Machine Learning in Education in 3 Hours: Part 1

ISEA Training Program
Weeks 5
Lief Esbenshade

Topics

Session 5

1. What is Machine Learning
    a. The field is moving fast
    b. Supervised, Unsupervised, Reinforcement, Generative
    c. Major Software Packages
2. Regression Models
    a. Gradient Descent Exercise
3. Binary Classification
4. Trees & Forests
    a. Tree Exercise
5. Overfitting, Test/Train Splits
6. Accuracy, Confusion Matrix

Session 6

7. Applied ML: Graduation/Dropout Risk Prediction
8. Model Ensembles
9. Neural Nets

1. What is Machine Learning?

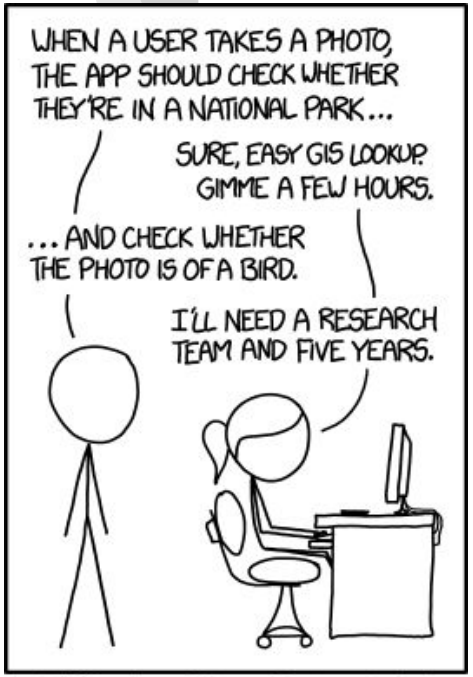
“In basic terms, ML is the process of training a piece of software, called a model, to make **useful predictions or generate content** from data.” - Google Intro to ML Course Accessed Feb 17 2025

"Machine learning is the science of getting computers to **act** without being explicitly programmed." - Andrew Ng (2011)

Resource: Google Glossary of Machine Learning Terms

September 2014

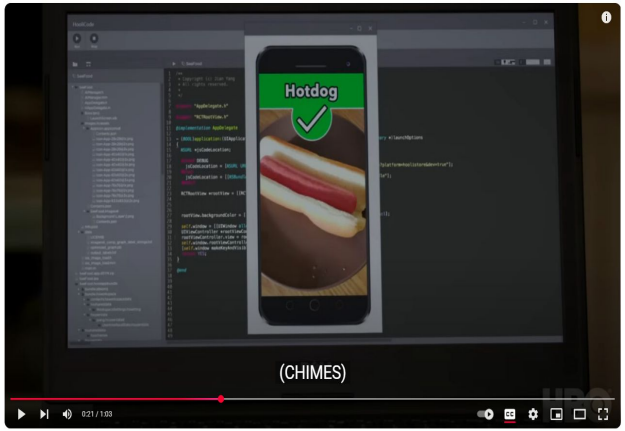
a joke about something we can't do



https://xkcd.com/1425/

3 Years

May 2017 - a joke about the current hot trend



Silicon Valley, S4E4

4 Years

January 2021

past classification, now generation

DALL-E 1

https://openai.com/index/dall-e/

4 Major Types of Machine Learning Models

- Supervised: Models learn from labeled data to make predictions or classifications based on known input-output pairs
  - OLS Regression, logistic regression, random forest, neural networks
  - Predicting student graduation outcomes, Email spam filters
- Unsupervised: Models identify patterns, structures, or groupings in unlabeled data without predefined outcomes.
  - Examples: K-means clustering, Principal Component Analysis (PCA)
  - Grouping students into learning clusters based on performance patterns to personalize instruction, Spotify music recommendations
- Reinforcement: Models learn optimal decision-making by interacting with an environment and receiving rewards or penalties.
  - Deep Q-Networks (DQN), Policy Gradient Methods
  - Developing intelligent tutoring systems that adapt learning paths based on student interactions (Duolingo), Google AlphaGo
- Generative: Models learn to generate new data similar to the training distribution, creating realistic text, images, or audio
  - Generative Adversarial Networks (GANs), Large Language Models (LLMs)
  - Colleague.AI, ChatGPT

Scikit-Learn

- Major open source package in python for Machine Learning
- Consistent API for ML work
- Transform Data: `X_scaled = StandardScaler().fit_transform(X)`
- Fit Models: `DecisionTreeClassifier().fit(X, y)`
- Predict Outcomes: `y_pred = model.predict(X)` P
- Evaluate Models: `acc = accuracy_score(y, y_pred)`
- Tune hyper-parameters:
  - ```
    param_grid = {"max_depth": [1, 2, 3, 4, 5, None]}
    search = GridSearchCV(model, param_grid, cv=5).fit(X_scaled, y)
    ```

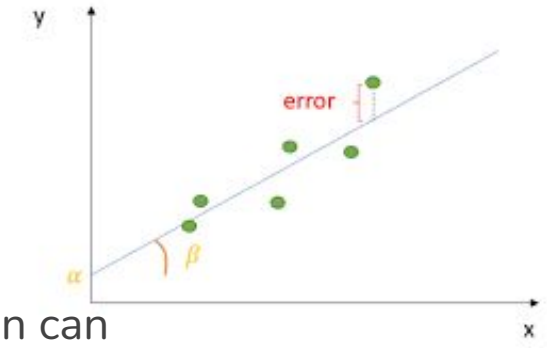
COLAB Example
    ```

Keras & Tensor Flow

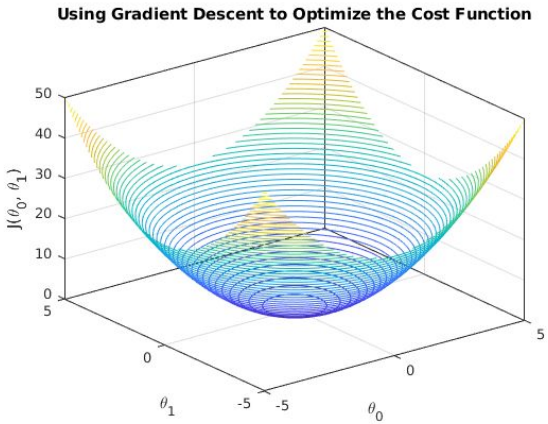
- Used for deep learning with ‘tensors’
- A tensor is an n-dimensional data structure
  - A single value (e.g. 9) is a 0 Dimensional tensor, a vector (e.g. [1, 2, 3, 4]) is 1D tensor, a matrix is 2D. If an image can be represented as a 2D matrix of pixels, how many dimensions is a GIF? What about one with sound?
- Keras started out independent and has been integrated into TensorFlow
- Consistent API for deep learning
- Tensors for x and y
- Layers
  - ```
    model = tf.keras.Sequential([
      tf.keras.layers.Dense(64, activation='relu'),
      tf.keras.layers.Dense(10, activation='softmax') ])
    ```
- Compilation and Optimization
  - ```
    model.compile(
      optimizer='adam',
      loss='sparse_categorical_crossentropy',
      metrics=['accuracy'])
    ```
- Fit, Evaluate, Predict
  - `model.fit(X_train, y_train, epochs=5)`
  - `test_loss, test_acc = model.evaluate(X_test, y_test)`
  - `predictions = model.predict(X_test)`

COLAB Example

OLS Regression and Gradient Descent



- Create a line through a distribution of x, y data points
  - Has a slope and an intercept
- Minimize the sum of the squared error (**“Cost Function”**)
- Coefficients can be calculated directly - but matrix inversion can be computationally expensive
  - Matrix inversion compute cost: $O(n^3)$
- Gradient descent use the slope of the cost function
  - Can be faster with big n: $O(k \times n^2)$ [k = iterations]
- Gradient Descent is a **generalizable optimization Framework**



Key Steps in Gradient Descent

- Goal: Learn parameters for slope and intercept
- Initialization: Start with random initial values for model parameters
- Iterative Updating of Parameters:
  - Compute the gradient of the cost function
  - Update parameters using negative gradient and a learning rate
- Convergence, repeat the process until:
  - The cost function converges to a minimum
  - specified number of iterations is reached

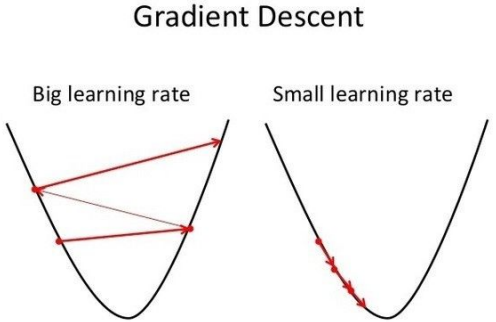
Colab Time

Split into groups after intro for 15 minutes

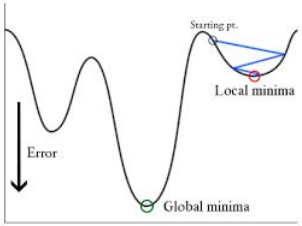
Reconvene at 12:50

5 minute break!

Return at 1:07



More on Gradient Descent



- Choosing the Learning Rate
  - Too small you take forever to converge, too large you can get divergent results
  - Use learning rate schedules (step decay, adaptive methods)
- Convergence Issues and Local Minima
  - Momentum technique, adaptive optimizers (Adam, RMSprop)
- Feature scaling
  - Gradient descent can have issues when features are on very different scales
  - Normalize or standardize your features (minmax, z-score)
- Batch Size
  - Batch (this is what we did) use the whole dataset to compute the gradient
    - Can’t use this if your dataset is enormous and you can’t fit the x’s all in memory
    - Can get stuck in local minima
  - Stochastic Gradient Descent (SGD) updates parameters one randomly selected input row at a time
    - very noisy, works with very large datasets
  - Mini-batch breaks the data set up into multiple groups (‘batches’) and updates parameters one batch at a time

Classification

- OLS, and many other models, create a numeric output, often we want to put data into a category, this is called Classification.
  - Spam / Not Spam, At Risk / Not at Risk
- OLS, Logistic Regression, Decision Trees, Random Forest, Support Vector Machine, Neural Networks are all models used in classification problems.
- “Thresholding” is the term for converting a continuous score into a class by setting a decision threshold
  - A simple version, predict "positive" if score > 0.5
  - You can examine how categories change with different thresholds
  - Can set multiple thresholds (high, medium, low risk)
  - Can vary thresholds for different groups (e.g. for demographic parity)

$$Gini = 1 - \sum p_i^2$$

Trees

- Split data into hierarchical if-then rules based on feature values
- Recursively split data to minimize impurity (e.g. Gini Index)
  - **Don’t use gradient descent**
- Non-linear!
  - Nice trick for missing data, just put a value far outside the normal data range
- Small trees are inherently interpretable
- Very prone to overfitting
- COLAB Example

Overfitting, Test/Train Splits

- In supervised machine learning, we have a set of X features and y outcomes. The goal is **not to predict the y outcomes we have already observed**, but to build a model that can generalize to new X inputs and predict unseen y outcomes. If the new set of X and y were identical to the training data, we wouldn't need ML, a lookup table would do!
- It's easy to forget this and overfit a model, meaning it learns patterns too perfectly in the training data, capturing noise or spurious relationships rather than real trends. This issue is particularly common in decision trees and neural networks but can also happen in regression when using excessive feature interactions or polynomial terms (do you really need x^1 to x^{13}?).
- To avoid overfitting, split data into training and test sets. Train the model using the training set, and evaluate its performance using the test set. The test set should only be used at the end; if you check accuracy on it too early, it essentially becomes part of the training data, leading to biased estimates of model performance.

A paper where I investigate overfitting questions in multi-year modelling scenarios

Random Forests

- Decision trees are prone to overfitting their training data and generalize poorly
- Random forests are ensembles (crudely averages) of many decision trees each trained on a random subset of the data.
- They tend to be much more stable and generalize much better

Measures of Classification Model Fit: Accuracy, Precision, Recall

- Accuracy: Number of correct predictions divided by the total number of cases predicted.
  - (True Positives + True Negatives) / Total Cases
  - Would you buy a model that predicts high school graduation rates with 90% accuracy?
  - 'Accuracy' is not useful alone - beware when that's the only evaluation number given
- Precision: are we over-identifying positive cases?
  - True Positives / (True Positives + False Positives)
- Recall: are we missing positive cases?
  - True Positives / (True Positives + False Negatives)
- Lots of measures that build off the confusion matrix framework!
- Also measures based on continuous scores:
  - Demographic Parity, Calibration, AUC-ROC

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Colab Simulation Comparing Fairness Metrics

Fairness

- Fairness is application specific, there is no one fairness measure to maximize
- What are the consequences of your model's predictions?
  - How will impacts of the model be evaluated? 'Fair' in the historical administrative data isn't the same as 'Fair' in use
    - Paradox: If a model predicts outcome X for user A but not for user B, and users take actions to prevent outcome X from occurring for A, but it then occurs for B instead, should a retrospective analysis conclude that the model was wrong?
- What is the data generating process (DGP) that your model relies on?
- How does the use of your model connect to the DGP?

More Fairness

- Demographic Parity
  - Model's predictions are independent of a sensitive attribute
  - Positive predictions at equal rates across demographic groups
    - Does this make sense in an unequal society?
- Calibration Fairness
  - Predicted risk (p') is equal to probability of outcome (p)
  - Measures quality of absolute risk score
- AUC ROC
  - Probability that a randomly selected dropout has a higher risk score than a randomly selected graduate
  - Measures quality of relative ranking

HW - Student Risk Predictions

- Rule based models like Allensworth & Easton (2009) “Chicago Model” were historically used in identifying students at risk of not graduating high school.
- Mid 2010’s explosion in research in using ML models to predict dropout/persistence.
  - Availability of longitudinal student administrative data
  - Newly accessible ML techniques
  - Lots of adoption and commercialization
- For next session there are several readings about risk prediction systems and you will train two models of your choice.
  - I recommend sticking with sci-kit learn
  - Look at the last section of the decision tree example if you are stuck getting started

Class Risk Prediction Models

- Comparison of accuracy
  - Logistic ~ 0.88, Trees ~0.84, RF ~ 0.89
- Did anyone dig into thresholding, or just use the default?
  - Logistic Regression is calculating a probability not a label!
  - What is RF doing to aggregate the trees?
- Which data to include? Generally speaking, graduation prediction models are excluding continued enrollment
  - Typical data setup is to have features for year N and outcomes from year N+X - so for 9th grade risk prediction you would observe features about 9th graders and collect outcomes from when they were supposed to graduate 4 years later.
  - Contexts where continuing enrollment as an outcome might be important
    - 5th year seniors in High School
    - Community Colleges with highly variable time to degree
- Time generalizability
  - Are the structural factors influencing whether a student graduates or not changing? How exactly is the model working?
  - See Esbenshade, L., Vitale, J., & Baker, R. S. (2024). Non-Overlapping Leave Future Out Validation (NOLFO): Implications for Graduation Prediction. In *Proceedings of the 17th International Conference on Educational Data Mining* (pp. 602-609).
    - Models were way more stable than I expected
    - Relative risk ranking, standardized features are the likely reasons why
- I like normalizing the confusion matrix (display percentages instead of counts)
- Be careful with FP / FN and what you are predicting
  - ‘Negative’ is aligned with the 0 class **NOT** the “bad” outcome
  - In graduation models you want to be clear on if you are predicting graduation or dropout it will flip around your FP/FN interpretations.
- Role of theory in selecting features
  - Applied ML is often aggressively a-theoretical and driven by empirical results - but maximizing fit metrics can get you into trouble when you need to explain and use the model - especially with education stakeholders!

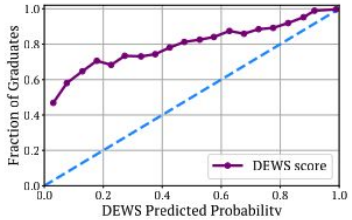
Key points from Baker et al 2023

- What is a ‘demographic’ variable
  - ‘Information that provides context about students’ backgrounds and experiences’
  - Non-mutable by the school (**but** ESL vs ELL, cohort-normed age)
- Demographic variables don’t generally lead to more accurate models
- Color blind racism?
  - Excluding demographic variables could reify the idea that individuals are solely responsible for their negative outcomes
  - OR including race could lead to minimization of factors that schools can change
- Demographics are not actionable
- Reinforcing bias in the training data
  - Early LAK paper: Final course grade *should* be a mechanical composite of assignment grades and participation grades, but a prediction model that included race did better than one without. This is strong empirical evidence that the final grades were biased. Should a prediction model include race to (accurately) predict that students of the disfavored group will get lower final grades?
  - When systems are biased, use demographics to show that the outcomes are biased and consider systematic interventions
- **Creating individual predictions because we want to intervene for the individual**

Key Points from Perdomo et al 2023

- Causal Estimation (Week 13 - quasi-experimental)
  - Uses a Regression Discontinuity Design - 2% to +12% treatment effect
    - This estimate is local to the risk cut off!
  - Measure effect estimate with all schools (intent to treat) or just schools that used it (treatment on the treated)
- Risk prediction is miscalibrated!
  - Dropout risk prediction is actually a sorting mechanism, we only care about **relative** risk, not **absolute** risk.
- “Are individual risk scores necessary for effectively targeting interventions? We propose a simple mechanism that only uses information about students’ environments—such as their communities, schools, and districts—and argue that this mechanism can target interventions just as efficiently as the individual risk score based mechanism.”
  - Community based interventions are easier and cheaper to implement
  - Week 14: economic evaluation
- Is this argument local to Wisconsin?
  - “10% of all state dropouts come from just five schools” (WI has ~514 high schools)
  - Wisconsin is one of the most segregated states - ranked 7th in one 2016 study (https://en.wikipedia.org/wiki/Education_segregation_in_Wisconsin)

Student Risk Prediction



1) Prediction
    a) Can we accurately predict which students are at risk? Yes, high AUC - this is the typically used metric
        i) “Because counselors do not see predicted probabilities, but rather ordered GRAD scores, we chose the area under the receiver operating characteristic curve (AUC) metric that measures whether students’ predictions are ordered in terms of actual risk” - Christie et al 2019
        ii) **BUT** models often miscalibrated - See perdomo fig 1 - **relative** risk vs **absolute** risk
    b) Are the predictions biased? Not very much, generally pretty similar across groups
2) Explainability
    a) Can the predictions be communicated to stakeholders? Yes, various risk buckets
        i) **BUT** explaining why a student is at risk is hard
3) Intervention
    a) Can individuals / schools do something with this data?
        i) BrightBytes tried integrating Risk prediction with Intervention Management tools
    b) Are these systems improving graduation rates?
        i) As of 2023, I know of no studies showing strong effects from graduation risk prediction systems
4) Structural vs Individual
    a) Is this a problem that can be solved at the individual level?
    b) Does making it individual hide structural problems (consider the role of behavior incidents in risk prediction)?

Side note for industry folks - Publish!

- Conferences often have Industry Track submissions, get your work out there!
- Let academics see how people are really using their methods
  - In my opinion, application is now the most important part of machine learning research in education
- Create a public record of what you are doing

Ensembles

- Combine multiple models together
- Can reduce overfitting, improve accuracy, improve generalizability
- BUT:
  - Increases compute cost
  - Makes the model even more difficult to explain
- “Bagging”: train models independently on different random data subsets (Random Forests)
- “Boosting”: train weak models sequentially, models fit to errors of prior model
- “Stacking”: train separate models and create a meta-classifier
- “Voting”: have multiple classifiers predict and take average or majority

Class Activity - Create an Ensemble
15 Minutes

1) In your breakout group take 5 minutes to share your HW models with each other
2) Implement an ensemble of 3 of the models
3) Use this Colab to get started
4) Finish early?
    a) Discuss ML applications in education that you have worked with
    b) Discuss supervised learning applications that you think would be useful
        i) What is the classification problem
        ii) What are the data requirements
        iii) What actions will happen because of the prediction

5 Minute Break

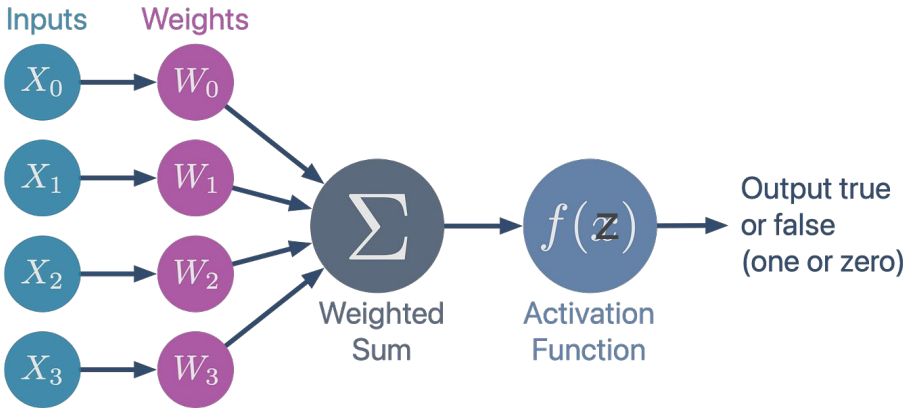
Return at 1:07

Introduction to Neural Networks

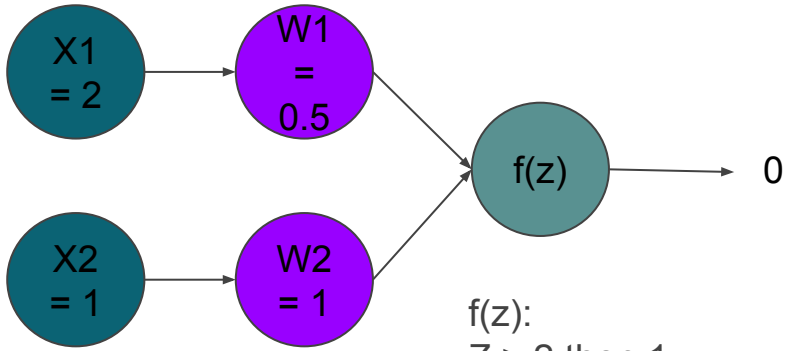
- What is a Neural Network?
  - A machine learning model inspired by the structure and function of the human brain.
  - Composed of layers of interconnected "neurons" that process and learn patterns from data.
- Biological Inspiration
  - The brain consists of neurons that receive, process, and transmit information.
  - Artificial neurons mimic this process using mathematical functions.
- Why Use Neural Networks?
  - Traditional algorithms struggle with complex pattern recognition (e.g., images, speech, natural language).
  - Neural networks excel at detecting patterns, generalizing from data, and making predictions.
- Key Components of a Neural Network
  - Input layer: Receives raw data (e.g., pixel values, words).
  - Hidden layers: Perform computations and extract features.
  - Output layer: Produces the final prediction (e.g., classification, probability).
  - Weights & biases: Parameters adjusted during training to improve accuracy.
  - Activation functions: Introduce non-linearity to model complex relationships.

The Perceptron

- Inputs are multiplied by weights and summed
- Activation function > step function converting sum into binary 0/1 based on a threshold
- Only works for linearly separable problems
- Cannot solve complicated problems - invented in 1958!



Take a minute to solve this:



f(z):
Z > 2 then 1
Z <= 2 then 0

Activation Functions

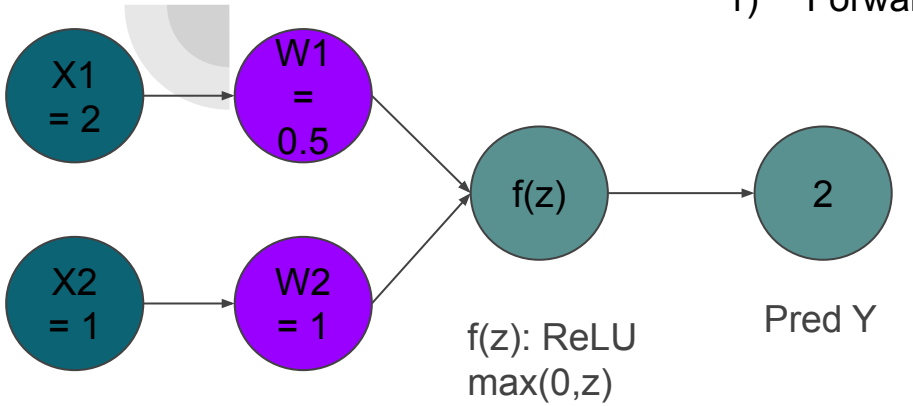
- SO, we have inputs, we multiply them by weights and get a sum. Perceptron with binary output is too simple.
- For neural networks we need to have an activation function
  - Why? If we didn't, we would just have a linear model, it would be just a really big logistic regression
  - Adding activation functions lets us learn really complex, non-linear patterns
- Sigmoid, Tanh are used
- Most common is ReLU $f(z) = \max(0, z)$
  - If the sum of weights in a neuron is less than 0, output 0, otherwise output the sum of weights
- Now we can solve problems that are more complex

Update the neural network with Backpropagation

- Forward pass: Compute outputs (Pred Y)
- Calculate Loss (lets use MSE again)
- Compute Gradients
- ‘Backpropagate’: update the weights
- And repeat

Backpropagation

1) Forward pass: Compute outputs (Pred Y)



$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_i}$$

2) Calculate Loss (lets use half MSE again)

½ (2 - 1)^2 = 0.5

3) Compute Gradients, this is more involved with the final output, the activation function and the weight

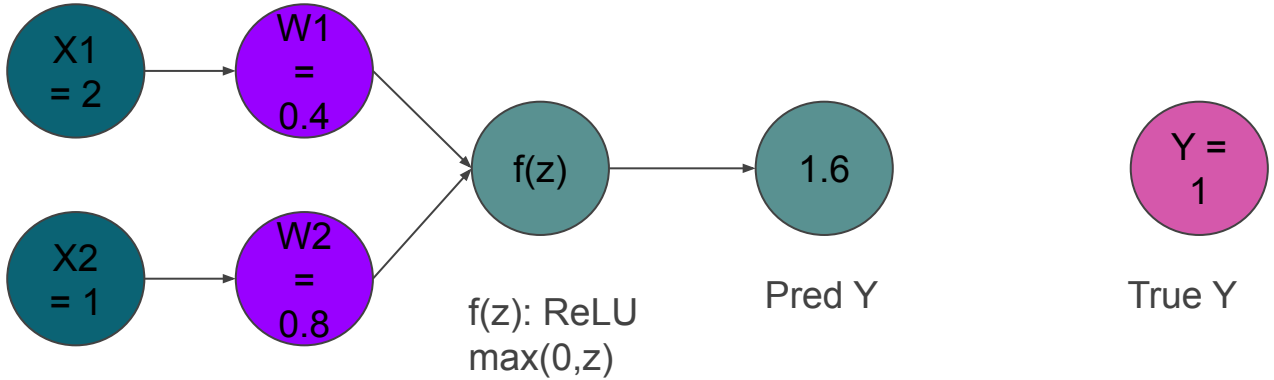
DW1 = -2 * 1 * 0.5 = -1

DW2 = -2 * 1 * 1 = -2

Note: the only difference in the gradient is the dz/dwi component, so with OLS we could initialize with uniform weights but with neural nets we must use random weights

Backpropagation

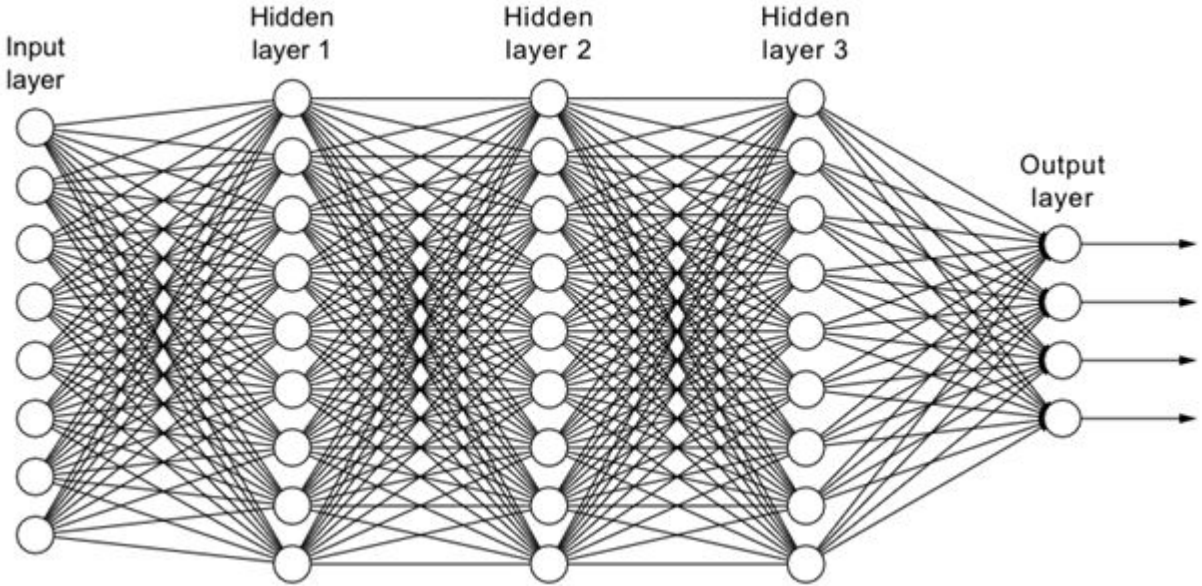
4) Update weights, using gradient and learning rate
W1 = -2 * 1 * 0.5 = -1
W2 = -2 * 1 * 1 = -2
a = 0.1



5) And repeat

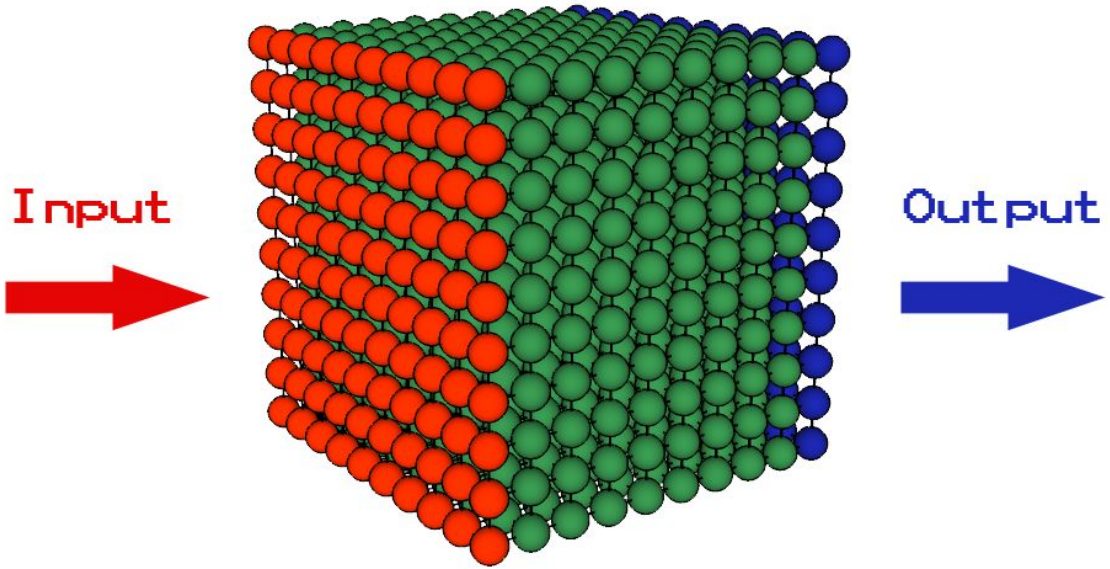
That was a simple network, and so is this

- Many input X variables
- The input layer represents variables, not rows!
- ‘Hidden layer’ are the nodes between the x input and y output
- You can have many output nodes
  - Multi-classification problems
  - Predict multiple outcomes simultaneously
  - Object detection
- Parameters ~= number of connections between nodes
  - 270 parameters in this model



N Dimensional Tensors

- The neural network structure easily generalizes to n dimensions
- More input nodes and connections between them



More advanced Neural Nets

- Convolutional
  - Images, videos represented in grid-like data
  - Moves a sliding window around over the data
- Recurrent
  - Time series, text analysis
  - Processes sequences, predicts next word and maintains a memory state of prior words
- Transformers
  - RNN strongly focused on order, process one at a time with memory state
  - Transformers use ‘Attention’ to process multiple items in parallel

Machine Learning in Education Key Points

- Supervised machine learning methods improved really fast
  - Standardized tools (Sklearn, TensorFlow, PyTorch etc)
  - Open datasets and competitions (Kaggle) to maximize metrics
- Application is the most important issue for education
  - What are you using ML for?
  - What actions are going to come out of your system?
  - Don't be like 'AlphaRoute' that stranded Kentucky students and caused school to be cancelled for a week because they wrecked bus routes.
- Fairness is really important and really hard
- Basic model creation is now pretty easy with LLMs
  - BUT - creating good models is still challenging, don't think that a quick LLM generated random forest is going to be good enough to use on students! Real danger of people making models that they don't understand. Edge cases matter when its real people you are impacting.