**1.**                                                              1 / 1 punto

# Diagnosing bias and variance

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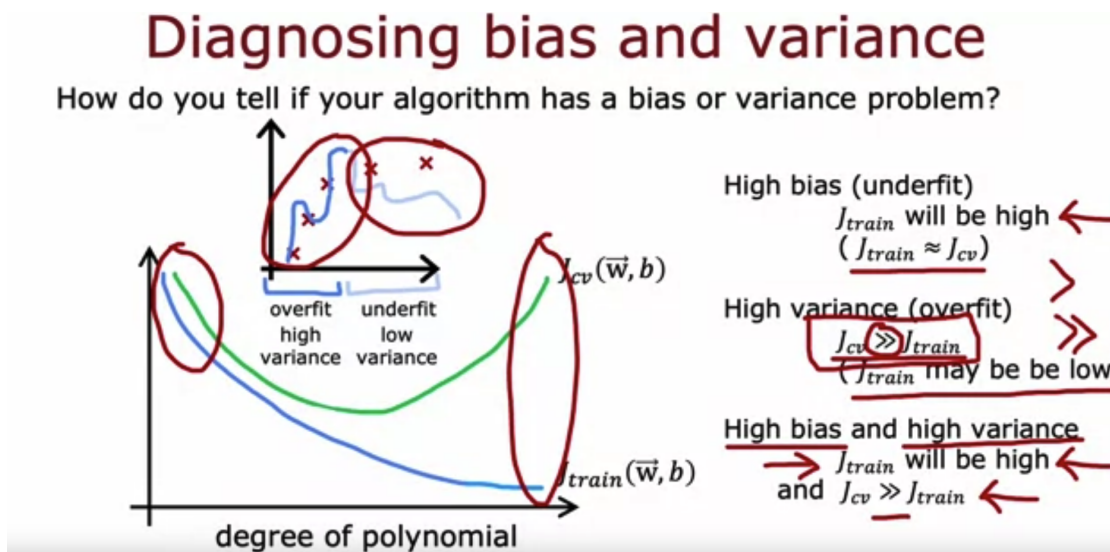


If the model's cross validation error $J_{cv}$ is much higher than the training error $J_{train}$, this is an indication that the model has…

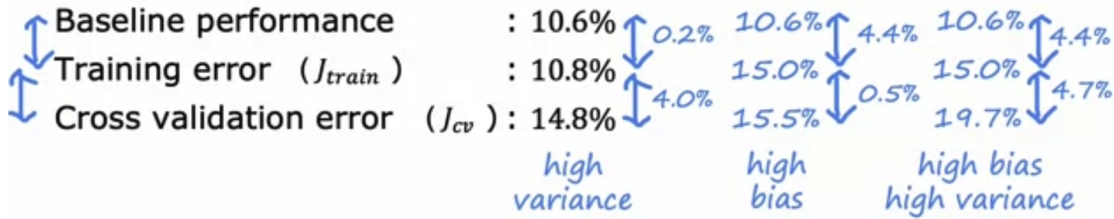- ⦿ high variance
- ○ high bias
- ○ Low bias
- ○ Low variance

> ✓ **Correcto**
>
> When $J_{cv} \gg J_{train}$ (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.

**2.**                                                              1 / 1 punto

## Bias/variance examples

Baseline performance       : 10.6%  $\uparrow$ 0.2%   10.6% $\uparrow$ 4.4%   10.6% $\uparrow$ 4.4%

Training error  ($J_{train}$)   : 10.8%         15.0%          15.0%

Cross validation error  ($J_{cv}$) : 14.8% $\downarrow$ 4.0%   15.5% $\downarrow$ 0.5%   19.7% $\downarrow$ 4.7%

                                      *high*         *high*        *high bias*

                                 *variance*     *bias*    *high variance*

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

○ Compare the training error to the cross validation error.

○ See if the training error is high (above 15% or so)

○ 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

> ✓ **Correcto**
>
> 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).

**3.**                                                              **1 / 1 punto**

## 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                   fixes high variance
→ Try smaller sets of features  $x, x^2, x^3, x^4, \ldots$    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 $\lambda$ ←                           fixes high bias
→ Try increasing $\lambda$ ←                           fixes high variance

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.

☐ Collect more training examples

☑ Decrease the regularization parameter $\lambda$ (lambda)

> ✓ **Correcto**
> Correct. Decreasing regularization can help the model better fit the training data.

☐ Remove examples from the training set

☑ Collect additional features or add polynomial features

> ✓ **Correcto**
> Correct. More features could potentially help the model better fit the training examples.

**4.**

**1 / 1 punto**

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.

☑ Increase the regularization parameter $\lambda$

> ✓ **Correcto**
> Yes, the model appears to have high variance (overfit), and increasing regularization would help reduce high variance.

☑ Collect more training data

> ✓ **Correcto**
> Yes, the model appears to have high variance (overfit), and collecting more training examples would help reduce high variance.

☐ Decrease the regularization parameter $\lambda$

☐ Reduce the training set size