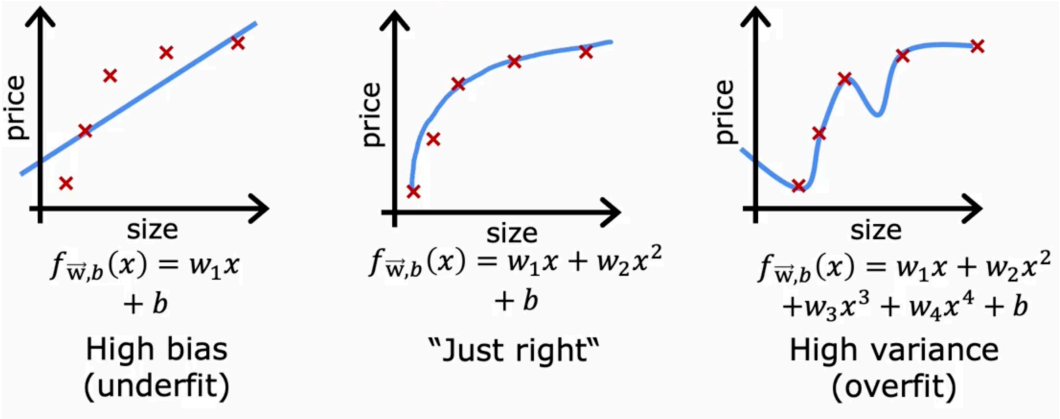
Bias and Variance

Diagnosis bias and variance

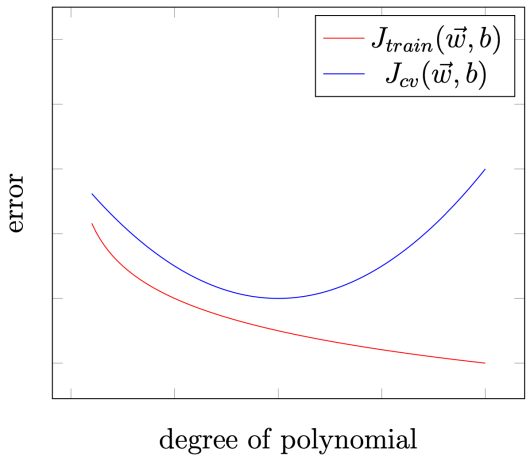
Instead of looking plots like this for determine the UNDERFITTING/OVERFITTING



A more **systematic** way to diagnose or to find out if your algorithm has high bias or high variance will be to look at the performance of your algorithm on the training set and on the cross validation set.

	High Bias (Underfit)	Just Right	High Variance (Overfit)
Degree	**d = 1**	**d = 2**	**d = 4**
J_{train}	High	Low	Low
J_{cv}	High	Low	High

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



High bias (underfit)

→ J_{train} will be high

($J_{\text{train}} \approx J_{\text{cv}}$)

High variance (overfit)

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

(J_{train} may be low)

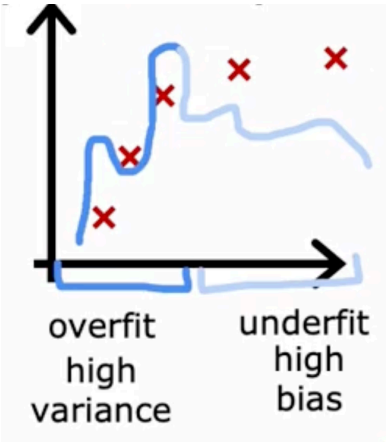
High bias and high variance

→ J_{train} will be high

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

High bias and high variance

- It doesn't really happen for linear models applied to 1D
- It would be as if for part of the input, you had a very complicated model that overfit, so it overfits to part of the inputs. But then for some reason, for other parts of the input, it doesn't even fit the training data well, and so it underfits for part of the input.

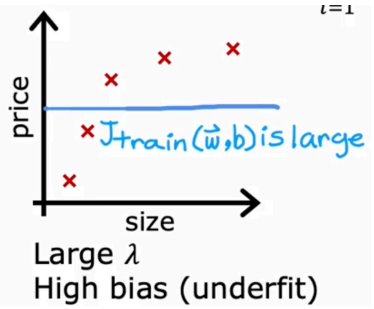


✨

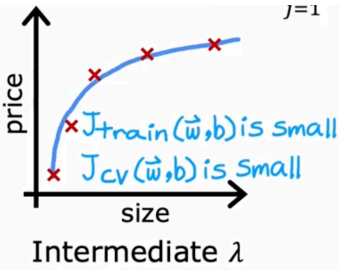
- **High bias** means poor performance on **training** data
- **High variance** means poor performance on **new** data.

Regularization and bias/variance

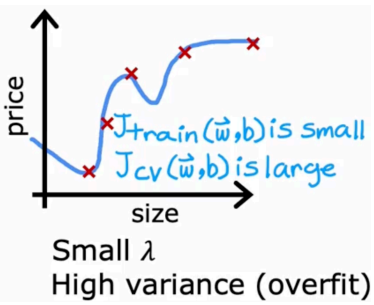
RECAP: Linear regression with regularization



Large λ
High bias (underfit)



Intermediate λ

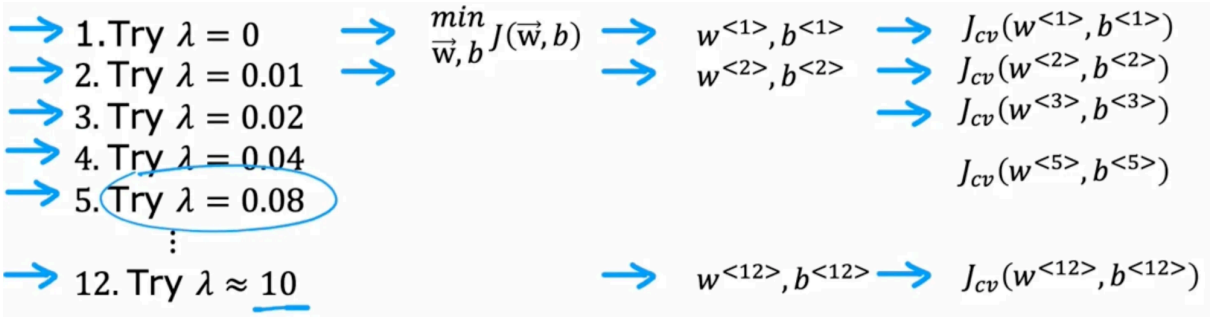


Small λ
High variance (overfit)

$\lambda \approx 10000 \Rightarrow w_1, w_2 \approx 0$ $f_{\vec{w},b}(\vec{x}) \approx b$	$\lambda = ?$	$\lambda \approx 0$

Choosing the regularization parameter λ

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



Then:

- **Pick** $w^{<5>}, b^{<5>}$
- **Report test error:** $J(w^{<5>}, b^{<5>})$

Bias and variance as a function of regularization parameter λ

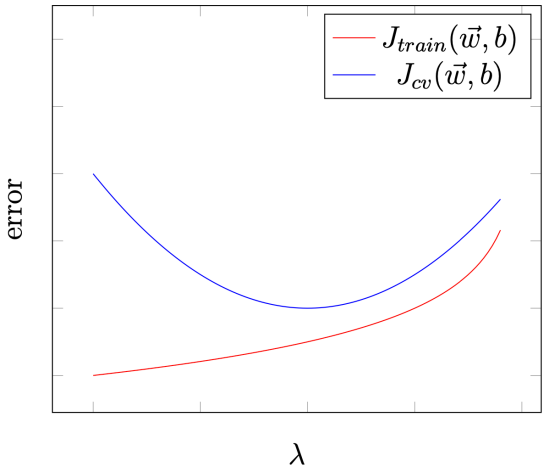
$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=1}^{m} \left(f_{\vec{w},b}\left(\vec{x}^{i}\right) - y^{i}\right)^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2$$

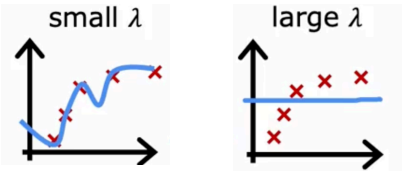
Small λ (approaching 0):

- **High variance, low bias**
- Model is complex and flexible
- Fits training data very well (low training error)
- Poor generalization to new data (high validation error)
- **Result:** Overfitting

Large λ (high values):

- **High bias, low variance**
- Model becomes too simple
- Poor fit to training data (high training error)
- Also poor performance on validation data (high validation error)
- **Result:** Underfitting





✨ **It conversely differs from 'degree of polynomial'**

Establishing a baseline level of performance

- Bias refers to the error due to overly simplistic assumptions in the learning algorithm, leading to poor performance on the training set.
- Variance refers to the error due to excessive complexity in the model, causing it to perform well on the training set but poorly on unseen data.

Why do we need a baseline error?

- Some tasks, like speech recognition, have *inherent noise or ambiguity*.
- Even humans make mistakes transcribing noisy or unclear audio.
- So, the **baseline error** (human-level error) tells you the *best possible* error you can realistically expect.
- For example, if humans make 10.6% errors on a noisy speech dataset, expecting your model to do better than that is unrealistic.

How to interpret the errors?

Scenario	Interpretation	What to check
J_{train} is close to baseline, but J_{cv} much higher	High variance (overfitting)	Model fits training data well but fails to generalize
J_{train} much higher than baseline, J_{cv} close to J_{train}	High bias (underfitting)	Model is not even fitting training data well enough
Both J_{train} and J_{cv} much higher than baseline, and J_{cv} much higher than J_{train}	Both high bias and high variance	Model is underfitting and overfitting simultaneously

Speech recognition example

- Human-level error (baseline): 10.6%
- Model training error: 10.8%
- Cross-validation error: 14.8%

Interpretation:

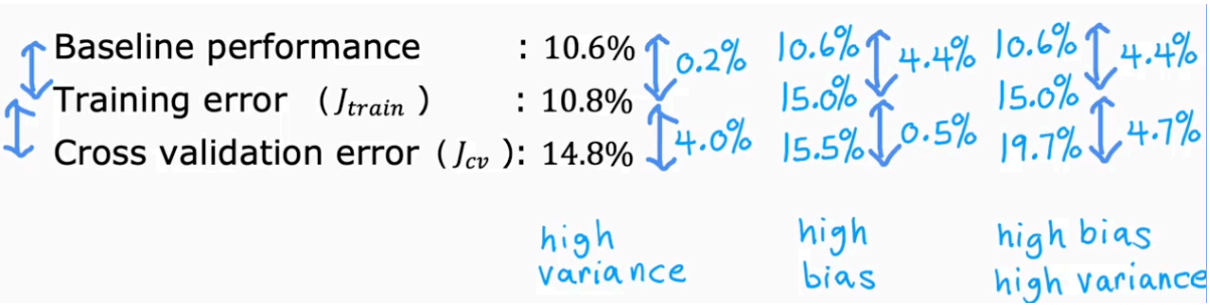
- Training error is only slightly worse than human baseline $(10.8\% - 10.6\% = 0.2\%)$ → *not high bias*
- But cross-validation error is significantly higher than training error $(14.8\% - 10.8\% = 4\%)$ → *high variance*
- So, the model overfits the training data but does not generalize well to new data.

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

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

⇒ When we've referred to **"high"** or **"low"** error up to this point, what exactly does that mean? To clearly define what counts as high or low, we need to set a baseline level of performance. Above are several ways we can establish this benchmark.

Bias/variance example



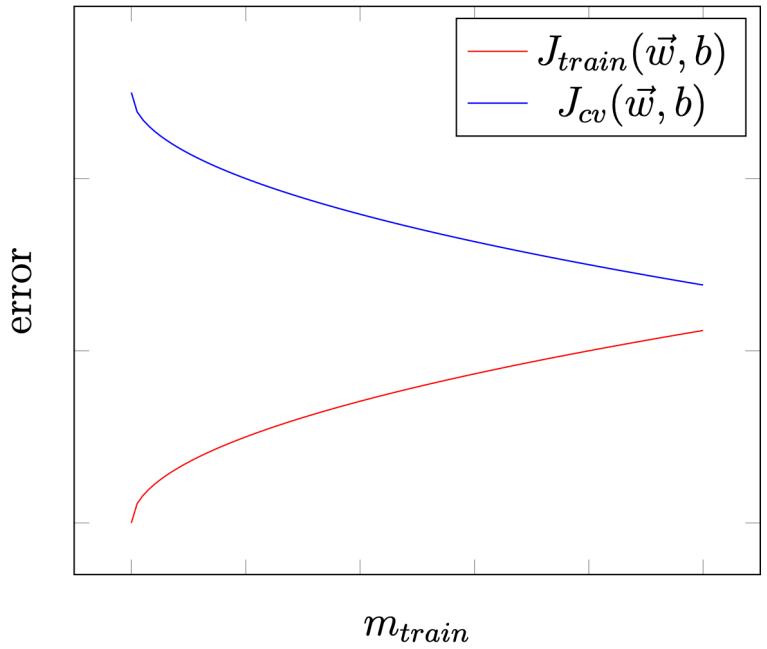
Summary of the diagnostic approach

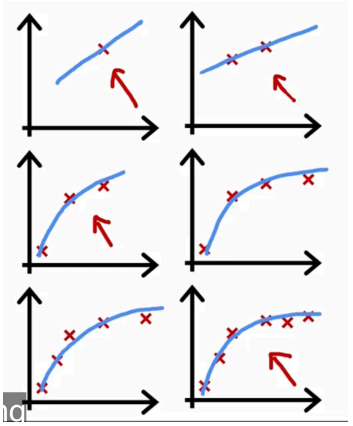
- **Bias problem:** Training error >> baseline
- **Variance problem:** Cross-validation error >> training error
- **Good fit:** Training error ≈ baseline and cross-validation error ≈ training error
- **Both problems:** Large gaps in both cases

✨ This approach is especially useful for **noisy, complex data** like speech, images, or text, where perfect accuracy is impossible.

- In simpler tasks with clean data, baseline error might be zero or near zero.
- You can also use other baselines such as previous models or competitors' results.
- The next step after diagnosing is to adjust your model accordingly:
  - For **high bias**, increase model complexity or reduce regularization.
  - For **high variance**, increase regularization, get more data, or simplify the model.

Learning curve



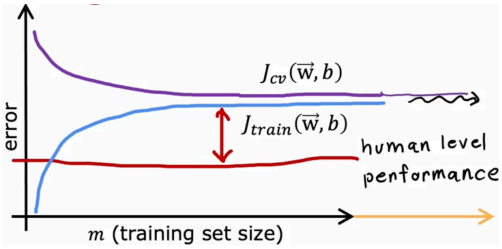


The training set **gets bigger**, the training error **increases** because it's **harder** to **fit** all of the training examples **perfectly**.

High Bias (Underfitting)

- The model is too simple to capture the underlying trend in the data (e.g., fitting a straight line to data that's actually curved).

Learning Curves



- Both training error and validation error are high.
- Both curves flatten out (plateau) at a high error.
- Gap between training and validation error is small.
- Human-level performance is much better than the model.

❓ **What happens if you add more data?**

⇒ **Not much improvement.** The model is too simple, so more data doesn't help.

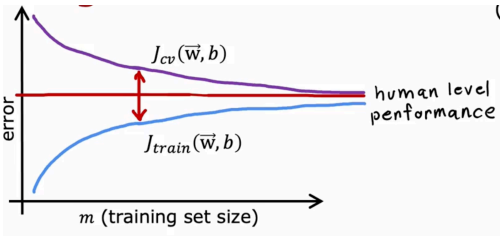
💡 **What should you do?**

- Use a **more complex model** (e.g., add more features, use a higher-order polynomial).
- Try to **reduce bias**.

High Variance (Overfitting)

- The model is too complex and fits the training data too closely, including noise (e.g., a wiggly curve that passes through every point).

Learning Curves (see first image)



- **Training error is low** (model fits training data very well).
- **Validation error is much higher** (model does poorly on new data).
- **Big gap between training and validation error.**
- **Human-level performance is closer to validation error.**

❓ **What happens if you add more data?**

⇒ **Validation error decreases.** The model starts to generalize better.

⇒ **Training error increases slightly** (harder to fit all points perfectly).

⇒ **Gap between training and validation error gets smaller.**

💡 **What should you do?**

- **Get more training data** (this helps the model generalize).
- Use **regularization** (to penalize overly complex models).

❓ NOTE: If a learning algorithm has high bias, adding more training data won't significantly improve its performance—the improvement will level off after a certain point. In contrast, if the algorithm has high variance, providing more training data is likely to enhance its performance.

Deciding what to try next revisited

Debugging a learning algorithm

You've implemented **regularized linear regression** on housing prices:

$$J(\vec{w}, b) = \frac{1}{2m} \sum_{i=0}^{m-1} (f_{\vec{w},b}(\vec{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=0}^{n-1} w_j^2$$

❓ 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 (simplify model) ⇒ **fixes high variance**
- Try getting additional features ⇒ **fixes high bias**
- Try adding polynomial features (x_1^2, x_2^2, x_1x_2, etc) ⇒ **fixes high bias**
- Try decreasing λ ⇒ **fixes high bias**
- Try increasing λ ⇒ **fixes high variance**

Bias/variance and neural network

Bias-Variance Tradeoff

Linear model (1st degree polynomial):

$f_{\vec{w},b}(x) = w_1x + b$

Simple model ⇒ **High Bias**

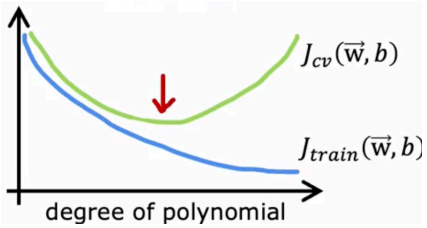
High-degree polynomial (e.g, 4th degree):

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

Complex model ⇒ **High Variance**

Therefore, our **INITIAL TRADEOFF** is using a better model (having degree bigger than 1 and lower than 4) like this:

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



At the point that $J_{cv}(\vec{w}, b)$ is **smallest**

- High bias leads to underfitting, while high variance results in overfitting, both negatively impacting model performance.
- Neural networks can help mitigate these issues, allowing for better model fitting without the need for strict tradeoffs.

Neural Networks and Bias/Variance

KEY-NOTE: Large neural networks are low-bias machines.

Debugging High Bias/Variance With Neural Network

1. Train the model on the training set.
2. Check Performance by J_{train}, if **HIGH** (high bias):
    a. Bigger network (more layer or units).
    b. Go to 1.
3. Check Performance by J_{cv}, if **HIGH** (high variance): (⇒ that mean J_{train} is good)
    a. Acquiring more data
    b. Go to 1.
4. If Performance is **good** ⇒ **DONE !!**

Bigger network helps fixing High Bias
If you have a dataset that follows a complex pattern (like a non-linear relationship), a small network may not be able to learn this pattern effectively, resulting in high bias. By increasing the size of the network, you allow it to learn the intricate details of the data, thus reducing bias.

What if my neural network is too big? Will that create a high variance problem?

When building neural networks, you might worry that making your network bigger (adding more layers or units) will cause it to overfit—meaning it will perform well on training data but poorly on new, unseen data. This is a common concern because bigger models can "**memorize**" the training data instead of learning to **generalize**.

⇒ If you use a technique called **regularization**, you can control this overfitting. Regularization works by adding a penalty to the loss function for large weights, which discourages the network from relying too heavily on any one feature or connection.

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

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_unreg = 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_reg = Sequential([layer_1, layer_2, layer_3])
```

KEY: using `kernel_regularizer=L2(0.01)` ($\lambda : 0.01$)

NOTE: A larger neural network can **slow down** your algorithm. But it shouldn't hurt your algorithm's **performance**.

So long as your training set **isn't** too **large, large neural networks** typically have **low bias** because they can model complex functions well. However, this often leads to **high variance**, making **overfitting** a common issue. As a result, when working with deep learning, practitioners usually focus more on **managing variance** than reducing bias. Despite this, analyzing **bias and variance** remains a useful strategy to guide model improvements.

❓ **QUES:** If the model's cross validation error J_{cv} is much higher than the training error J_{train}, this is an indication that the model has...

- High bias
- High variance 👍
- Low variance
- Low bias

Explain: When $J_{cv} >> J_{train}$ (whether J_{train} is also high or not, this is a sign that the model is overfitting to the training data and performing much worse on new examples).

❓ **QUES:** Which of these is the best way to determine whether your model has high bias (has underfit the training data)?

- See if the cross validation error is high compared to the baseline level of performance
- Compare the training error to the baseline level of performance 👍
- See if the training error is high (above 15% or so)
- Compare the training error to the cross validation error.

Explain: Correct. If comparing your model's training error to a baseline level of performance (such as human level performance, or performance of other well-established models), if your model's training error is much higher, then this is a sign that the model has high bias (has underfit).

❓ **QUES:** You find that your algorithm has high bias. Which of these seem like good options for improving the algorithm's performance? Hint: two of these are correct.

- [x] ~~Decrease the regularization parameter λ~~
- [ ] Remove examples from the training set
- [x] ~~Collect additional features or add polynomial features~~
- [ ] Collect more training examples

❓ **QUES:** You find that your algorithm has a training error of 2%, and a cross validation error of 20% (much higher than the training error). Based on the conclusion you would draw about whether the algorithm has a high bias or high variance problem, which of these seem like good options for improving the algorithm's performance? Hint: two of these are correct.

- [ ] Decrease the regularization parameter λλlambda
- [ ] Reduce the training set size
- [x] ~~Increase the regularization parameter λλlambda~~
- [x] ~~Collect more training data~~