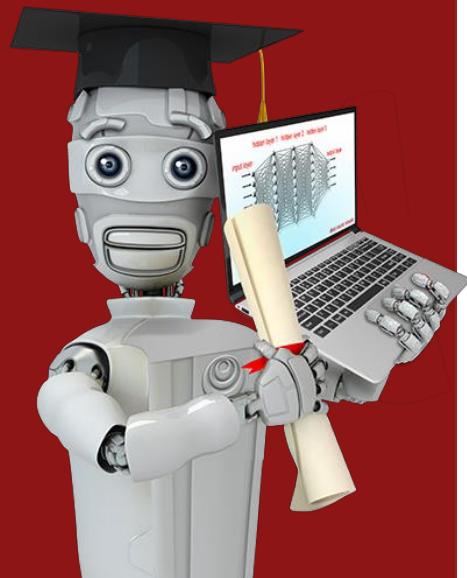


Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



Advice for applying machine learning

Deciding what to try next

Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

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

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

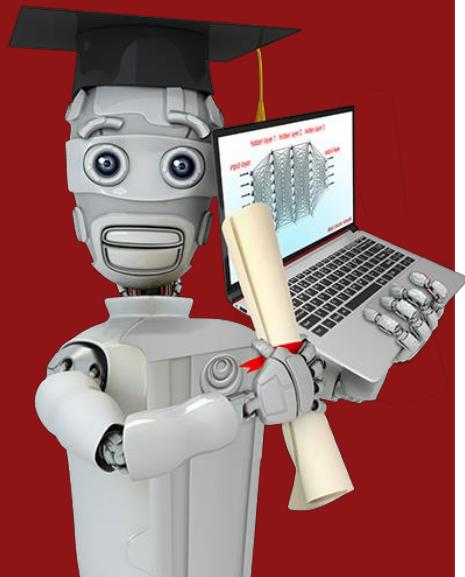
- Get more training examples
- Try smaller sets of features
- Try getting additional features
- Try adding polynomial features ($x_1^2, x_2^2, x_1x_2, etc$)
- Try decreasing λ
- Try increasing λ

Machine learning diagnostic

Diagnostic: A test that you run to gain insight into what is/isn't working with a learning algorithm, to gain guidance into improving its performance.

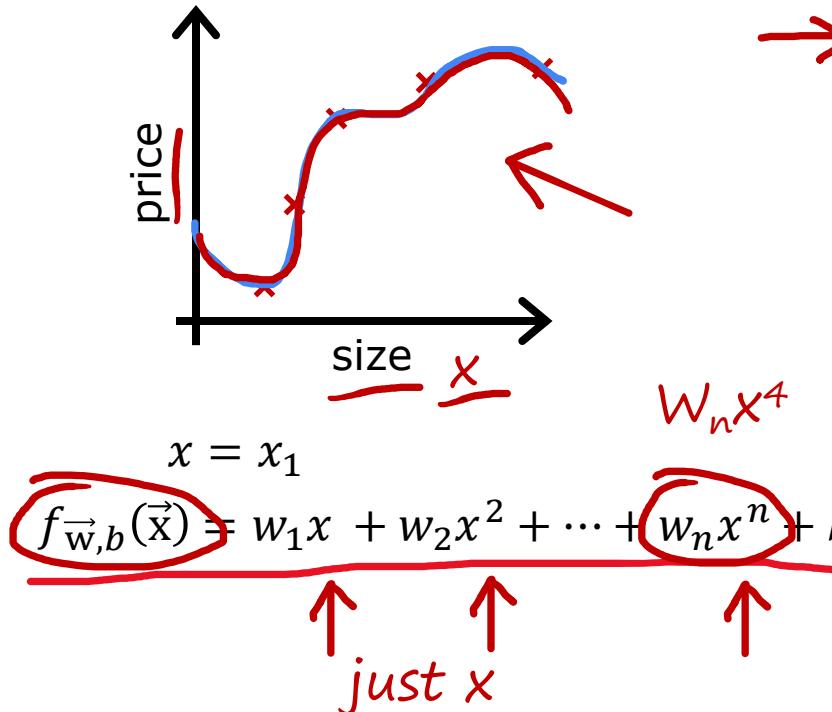
Diagnostics can take time to implement but doing so can be a very good use of your time.

Evaluating and choosing models



Evaluating a model

Evaluating your model



→ Model fits the training data well but will fail to generalize to new examples not in the training set.

- x_1 = size in feet²
- x_2 = no. of bedrooms
- x_3 = no. of floors
- x_4 = age of home in years

Evaluating your model

Dataset:

	size	price	
70%	2104	400	$(x^{(1)}, y^{(1)})$
	1600	330	$(x^{(2)}, y^{(2)})$
	2400	369	\vdots
	1416	232	$(x^{(m_{train})}, y^{m_{train}})$
	3000	540	
	1985	300	
	1534	315	
30%	1427	199	$(x_{test}^{(1)}, y_{test}^{(1)})$
	1380	212	\vdots
	1494	243	$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$

Diagram illustrating the splitting of a dataset into training and test sets. A red bracket labeled "training set" groups the first 7 rows (70% of the data). A red bracket labeled "test set" groups the last 3 rows (30% of the data). The training set is mapped to a sequence of pairs $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m_{train})}, y^{m_{train}})$. The test set is mapped to a sequence of pairs $(x_{test}^{(1)}, y_{test}^{(1)}), (x_{test}^{(2)}, y_{test}^{(2)}), \dots, (x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$.

m_{train} = no. training examples = 7

m_{test} = no. test examples = 3

Train/test procedure for linear regression (with squared error cost)

Fit parameters by minimizing cost function $J(\vec{w}, b)$

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

Compute test error:

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

Compute training error:

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

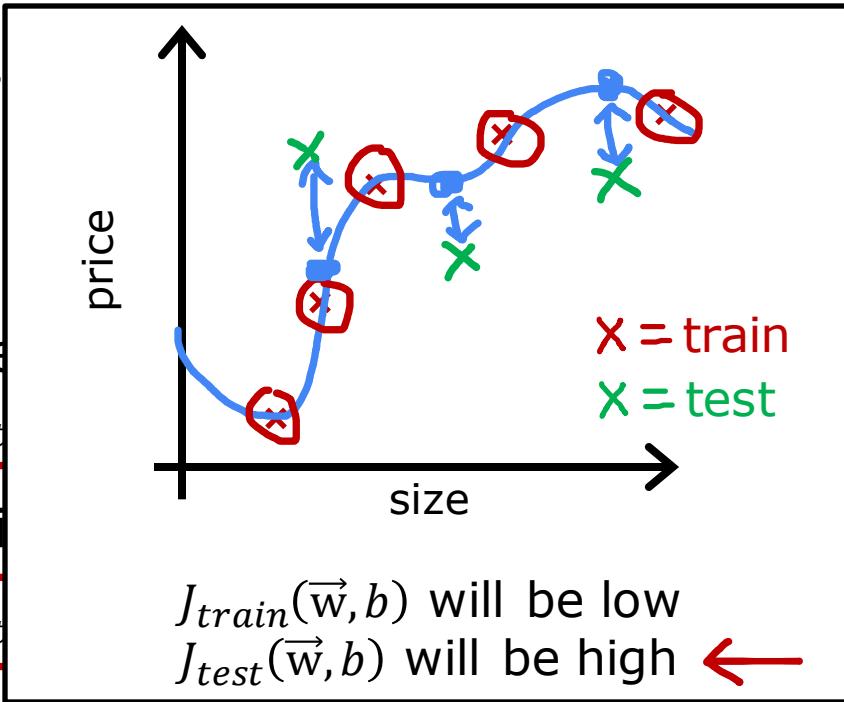
Train/test procedure for linear regression (with squared error cost)

Fit parameters

$$\rightarrow J(\vec{w}, b) = \min_{\vec{w}, b}$$

Compute test error

Compute training error



$b)$

$$\left[\sum_{j=1}^n w_j^2 \right]$$

$$\left[(i) \right. \left. est \right]^2$$

$$\left[-y_{train}^{(i)} \right]^2 \left. \right]$$

Train/test procedure for classification problem

O / |

Fit parameters by minimizing $J(\vec{w}, b)$ to find \vec{w}, b

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

Compute test error:

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

Compute train error:

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

Train/test procedure for classification problem

O / 1

Fit parameters by minimizing $J(\vec{w}, b)$ to find \vec{w}, b

E.g.,

$$J(\vec{w}, b) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))]$$

Compute test error:

$$J_{test}(\vec{w}, b) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} [y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))]$$

Compute train error:

$$J_{train}(\vec{w}, b) = -\frac{1}{m_{train}} \sum_{i=1}^{m_{train}} [y^{(i)} \log(f_{\vec{w}, b}(\vec{x}^{(i)})) + (1 - y^{(i)}) \log(1 - f_{\vec{w}, b}(\vec{x}^{(i)}))]$$

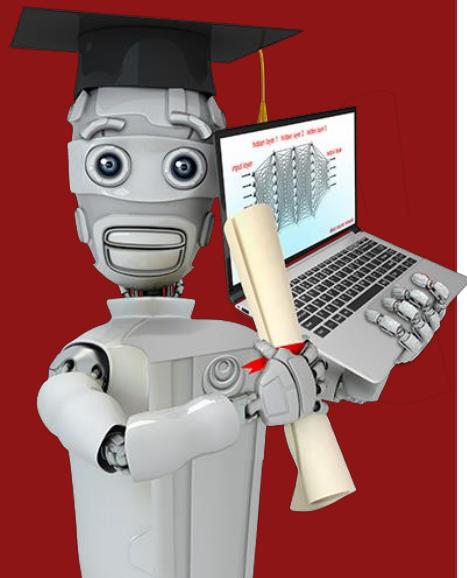
fraction of the test set and the fraction of the train set that the algorithm has misclassified.

$$\hat{y} = \begin{cases} 1 & \text{if } f_{\vec{w}, b}(\vec{x}^{(i)}) \geq 0.5 \\ 0 & \text{if } f_{\vec{w}, b}(\vec{x}^{(i)}) < 0.5 \end{cases}$$

count $\hat{y} \neq y$

$J_{test}(\vec{w}, b)$ is the fraction of the test set that has been misclassified.

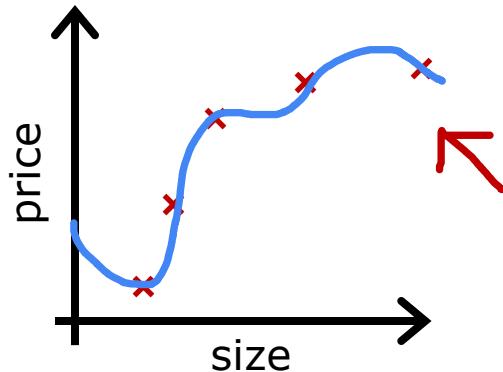
$J_{train}(\vec{w}, b)$ is the fraction of the train set that has been misclassified.



Evaluating and choosing models

Model selection and training/cross validation/test sets

Model selection (choosing a model)



$$f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

Once parameters \vec{w}, b are fit to the training set, the training error $J_{train}(\vec{w}, b)$ is likely lower than the actual generalization error.

$J_{test}(\vec{w}, b)$ is better estimate of how well the model will generalize to new data than $J_{train}(\vec{w}, b)$.

Model selection (choosing a model)

$d=1$

$$1. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + b$$

$d=2$

$$2. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + b$$

$d=3$

$$3. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + w_3 x^3 + b$$

\vdots

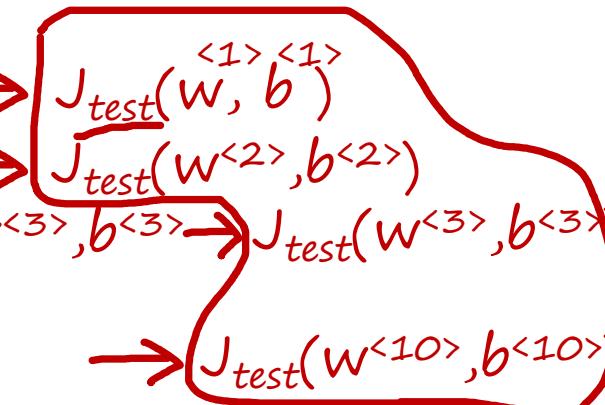
$d=10$

$$10. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + \dots + w_{10} x^{10} + b$$

$$\xrightarrow{w, b^{<1>} \atop w, b^{<2>}} \xrightarrow{w, b^{<1>} \atop w, b^{<2>}}$$

$$\xrightarrow{w^{<3>}, b^{<3>}} \xrightarrow{w^{<3>}, b^{<3>}}$$

$$\xrightarrow{w^{<10>}, b^{<10>}}$$



Choose $w_1 x_1 + \dots + w_5 x^5 + b$ $d=5$

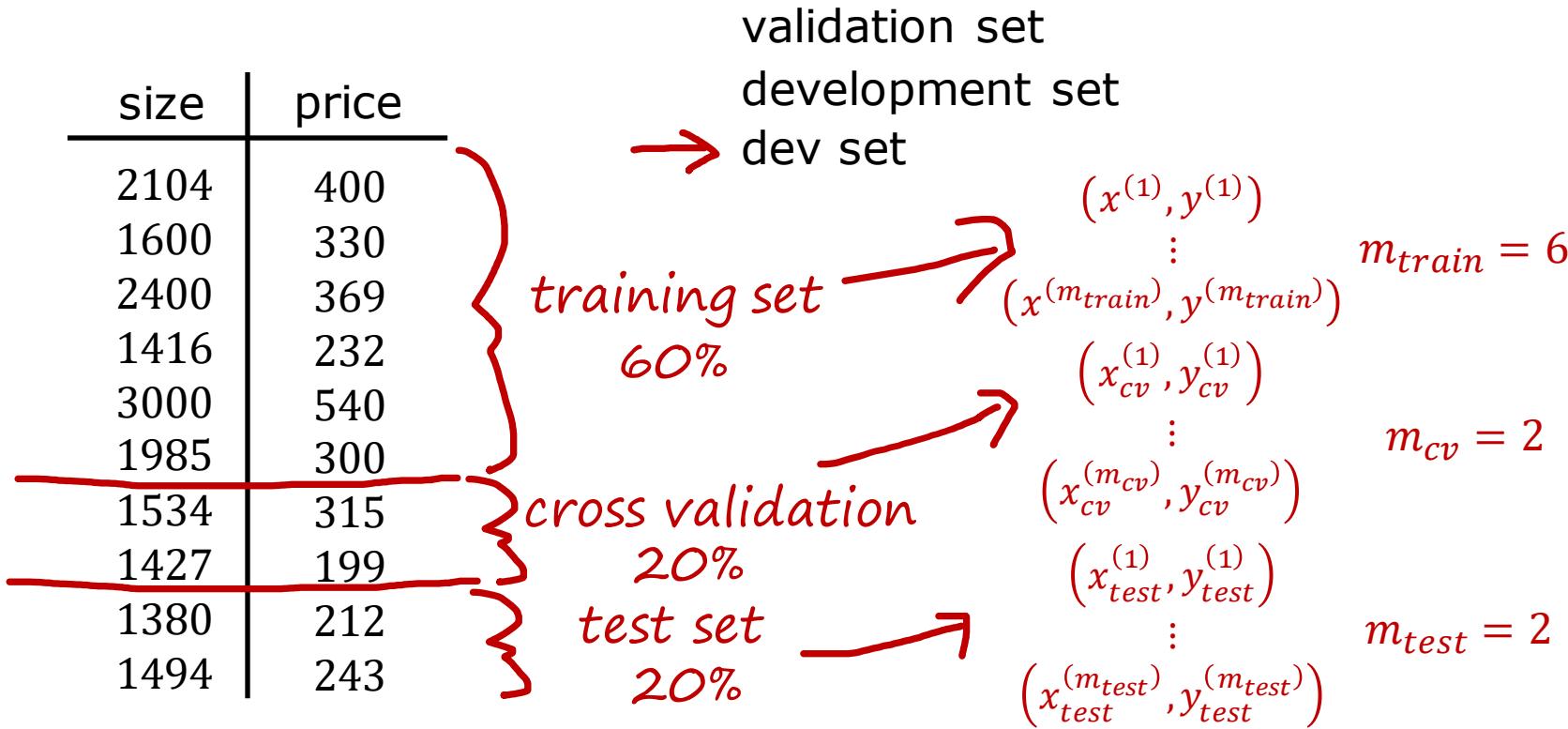
$$J_{test}(w^{<5>}, b^{<5>})$$

How well does the model perform? Report test set error $J_{test}(w^{<5>}, b^{<5>})$?

The problem is $J_{test}(w^{<5>}, b^{<5>})$ is likely to be an optimistic estimate of generalization error. Ie: An extra parameter d (degree of polynomial) was chosen using the test set.

w, b

Training/cross validation/test set



Training/cross validation/test set

Training error:
$$J_{train}(\vec{w}, b) = \frac{1}{2m_{train}} \left[\sum_{i=1}^{m_{train}} (f_{\vec{w}, b}(\vec{x}^{(i)}) - y^{(i)})^2 \right]$$

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

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

Model selection

$d=1$

$$1. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + b$$

$d=2$

$$2. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + b$$

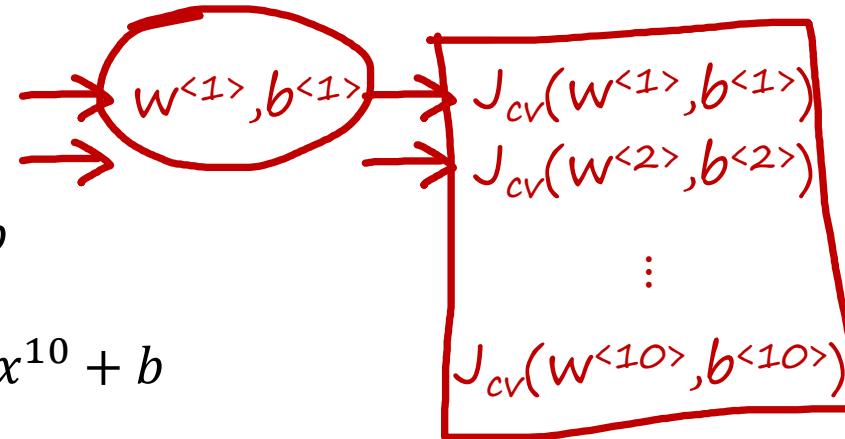
$d=3$

$$3. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + w_3 x^3 + b$$

:

$d=10$

$$10. f_{\vec{w}, b}(\vec{x}) = w_1 x_1 + w_2 x^2 + \dots + w_{10} x^{10} + b$$



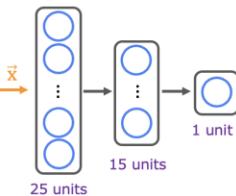
→ Pick $w_1 x_1 + \dots + w_4 x^4 + b$

$(J_{cv}(w^{<4>}, b^{<4>}))$

Estimate generalization error using test the set: $J_{test}(w^{<4>}, b^{<4>})$

Model selection – choosing a neural network architecture

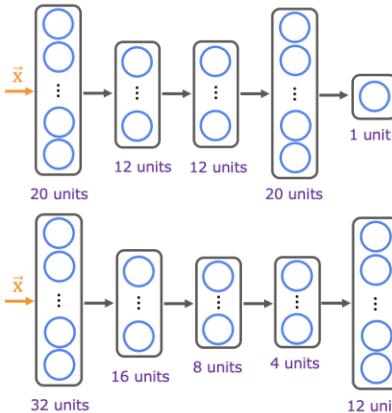
1.



$$w^{(1)}, b^{(1)}$$

$$\underline{J_{cv}(\mathbf{W}^{(1)}, \mathbf{B}^{(1)})}$$

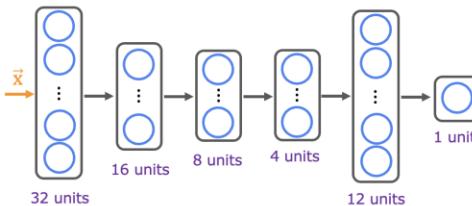
→ 2.



$$w^{(2)}, b^{(2)}$$

$$\textcircled{J_{cv}(\mathbf{W}^{(2)}, \mathbf{B}^{(2)})}$$

3.



$$w^{(3)}, b^{(3)}$$

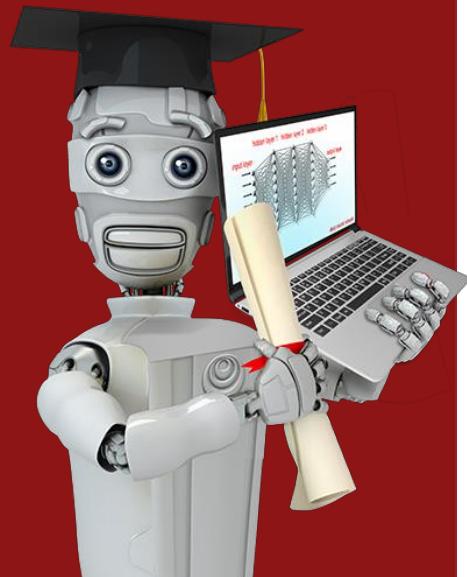
$$J_{cv}(\mathbf{W}^{(3)}, \mathbf{B}^{(3)})$$

Pick $\mathbf{W}^{(2)}, \mathbf{B}^{(2)}$

Train, CV

Estimate generalization error using the test set:

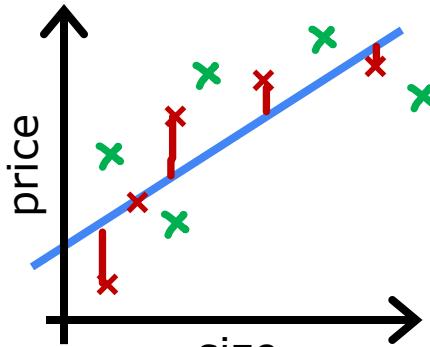
$$\boxed{J_{test}(\mathbf{W}^{(2)}, \mathbf{B}^{(2)})}$$



Bias and variance

Diagnosing bias and variance

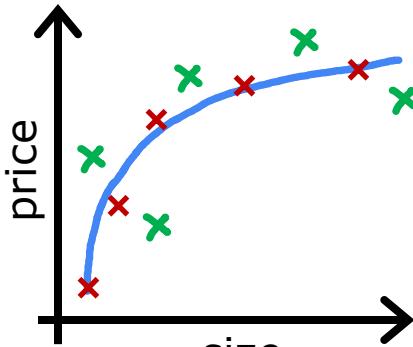
Bias/variance



$$f_{\vec{w}, b}(x) = w_1 x + b$$

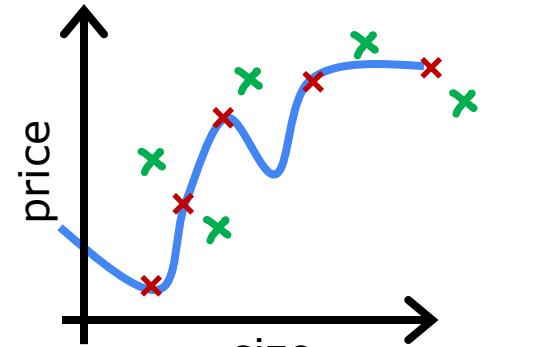
→ High bias
(underfit)

$d = 1$ J_{train} is high
 J_{cv} is high



$$f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + b$$

"Just right"



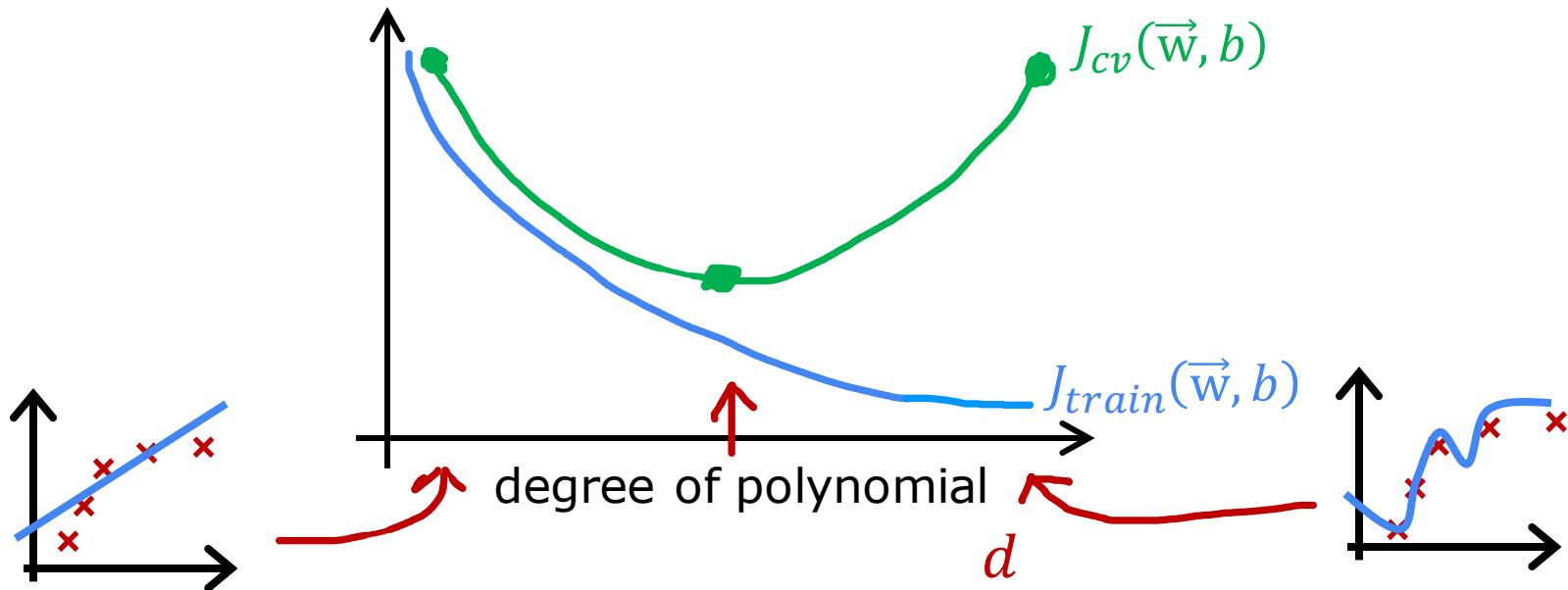
$$f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

High variance
(overfit)

$d = 2$ J_{train} is low
 J_{cv} is low

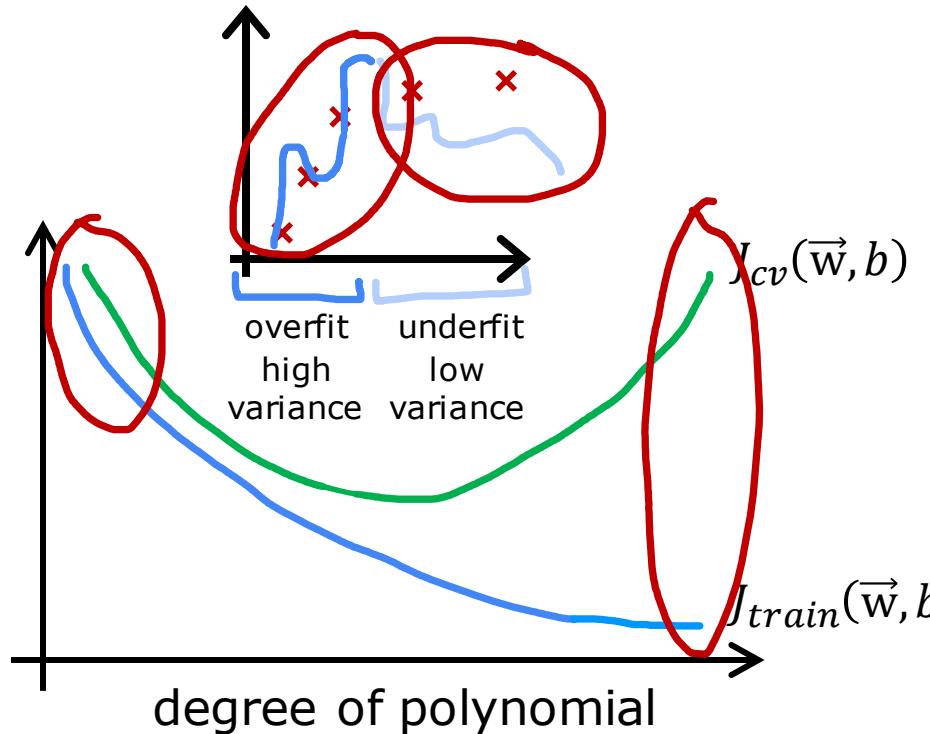
$d = 4$ J_{train} is low
 J_{cv} is high

Understanding bias and variance



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)

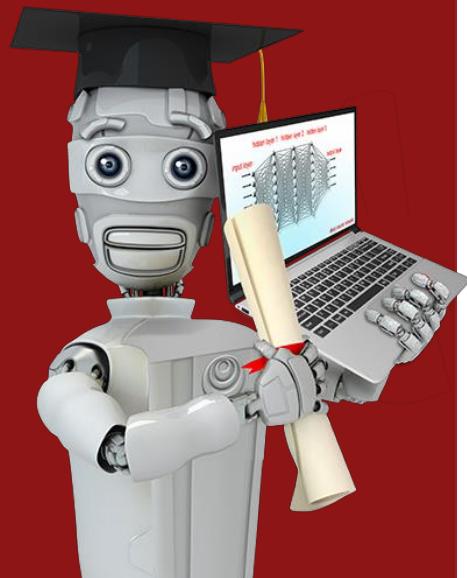
$J_{cv} \gg J_{train}$

(J_{train} may be low)

High bias and high variance

J_{train} will be high

and $J_{cv} \gg J_{train}$



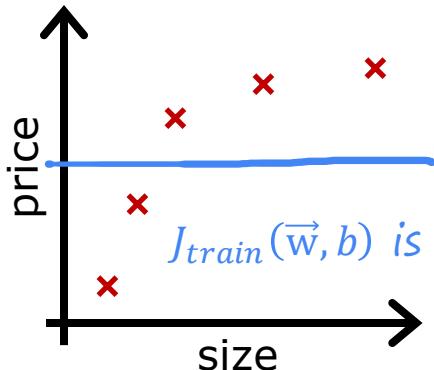
Bias and variance

Regularization and
bias/variance

Linear regression with regularization

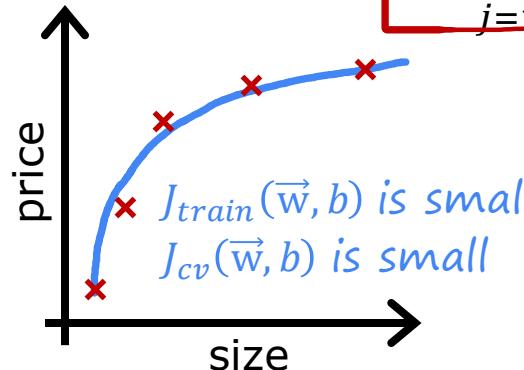
Model: $f_{\vec{w}, b}(x) = \underbrace{w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b}_m$

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



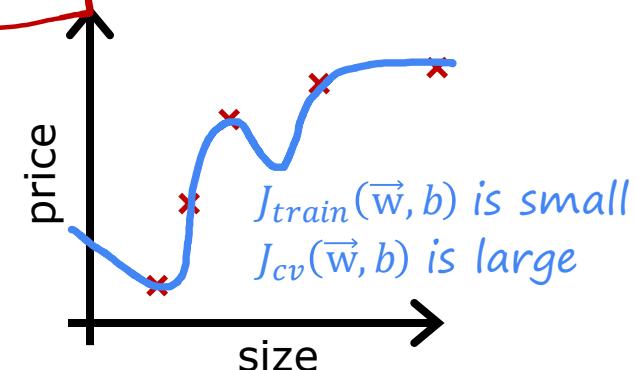
Large λ
High bias (underfit)

$$\lambda = 10,000 \quad w_1 \approx 0, w_2 \approx 0$$
$$f_{\vec{w}, b}(\vec{x}) \approx b$$



Intermediate λ

$$\lambda$$

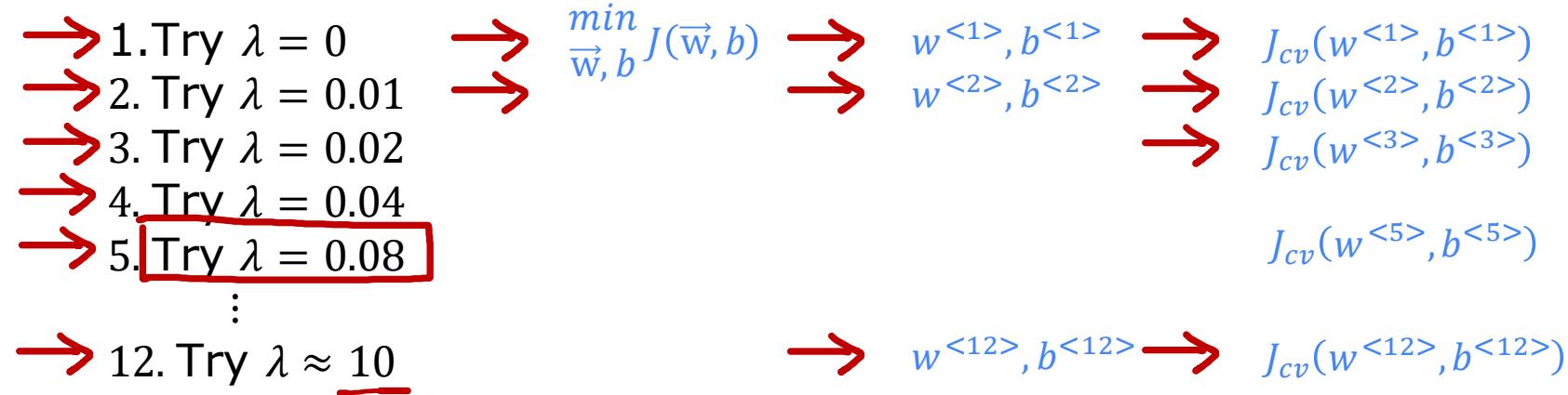


Small λ
High variance (overfit)

$$\lambda = 0$$

Choosing the regularization parameter λ

Model: $f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$

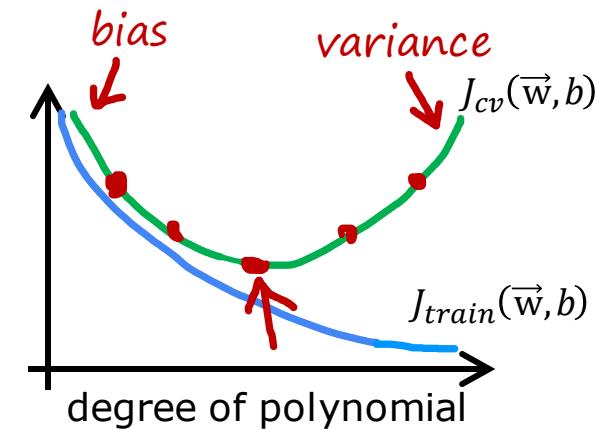
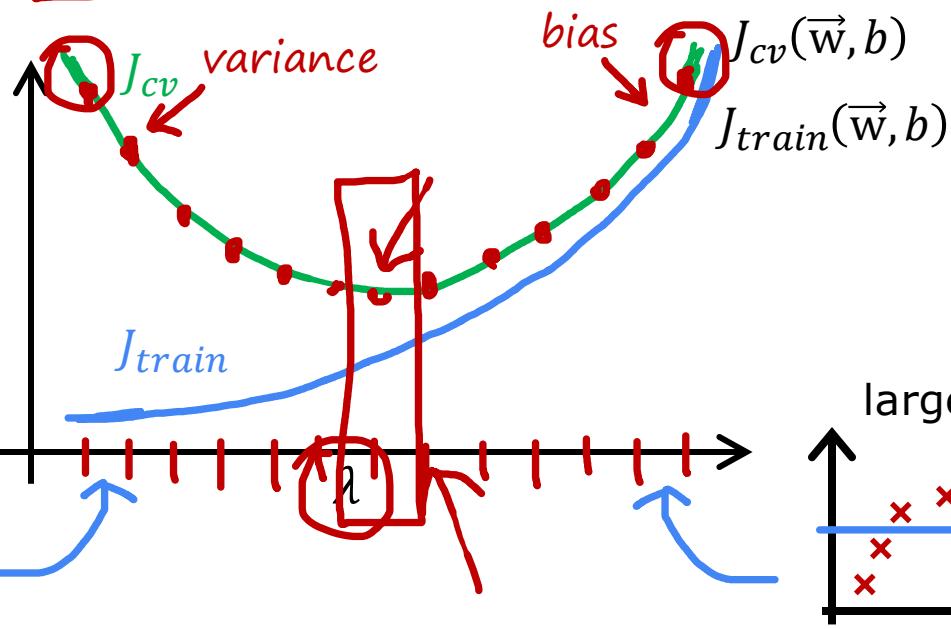


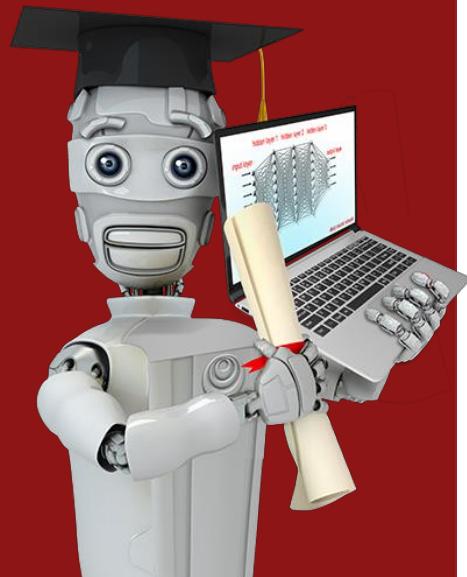
Pick $w^{<5>}, b^{<5>}$

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

Bias and variance as a function of regularization parameter λ

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





Bias and variance

Establishing a baseline level of performance

Speech recognition example



Human level performance

: 10.6% 0.2% 4.0%

Training error J_{train}

: 10.8%

Cross validation error J_{cv}

: 14.8%

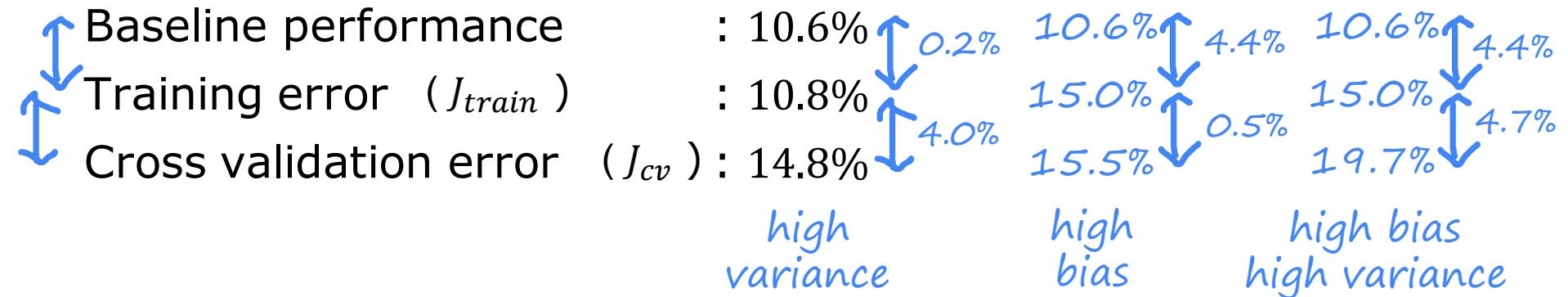


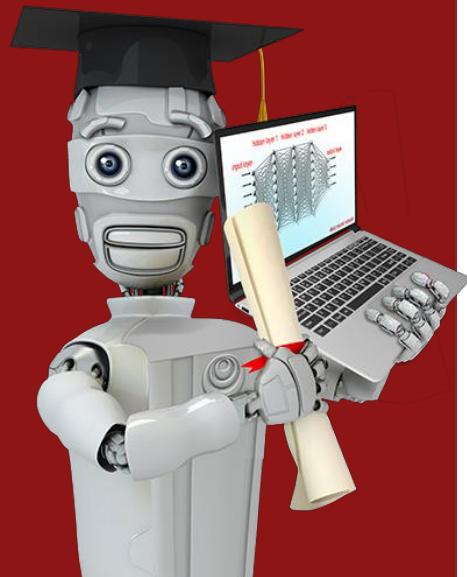
Establishing a baseline level of performance

What is the level of error you can reasonably hope to get to?

- Human level performance
- Competing algorithms performance
- Guess based on experience

Bias/variance examples





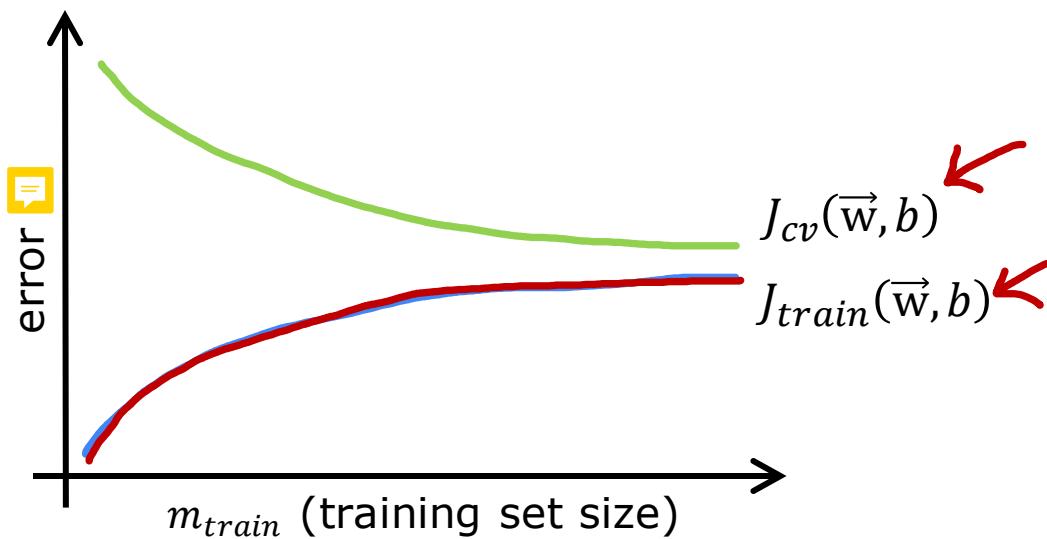
Bias and variance

Learning curves

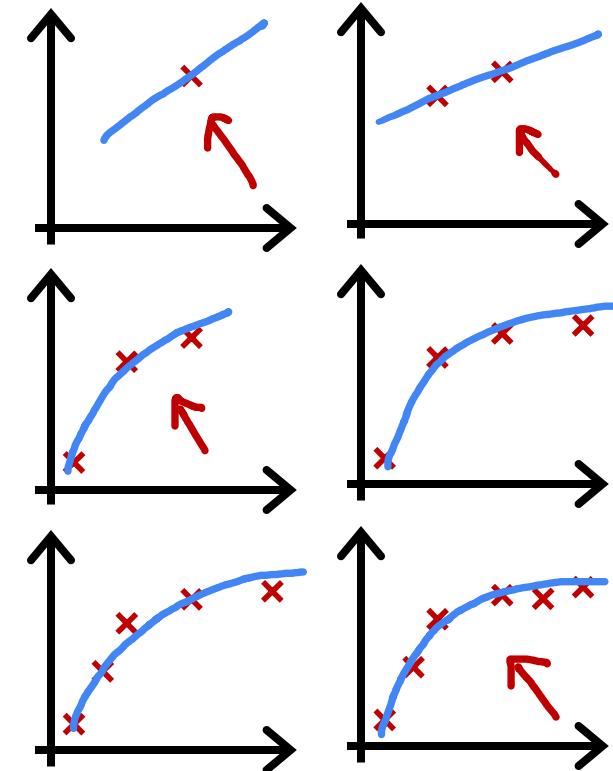
Learning curves

Training error vs training set size

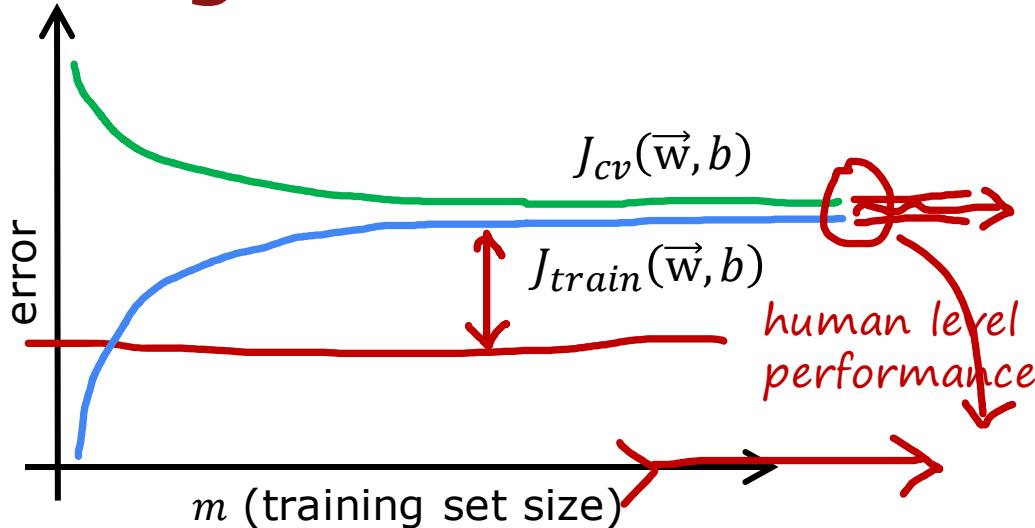
Cross-validation error vs training set size



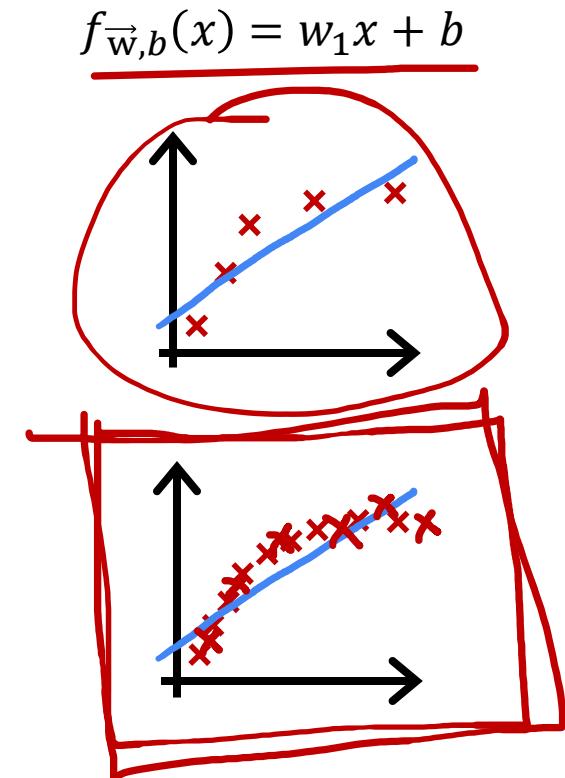
$$f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + b$$



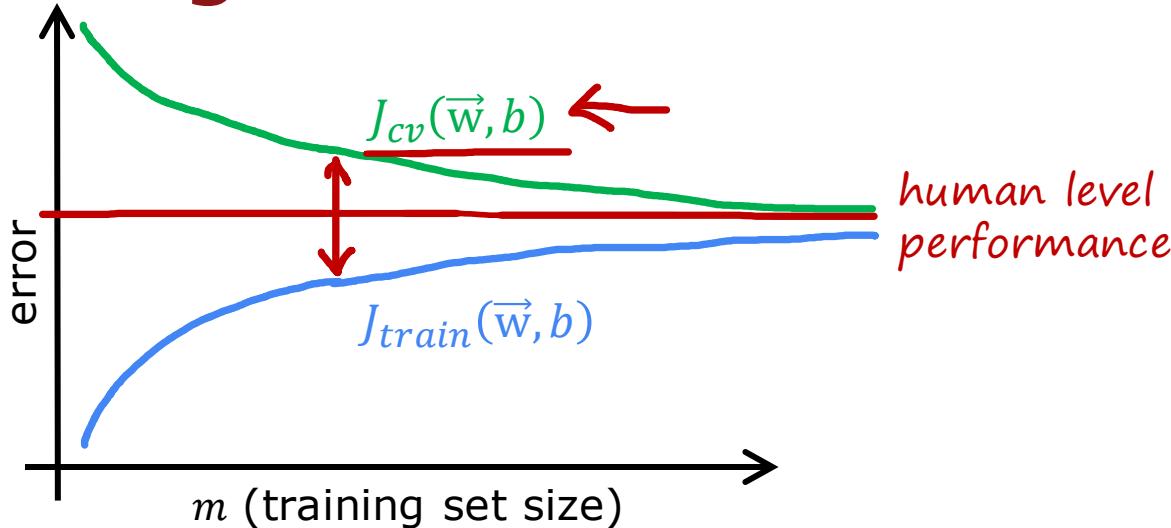
High bias



if a learning algorithm suffers from high bias, getting more training data will not (by itself) help much.

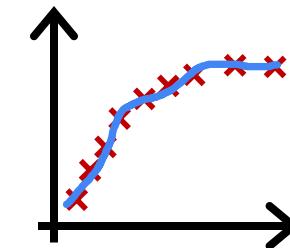
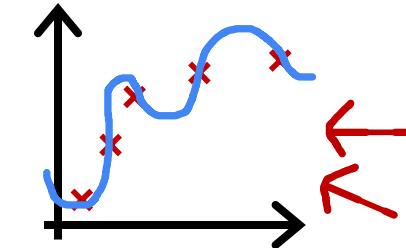


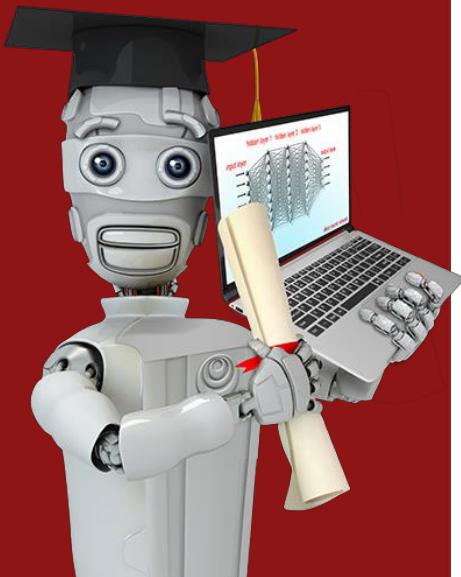
High variance



if a learning algorithm suffers from high variance, getting more training data is likely to help.

$$f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b \quad (\text{with small } \lambda)$$





Bias and variance

Deciding what to try next
revisited

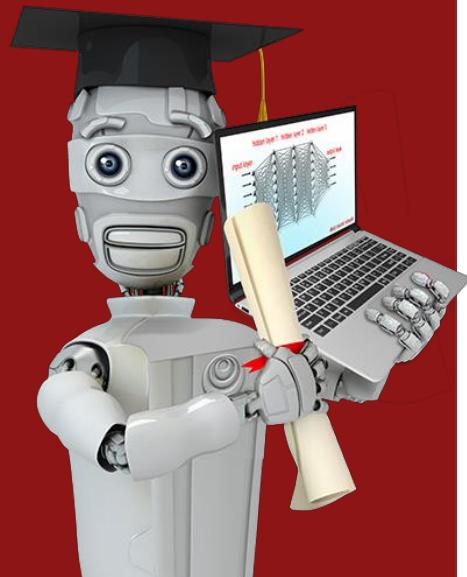
Debugging a learning algorithm

You've implemented regularized linear regression on housing prices

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

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 $x, x^2, \cancel{x}, \cancel{x^3}, \cancel{x^4}, \dots$ fixes high variance
- Try getting additional features ← fixes high bias
- Try adding polynomial features $(x_1^2, x_2^2, x_1 x_2, \text{etc})$ fixes high bias
- Try decreasing λ ← fixes high bias
- Try increasing λ ← fixes high variance



Bias and variance

**Bias/variance and
neural networks**

The bias variance tradeoff

$$f_{\vec{w}, b}(x) = w_1 x + b$$

Simple model

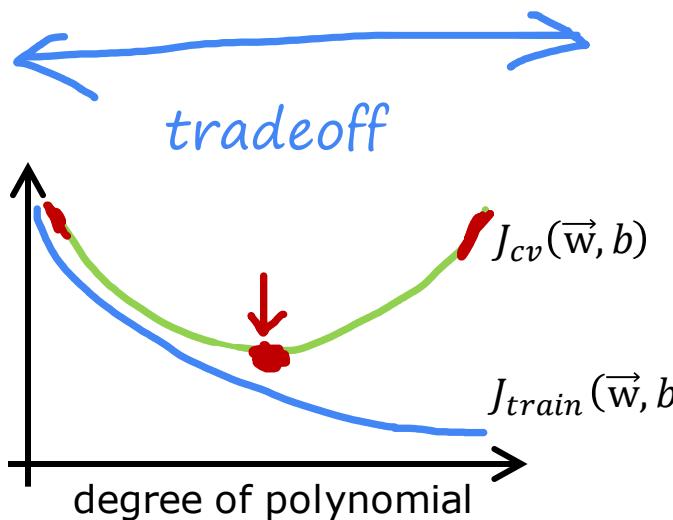
High bias

$$f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + b$$

$$f_{\vec{w}, b}(x) = w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + b$$

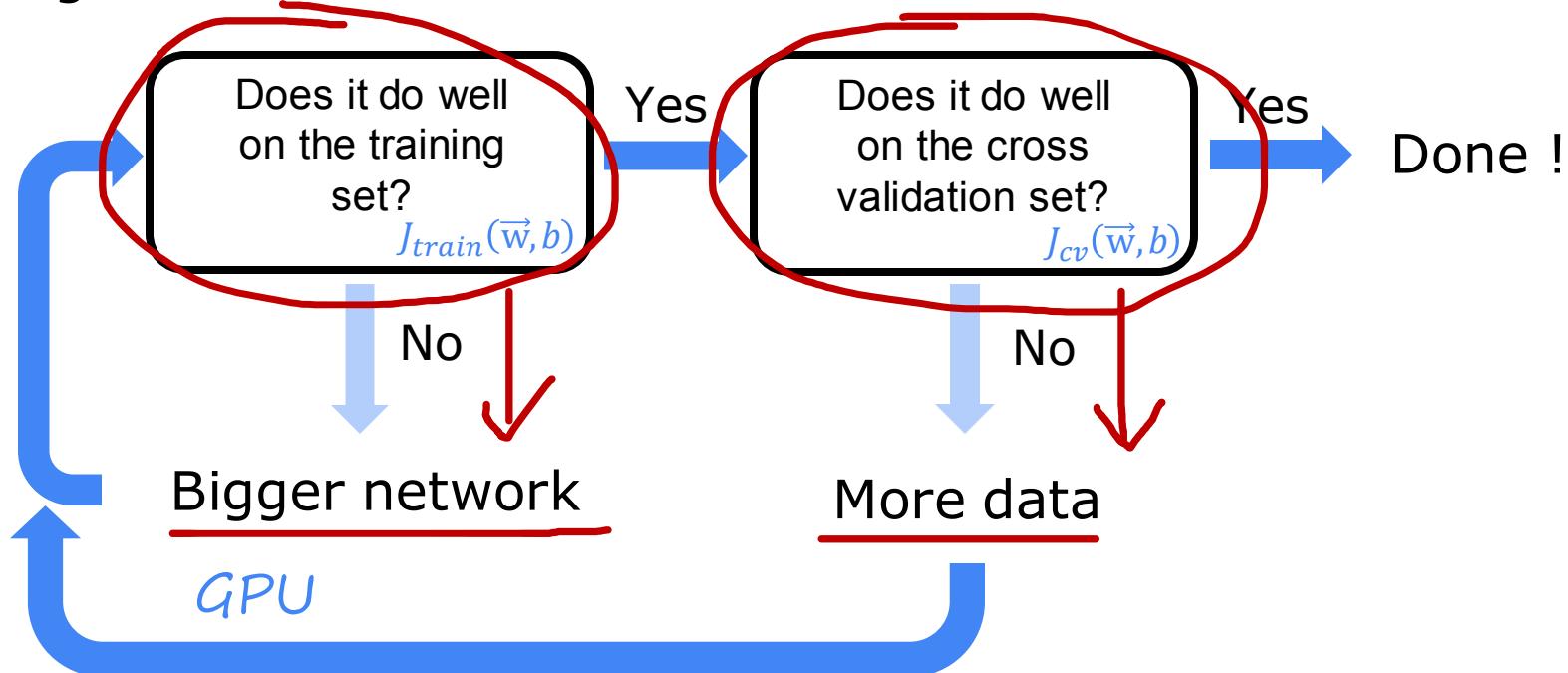
Complex model

High variance

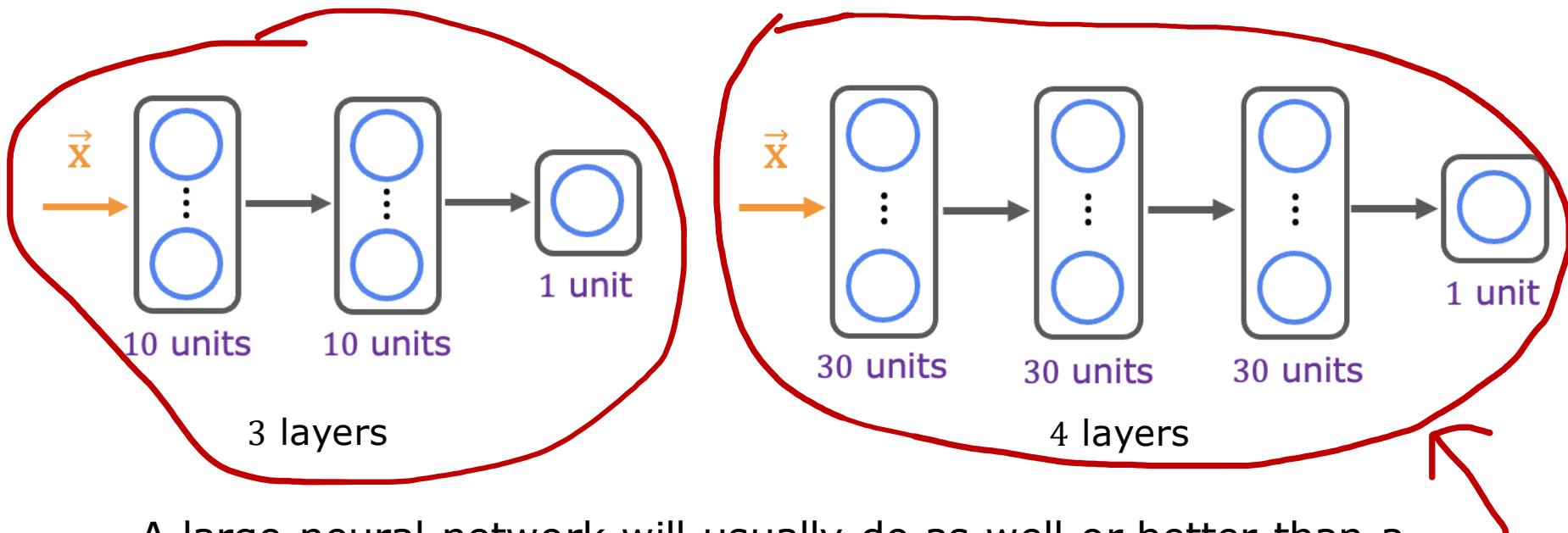


Neural networks and bias variance

Large neural networks are low bias machines



Neural networks and regularization



A large neural network will usually do as well or better than a smaller one so long as regularization is chosen appropriately.

Neural network regularization

$$J(\mathbf{W}, \mathbf{B}) = \frac{1}{m} \sum_{i=1}^m L(f(\vec{x}^{(i)}), y^{(i)}) + \frac{\lambda}{2m} \sum_{all\ weights\ \mathbf{W}} (w^2)$$

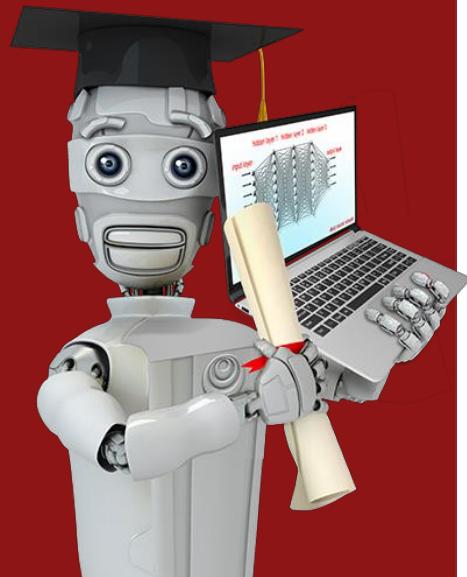
Unregularized MNIST model

```
layer_1 = Dense(units=25, activation="relu")
layer_2 = Dense(units=15, activation="relu")
layer_3 = Dense(units=1, activation="sigmoid")
model = Sequential([layer_1, layer_2, layer_3])
```

Regularized MNIST model

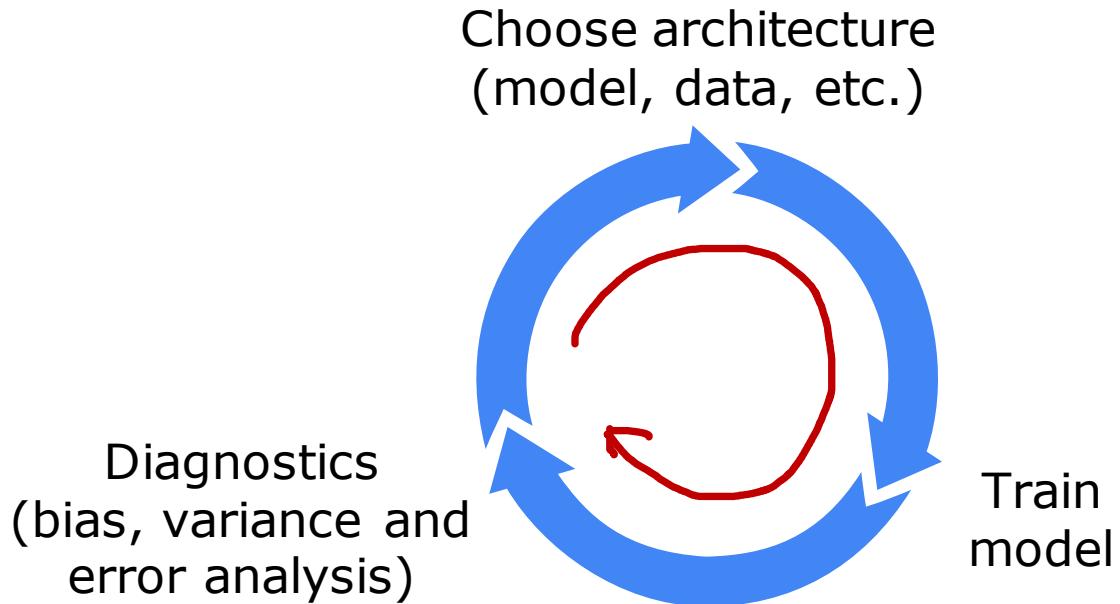
```
layer_1 = Dense(units=25, activation="relu", kernel_regularizer=L2(0.01))
layer_2 = Dense(units=15, activation="relu", kernel_regularizer=L2(0.01))
layer_3 = Dense(units=1, activation="sigmoid", kernel_regularizer=L2(0.01))
model = Sequential([layer_1, layer_2, layer_3])
```

Machine learning development process



**Iterative loop of
ML development**

Iterative loop of ML development



Spam classification example

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.

From: Alfred Ng
To: Andrew Ng
Subject: Christmas dates?

Hey Andrew,
Was talking to Mom about plans
for Xmas. When do you get off
work. Meet Dec 22?
Alf

Building a spam classifier

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

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

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

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

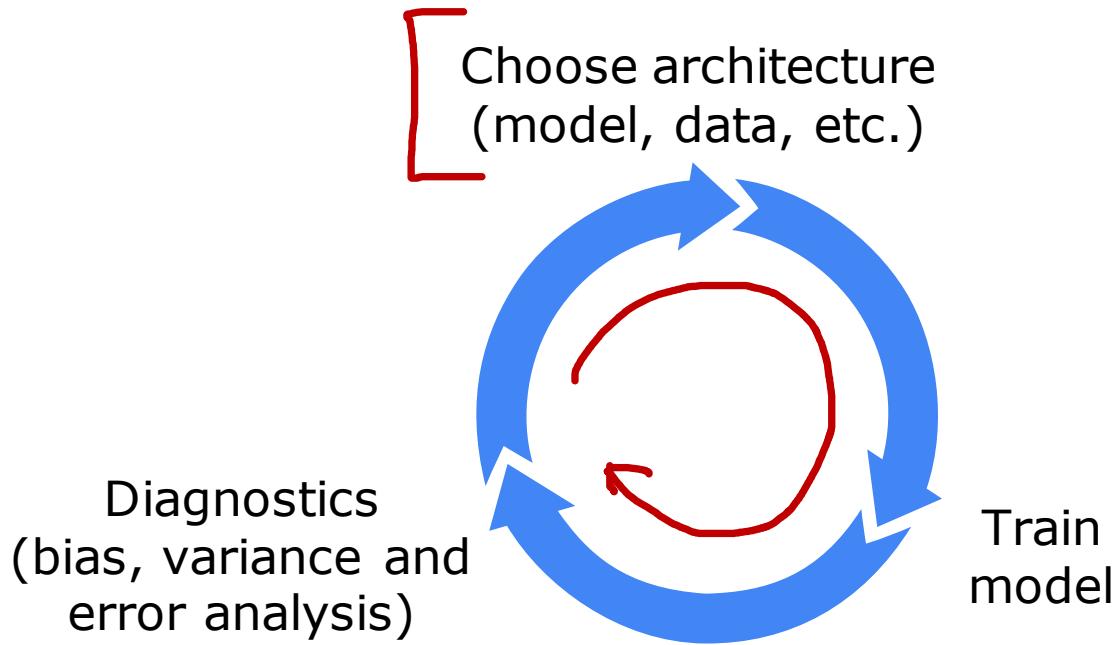
Deal of the week! Buy now!
Rolex w4tchs - \$100
Medicine (any kind) - £50
Also low cost M0rgages available.

Building a spam classifier

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

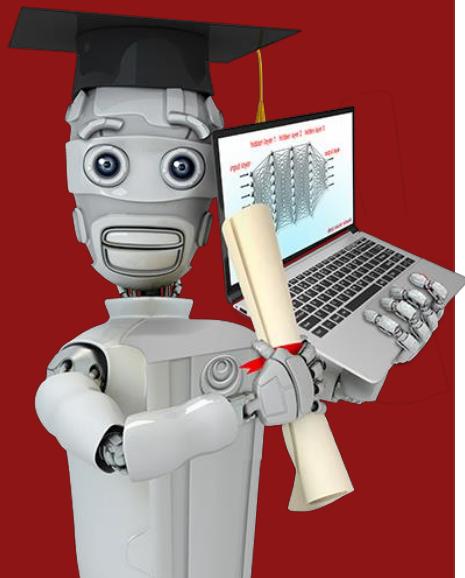
- Collect more data. E.g., “Honeypot” project.
 - Develop sophisticated features based on email routing (from email header).
 - Define sophisticated features from email body.
E.g., should “discounting” and “discount” be treated as the same word.
 - Design algorithms to detect misspellings.
E.g., w4tches,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

Iterative loop of ML development



Machine learning development process

Error analysis



Error analysis

$m_{cv} = \frac{500}{5000}$ examples in cross validation set.

Algorithm misclassifies 100 of them.

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

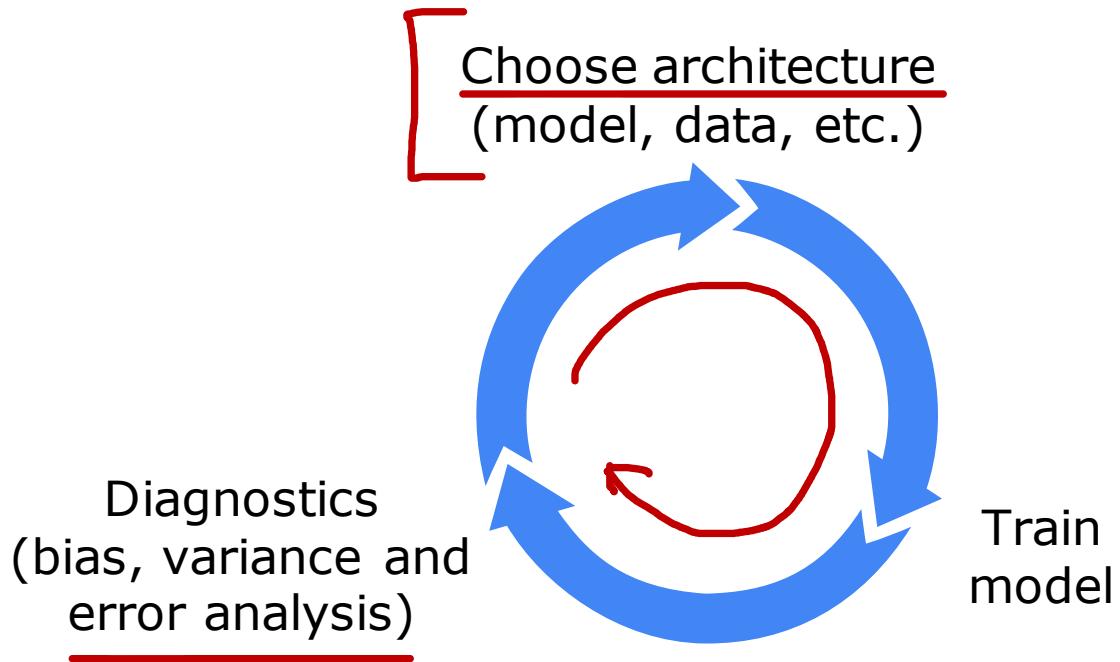
- Pharma: 21 → more data features
- Deliberate misspellings (w4tches, med1cine): 3
- Unusual email routing: 7
- Steal passwords (phishing): 18 → more data features
- Spam message in embedded image: 5

Building a spam classifier

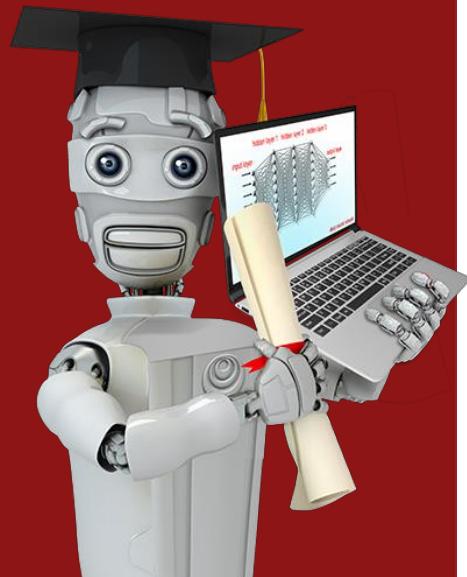
How to try to reduce your spam classifier's error?

- 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.

Iterative loop of ML development



Machine learning development process



Adding data

Adding data

- Add more data of everything. E.g., “Honeypot” project.
- Add more data of the types where error analysis has indicated it might help.

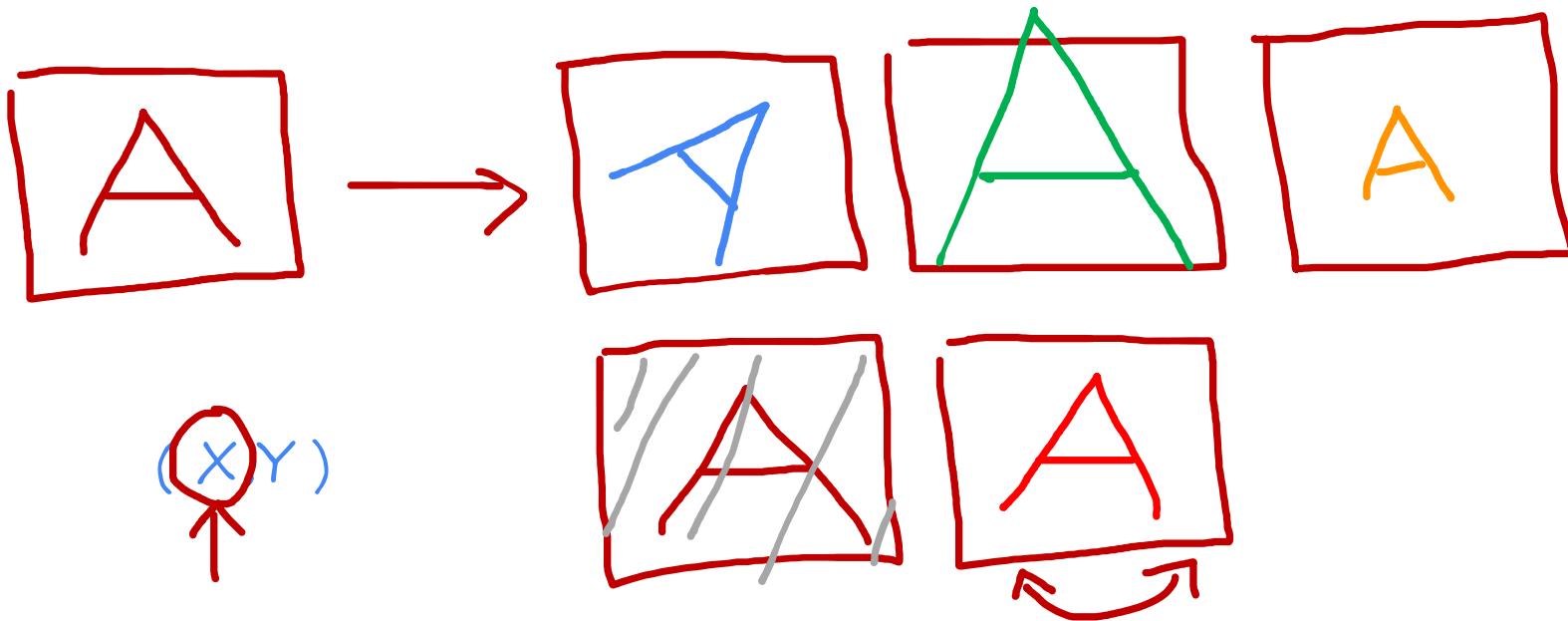
Pharma spam

E.g., Go to unlabeled data and find more examples of Pharma related spam.

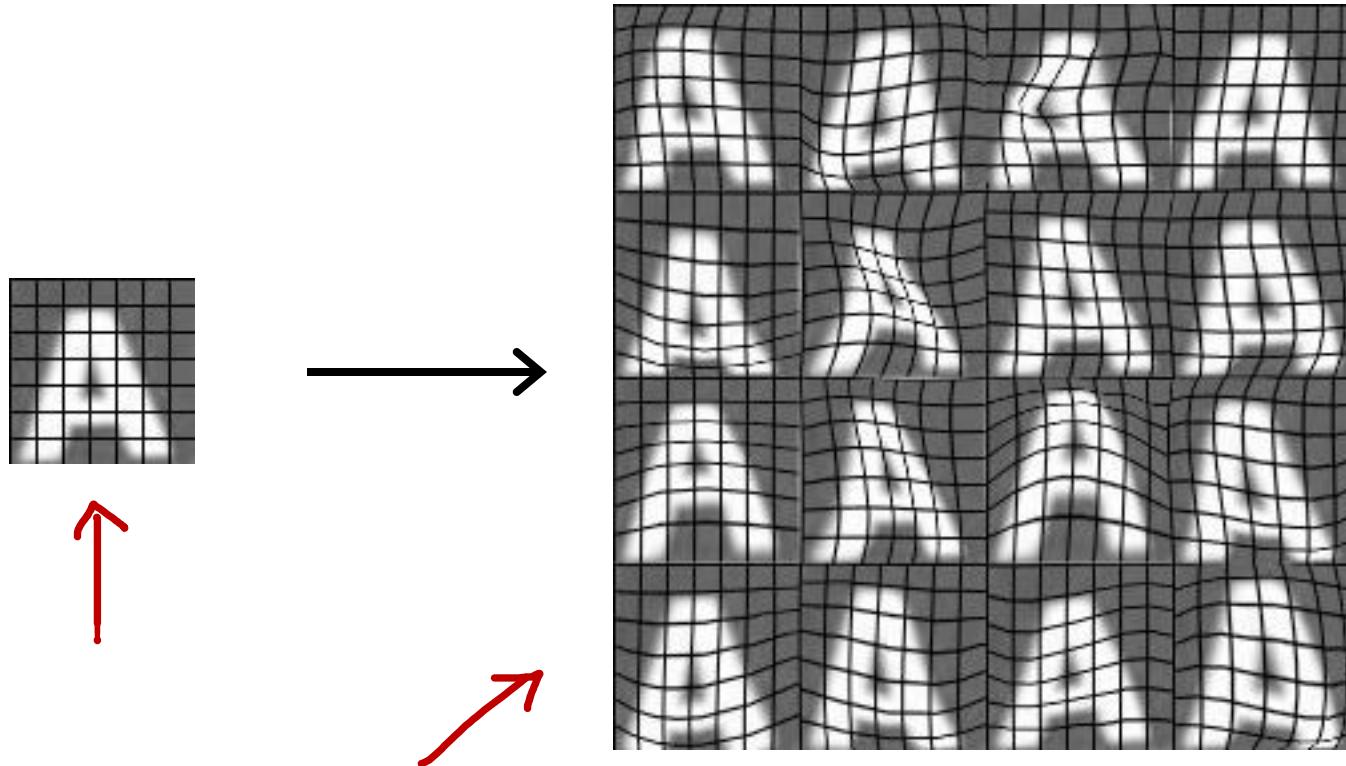
(X,Y)

Data augmentation

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

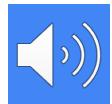


Data augmentation by introducing distortions



Data augmentation for speech

Speech recognition example



Original audio (voice search: "What is today's weather?")



+ Noisy background: Crowd



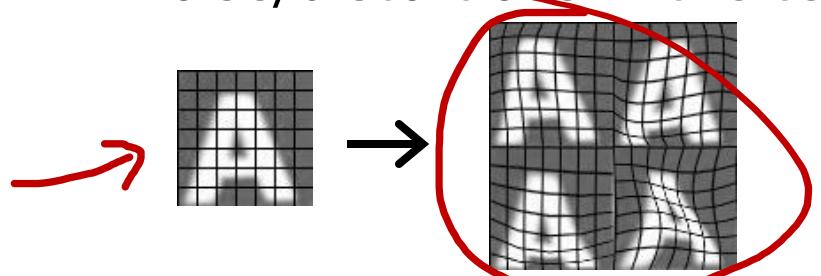
+ Noisy background: Car



+ Audio on bad cellphone connection

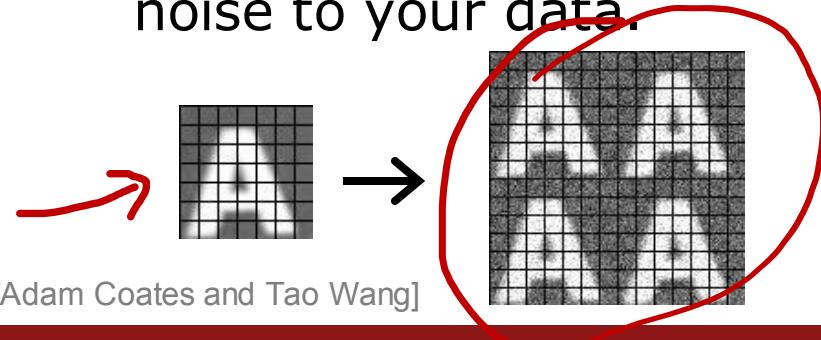
Data augmentation by introducing distortions

Distortion introduced should be representation of the type of noise/distortions in the test set.



Audio:
Background noise,
bad cellphone connection

Usually does not help to add purely random/meaningless noise to your data.



x_i =intensity (brightness) of pixel i
 $x_i \leftarrow x_i + \text{random noise}$

[Adam Coates and Tao Wang]

Data synthesis

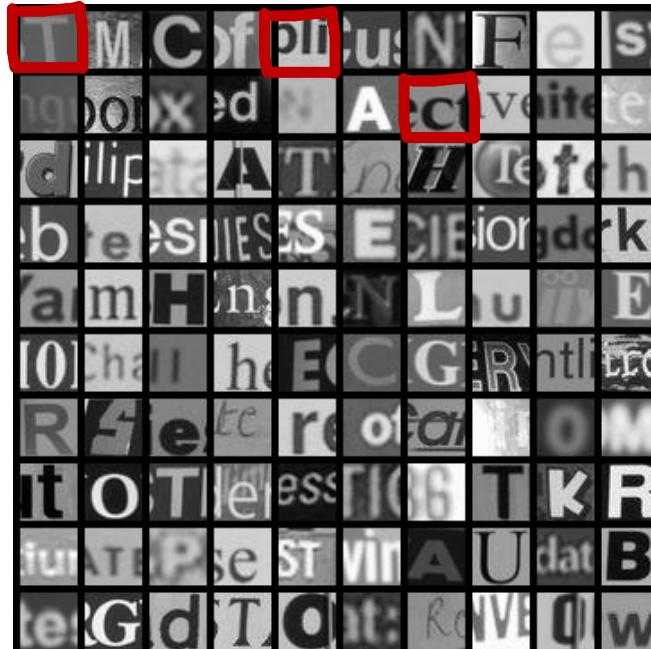
Synthesis: using artificial data inputs to create a new training example.

Artificial data synthesis for photo OCR



[<http://www.publicdomainpictures.net/view-image.php?image=5745&picture=times-square>]

Artificial data synthesis for photo OCR



Real data

[Adam Coates and Tao Wang]

Abcdefg

Abcdefg

A
bc
def
g

A
B
c
d
e
f
g

A
bc
def
g

Artificial data synthesis for photo OCR



Real data



Synthetic data

[Adam Coates and Tao Wang]

Engineering the data used by your system

Conventional
model-centric
approach:

$$\text{AI} = \text{Code} + \text{Data}$$

(algorithm/model)



Work on this

Data-centric
approach:

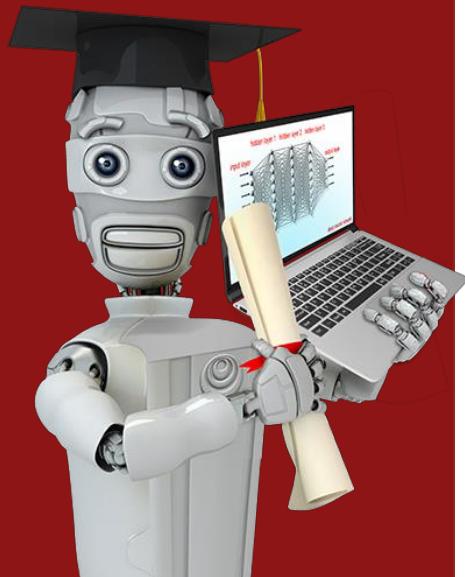
$$\text{AI} = \text{Code} + \text{Data}$$

(algorithm/model)



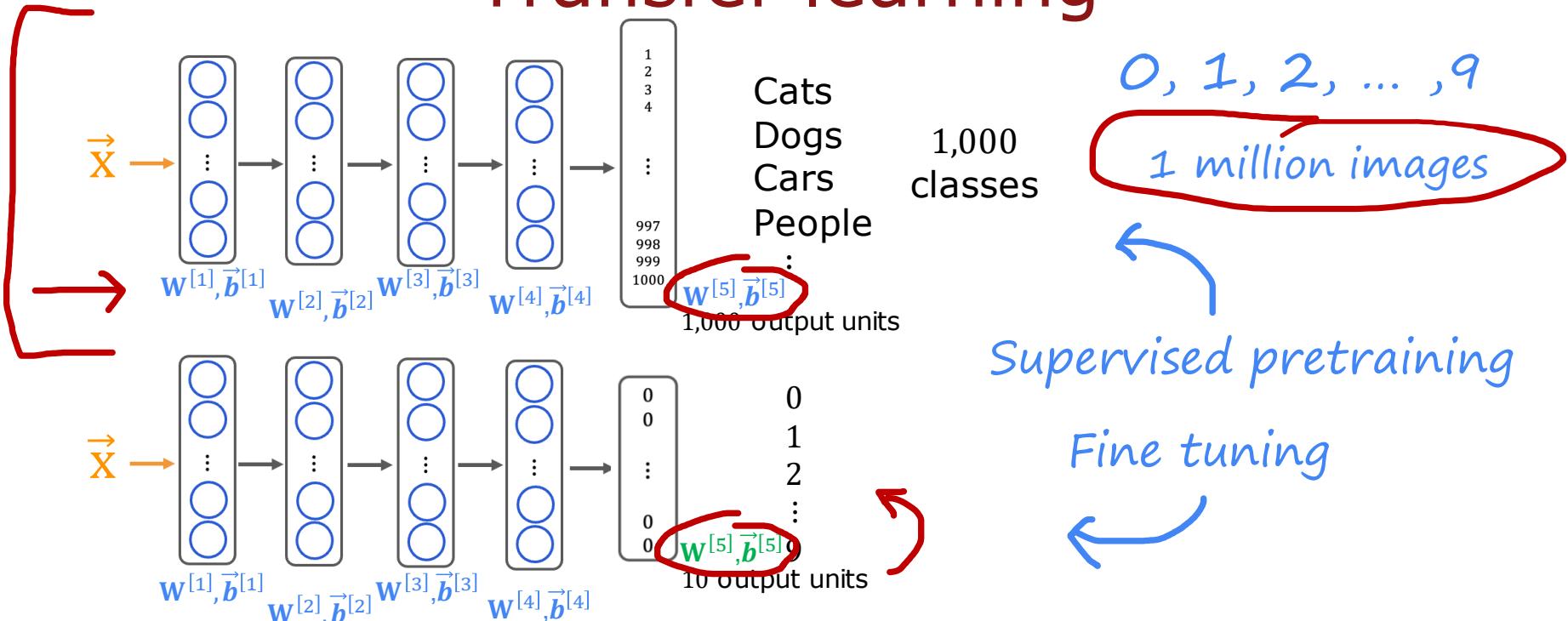
Work on this

Machine learning development process



**Transfer learning: using data
from a different task**

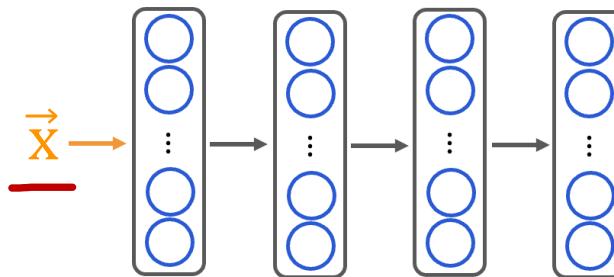
Transfer learning



Option 1: only train **output layers** parameters.

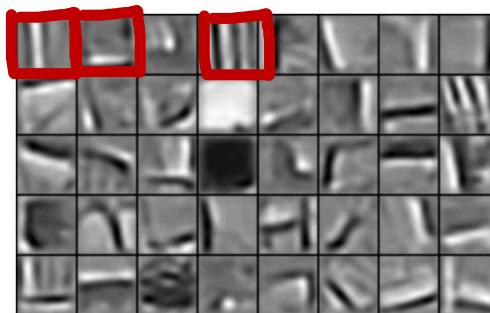
Option 2: **train all parameters**.

Why does transfer learning work?

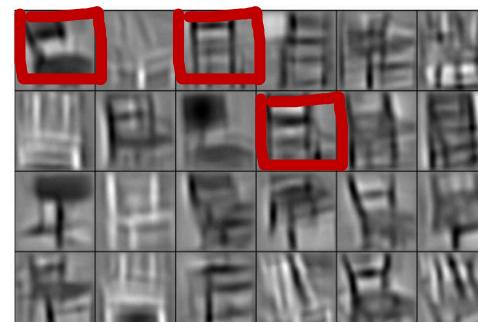


detects
edges detects
corners detects
curves/basic shapes

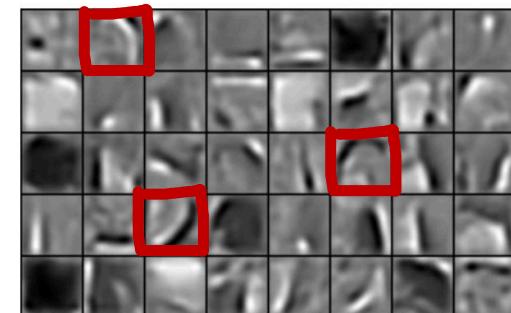
use the same input type



Edges



Corners

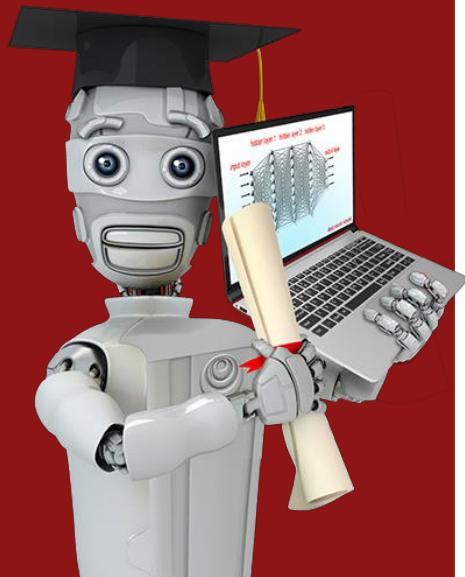


Curves / basic shapes

Transfer learning summary

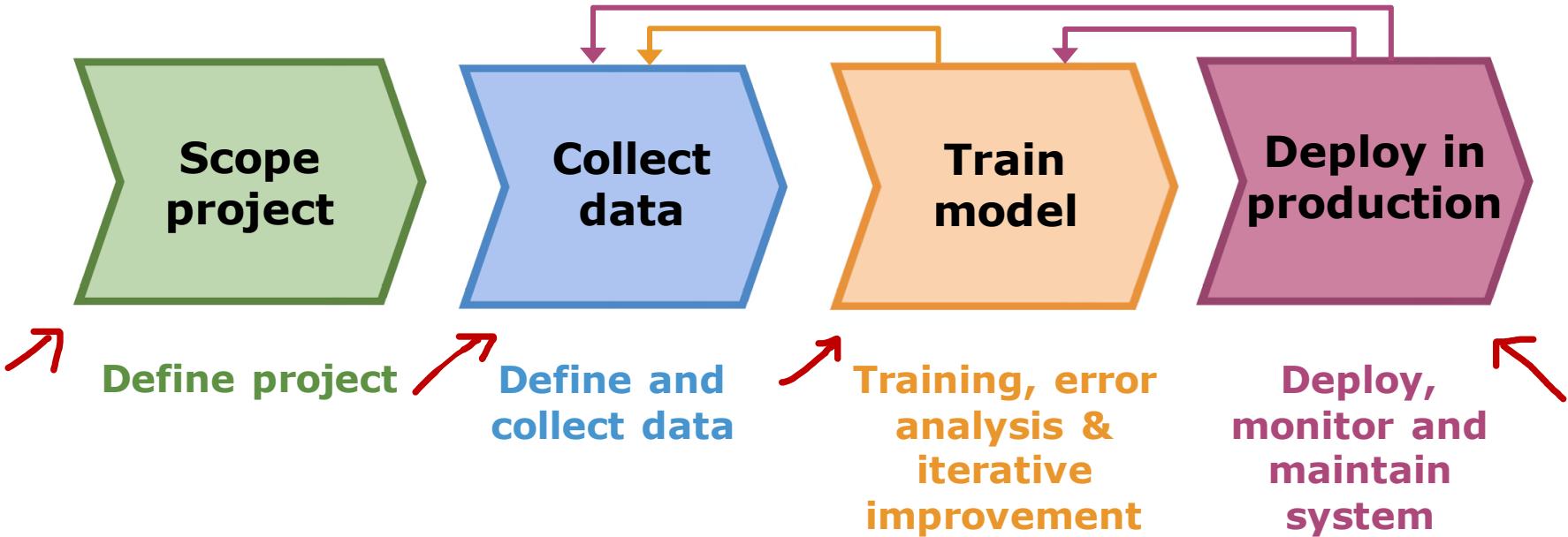
- 1. Download neural network parameters pretrained on a large dataset with same input type (e.g., images, audio, text) as your application (or train your own). *1M*
- 2. Further train (fine tune) the network on your own data. *1000*
50

Machine learning development process

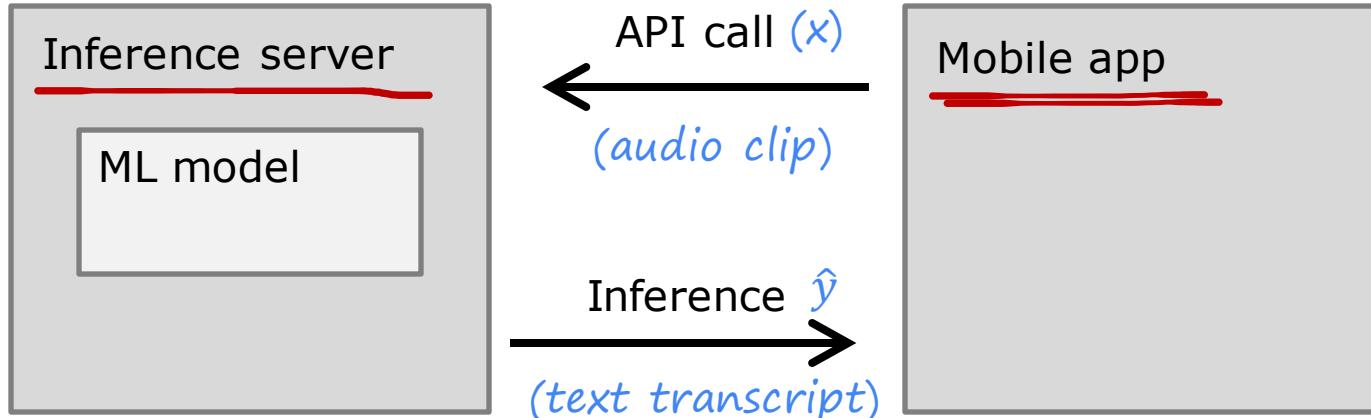


Full cycle of a
machine learning project

Full cycle of a machine learning project



Deployment



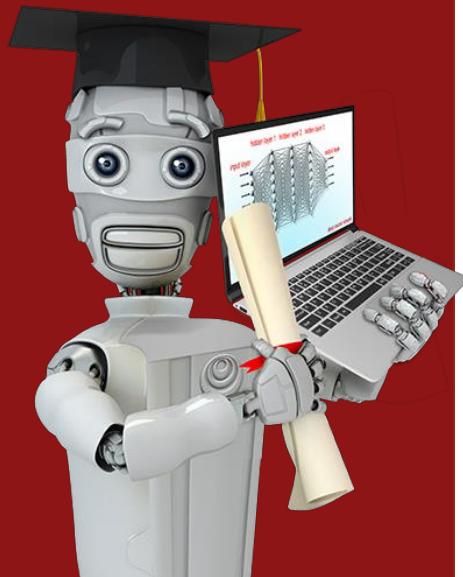
→ Software engineering may be needed for:

- Ensure reliable and efficient predictions
- Scaling
- Logging
- System monitoring
- Model updates

MLOps
machine learning
operations

Machine learning development process

Fairness, bias, and ethics



Bias

Hiring tool that discriminates against women.

Facial recognition system matching dark skinned individuals to criminal mugshots.

Biased bank loan approvals.

Toxic effect of reinforcing negative stereotypes.

Adverse use cases

Deepfakes

Spreading toxic/incendiary speech through optimizing for engagement.

Generating fake content for commercial or political purposes.

Using ML to build harmful products, commit fraud etc.

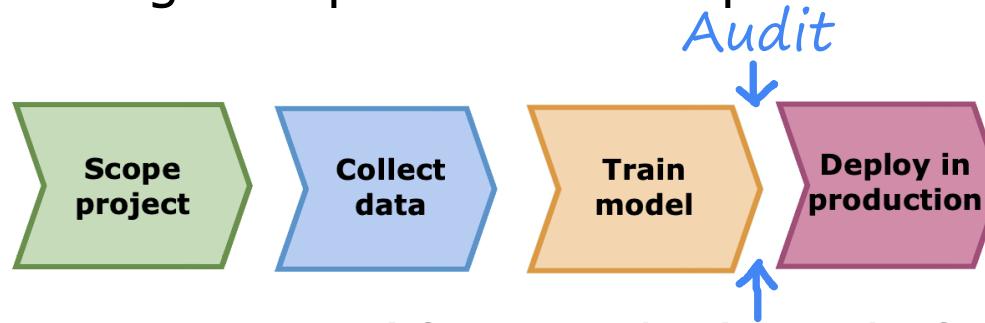
Spam vs anti-spam : fraud vs anti-fraud.

Guidelines

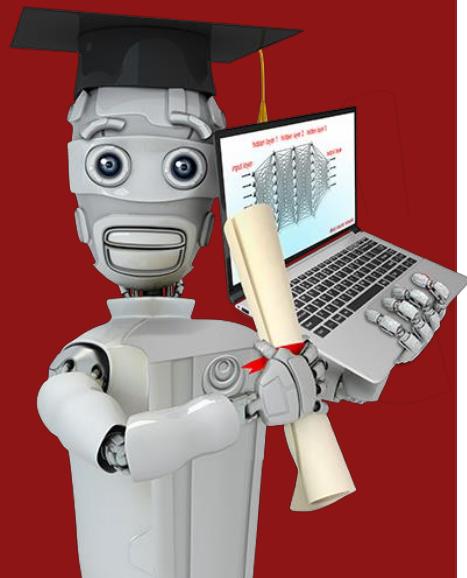
Get a diverse team to brainstorm things that might go wrong, with emphasis on possible harm to vulnerable groups.

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

Audit systems against possible harm prior to deployment.



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



Skewed datasets (optional)

Error metrics for
skewed datasets

Rare disease classification example

Train classifier $f_{\vec{w}, b}(\vec{x})$

($y = 1$ if disease present,
 $y = 0$ otherwise)

Find that you've got 1% error on test set
(99% correct diagnoses)

Only 0.5% of patients have the disease

print ("y=0")

99.5% accuracy

0.5% error

1%

1.2%

Precision/recall

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

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

`print("y=0")`

Precision:

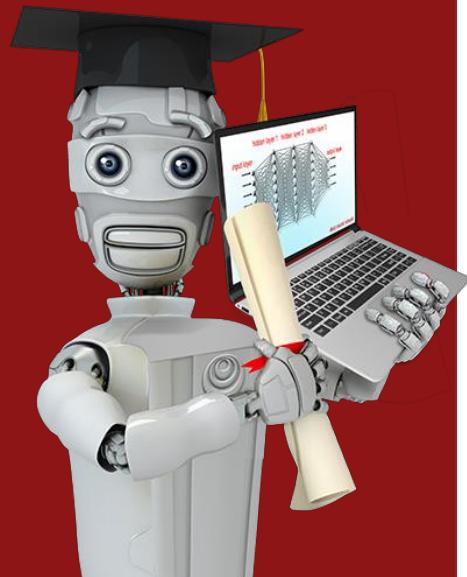
(of all patients where we predicted $y = 1$, what fraction actually have the rare disease?)

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

Recall: ←

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

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



Skewed datasets (optional)

Trading off precision
and recall

Trading off precision and recall

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

- Predict 1 if $f_{\vec{w}, b}(\vec{x}) \geq \cancel{0.5}$ ~~0.7~~ ~~0.9~~ ~~0.3~~
- Predict 0 if $f_{\vec{w}, b}(\vec{x}) < \cancel{0.5}$ ~~0.7~~ ~~0.9~~ ~~0.3~~

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

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

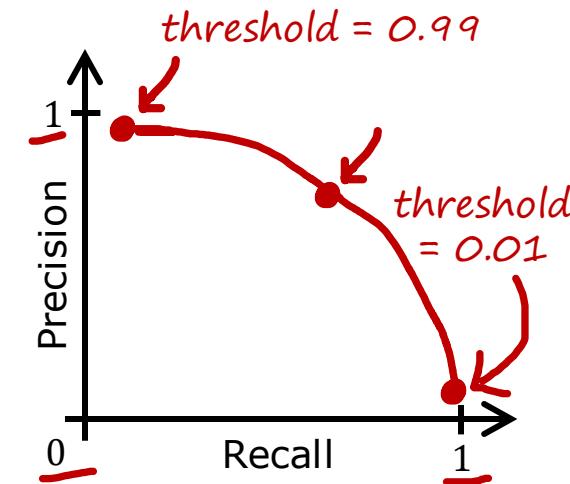
Suppose we want to predict $y = 1$ (rare disease) only if very confident.

→ higher precision, lower recall.

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

→ lower precision, higher recall.

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



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")`

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

$$F1 \text{ score} = \frac{1}{2} \left(\frac{1}{P} + \frac{1}{R} \right) = 2 \frac{PR}{P+R}$$