Introduction to Modern AI
Week 1: Introduction to Learning

Gavin Hartnett

PRGS, Winter Quarter 2021

Goals of this course

- Find the right balance between *depth* and *breadth*
  - go deep on the foundamental concepts underpinning modern AI
  - give a broad survey of how these concepts facilitate the many modern applications of AI and machine learning across different domains (computer vision, natural language processing, reinforcement learning, etc)
- Find the right balance between *theory* and *practice*
  - understand the key ideas behind different learning algorithms and models
  - be able to implement these in Python

Review of Syllabus

- Class schedule and attendance policy
- Evaluation criteria and homework
- Expected topics to cover
- Heads up: expect some deviations from syllabus
- Google Colab

Final Project

- The final project is a presentation and a short report on an approved topic
- Will account for 40% of your grade
- The project could be one of the following
  - A technical review on an ML method not discussed in class
  - A proposal for an ML approach to a policy problem, along with preliminary analysis
  - A research paper on policy problems associated with ML methods

Teacher and Student Introductions

- Me!
- You!

One Last Thing...

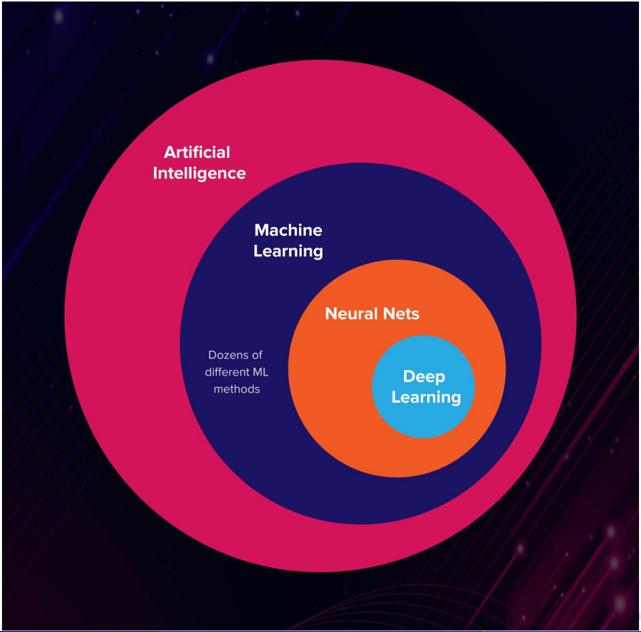
- There will be a lot of math and mathematical notation in this class
- I will try to be consistent in my notation, but I can guarantee that there will be mistakes and typos
- I will also try to be economical in my use of math
- If you are confused about a specific issue, please interrupt me and ask
- If you feeling lost more generally, please come see me outside of class. My challenge in teaching this class is to present the material to a diverse audience (you), and you're feedback as we go will be very valuable

Broad Overview

What is Machine Learning?

- A sub-field of AI where the intelligence is learned, rather than explicitly programmed
- This requires data, and generally speaking the more data the better
- Of course, data quality also matters
- In order to learn complex relationships from data, flexible families of functions are used
- (Artificial) neural networks are the most famous of these
- When the neural networks consist of many layers, the term Deep Learning is often used

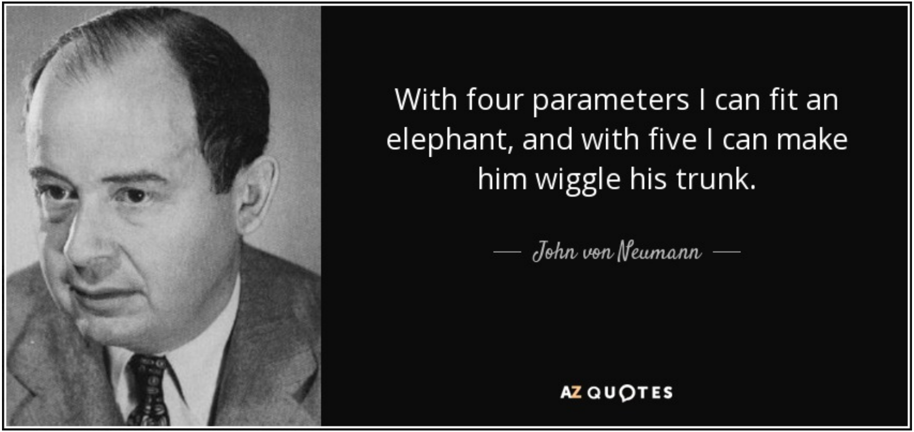
AI Taxonomies



AI Taxonomies

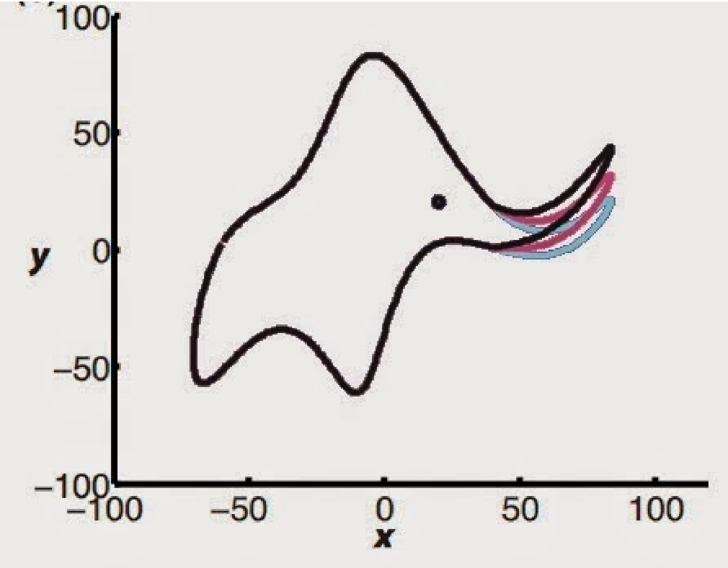
- AI and ML are often used synonymously, but in past years ML was just a rather small subfield of AI
- This class could also be called simply "Introduction to Machine Learning", because I will not cover more traditional approaches to AI
- ML has many overlaps and connections with other fields, especially statistics
- Some folks get quite excited about how to find the boundary where one field ends and another begins
- Is this an important (or helpful) question?

Is ML just curve-fitting?



https://www.azquotes.com/quote/653927

Is ML just curve-fitting?



Mayer et al. American Journal of Physics 78.6 (2010): 648-649.

Is ML just curve-fitting?

- Even if it is, perhaps curve-fitting and function approximation are problems with enough universality to allow us to wondrous things...
- What happens when we build models with hundreds of billions of parameters?
- This is the case for the latest trend in NLP, the so-called Large Language Models
- Example: GPT-2 (1.5B params), GPT-3 (175B params)
- Others argue that a new paradigm is needed that combines symbolic reasoning and the statistical approach of ML

Supervised Learning

- Goal is to learn the input/output relationship between some variables
- The learning is supervised by the known input/output examples
- Dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1,\ldots,N}$
  - $\boldsymbol{x}_i \in \mathbb{R}^p$: independent variables
  - y_i: dependent variables/targets/labels
- Target variables y_i could be real-valued (regression) or categorical (classification)
- Simple examples: linear regression and logistic regression
- Generally assume $y = f(\boldsymbol{x})$, or $y = f(\boldsymbol{x}) + \epsilon$, with ϵ some random variable (noise)
- Real f is unknown, goal is to find a good approximation using the data
- Key question: how should f be modeled?

Unsupervised Learning

- Have unstructured or unlabeled dataset, goal is to learn useful properties of this data
- Compared to supervised learning, there is greater variability among the types of tasks unsupervised algorithms perform
- Examples include:
  - Clustering
  - Dimensionality reduction
  - Visualizations
  - Density estimation
- Warning: sometimes the line separating supervised and unsupervised are fuzzy
- There are also other paradigms such as self-supervised, semi-supervised, etc

Reinforcement Learning

- How should agents behave in an environment so as to maximize their reward/utility?
- Environment could be stochastic (random) as well as changing overtime
- Could also include many other agents
- High-profile examples: Deepmind's AlphaGo, or OpenAI's DOTA bot
- Although RL uses many concepts from supervised/unsupservised ML, in many ways it deserves to be taught separately
- We will devote Week 8 to RL

Linear and Logistic Regression

Linear Regression

- Given: a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1,\ldots,N}$, with $\boldsymbol{x}_i \in \mathbb{R}^d$, and $y \in \mathbb{R}$
- Goal: develop (learn) a model that can predict the output y given a previously unseen input $\boldsymbol{x}$
- Assume a linear relationship:

$$\hat{y} = f(\boldsymbol{x}), \qquad f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$$

- $\hat{y}$ is our estimate or prediction for the "true" y
- $\boldsymbol{w} \in \mathbb{R}^d$ is a weight vector
- $b \in \mathbb{R}$ is the bias
- f is the model
- $(\boldsymbol{w}, b)$ are the model parameters

Linear Regression

- With these assumptions, the problem now amounts to: how to pick the parameters $(\boldsymbol{w}, b)$?
- Ordinary Least Squares (OLS): pick the parameter values that minimize the Mean Squared Error (MSE):

$$\mathrm{MSE}(\boldsymbol{w}, b) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(f(\boldsymbol{x}_i; \boldsymbol{w}, b) - y_i)^2$$

- The MSE is an example of what is known as a loss or objective function (sometimes also cost function)
- It allows us to compare two different models - in general the one with a smaller MSE value is preferred
- The model with the smallest MSE is then the best, or optimal model

Linear Regression

How to find the best model (aka how to fit the model or how to learn the model)

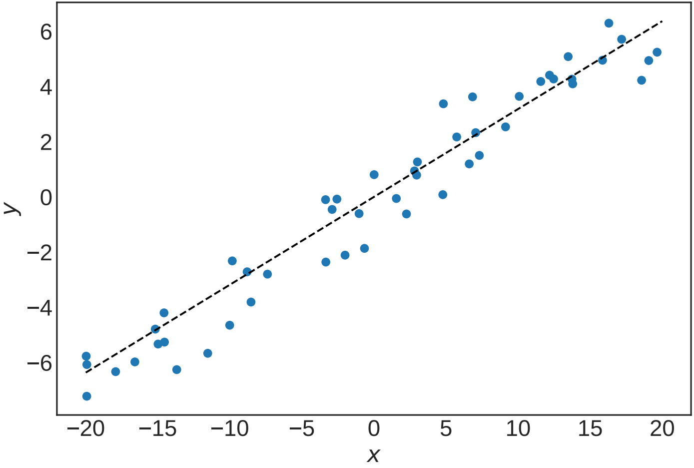
- Want $(\boldsymbol{w}, b)$ that minimize the loss function, call these $(\boldsymbol{w}_*, b_*)$
- The gradient of the loss function will vanish at the minimum
- Though note that the converse is not true, a vanishing gradient does not necessarily imply the minimum is attained
- Vanishing gradient:

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}(\boldsymbol{w}, b)\Big|_{(\boldsymbol{w}^*, b^*)} = 0\,, \qquad \nabla_{b} \mathrm{MSE}(\boldsymbol{w}, b)\Big|_{(\boldsymbol{w}^*, b^*)} = 0$$

Linear Regression

Example:

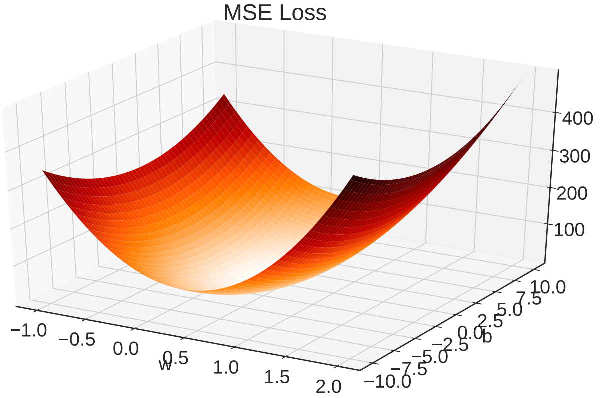
$$y = f_{\text{true}}(x) + \epsilon = x/\pi + \epsilon\,, \qquad \epsilon \sim \mathcal{N}(0,1)\,.$$



Linear Regression

Loss function is quadratic polynomial in $\boldsymbol{w}, b$:

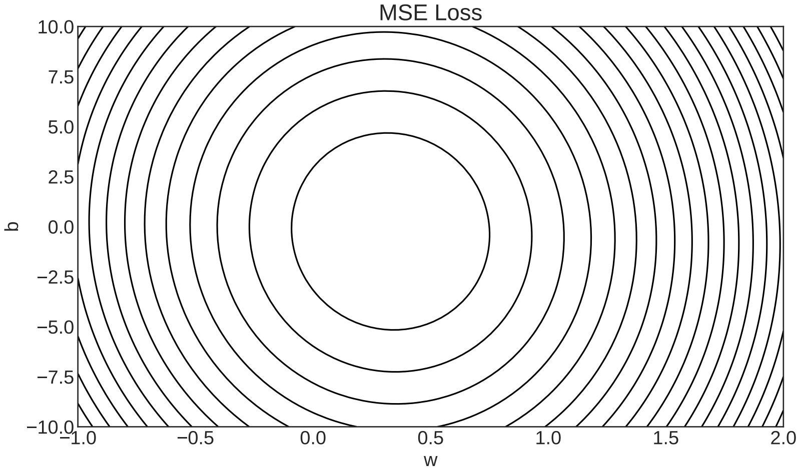
$\text{MSE}(\boldsymbol{w}, b) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i; \boldsymbol{w}, b) - y_i\right)^2$



Linear Regression

Loss function is quadratic polynomial in $\boldsymbol{w}, b$:

$\mathrm{MSE}(\boldsymbol{w}, b) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\boldsymbol{x}_i; \boldsymbol{w}, b) - y_i\right)^2$



Linear Regression

Some nice things about OLS linear regression

- There is a unique optimal solution
- The loss surface is convex - there are no local minima or even saddles
- The optimization problem can be solved exactly (by hand)
- The simplicity of the model (i.e., linearity) means that the parameters are interpretable
- There is a well-developed theory on the variance of the OLS estimators

Linear Regression

Teaser: much of supervised learning corresponds to promoting f to a much more complicated function

- The optimization problem is much harder now - but thankfully there are great software libraries and specialized computer hardware for solving it
- Formal guarantees go out the window
- Nevertheless, things tend to work well in practice

Logistic Regression (binary classification)

- Given: a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1,\ldots,N}$, with $\boldsymbol{x}_i \in \mathbb{R}^d$, and $y \in \{0, 1\}$ a *binary* variable
- Goal: develop (learn) a model that can predict the output y given a previously unseen input $\boldsymbol{x}$
- Because y is binary, it will be easier to model the probability $P(y|\boldsymbol{x})$
- For a given $\boldsymbol{x}$, the distribution is fully characterized in terms of a single number, say $p(y = 1|\boldsymbol{x})$, since

$$p(y = 0|\boldsymbol{x}) + p(y = 1|\boldsymbol{x}) = 1$$

- As before, we will assume a linear relationship (this time for the log-odds):

$$\ln\left(\frac{p(y = 1|\boldsymbol{x})}{1 - p(y = 1|\boldsymbol{x})}\right) = f(\boldsymbol{x})\,, \qquad f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$$

Logistic Regression (binary classification)

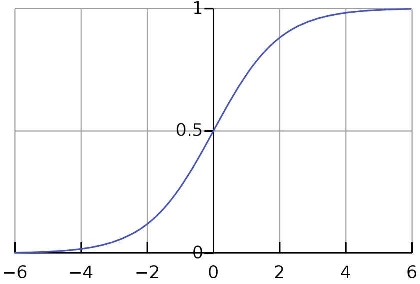
- Solve for the probability:

$$\ln\left(\frac{p(y=1|\boldsymbol{x})}{1-p(y=1|\boldsymbol{x})}\right) = f(\boldsymbol{x})$$

$$p(y=1|\boldsymbol{x}) = \frac{1}{1+e^{-f(\boldsymbol{x})}}$$

- This function is important enough to name: *logistic function*:

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



Logistic Regression (binary classification)

- With these assumptions, the problem now amounts to: how to pick the parameters $(\boldsymbol{w}, b)$? What is a good loss function?
- Answer: the Binary Cross Entropy (BCE) loss function

$$\mathrm{BCE}(\boldsymbol{w}, b) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \ln p(y=1|\boldsymbol{x}_i) + (1-y_i)\ln(1-p(y=1|\boldsymbol{x}_i))\right)$$

- The optimal parameters are found just as before - set the gradient to zero

$$\nabla_{\boldsymbol{w}}\mathrm{MSE}(\boldsymbol{w}, b)\Big|_{(\boldsymbol{w}^*, b^*)} = 0\,, \qquad \nabla_{b}\mathrm{MSE}(\boldsymbol{w}, b)\Big|_{(\boldsymbol{w}^*, b^*)} = 0$$

- Now the problem is more complicated, can't find solution analytically (closed-form)
- Thankfully, unique solution is guaranteed though
- Q: Why is this approach for classification called logistic *regression*?

Logistic Regression (K-ary classification)

- Can this approach be modified to handle the multi-class problem?
- Let $K > 2$ be the number of classes, $y \in \{0, 1, ..., K-1\}$
- Note: these numbers just index the label ($y = 2$ is not twice $y = 1$)
- Use a log-linear model for each class probability

$$p(y|\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_k^T \boldsymbol{x} + b_k)}{\sum_{k'=0}^{K-1} \exp(\boldsymbol{w}_{k'}^T \boldsymbol{x} + b_{k'})} = \text{softmax}\left(\boldsymbol{w}_k^T \boldsymbol{x} + b_k\right)$$
$$= \text{softmax}\left(\boldsymbol{w}_k^T \boldsymbol{x} + b_k\right)$$

- softmax is the multi-class extension of the logistic function
- converts unconstrained outputs to outputs that may be interpreted as probabilities (positive, sum to 1)
- loss function:

$$\text{loss}(\boldsymbol{w}, b) = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=0}^{K-1} \delta_{y_i,k} \ln p(y = k|\boldsymbol{x})$$

Recap

- In supervised learning the goal is to model the input ($\boldsymbol{x}$)/output (y) relationship by learning from a dataset of examples
- Real-valued outputs: regression
- Categorical outputs: classification
- Simplest regression example: OLS linear regression
- Simplest classification example: logistic regression (binary classification)
- Although different, both linear and logistic regression follow the same pattern:
  - Dataset $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1,\ldots,N}$
  - A linear model $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$
  - A loss function
  - Parameters are fit by minimizing the loss function

Theoretical Motivation

Empirical Risk Minimization

- We would like to have a better motivation for the loss functions we have chosen
- Empirical Risk Minimization is a broad theoretical framework which addresses this
- Set-up:
  - We have random variables x, y with joint probability $p(x, y)$
  - Want to learn a function from x to y: $\hat{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$
  - Have a per-example loss function $L(\hat{y}, y)$
- Want to minimize the loss averaged over the joint distribution:

$$\text{loss}(\boldsymbol{\theta}) = \int \mathrm{d}x \, \mathrm{d}y \, p(x, y) L(f(\boldsymbol{x}; \boldsymbol{\theta}), y)$$

- (this is sometimes called the risk in statistical learning theory)

Empirical Risk Minimization

$$\mathrm{loss}(\boldsymbol{\theta}) = \int \mathrm{d}x\,\mathrm{d}y\; p(x,y) L(f(\boldsymbol{x};\boldsymbol{\theta}), y)$$

- Issue $\#1$: rarely have access to $p(x, y)$, instead have a sample (dataset)
- Solution: minimize the *empirical* risk (subscript is often dropped):

$$\mathrm{loss}_{\mathrm{empirical}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\boldsymbol{x}_i;\boldsymbol{\theta}), y_i)\,, \quad \mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$$

- Careful: if data is high-dimensional and model has high capacity, we could overfit

Empirical Risk Minimization

$$\text{loss}(\boldsymbol{\theta}) = \int \mathrm{d}x\, \mathrm{d}y\, p(x, y) L(f(\boldsymbol{x}; \boldsymbol{\theta}), y)$$

- Issue #2: Often the loss function we directly care about is not amenable to optimization
- Example: classification
  - We typically care about the accuracy, not something called the BCE
  - Accuracy is the average of the 0-1 loss:

$$L(\hat{y}, y) = \begin{cases} 0 & \hat{y} = y \\ 1 & \hat{y} \neq y \end{cases}$$

  - But this is not differentiable. Also, no learning signal (all or nothing).
- Solution: introduce a *surrogate loss function* such as BCE
- Should be well-motivated, serves as useful proxy for evaluation metric we actually care about

Maximum Likelihood Estimation (MLE)

- We would like to have a better motivation for the loss functions we have chosen
- MLE: select the model parameters that maximize the likelihood of the observed data, assuming that the data was generated from the model
- For flexible, expressive models, this is a good approach
- Doesn't work well if the model is biased, unable to faithfully represent the data *for any parameters*
- Introduce likelihood function $\mathcal{L}(\theta)$ (log-likelihood $\ell(\theta) = \ln \mathcal{L}(\theta)$)
- MLE estimate: $\hat{\theta} = \operatorname{argmax}_\theta \ell(\theta)$

Maximum Likelihood Estimation (MLE)

Example: linear regression

- We are modeling $\hat{y} = f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^T \boldsymbol{x} + b$
- Assume true relationship is $y = f(\boldsymbol{x}; \boldsymbol{w}, b) + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- Then y is also Gaussian (normal) but with a non-zero mean:

$$p(y|\boldsymbol{x}) = \frac{\exp\left(-\frac{(f(\boldsymbol{x};\boldsymbol{w},b)-y)^2}{2\sigma^2}\right)}{\sqrt{2\pi\sigma^2}}$$

- This is for a single (x, y) pair. For the dataset $\mathcal{D}$ of observations,

$$\mathcal{L}(\boldsymbol{w}, b) = \prod_i p(y_i|\boldsymbol{x}_i)$$

and so

$$\ell(\boldsymbol{w}, b) = -\sum_i \frac{(f(\boldsymbol{x}; \boldsymbol{w}, b) - y)^2}{2\sigma^2} + \text{const}$$

- Conclusion: maximizing $\mathcal{L}$ (or ℓ) is equivalent to minimizing MSE!

Maximum Likelihood Estimation (MLE)

- Another way to interpret MLE is that it is minimizing the difference between the empirical distribution and the model distribution
- Empirical distribution: typically unobserved distribution that gave rise to data
- Model distribution: how $(\boldsymbol{x}, y)$ would be distributed if our model were true (may require an assumption on how errors are distributed)
- Need notion of distance between two distributions - this is provided by the Kullbeck-Liebler divergence (later lecture)
- For now, it is enough to simply state that maximizing the likelihood minimizes the KL divergence

The Ingredients of an ML Algorithm

Preliminary Stuff

- Put the problem into a form addressable via one of the many ML approaches (regression, classification, etc)
- Preprocess (wrangle) the data

Machine Learning

- Select a model (the function, $f(\boldsymbol{x}; \boldsymbol{\theta})$)
- Select a loss function $\text{loss}(\boldsymbol{\theta})$
- Select a learning algorithm (how to solve $\nabla_{\boldsymbol{\theta}} \text{loss}(\boldsymbol{\theta}) = 0$)
- Reflect on what learning principle is being implemented. What is the model actually learning?