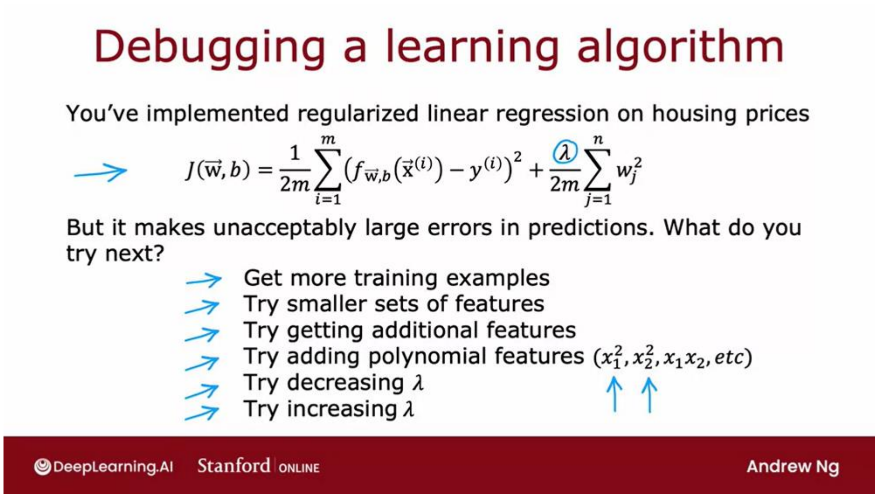
Week 3

Advice For Applying Machine Learning

Next week: Decision trees.

***You got tools now but how do you use these tools effectively?** You need to make good decisions on what to do next in a project.

Example:



Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

$$J(\vec{w}, b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

But it makes unacceptably large errors in predictions. What do you try next?

- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features $(x_1^2, x_2^2, x_1x_2, etc)$
- Try decreasing λ
- Try increasing λ

In a situation like this, there are multiple actions you can take. But while some of them are fruitful, some of them are.

Difference is knowing which is which, along with when do to them.

****Diagnostic:** A test run to gain insight about what is/isn't working with the algorithm. They can time to implement but doing so can be a very good use of your time.

Evaluating a Model

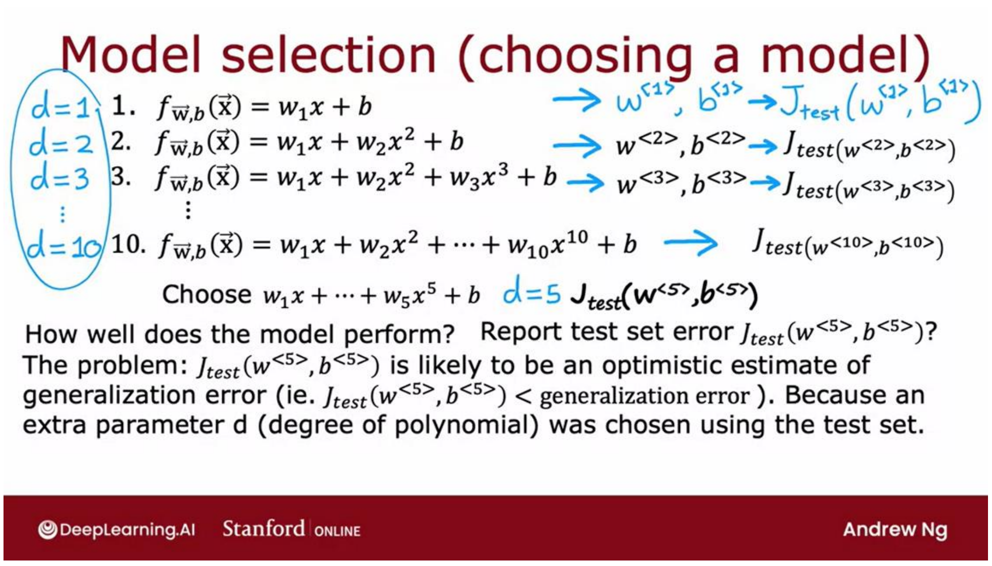
If your model depends on 1 variable, you can easily plot it to detect anomalies, but what if you have 4 independent variables? → training/cross validation/test splits

*Test error computation doesn't include regularization term. Cost function we are aiming to minimize in the training does. (training error does not include reg term too)

Evaluation is made not with the value of cost, but the amount of samples misclassified.

Next: How to choose a model?

Theoretically, you could try this: **WITH AN IMPORTANT, MUST READ NOTE**



Model selection (choosing a model)

$d=1$ 1. $f_{\vec{w},b}(\vec{x}) = w_1x + b \rightarrow w^{<1>}, b^{<1>} \rightarrow J_{test}(w^{<1>}, b^{<1>})$

$d=2$ 2. $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + b \rightarrow w^{<2>}, b^{<2>} \rightarrow J_{test}(w^{<2>}, b^{<2>})$

$d=3$ 3. $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + w_3x^3 + b \rightarrow w^{<3>}, b^{<3>} \rightarrow J_{test}(w^{<3>}, b^{<3>})$

$\vdots$

$d=10$ 10. $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + \cdots + w_{10}x^{10} + b \rightarrow J_{test}(w^{<10>}, b^{<10>})$

Choose $w_1x + \cdots + w_5x^5 + b$ $d=5$ $J_{test}(w^{<5>}, b^{<5>})$

How well does the model perform? Report test set error $J_{test}(w^{<5>}, b^{<5>})$?
The problem: $J_{test}(w^{<5>}, b^{<5>})$ is likely to be an optimistic estimate of generalization error (ie. $J_{test}(w^{<5>}, b^{<5>}) <$ generalization error). Because an extra parameter d (degree of polynomial) was chosen using the test set.

DeepLearning.AI Stanford ONLINE Andrew Ng

FLAW OF THIS APPROACH:

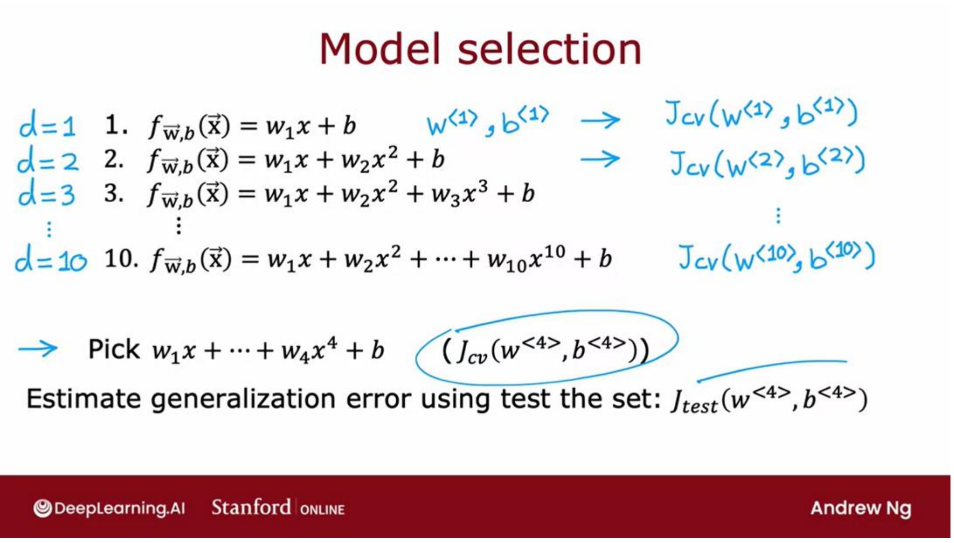
We said that J_train was less than J_test becase parameters were optimized to make cost lower in the training set, not general accuracy. This approach uses a new parameter d, to optimize in (and only in) the test set. It doesn't necessarily mean it improves general quality. SO THIS APPROACH IS FLAWED.

So how to choose?

Split the data into 3, instead of 2: training, **cross validation (or validation or development)**, test.

Now we have 3 J's, each for a different subset (their formula are the same). None of them include the reg. term.

After that, do the procedure above again, but this time choose the model depending on J_cv.



Model selection

$d=1$ 1. $f_{\vec{w},b}(\vec{x}) = w_1x + b$ $w^{<1>}, b^{<1>} \rightarrow J_{cv}(w^{<1>}, b^{<1>})$

$d=2$ 2. $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + b$ $\rightarrow J_{cv}(w^{<2>}, b^{<2>})$

$d=3$ 3. $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + w_3x^3 + b$

$\vdots$

$d=10$ 10. $f_{\vec{w},b}(\vec{x}) = w_1x + w_2x^2 + \cdots + w_{10}x^{10} + b$ $J_{cv}(w^{<10>}, b^{<10>})$

$\rightarrow$ Pick $w_1x + \cdots + w_4x^4 + b$ $(J_{cv}(w^{<4>}, b^{<4>}))$
Estimate generalization error using test the set: $J_{test}(w^{<4>}, b^{<4>})$

DeepLearning.AI Stanford ONLINE Andrew Ng

Training set: Optimize parameters w and b.

Dev set: Optimize d.

Test set: We didn't use test set to optimize anything.

So we can use it to simulate a real world situation and test generalization error.

IMPORTANT NOTES FROM LABS

You can then compute the MSE for the cross validation set with basically the same equation:

$$J_{cv}(\vec{w}, b) = \frac{1}{2m_{cv}}\left[\sum_{i=1}^{m_{cv}}(f_{\vec{w},b}(\vec{x}_{cv}^{(i)}) - y_{cv}^{(i)})^2\right]$$

As with the training set, you will also want to scale the cross validation set. An *important* thing to note when using the z-score is you have to use the mean and standard deviation of the **training set** when scaling the cross validation set. This is to ensure that your input features are transformed as expected by the model. One way to gain intuition is with this scenario:

- Say that your training set has an input feature equal to `500` which is scaled down to `0.5` using the z-score.
- After training, your model is able to accurately map this scaled input `x=0.5` to the target output `y=300`.
- Now let's say that you deployed this model and one of your users fed it a sample equal to `500`.
- If you get this input sample's z-score using any other values of the mean and standard deviation, then it might not be scaled to `0.5` and your model will most likely make a wrong prediction (i.e. not equal to `y=300`).

You will scale the cross validation set below by using the same `StandardScaler` you used earlier but only calling its `transform()` method instead of `fit_transform()`.

```python
# Scale the cross validation set using the mean and standard deviation of the training set
X_cv_scaled = scaler_linear.transform(x_cv)

print(f"Mean used to scale the CV set: {scaler_linear.mean_.squeeze():.2f}")
print(f"Standard deviation used to scale the CV set: {scaler_linear.scale_.squeeze():.2f}")

# Feed the scaled cross validation set
yhat = linear_model.predict(X_cv_scaled)

# Use scikit-learn's utility function and divide by 2
print(f"Cross validation MSE: {mean_squared_error(y_cv, yhat) / 2}")
```

- When scaling, scale train and dev sets WITH TRAINING SET'S MEAN AND STD DEV. Because the model expects it to be scaled that way.

Adding Polynomial Features

From the graphs earlier, you may have noticed that the target `y` rises more sharply at smaller values of `x` compared to higher ones. A straight line might not be the best choice because the target `y` seems to flatten out as `x` increases. Now that you have these values of the training and cross validation MSE from the linear model, you can try adding polynomial features to see if you can get a better performance. The code will mostly be the same but with a few extra preprocessing steps. Let's see that below.

Create the additional features

First, you will generate the polynomial features from your training set. The code below demonstrates how to do this using the `PolynomialFeatures` class. It will create a new input feature which has the squared values of the input `x` (i.e. degree=2).

```python
# Instantiate the class to make polynomial features
poly = PolynomialFeatures(degree=2, include_bias=False)

# Compute the number of features and transform the training set
X_train_mapped = poly.fit_transform(x_train)
```

- Here, way to add features is demonstrated. We utilized **"PolinomialFeatures" from sklearn.**
- https://www.coursera.org/learn/advanced-learning-algorithms/ungradedLab/7aoYQ/optional-lab-model-evaluation-and-selection/lab?path=%2Fnotebooks%2FC2W3_Lab_01_Model_Evaluation_and_Selection.ipynb
This lab shows how to compare different models using a for loop. Go check it later, you'll need it for sure.

*****THIS SAME IDEA CAN BE USED FOR OTHER THINGS LIKE NEURAL NETWORK ARCHITECTURE.**

This is a very widely used procedure.

BIAS AND VARIANCE

- "Typical workflow is: you have an idea and train a model but almost always it doesnt work that well the first time."
- Key is how to improve this performance

Looking at bias and variance gives you direction.

We can't visualize when working with more than x's. So we examine performance on train and dev sets.

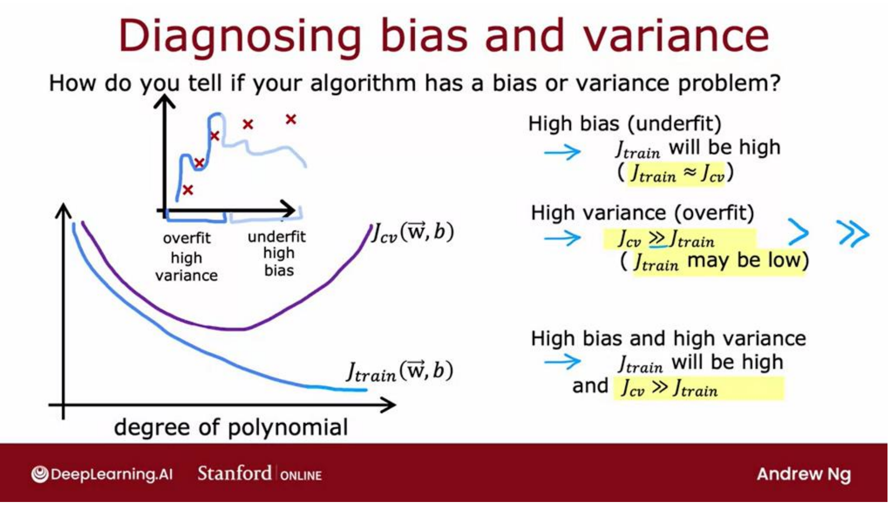
Underfit (**high bias**) → J_train is high, J_cv is high

Overfit (**low bias**) → J_train is low, J_cv is high

Just right → They are both low.

What changes between those situations: parameter d. "d" means order of the polinomial we are trying to fit.

Plot of d – loss:



Diagnosing bias and variance

How do you tell if your algorithm has a bias or variance problem?

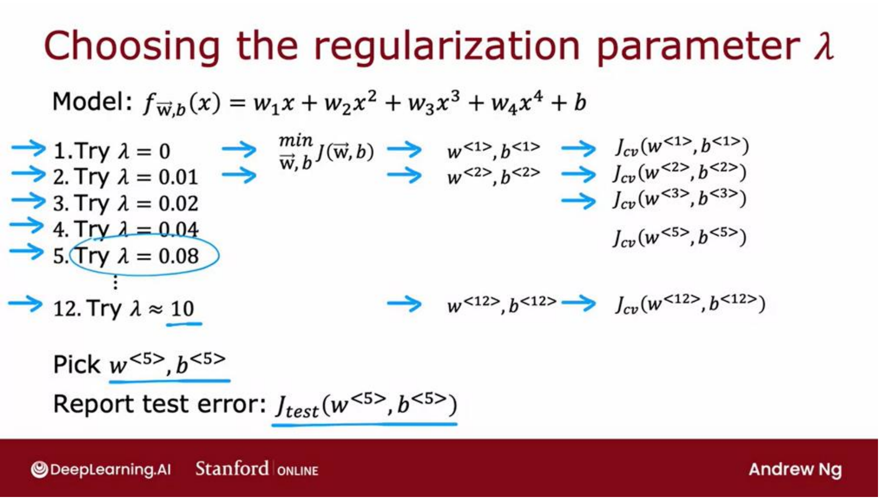
High bias (underfit)
→ J_{train} will be high
($J_{train} \approx J_{cv}$)

High variance (overfit)
→ $J_{cv} \gg J_{train}$
(J_{train} may be low)

High bias and high variance
→ J_{train} will be high
and $J_{cv} \gg J_{train}$

Last situation is shown in the graph on top. You won't see it a lot, especiall not with linear regression.

Effect of regularization and when to use it (together with ***lambda*** choice)



Choosing the regularization parameter λ

Model: $f_{\vec{w},b}(x) = w_1x + w_2x^2 + w_3x^3 + w_4x^4 + b$

1. Try $\lambda = 0$ → $\min_{\vec{w},b} J(\vec{w},b)$ → $w^{<1>}, b^{<1>}$ → $J_{cv}(w^{<1>}, b^{<1>})$
2. Try $\lambda = 0.01$ → $w^{<2>}, b^{<2>}$ → $J_{cv}(w^{<2>}, b^{<2>})$
3. Try $\lambda = 0.02$ → $J_{cv}(w^{<3>}, b^{<3>})$
4. Try $\lambda = 0.04$
5. Try $\lambda = 0.08$ → $J_{cv}(w^{<5>}, b^{<5>})$

⋮

12. Try $\lambda \approx 10$ → $w^{<12>}, b^{<12>}$ → $J_{cv}(w^{<12>}, b^{<12>})$

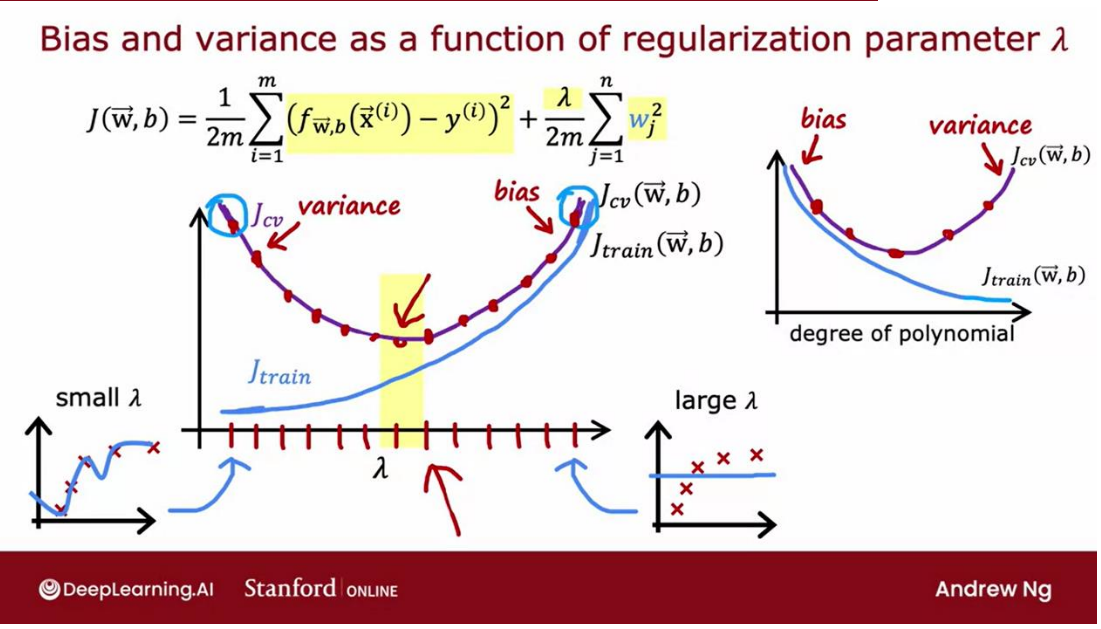
Pick $w^{<5>}, b^{<5>}$

Report test error: $J_{test}(w^{<5>}, b^{<5>})$

DeepLearning.AI Stanford ONLINE Andrew Ng

If lambda is too large, you would fit a "y = c" line, which indicates high bias (underfitting). If lambda is 0, this means no regularization and this causes overfitting, high variance. There is an intermediate value. Cross validation gives you a way to choose.

Procedure is same as choosing "d". Every choice of lambda gives you a set of w and b, and those parameters result in a J_cv.



Bias and variance as a function of regularization parameter λ

$$J(\vec{w},b) = \frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{j=1}^{n} w_j^2$$

As lambda increases, J_train will tend to increase. J_cv will start high, it'll get smaller untill it hits the bottom. After an intermediate value of lambda, it starts to get bigger.

What is high and what is low? How should i judge while looking at those numbers.

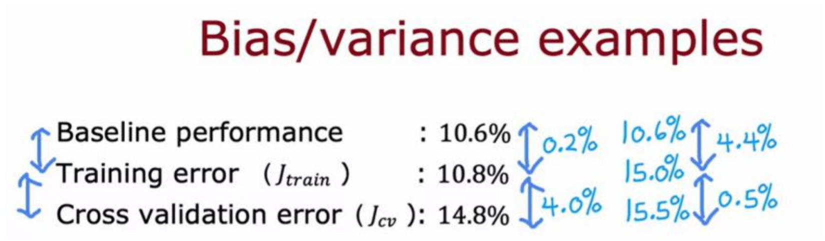
If human level performance has %10.6 error, training error of %10.8 is good. If J_cv is %14.8, then there is a big gap in dev set. It might have a variance problem rather than bias problems (Even though at first look, %10.8 error feels too much and makes you wonder if it's about bias.)

Solution: Establish baseline of performance

What is a good performance for this task? Examine the following:

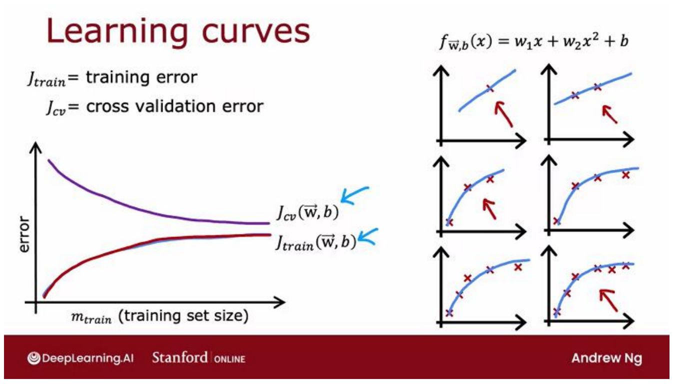
- Human level performance
- Performance of other algorithms
- Guess based on experience

Then look at the gap between baseline performance and J_train. Then take o look at the gap between J_train and J_cv. First indicates high bias, and second high variance.



Bias/variance examples

Baseline performance	: 10.6%		10.6%
		0.2%		4.4%
Training error (J_{train})	: 10.8%		15.0%
		4.0%		0.5%
Cross validation error (J_{cv})	: 14.8%		15.5%



Learning Curves

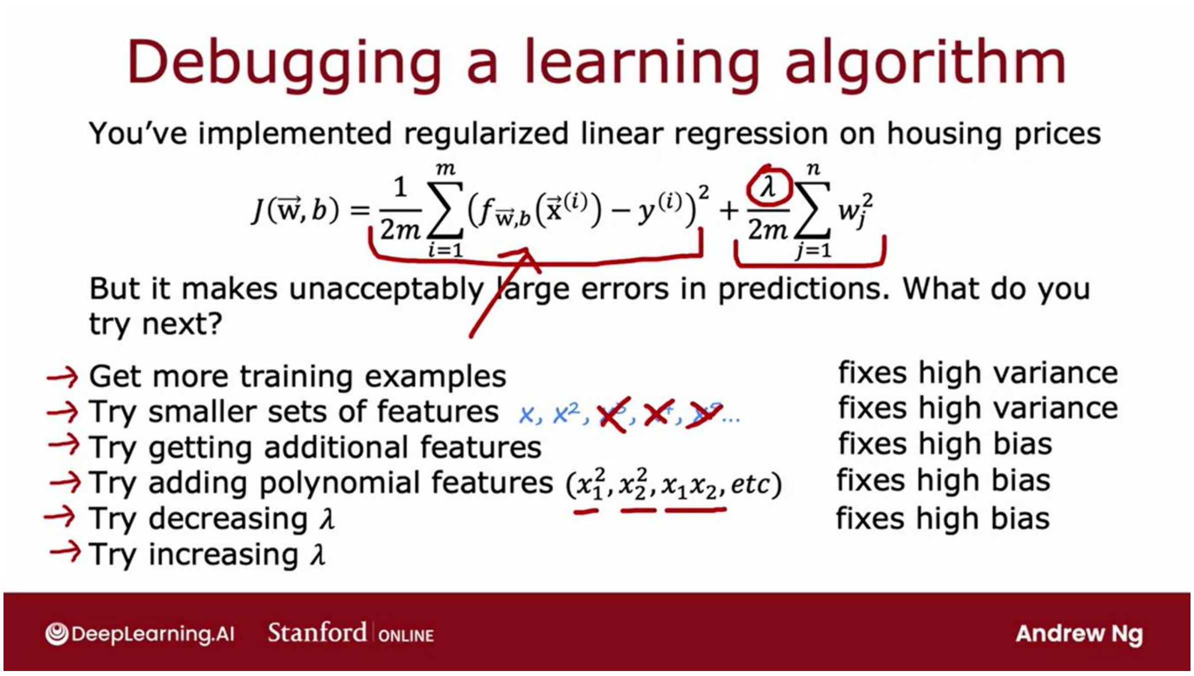
How is the model performing depending on it's experience.

As number of train examples increase, it gets harder to model all of them. So as m increases, J_train increases as well but J_cv will decrease.

Note that J_cv is always higher. (remember optimistic estimation thingie)

- In high bias scenario: As m increases, J_train will increase till it flattens. "It's just a line, what else can it do?" There will be a large gap between desired performance and J_train.
  So if you have high bias, more data won't change much.
  An example that shows throwing more data is not always the solution.
- In high variance scenario: As m increaases J_train increases and J_cv decreases. These two values being too distant, refers to overfitting. As m increases these values kind of meet in the middle, where desired performance lies.
  So if you hace high variance, more data helps a lot.
- If your dataset includes 1000 samples, you could use 100/200 to plot J_train and J_cv with different models. This would guide you to detect high bias and variance, and avoid them.
  One downside, computationally expensive to train so much with different subsets. Not done that often but important to think sometimes.

This procedure above gives you a good way to decide what to do next.



Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

$$J(\vec{w},b) = \underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)}\right)^2}_{} + \underbrace{\frac{\boxed{\lambda}}{2m}\sum_{j=1}^{n} w_j^2}_{}$$

But it makes unacceptably large errors in predictions. What do you try next?

→ Get more training examples	fixes high variance
→ Try smaller sets of features $x, x^2, \cancel{x^3}, \cancel{x^4}, \cancel{x^5} \ldots$	fixes high variance
→ Try getting additional features	fixes high bias
→ Try adding polynomial features $(\underline{x_1^2}, \underline{x_2^2}, \underline{x_1 x_2}, etc)$	fixes high bias
→ Try decreasing λ	fixes high bias
→ Try increasing λ	

DeepLearning.AI Stanford ONLINE Andrew Ng

PRACTICE MAKES PERFECT

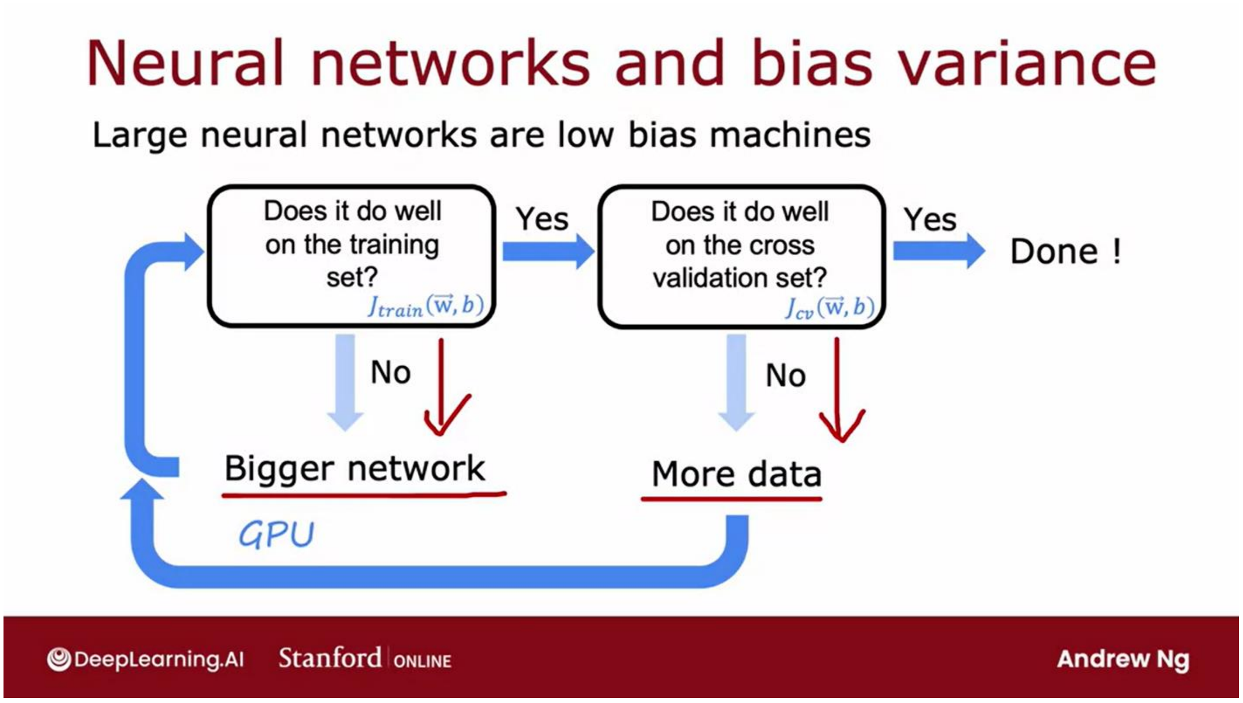
Bias/Variance and Neural Networks

NN's are so powerfull because they, together with having big data, have given us ways to address bias and variance.

Before NN's, people needed to find the optimum point in bias-variance (simple model – complex model) trade-off. Now, we don't.

Large neural networks are low bias machines. If it's large enough, you can fit the data. (unless it's enormous) Don't need to trade-off.

- First check bias: If high, increase size of the network, meaning more layers or more units.
- Then check variance: If high, get more data and retun to first objective.
- **So this approach requires computational capacity. That's why rise of deep learning was tied to improvements in hardware accelerators like GPU.**


Neural networks and bias variance
Large neural networks are low bias machines
Does it do well on the training set?
$J_{train}(\vec{w}, b)$
Yes
Does it do well on the cross validation set?
$J_{cv}(\vec{w}, b)$
Yes
Done !
No
Bigger network
GPU
No
More data
DeepLearning.AI
Stanford ONLINE
Andrew Ng

- As long as regularization is chosen appropriately, using large neural networks do as well as or better than small ones, without risk of overfitting.
- For neural network regularization, just add “kernel_regularization = L2(*lambda*)” parameter to that layer while initializing.

- **IT HARDLY EVER HURTS IF YOU ARE REGULARIZING CORRECTLY, ALL IT MIGHT DO IS TO SLOW YOUR PROGRESS. THEY ARE LOW BIAS MACHINES.**
- **“**With this, i more often find myself dealing with variance problems rather than bias.”

Machine Learning Development Process

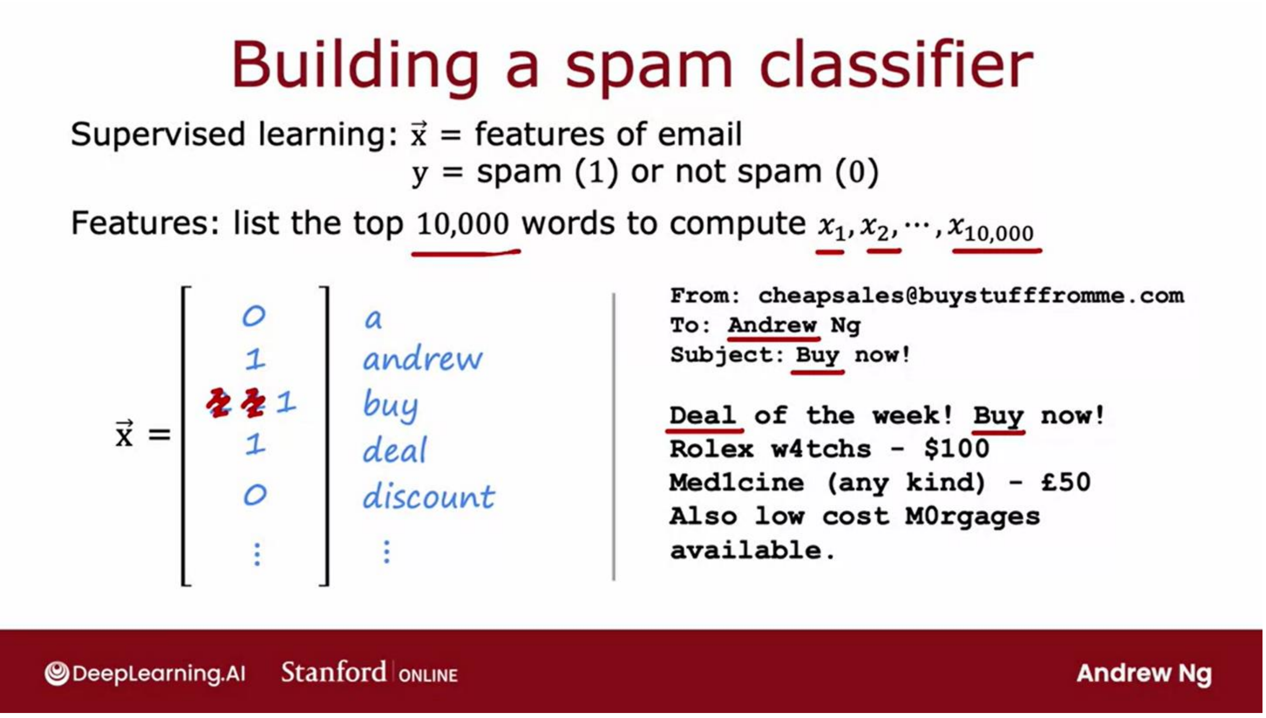
What is it like to live the process? → Need to make good choices.

There is an iterative loop, like the following:

1. **Choose architecture:** Model selection, type of data, hyperparameters…
2. **Train model:** First training will almost always not give you what you really want
3. **Diagnostics:** Do bias, variance and error analysis. **Then return to 1.**

Example: Spam email classifier → That is a text classification task

- Suggested one hot encoding. Also can do: use number of occurences of that word.



Building a spam classifier

Supervised learning: $\vec{x}$ = features of email
y = spam (1) or not spam (0)

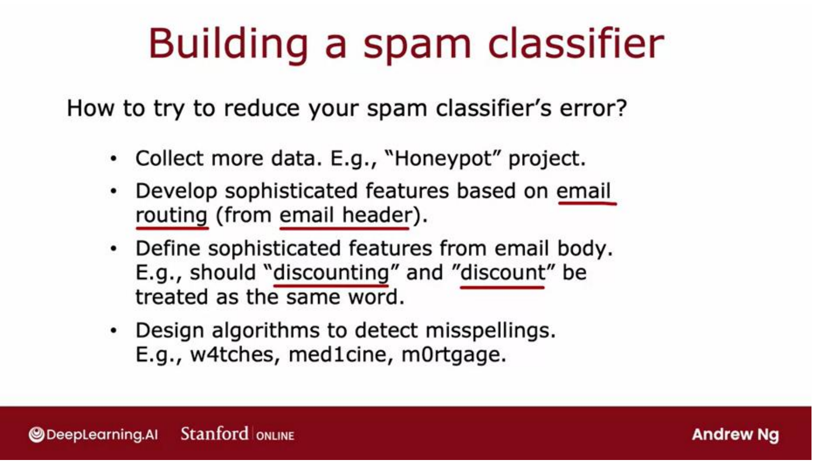
Features: list the top 10,000 words to compute $x_1, x_2, \cdots, x_{10,000}$

$$\vec{x} = \begin{bmatrix} 0 \\ 1 \\ \cancel{2}\cancel{2}1 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \begin{array}{l} a \\ andrew \\ buy \\ deal \\ discount \\ \vdots \end{array}$$

```
From: cheapsales@buystufffromme.com
To: Andrew Ng
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - $100
Med1cine (any kind) - £50
Also low cost M0rgages
available.
```

DeepLearning.AI Stanford ONLINE Andrew Ng



Building a spam classifier

How to try to reduce your spam classifier's error?

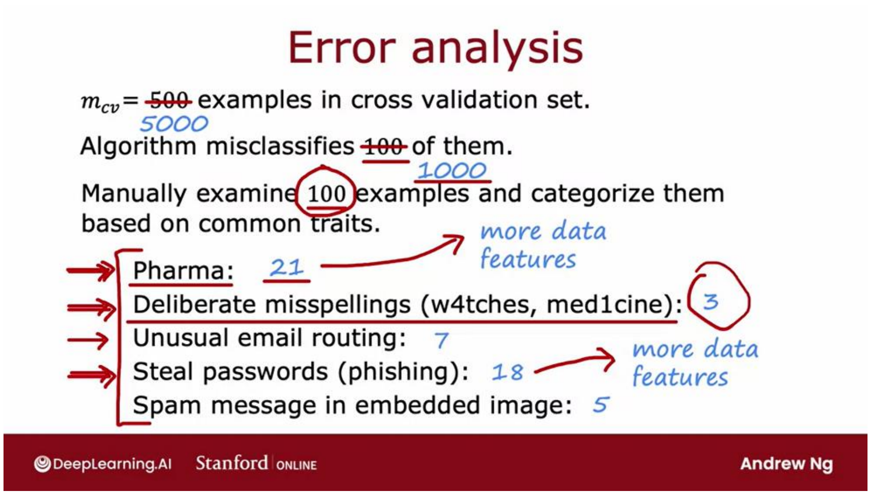
- Collect more data. E.g., "Honeypot" project.
- Develop sophisticated features based on email routing (from email header).
- Define sophisticated features from email body. E.g., should "discounting" and "discount" be treated as the same word.
- Design algorithms to detect misspellings. E.g., w4tches, med1cine, m0rtgage.

DeepLearning.AI Stanford ONLINE Andrew Ng

These ideas are all unique to your problem, although they might be identical moves in different problems.

Error Analysis

Andrew: "In diagnostics, bias and variance is the most important thing to consider, then comes error analysis."



Error analysis

m_{cv} = ~~500~~ 5000 examples in cross validation set.

Algorithm misclassifies ~~100~~ 1000 of them.

Manually examine 100 examples and categorize them based on common traits.

- Pharma: 21 $\rightarrow$ more data features
- Deliberate misspellings (w4tches, med1cine): 3
- Unusual email routing: 7
- Steal passwords (phishing): 18 $\rightarrow$ more data features
- Spam message in embedded image: 5

DeepLearning.AI Stanford ONLINE Andrew Ng

Let's say we have m_cv = 500 and algorithm misclassifies 100 of them. Examine why every sample is missclaffied. If only 3 of them are caused by same reason, it might not be worth it to focus on it. If 20-30 of them are close, focus on it. **It's just prioritizing.** (These error classes don't need to be disjoint/mutually exclusive.)

*If m_cv was 5000 and 1000 was error, i would sample subsets of that 1000 and examine them."

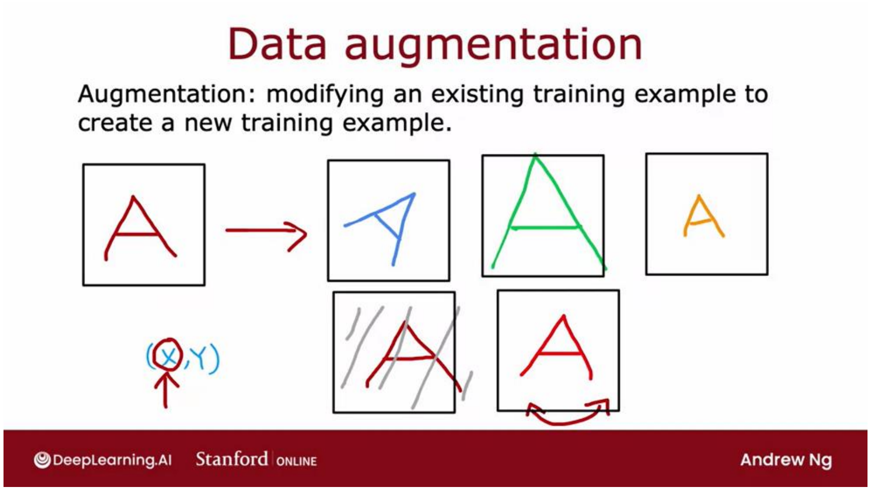
Limitation of Error Analysis: Can be hard for tasks that humans aren't good at. E.g. what adds will the customer click on? (How would i know?) **It's extremely important for other tasks though.**

Adding Data (tips on adding, collecting and creating more data)

"We applying ml algorithms we almost always wish we had more data. But data of all types can be ecpensive. **Instead you can add more data of the types that error analysis indicates it might help.** If you can't classify spam mails which include adds, you need to get more examples of add spams.

E.g. go to unlabeled data and ask labors to find related spam."

Another technique: DATA AUGMENTATION

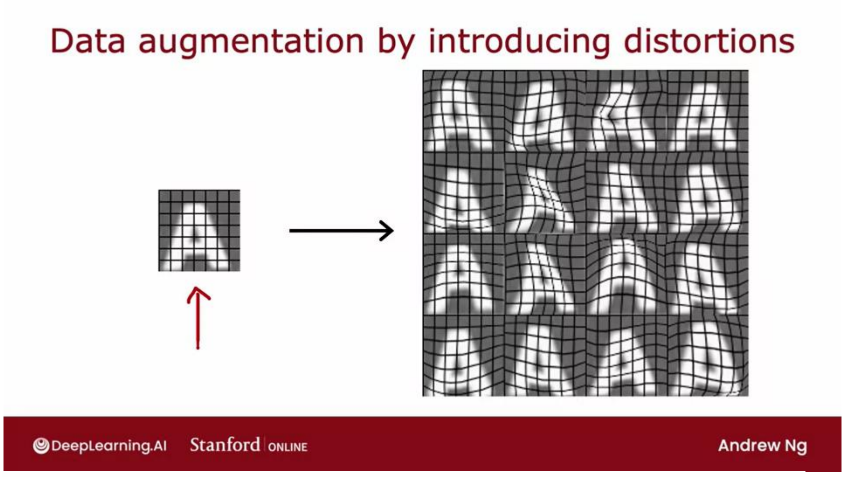


Data augmentation

Augmentation: modifying an existing training example to create a new training example.

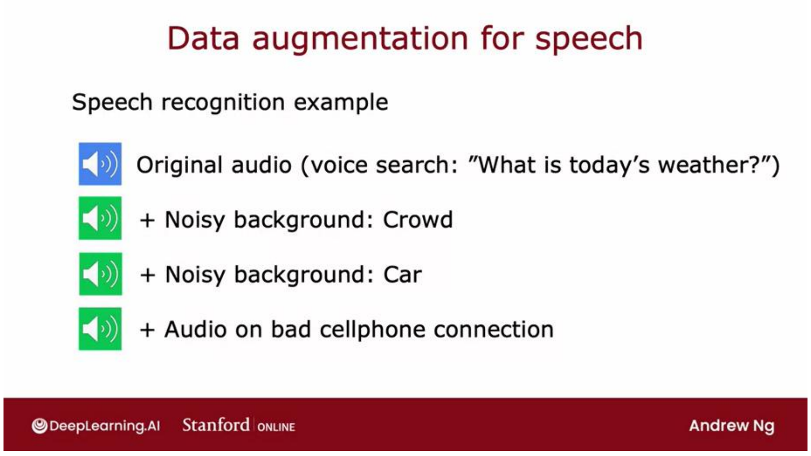
Widely used in images and audio data. Take existing sample to create new training example.

Zoom in and out, take mirror, play with background, **distortion**…



Data augmentation by introducing distortions

This also works for audio recognition.



Data augmentation for speech

Speech recognition example

- Original audio (voice search: "What is today's weather?")
- + Noisy background: Crowd
- + Noisy background: Car
- + Audio on bad cellphone connection

DeepLearning.AI Stanford ONLINE
Andrew Ng

IDEA IS: Do minor changes to boost undestanding of the model.

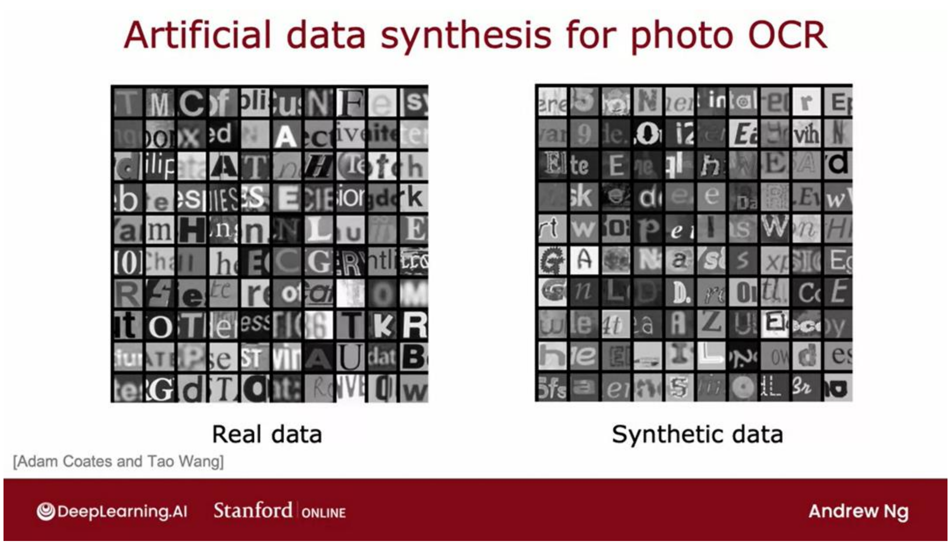
Here, we got 4 samples instead of 1. Noises that you add should be representative of the ones in your test set.

It doesn't help to add purely random noise.

Another tenchnique: Data synthesis

Using artificial data inputs to create new training examples.

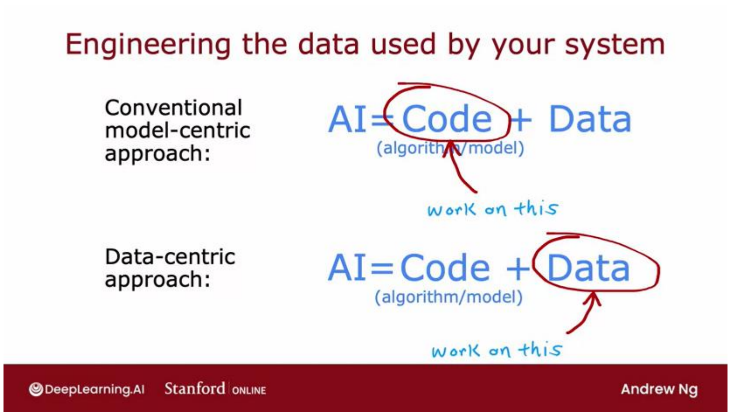
(Photo OCR: Reading text that appears in an image)


Artificial data synthesis for photo OCR
Real data
Synthetic data
[Adam Coates and Tao Wang]
DeepLearning.AI Stanford ONLINE
Andrew Ng

Change fonts and backgrounds to generate more data.

This is done mostly in computer vision tasks.

Everything we see in this video tries to engineer the data somehow to make it more usefull. You'll need to engineer your data yourself.



Engineering the data used by your system

Conventional model-centric approach: $AI = \text{Code} + \text{Data}$ (algorithm/model)

work on this

Data-centric approach: $AI = \text{Code} + \text{Data}$ (algorithm/model)

work on this

DeepLearning.AI Stanford ONLINE
Andrew Ng

You have 2 things to focus on. Best approach and optimum distribution of effort can vary.

TRANSFER LEARNING (use data from a different, barely related task)

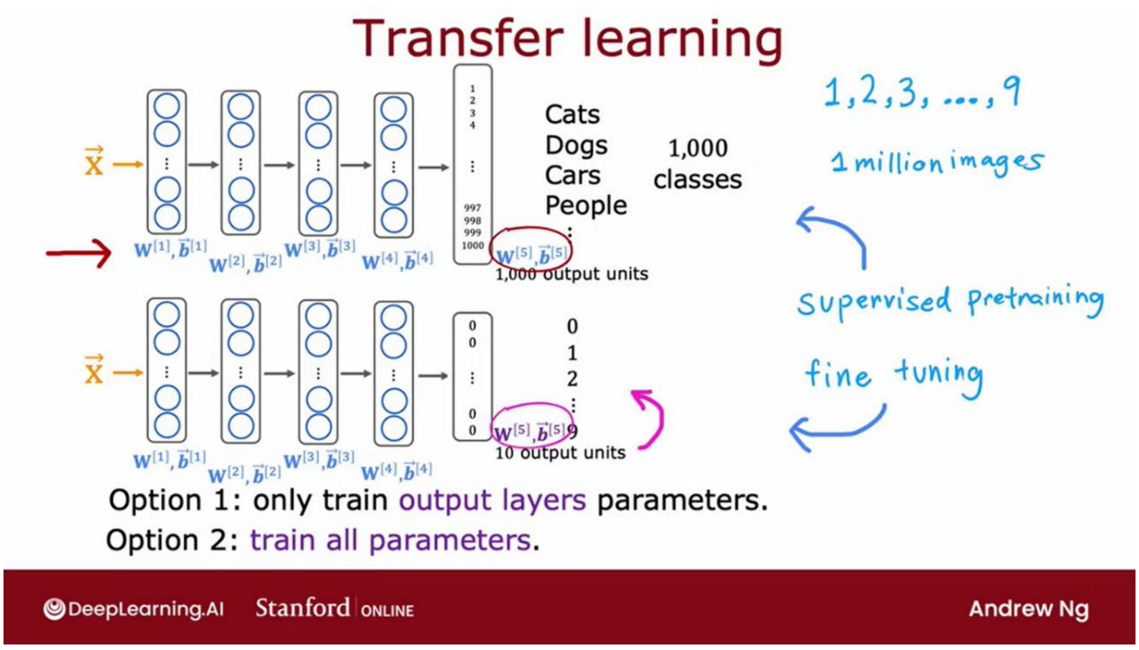
Used for applications where you don't have much data and/or collecting more data is not an option.

HOW??? "This is one of the tenchniques i use a lot."

This part introduces "pre-training" and "fine-tuning".

If you don't have digit examples, train a large neural network on classifying other images. Train it to adjust parameters of layers. Then change, only and only, the output layer to your needs. 2 OPTIONS:

1. Keep parameters, only train the output layer.
2. Initialize parameters with values from pre-training, and train them further together with the output layer.



- Other people publish pretrained models, and collaborating makes all of us get better results. They post their neural networks online (like HuggingFace) and you can download them.

Why does transfer learning ever work? Aren't they different?

Base objectives of these objectives are actually similar. They include detecting low level features like edges, corners, curves/basic shapes… Just like cat, dog, person include generic features of an image, a handwritten image has these features.

One restriction of pretraining though, input of pretraining and fine-tuning need to be same. They both need to be images (or sth else), they need to have same dimensions etc.

**Transfer learning helps a lot when data for your application isn't large.

"1. Download neural network parameters pretrained on a large dataset with same input type (e.g., images, audio, text) as your application (or train your own). → 1 million image

2. Further train (fine tune) the network on your own data. → 1000 images (sometimes few as 50, rare though)

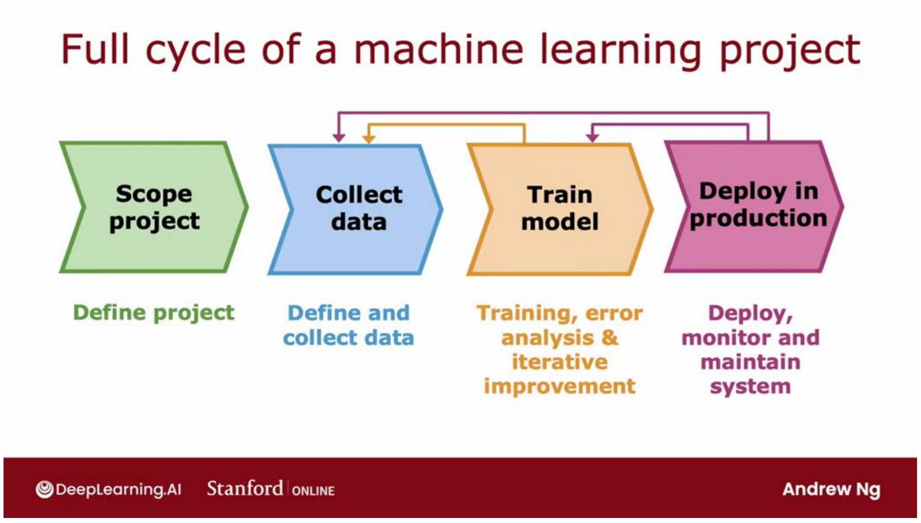
GPT-3, BERT, ImageNet → Successful pretrained models, trained with huge datasets

Full cycle of a machine learning project

What is the life cycle? Model training and data handling are just parts of it.

Example. Speech recognition

1. Define project (What is the goal) → search with your voide
2. Collect data (define it and then collect)
3. Train model, make analysis and improve iteratively (might return to 2)
    - Collect more general data or data referring to your errors.
    - **If audio with car noise can't get interpreted, collect more of it. (can use data augmentation)**
4. Deploy in production (monitor and maintain system)

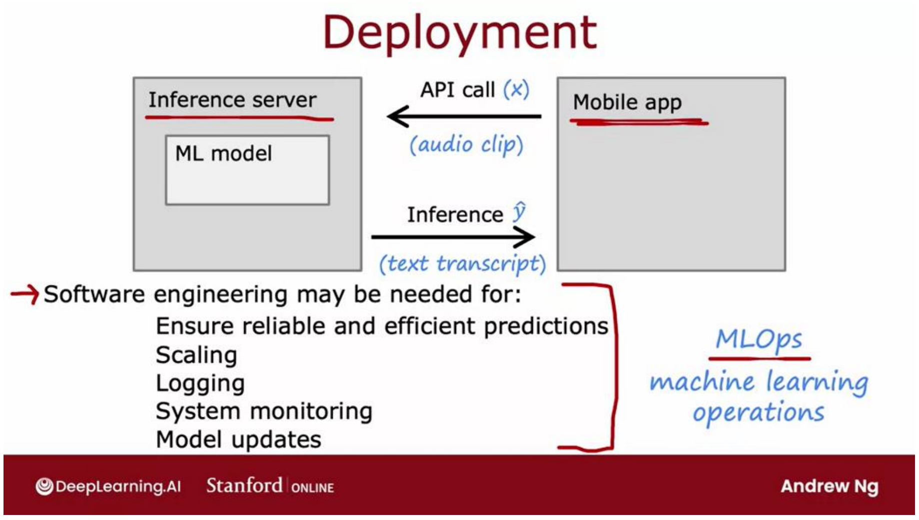


Can return to 2 or 3 from 4.

*****Data from users can give you more intuition that you might've possible missed.**

What is deployment like?

After training a high perfomance model, you implement it in a server. Server calls you ml model to make predictions. Consider it's in a mobile app. **App makes an "api call" with audio clip and server returns text transcript in return.**



Note that if user behaviour changes drastically, say people start searching for a new celebrity a lot, you model accuracy might decline and you might need to do trainig again. That is why monitoring is needed.

MLOps include: training, sustaining, keeping up to date, monitoring, keeping computation cost low with high accuracy…

Some issues related to fairness, bias and ethics

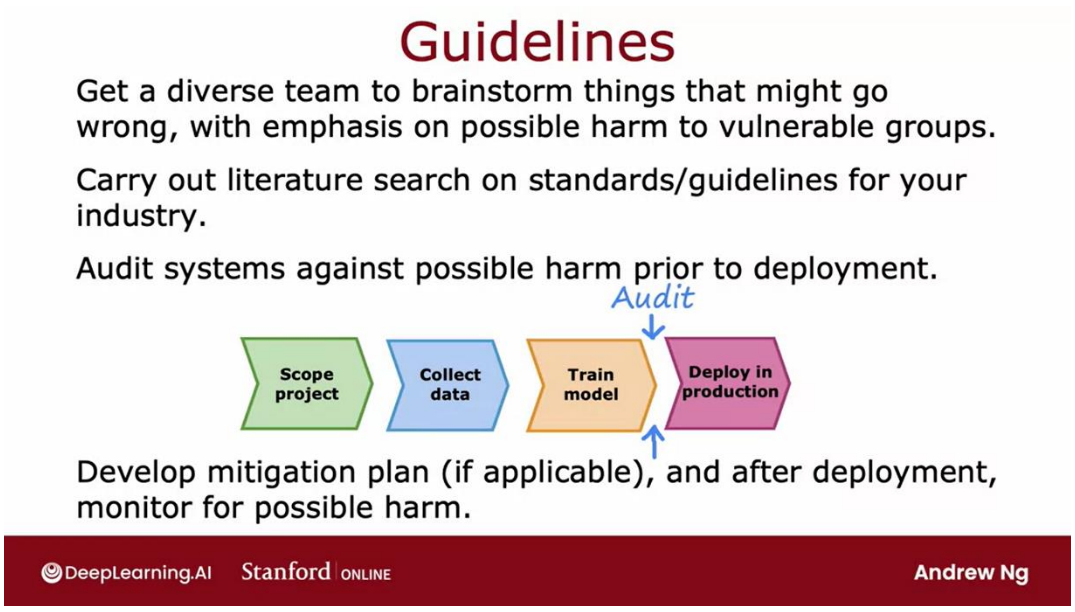
Fiar treatment of individuals is a necessity.

Fake products (video, audio) pose danger.

"Many times, i killed projects that were financially promising, just because they would make the world a worse place. They use these tools too. Don't be among them, be against them."

"I read a lot of ethics related books and there isn't a checklist for ethics."

- **I found that having more diverse teams actually causes a team collectively to be better at coming up with ideas about things that might go wrong.**



Guidelines

Get a diverse team to brainstorm things that might go wrong, with emphasis on possible harm to vulnerable groups.

Carry out literature search on standards/guidelines for your industry.

Audit systems against possible harm prior to deployment.

Audit
Scope project
Collect data
Train model
Deploy in production

Develop mitigation plan (if applicable), and after deployment, monitor for possible harm.

DeepLearning.AI Stanford ONLINE Andrew Ng

Skewed datasets (detecting rare classes)

If your data distribution is far from 50-50, our general metrics like accuracy don't work that well.

Example: Binary classifier for rare disease (y is rarely equal to 1)

You find out that test error is %1 and it works %99 of the time. Is this a good score?

If disease is already rare, say %0.5 people have it, then even a non-learning "Say healthy to everyone" algorithm will have a %99.5 accuracy. This means %1 error is too much.

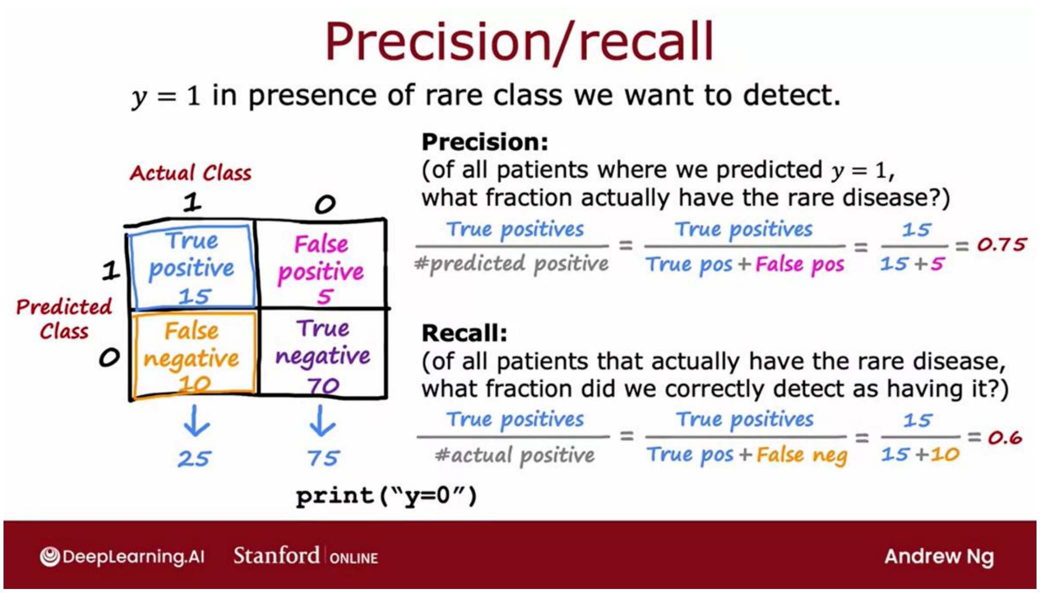
***Only looking at accuracy is not meaningful. %0.5 error might label patients as healthy, while %1 error labels healthy people as patients.

***In skewed datasets, we usually use a different error metric rather than jsut classification error.**

- **Common pair of error metrics: PRECISION and RECALL**

- **Precision:** When the model says person has the disease, what is the probability of that person having the disease?

- **Recall:** When a person has the disease, what is the probability of the disease being diagnosed?



Precision/recall

$y = 1$ in presence of rare class we want to detect.

		Actual Class 1	Actual Class 0
Predicted Class	1	True positive 15	False positive 5
	0	False negative 10	True negative 70
		↓ 25	↓ 75

`print("y=0")`

Precision:
(of all patients where we predicted $y = 1$, what fraction actually have the rare disease?)

$$\frac{\text{True positives}}{\#\text{predicted positive}} = \frac{\text{True positives}}{\text{True pos} + \text{False pos}} = \frac{15}{15+5} = 0.75$$

Recall:
(of all patients that actually have the rare disease, what fraction did we correctly detect as having it?)

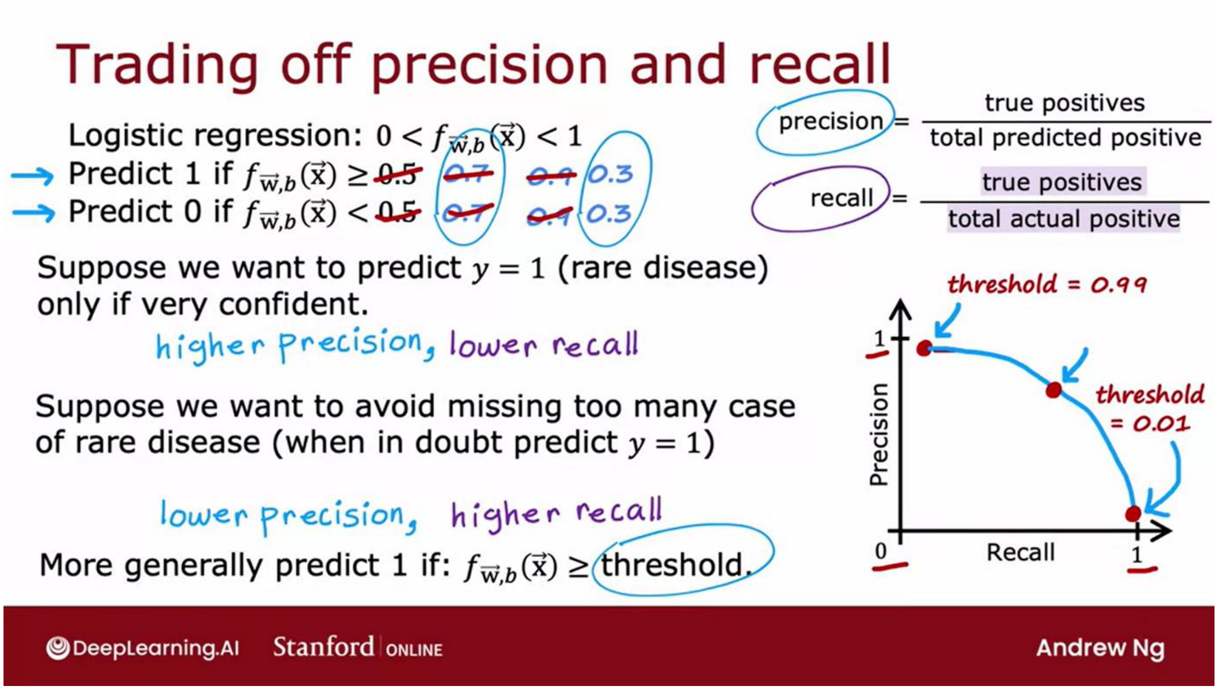
$$\frac{\text{True positives}}{\#\text{actual positive}} = \frac{\text{True positives}}{\text{True pos} + \text{False neg}} = \frac{15}{15+10} = 0.6$$

DeepLearning.AI Stanford ONLINE Andrew Ng

Trading off precision and recall to optimize performance of algorithm

In practice, there is often a trade-off between these two.

- In case of a disease, we might want to only consider the patient sick if we are really confident of it. This means decision boundary moves up.



Trading off precision and recall

Logistic regression: $0 < f_{\vec{w},b}(\vec{x}) < 1$

→ Predict 1 if $f_{\vec{w},b}(\vec{x}) \geq$ ~~0.5~~ ~~0.7~~ ~~0.9~~ 0.3

→ Predict 0 if $f_{\vec{w},b}(\vec{x}) <$ ~~0.5~~ ~~0.7~~ ~~0.9~~ 0.3

Suppose we want to predict $y = 1$ (rare disease) only if very confident.

higher precision, lower recall

Suppose we want to avoid missing too many case of rare disease (when in doubt predict $y = 1$)

lower precision, higher recall

More generally predict 1 if: $f_{\vec{w},b}(\vec{x}) \geq$ threshold.

$$\text{precision} = \frac{\text{true positives}}{\text{total predicted positive}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{total actual positive}}$$

DeepLearning.AI Stanford ONLINE Andrew Ng

IMPORTANT IMAGE

Threshold affects precision and recall. Choice of threshold will vary from problem to problem. **What you might do, is to plot the "precision-recall graph" and pick a value in between for the threshold.**

Unlike "d selection with cross validation error", you need to pick the threshold manually. Because there isn't a point with minimum cost and a trade-off happens between precision and recall.

F1 SCORE → Used to pick threshold automatically

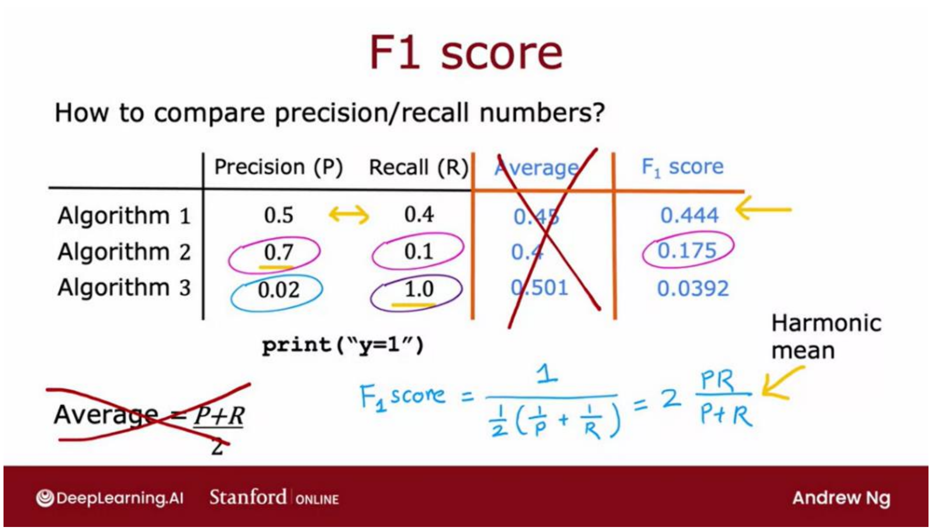
You're now evaluating your model with 2 different metrics. How should one compare them?

Way is to **combine these two metrics** into a single score. We can then just look at the score.

Note: Getting average is not a good strategy.

Note 2: If one of these metrics are too small, it's a big problem and this needs to be addressed when evaluating performance.

Solution: F1 score → harmonic mean: 2 / (1/P + 1/R) → 2.P.R / (P + R)



F1 score

How to compare precision/recall numbers?

	Precision (P)	Recall (R)	~~Average~~	F_1 score
Algorithm 1	0.5	0.4	~~0.45~~	0.444
Algorithm 2	0.7	0.1	~~0.4~~	0.175
Algorithm 3	0.02	1.0	~~0.501~~	0.0392

print("y=1")

~~Average = $\frac{P+R}{2}$~~

Harmonic mean

$$F_1 \text{score} = \frac{1}{\frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right)} = 2\frac{PR}{P+R}$$

DeepLearning.AI Stanford ONLINE Andrew Ng

Harmonic mean, is a way of getting average while paying more attention to lower values. (See the table, f1 is closer to the lower value.)