# A brief note on model learning

Matthew Engelhard

# How does training, or learning, work?

I want to find the *best* model parameters.

First, I need to define what *best* means.

-> define the *loss*

-> the *best* parameters are those that minimize it

Then, I need a strategy for adjusting my parameters to make my model *better…*

-> …in other words, a strategy for reducing the *loss*

-> this is the field of *optimization*

# I want to find the *best* dinner recipe.

This doesn't make sense until I define what *best* means.

-> My measure of *best*: highest possible $\frac{\text{calories}}{\text{cost x time}}$

-> "a lot food for cheap, *right now*"

Then, I need a strategy for adjusting my ingredients and prep.

-> increase cheap, high calorie ingredients

-> decrease ingredients with longer prep/cook time

# I want to find the *best* dinner recipe.

This doesn't make sense until I define what *best* means.

-> My measure of *best*: highest possible $\frac{\text{calories}}{\text{cost x time}}$

-> "a lot food for cheap, *right now*"

Then, I need a strategy for adjusting my ingredients and prep.

-> increase cheap, high calorie ingredients

-> decrease ingredients with longer prep/cook time

# How does training, or learning, work?

I want to find the *best* model parameters.

First, I need to define what *best* means.

-> for binary classification, we use the *cross-entropy loss*

-> the *best* parameters are those that minimize it

Then, I need a strategy for adjusting my parameters to make my model *better…*

-> our strategy for reducing it is called *stochastic gradient descent*

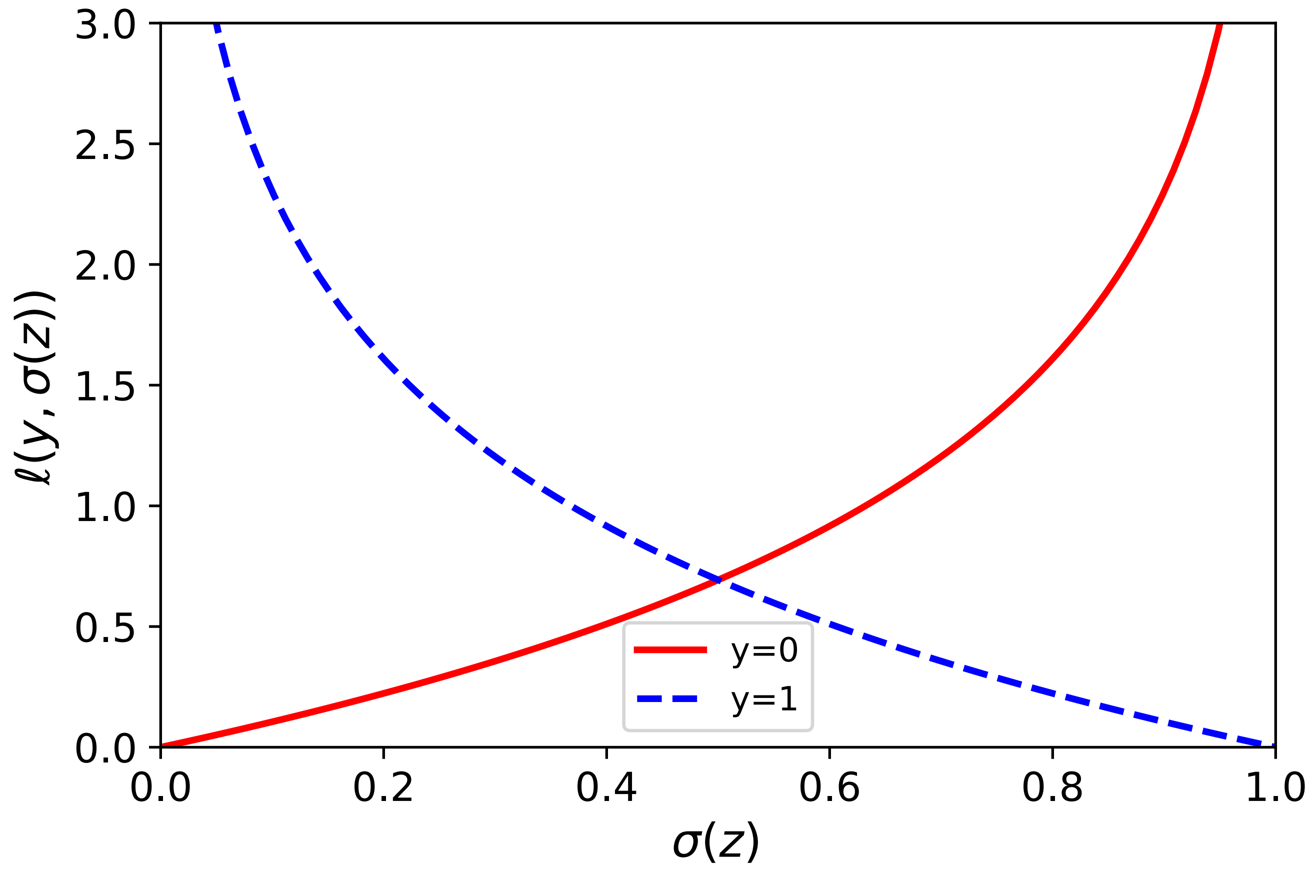
-> this is an example of an *optimization algorithm*

# Loss Function for Binary Classification

*"cross-entropy loss"*

When the outcome $y$ is 0, the model-predicted probability $p$ must be close to 0, otherwise the loss (i.e., the penalty) is large

When the outcome $y$ is 1, the model-predicted probability $p$ must be close to 1, otherwise the loss (i.e., the penalty) is large



# A Simple Strategy for Optimizing Parameters

Here’s our game.

1. We want to reduce the loss as much as we can by adjusting our model’s parameters.

2. We can move any parameter up or down from its current value.

3. If the loss increases, *we’re going the wrong way*, and we should try moving that same parameter in the opposite direction.

4. If the loss decreases, *we’re going the right way*, and we should keep moving that parameter in that same direction until the loss stops decreasing.

5. We will keep trying this for all of the parameters until we can no longer decrease the loss; then we’re done.

# Let’s try this strategy on a small dataset.

**MORTALITY PREDICTION WORKSHEET**

**COVARIATES**

| patient | age | age_normalized | female | temp | temp_normalized |
|---|---|---|---|---|---|
| 0 | 30.5 | -0.5 | 0 | 105.0 | 2.4 |
| 1 | 74.0 | 1.1 | 1 | 96.7 | -0.8 |
| 2 | 27.4 | -0.6 | 0 | 96.1 | -1.0 |
| 3 | 0.1 | -1.5 | 1 | 98.5 | -0.1 |
| 4 | 0.7 | -1.5 | 1 | 96.5 | -0.9 |
| 5 | 49.9 | 0.2 | 1 | 97.1 | -0.6 |
| 6 | 72.9 | 1.0 | 1 | 100.1 | 0.5 |
| 7 | 29.1 | -0.5 | 1 | 99.6 | 0.3 |
| 8 | 83.5 | 1.4 | 1 | 100.6 | 0.7 |
| 9 | 82.3 | 1.4 | 1 | 95.2 | -1.3 |
| 10 | 23.7 | -0.7 | 0 | 99.4 | 0.2 |
| 11 | 12.9 | -1.1 | 0 | 96.6 | -0.8 |
| 12 | 53.9 | 0.4 | 1 | 100.3 | 0.6 |
| 13 | 18.8 | -0.9 | 0 | 98.6 | 0.0 |
| 14 | 51.8 | 0.3 | 0 | 98.5 | -0.1 |
| 15 | 3.3 | -1.4 | 0 | 94.6 | -1.6 |
| 16 | 69.7 | 0.9 | 0 | 99.1 | 0.1 |
| 17 | 60.4 | 0.6 | 1 | 104.2 | 2.1 |
| 18 | 73.6 | 1.1 | 1 | 99.1 | 0.1 |
| 19 | 53.3 | 0.3 | 1 | 99.1 | 0.1 |

**OUTCOMES AND PREDICTIONS**

| mortality | predicted_log_odds | predicted_prob | prediction | correct? | loss |
|---|---|---|---|---|---|
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 1 | 0.00 | 0.50 | 0 | 0 | 0.3010 |
| 0 | 0.00 | 0.50 | 0 | 1 | 0.3010 |

**PARAMETERS**

| | b_age | b_female | | b_temp | bias |
|---|---|---|---|---|---|
| guess | 0.00 | 0.00 | | 0.00 | 0.00 |
| optimal | [redacted] | | | | |

**PERFORMANCE**

| accuracy | avg_loss |
|---|---|
| 0.65 | 0.3010 |

# Parameter Optimization by Gradient Descent

- We are trying to get to the great valley (minimum loss).
- We do not know where it is (cannot test all parameter values).
- We cannot see clearly in any direction (complex, high-dimensional function).
- So, we test the ground around us, and walk downhill (use calculus / calculate gradient).

# Parameter Optimization by stochastic Gradient Descent

- We are trying to get to the great valley (minimum loss).
- We do not know where it is (cannot test all parameter values).
- We cannot see clearly in any direction (complex, high-dimensional function).
- So, we test the ground around us, and stumble downhill (use calculus / calculate gradient).

# Parameter Optimization by Gradient Descent

- We are trying to get to the great valley (minimum loss).
- We do not know where it is (cannot test all parameter values).
- We cannot see clearly in any direction (complex, high-dimensional function).
- So, we test the ground around us, and walk downhill (use calculus / calculate gradient).