Module 4

Lecture 5

Introduction to Machine Learning: Supervised Learning

Sarah Groves, Ph.D.

Shayn Peirce-Cottler, Ph.D.

Pull from upstream remote!

If you get a merge conflict on .gitignore, choose YOUR version

Where we are so far…

Week 1

STEP 1: Learn about cancer (and a specific cancer type if desired!)

STEP 2: Explore the data-set (you will be given one)

STEP 3: Ask a question related to the data set that you are curious about

Week 2

STEP 4: Visualize the data with **dimensionality reduction** (unsupervised)

STEP 5: Implement clustering, **regression**, or **classification** using Python/scikit-learn to answer the question (supervised)

Week 3

STEP 6: Verify and validate the answer using **ML metrics**

STEP 7: Draw conclusions and **reflect on the ethical implications**

Week 4

STEP 8: Communicate your question's background, methods, conclusions, limitations, implications, and future work

Due Thursday April 9 by class:
Cancer Hallmark ABM simulations activity

Due Friday April 10 midnight:
GitHub Check-in #1
Due Tuesday April 14 by class:
Video comprehension quiz

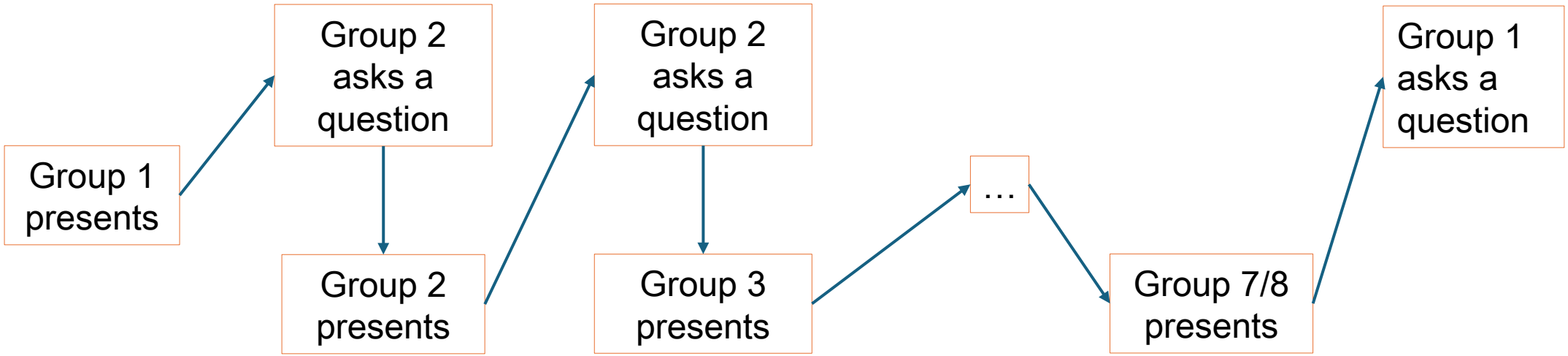
Due SUNDAY APRIL 19 midnight:
GitHub Check-in #2

Due SUNDAY APRIL 26
midnight:
GitHub Check-in #3

Due Tuesday May 5 midnight:
Final Project
Presentations: May 5 2pm (Final exam block)

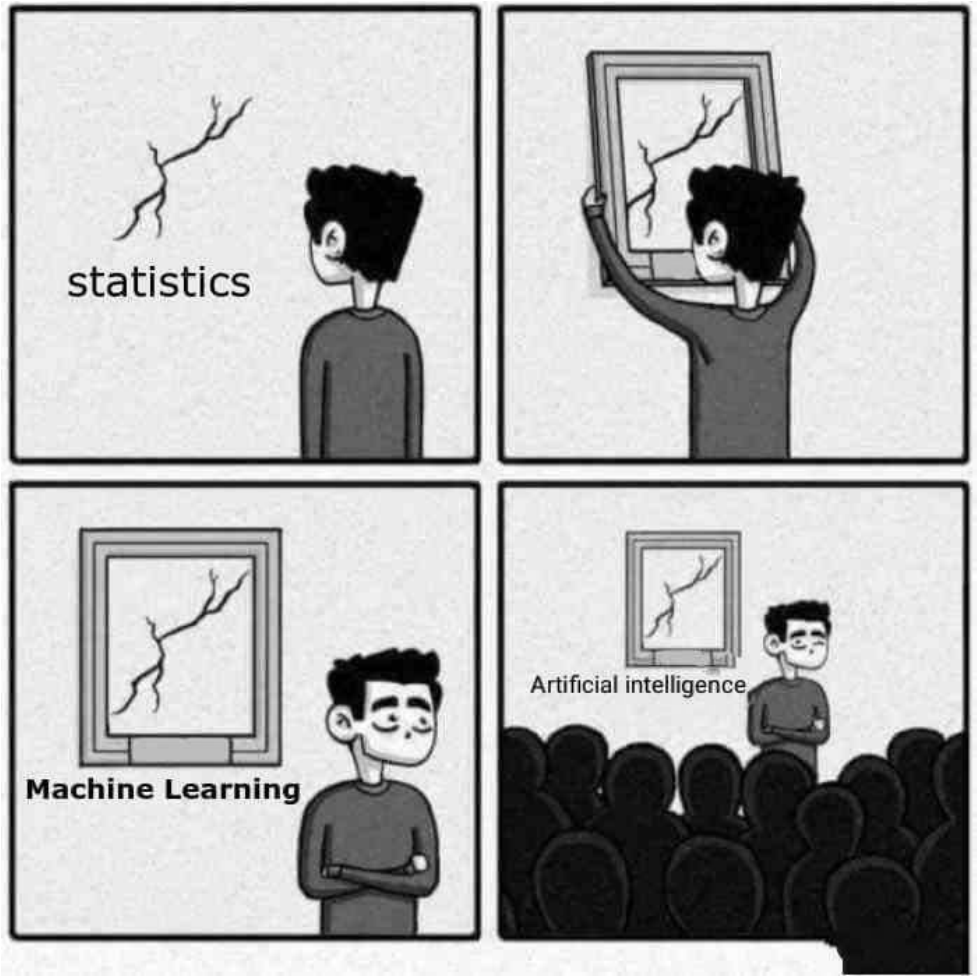
Project check-in- in class

- Share with your team what you are planning for your project, including your project goal and the specific method you will use to answer it (classification or regression)
- If you are not sure how to choose your target variable, now is a GREAT time to get advice!
- The next team to present should first ask a question of the previous team
- The team who went first should ask a question of the team that goes last



Supervised learning methods

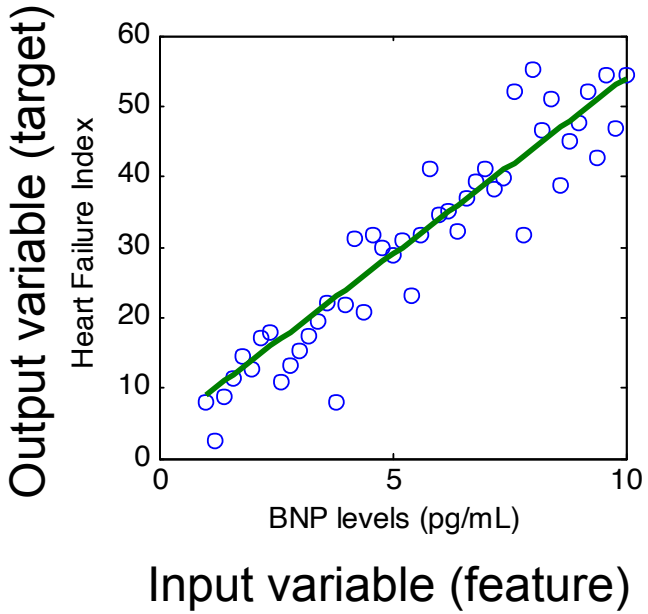
- **I have "labeled" data**
  - I am trying to predict something about new data and have a dataset with the "true answer/label/target variable"
  - Predicting survival from gene expression
  - Predicting expression of one gene from another
  - Predicting subtype/stage based on gene expression
- Regression
  - Linear regression
    - Optimizing the SSE using gradient descent
- Classification
  - Logistic regression
    - Optimizing logistic cost function
  - SVMs
  - Naïve Bayes
  - Decision trees



https://scikit-learn.org/stable/supervised_learning.html

Linear regression (2.0)

Predict a number from infinitely many possible outputs

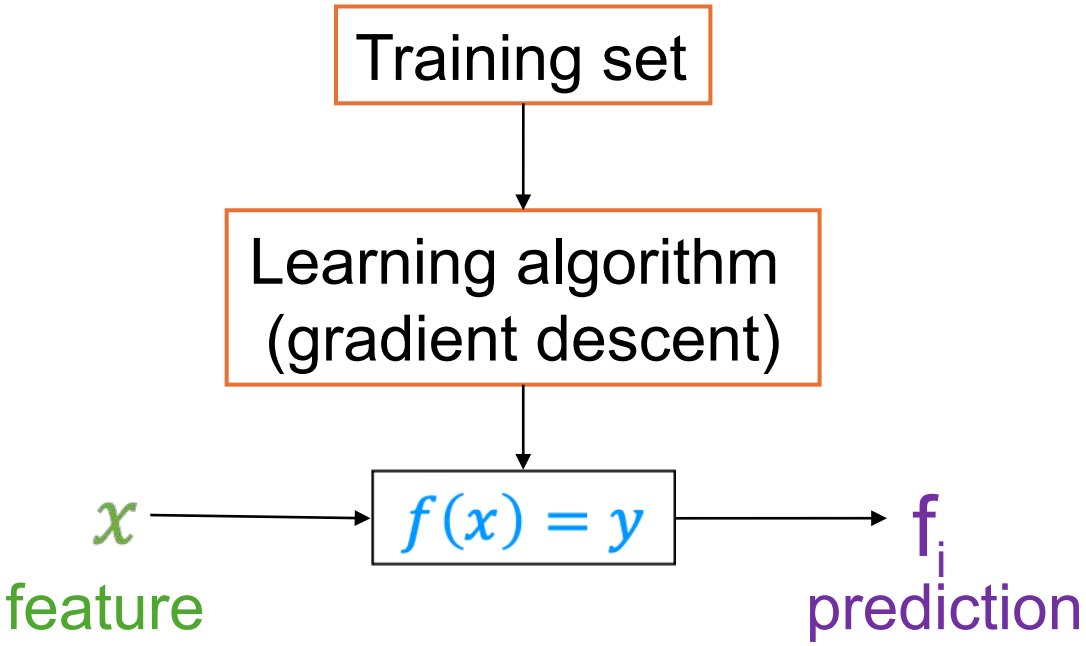


Goal: find the line (f) that best fits the data (xi, yi) by minimizing the sum of squared errors

$$f_i(x_i) = b + w_i\,(x_i)$$

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - f_i)^2$$

m = number of training examples (number of datapoints with x and y information)



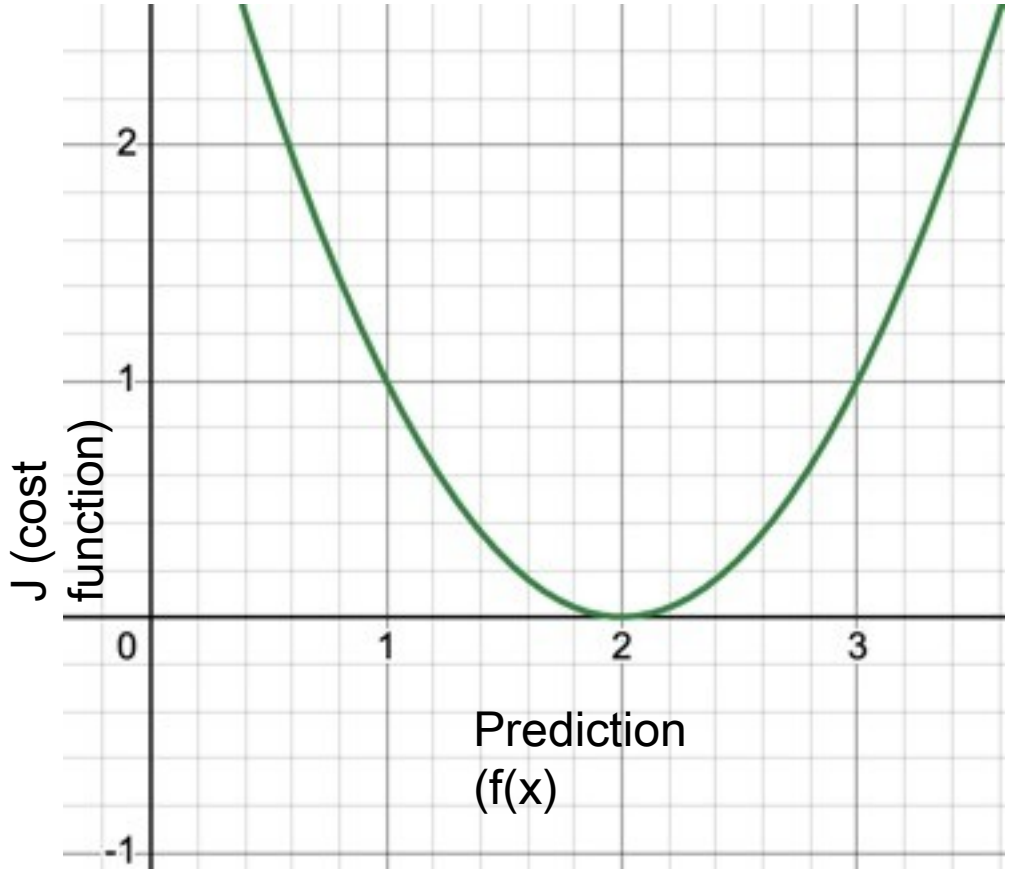
Linear regression (2.0): Optimizing a cost function

- How well is our model doing? (And how can the computer make it better?)
- Our model parameters: b, w
- Our cost function is the average SSE:

$$J(b,w) = \frac{1}{2m} SSE = \frac{1}{2m} \sum_{i=1}^{m} (y_i - f_i)^2$$

- GOAL:

$$\min_{b,w} J(b,w)$$



Linear regression (2.0): Gradient descent (optimization)

- Start with some b, w
- Keep changing b, w to reduce J
  - Use the gradient in each direction and a step size to change each
- Until we settle at or near a minimum

Steepest descent (numerical method)

$$x_{i+1} = x_i + hf'(x_i)$$

- Proceed along the path of steepest descent, i.e. direction with the greatest negative change
  - In 1D, there are only 2 ways to go
- Smaller steps as xi approaches xmax
- If h is too small, it will converge slowly
- If h is too big, it may overshoot the maximum

Gradient descent (ML implementation)

Updates (simultaneously):

$$b = b - \alpha \frac{\delta}{\delta b} J(b,w) = \frac{1}{m} \sum_{i=1}^{m} (y_i - f_i) x_i$$

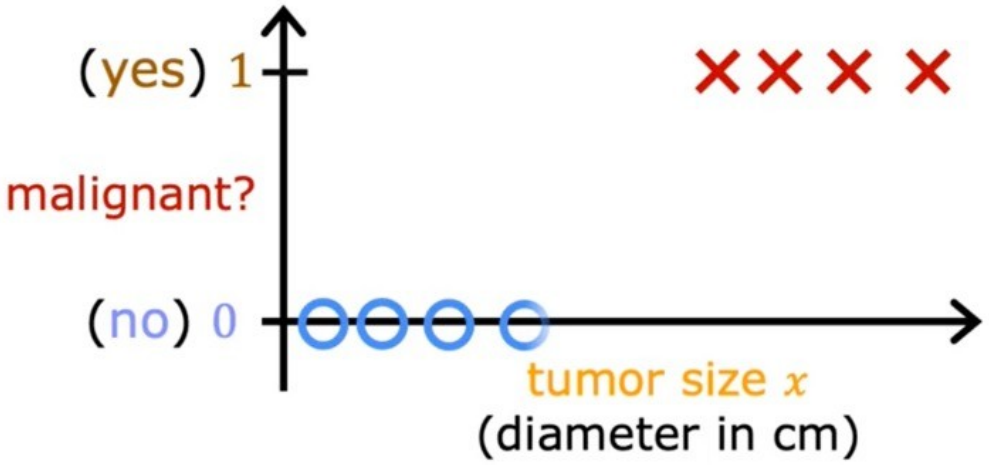
$$w = w - \alpha \frac{\delta}{\delta w} J(b,w) = \frac{1}{m} \sum_{i=1}^{m} (y_i - f_i)$$

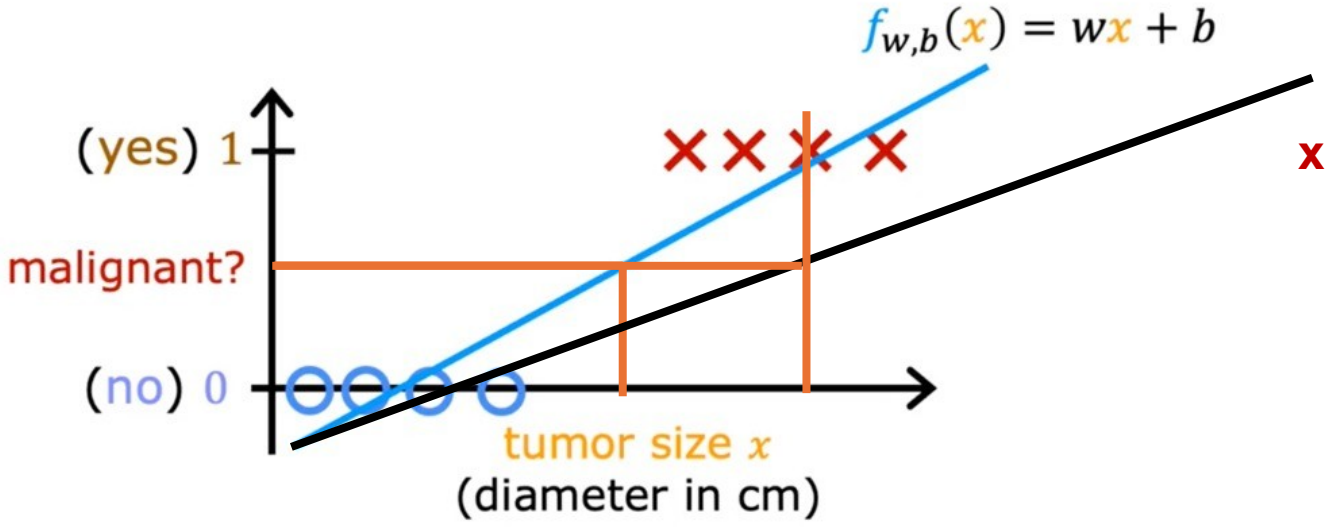
α = learning rate (step size)

Classification

- Binary classification: two possible classes

Predict a category from a small number of possible outputs



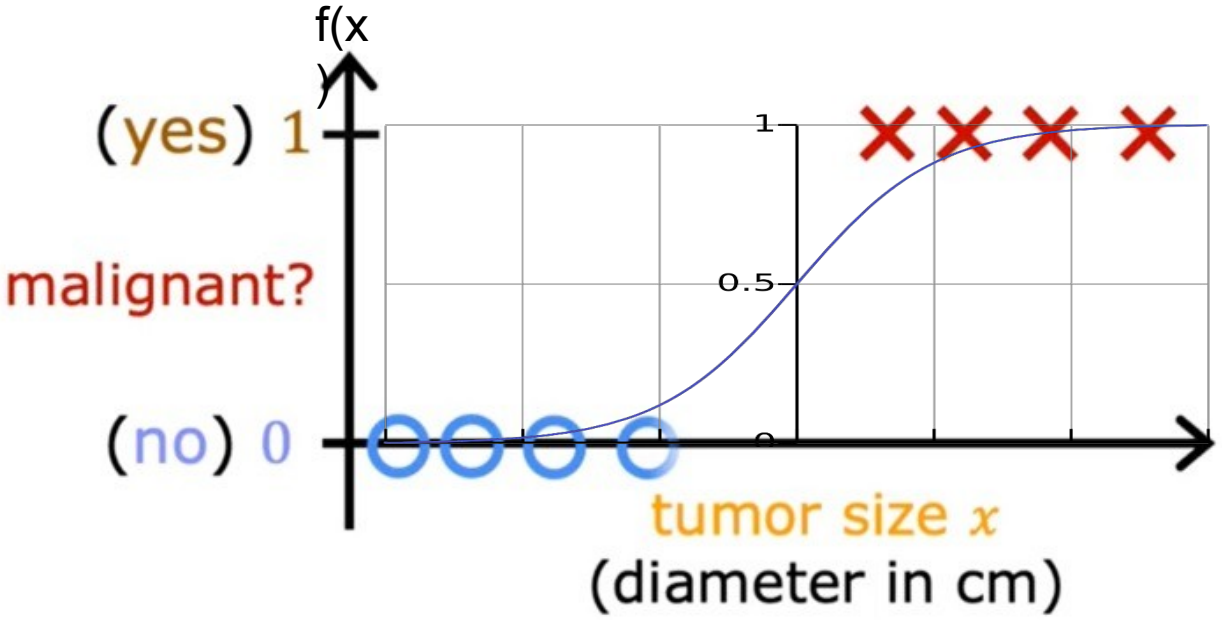


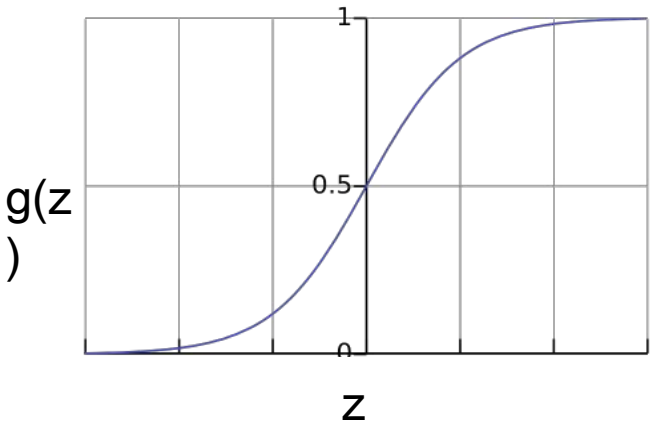
Linear regression with threshold?

What if we have a sample with very high tumor size?

Adapted from Coursera, Andrew

Logistic regression





- Sigmoidal/logistic function

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$z(x) = b + w\,(x)$$

$$f(x) = g(z(x)) = \frac{1}{1 + e^{-(b+w(x))}}$$

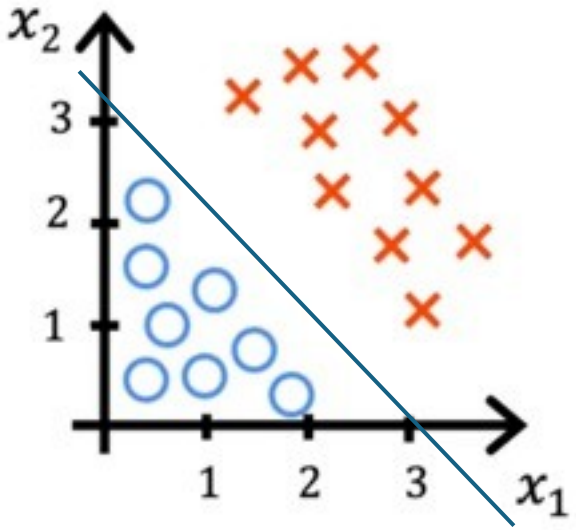
Outputs between 0 and 1

When is f(x) ≥ 0.5?

When z ≥ 0

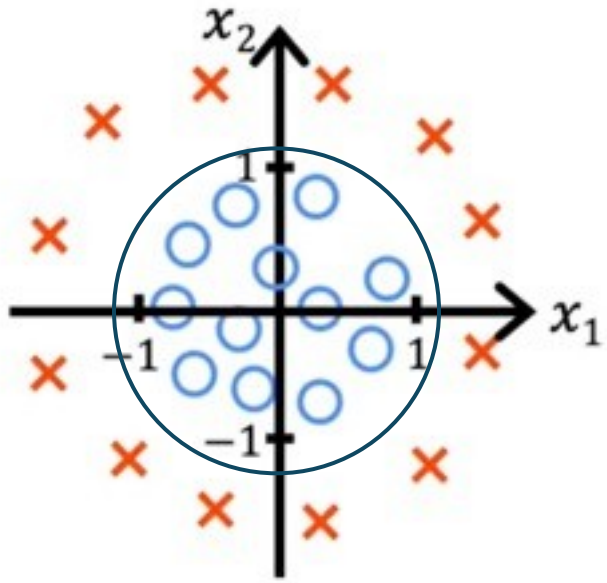
When $b+w(x) \geq 0$

Decision boundary: where $z = 0$



$$z(x) = b + w_i\,(x_i)$$

$$z(x) = 0 = x_1 + x_2 - 3$$



$$z(x) = 0 = {x_2}^2 + {x_1}^2 - 1$$

Output of function will still be 0 or 1! Now “inside” or “outside”

Cost function → logistic loss function

Linear regression

Loss, L

$$J = \frac{1}{m}\sum_{i=1}^{m}\boxed{\frac{1}{2}(y_i - f_i)^2}$$

Cost function

$$J = \frac{1}{m}\sum_{i=1}^{m} L(f_i, y_i)$$

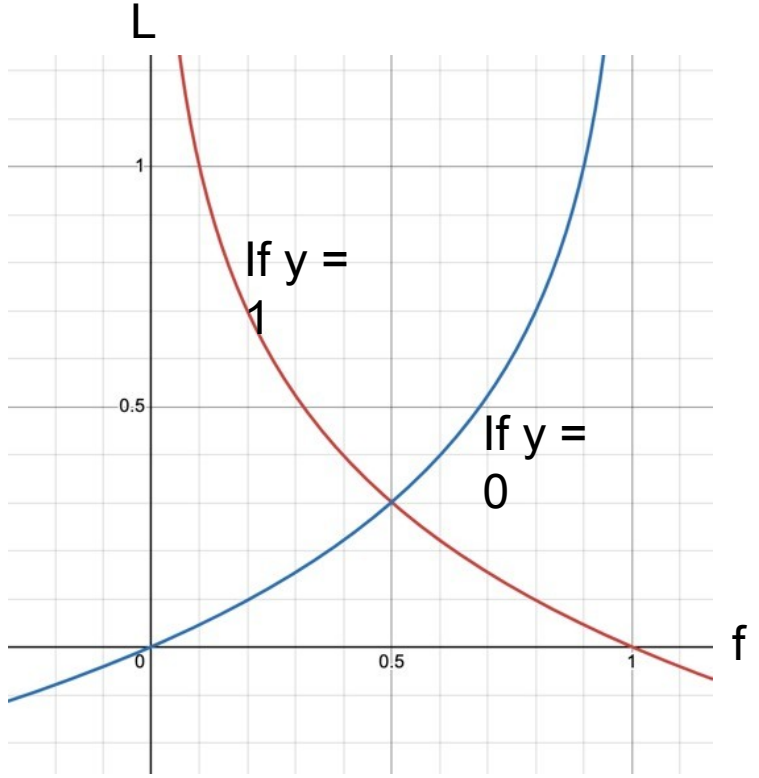
But this optimization problem would not be easy to solve with gradient descent (convex) if we plug the logistic f(x) in to the average SSE cost function!

$$f(x_i) = \frac{1}{1 + e^{-(b+w(x_i))}}$$

Logistic regression

Loss, L

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & \text{if } y^{(i)} = 0 \end{cases}$$



p = f(x) = predicted probability that y=1

$$L_{\log}(y, p) = -(y \log(p) + (1 - y)\log(1 - p))$$

Decision tree classifier

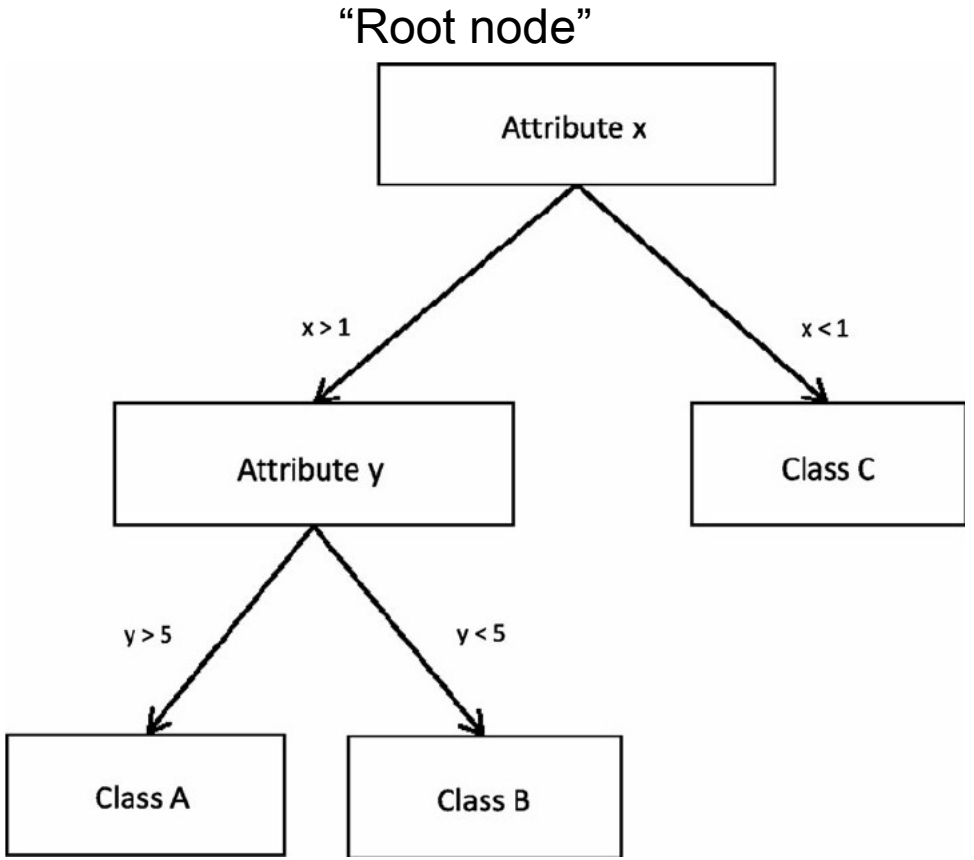
Go through each variable and set a simple decision rule based on the features to predict value of a target variable

- Can predict multiple classes
- May be prone to overfitting (next week!)
- Predictions are not smooth/continuous so they aren't good at extrapolation
- Greedy exhaustive search over all features and possible thresholds to minimize the *criterion* function

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

https://scikit-learn.org/stable/modules/tree.html



Keep splitting until each branch is "pure" (or close to pure)

The gini coefficient for optimization

For two classes with class proportions p_1 and p_2:

$$\text{Gini}(P) = p_1p_2 + p_2p_1 = 2p_1p_2$$

$$\text{Gini}(P) = 1 - \sum_{i=1}^{n}(p_i)^2$$

- A measure of how mixed classes are in a region
- Probability of being correctly assigned * probability of misclassification for each class
- Minimum value = 0 when node is pure
- Maximum value = 0.5 (probability of two classes are the same)



The gini coefficient for optimization

For two classes with class proportions p_1 and p_2:

$$\text{Gini}(P) = p_1p_2 + p_2p_1 = 2p_1p_2$$

$$\text{Gini}(P) = 1 - \sum_{i=1}^{n}(p_i)^2$$

- A measure of how mixed classes are in a region
- Probability of being correctly assigned * probability of misclassification for each class
- Minimum value = 0 when node is pure
- Maximum value = 0.5 (probability of two classes are the same)



Logistic regression vs decision trees

- Logistic regression
  - Smooth linear or curved boundaries
  - Good for small datasets
  - Can use feature engineering to make non-linear features (like circle boundary)
- Decision trees
  - Complex, nonlinear interactions between features
  - No feature engineering needed
  - Easy to interpret
  - Easy to overfit to your data; needs more data to avoid this

Using sci-kit learn

Check out the User Guide on sklearn website!

```
conda activate BME2315
pip install scikit-learn


from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

In class work time on your projects

- Based on your PCA visualizations, your project goal, and your chosen method, **decide on a model to use**
  - You will have a chance to make a better model next week, but feel free to ask for guidance on choosing a model!
- **Fit your model** to your training dataset
- **Predict the target variable** in the **validation** dataset
- Choose some **metrics** to determine if your model is performing well on the validation dataset
  - Visualize them with a plot (ROC curve? Confusion matrix? Something else?)
  - **Report in-sample error** (on training set) and **out-of-sample error** (on validation set) and write out your thought process & **conclusions**: is your model performing well?

3rd check-in: All the red should be completed!