

Bias and Variance

High bias (under-fitting) means the algorithm doesn't fit the training data well, while high variance (overfitting) means the algorithm fits the training data too well but performs poorly on new data.

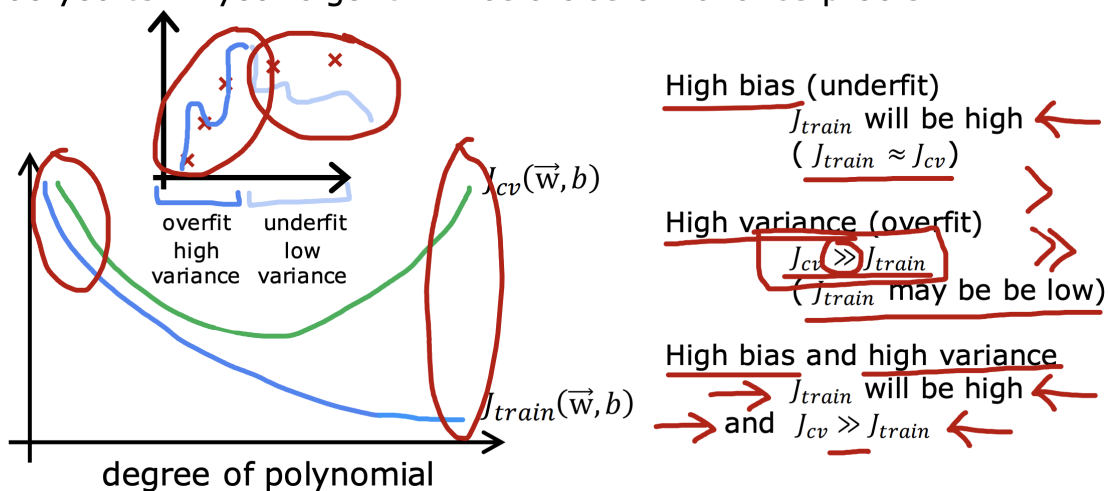
The performance of the algorithm on the training set and the cross-validation set can help diagnose if the algorithm has high bias or high variance.

Summary

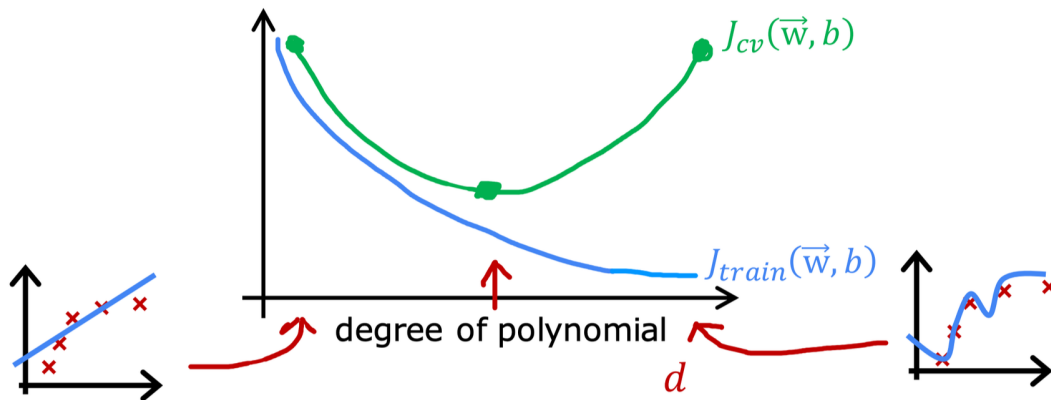
	High Bias	High Variance
Degree of Polynomial	Low (Underfitting)	High (Overfitting)
Regularization Parameter (Lambda)	High	Low
Number of Training Examples (m)	Less Sensitive	More Examples Help

Diagnosing bias and variance

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

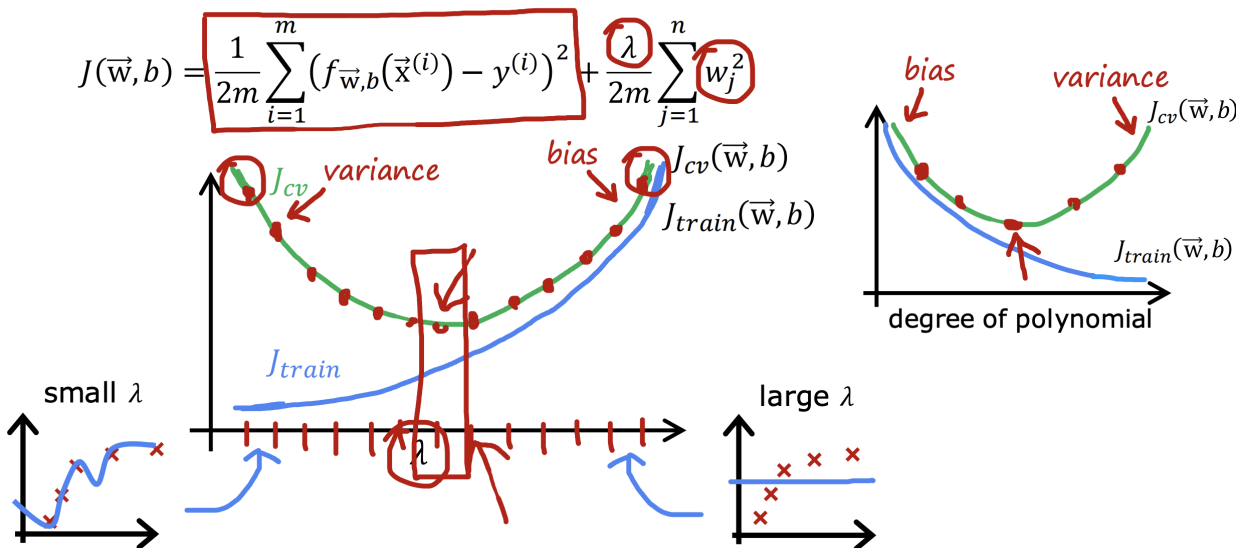


The degree of the polynomial fitted to the data can affect the bias and variance. A low degree can lead to under-fitting (high bias), a high degree can lead to overfitting (high variance), and an intermediate degree can provide a good fit (low bias and low variance).



Regularisation and bias/variance

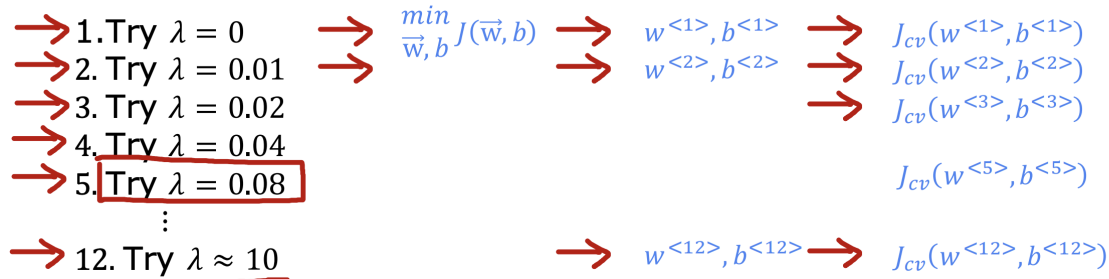
Bias and variance as a function of regularization parameter λ



A large value of Lambda can lead to underfitting (high bias), while a small value can lead to overfitting (high variance). An intermediate value of Lambda can provide a good fit (low bias and low variance).

Choosing the regularization parameter λ

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



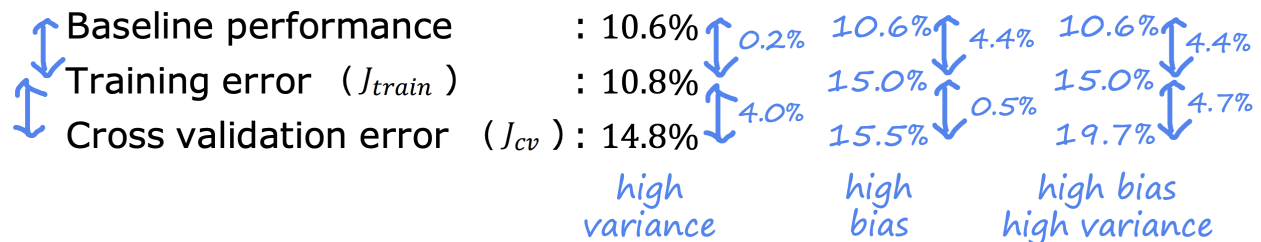
Pick $w^{<5>}, b^{<5>}$

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

To choose a good value for Lambda, you can use cross-validation. This involves trying out a range of possible values for Lambda, fitting parameters using those different regularisation parameters, and then evaluating the performance on the cross-validation set. The value of Lambda that results in the lowest cross-validation error can be chosen as the best value.

Baseline level of Performance

Bias/variance examples



In the example given, the training error was 10.8% and the cross-validation error was 14.8%. Initially, it might seem like the algorithm has a high bias problem because the training error is high. However, when compared to the human level error of 10.6%, the training error is only slightly worse, suggesting that the algorithm is doing quite well on the training set.

The difference between the training error and the baseline level of performance can indicate a high bias problem, while the difference between the cross-validation error and the training error can indicate a high variance problem.

In some cases, an algorithm can have both high bias and high variance. This can be identified if both the gap between the baseline and the training error, and the gap between the training error and cross-validation error are large.



Reasonably choices for Baseline level

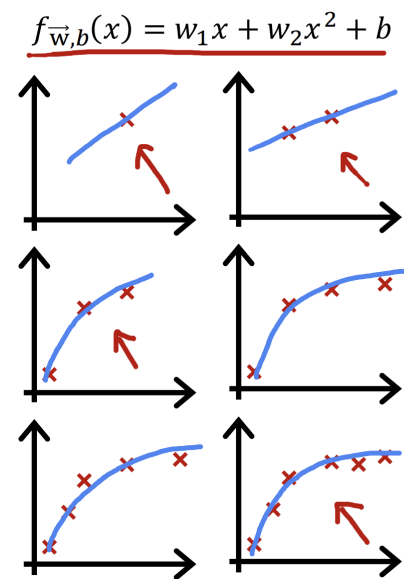
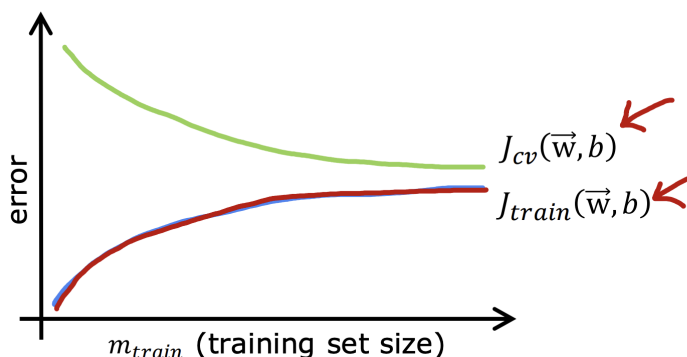
- Human level performance
- Competing algorithms performance
- Guess based on experience

Learning curves are graphs that plot the performance of a model on both the training and validation datasets over the number of training instances.

Learning curves

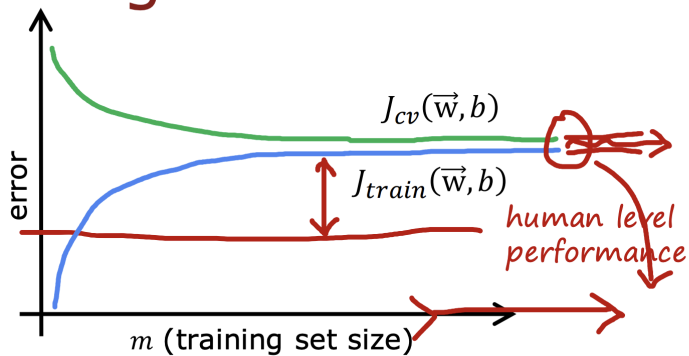
J_{train} = training error

J_{cv} = cross validation error



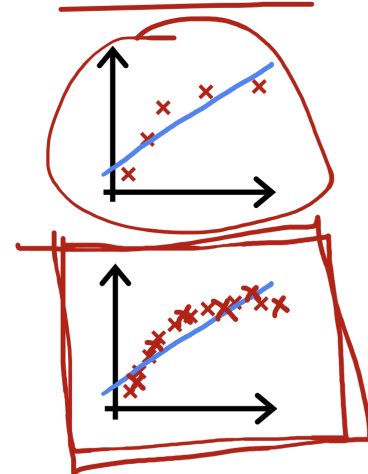
As the number of training examples (m_{train}) increases, the cross-validation error decreases, indicating that the model is learning better. However, the training error increases as the number of training examples increases. This is because it becomes harder for the model to fit all the training examples perfectly as the number of examples increases.

High bias

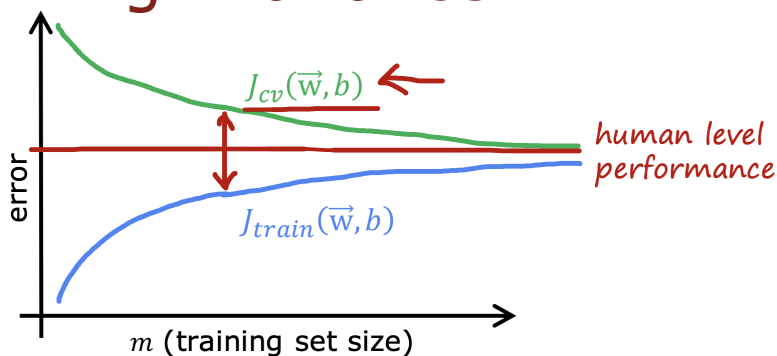


if a learning algorithm suffers from high bias, getting more training data will not (by itself) help much.

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



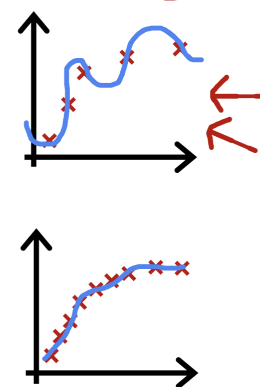
High variance



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

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

(with small λ)

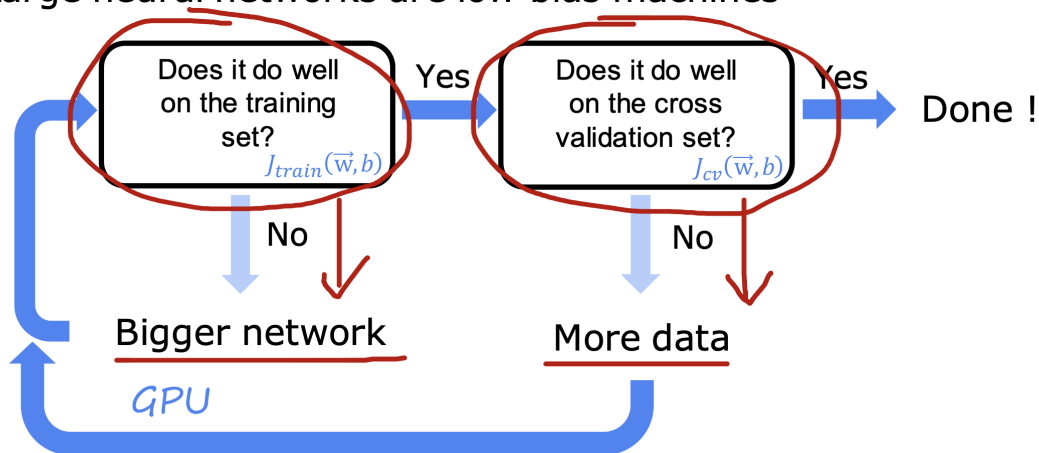


Debugging a learning algorithm

To Do	Affect	Reason
Getting more training examples	Fixes high variance	Reduces the algorithm's tendency to overfit to a small training set
Trying a smaller set of features	Fixes high variance	Reduces the complexity of the model and prevents overfitting
Getting additional features	Fixes high bias	Allows the algorithm to better fit the training data
Adding polynomial features	Fixes high bias	Allows the algorithm to fit more complex patterns in the data
Decreasing the regularisation parameter (Lambda)	Fixes high bias	Allows the algorithm to pay more attention to fitting the training data
Increasing the regularisation parameter (Lambda)	Fixes high variance	Forces the algorithm to fit a smoother, less complex function

Neural networks and bias variance

Large neural networks are low bias machines



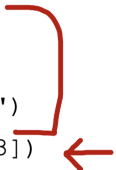
Note :- A large neural network will usually do as well or better than a smaller one so long as regularisation is chosen appropriately.

Neural network regularization

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

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])
```

