
1. Student simulators
2. Reasoning about what a student knows
3. Adaptive tests
Q3: If you pick an answer to this question at random, what is the chance that you will be correct?

a) 25%  
b) 60%  
c) 50%  
d) 25%
If student $i$ attempts problem $j$, the likelihood they answer it correctly is...

$$p_{i,j} = \sigma(a_i - d_j)$$
\[ p_{i,j} = \sigma(a_i - d_j) \]

Learner ability: \( a_i \)

Question difficulty: \( d_j \)

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

Problem is way too easy for the student

Problem is way too hard

Zone of proximal development
Probabilistic Generative Model – A way to simulate

\[ \begin{align*}
&d_1 \\
&d_2 \\
&d_3
\end{align*} \]

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\[ d_j \text{ is given} \]

Gives you a method to simulate.
Probabilistic Generative Model – A way to simulate

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\[ d_j \text{ is given} \]

Gives you a method to simulate.
Probabilistic Generative Model – A way to simulate

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\[ d_j \text{ is given} \]

Gives you a method to simulate.
Probabilistic Generative Model – A way to simulate

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\( d_j \) is given

Gives you a method to simulate.
Probabilistic Generative Model – A way to simulate

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\[ d_j \text{ is given} \]

Gives you a method to simulate.
Probabilistic Generative Model

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\[ d_j \text{ is given} \]

Gives you a method to simulate.
Probabilistic Generative Model

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]

\[ d_j \text{ is given} \]

\[ q_1 \]
\[ q_2 \]
\[ q_3 \]
\[ q_4 \]
Create an imaginary problem, and an imaginary student and simulate the response
Is this the **full** story?
A brief history of IRT

Charles Darwin writes Origin of the Species, 1859

Francis Galton is inspired: creates psychometrics, 1889

Education Testing Services create IRT, 1950

Computers become terribly clever, 2019
If student \( i \) attempts problem \( j \), the likelihood they answer it correctly is...

\[
\sigma(x) = \frac{1}{1 + e^{-x}}
\]

\[
p_{i,j} = \sigma(a_i - d_j)
\]
If student $i$ attempts problem $j$, the likelihood they answer it correctly is...

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Guess: probability correct by chance

$$p_{i,j} = c + [1 - c] \cdot \sigma(a_i - d_j)$$

Probability correct

Ability of student $i$

Difficulty of problem $j$
If student $i$ attempts problem $j$, the likelihood they answer it correctly is...

$$p_{i,j} = c + [1 - c] \cdot \sigma(k_j[a_i - d_j])$$

**Guess:** probability correct by chance

**Ability of student $i$**

**Difficulty of problem $j$**

**Discrimination of problem $j$**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
If student $i$ attempts problem $j$, the likelihood they answer it correctly is...

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]

Guess: probability correct by chance

\[ \rho_{i,j} = c + [1 - c] \cdot \sigma(a_i - d_j) \]

Probability correct

Ability of student $i$

Difficulty of problem $j$
Update your simulation the response
If student $i$ attempts problem $j$, the likelihood they answer it correctly is...

$$p_{i,j} = \sigma(a_i - d_j)$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- $p_{i,j}$: Probability correct
- $\sigma$: Squashing function
- $a_i$: Ability of student $i$
- $d_j$: Difficulty of problem $j$
Probabilistic Generative Model

\[ a_1 \quad a_2 \quad \ldots \quad a_n \]

\begin{array}{ccc}
  d_1 & d_2 & d_3 \\
  \bullet & \bullet & \bigcirc \\
  \bullet & \bigcirc & \bullet \\
  \bigcirc & \bullet & \bullet \\
\end{array}

Unobserved variables
Probabilistic Generative Model

Observed variables
Probabilistic Generative Model

<table>
<thead>
<tr>
<th></th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td>$a_2$</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$a_n$</td>
<td>[ ]</td>
<td>[ ]</td>
<td>[ ]</td>
</tr>
</tbody>
</table>

Gives you a method to simulate.

You can calculate abilities (unobserved) given responses (observed).
Estimate ability for a single student

\[ a_i \sim N(\mu = 0, \text{var} = 1) \]
Why sigmoid?
$$p_{i,j} = \sigma(a_i - d_j)$$

Learner ability: $$a_i$$

Question difficulty: $$d_j$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Problem is way too easy for the student

Problem is way too hard

Zone of proximal development
Probability Correct

Letter Size (arc mins)

Floored Exponential

Observed

Sigmoid

$c$ (guess prob.)

20/20

20/60

20/100

$20/100$ $k_0$ $20/60$ $20/20$
Floored Exponential

\[ p(x) = \max \left\{ c, 1 - e^{-\lambda_i(x-a_i)} \right\} \]

Guess: probability correct by chance

Discernibility

Base ability

Letter size
1. Take an eye exam on this website

2. Connect your phone

3. Visualize the math