When there is a large error in its predictions, you should try:

① Get more training examples.
② Try smaller sets of features
③ Try getting additional features
④ Try getting polynomial features (x_1^2, x_2^2, x_1x_2)
⑤ Try decreasing / increasing λ.

{ we shouldn't randomly choose them to implement.

"machine learning diagnostic".

Evaluate a Hypothesis → overfitting.

split training set into trainging set and test ~~(validation)~~ set, (70% : 30%).

- Learning parameter θ from training data (minimize $J(\theta)$)
- Compute test error.
  - linear regression: $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (h_\theta(x_{test}^{(i)}) - y_{test}^{(i)})^2$
  - logistic regression (classification): $J_{test}(\theta) = -\frac{1}{m_{test}} \sum y_{test}^{(i)} \log(h_\theta(x_{test}^{(i)})) + (1-y_{test}^{(i)}) \log(1-h_\theta(x_{test}^{(i)}))$
- misclassification error (0/1 missclassification error).

$$\text{error}(h_\theta(x), y) = \begin{cases} 1, & \text{if } h_\theta(x) \geq 0.5 \text{ and } y=0 \text{ or if } h_\theta(x) \leq 0.5 \text{ and } y=1 \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{Test error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \text{error}(h_\theta(x_{test}^{(i)}), y_{test}^{(i)})$$

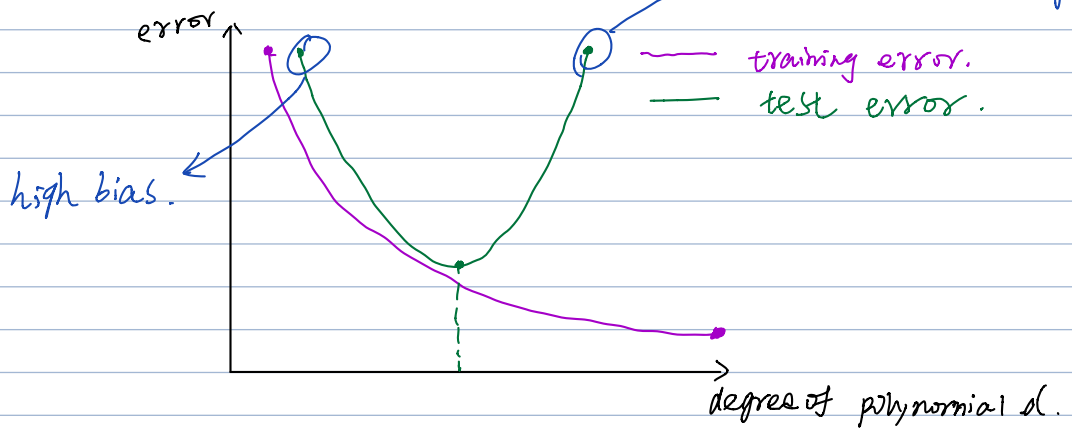
Model selection:

split training sets into training, validation, test sets.

① optimize the parameters in θ using the training set for each polynomial degree
② Find the polynomial degree d with least error using the cross validation set.
③ Estimate the generalization error using the test set with $J_{test}(\theta^{(d)})$

Bias and variance.

High bias → underfit
High variance → overfit.



Bias problem (underfit):
- $J_{train}(\theta)$ will be high
- $J_{cv}(\theta) \approx J_{train}(\theta)$

Variance (overfit):
- $J_{train}(\theta)$ will be low
- $J_{cv}(\theta) >> J_{train}(\theta)$

Regularization & bias/variance.

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2$$

large λ → underfitting → high bias.

small λ → overfitting → high variance.

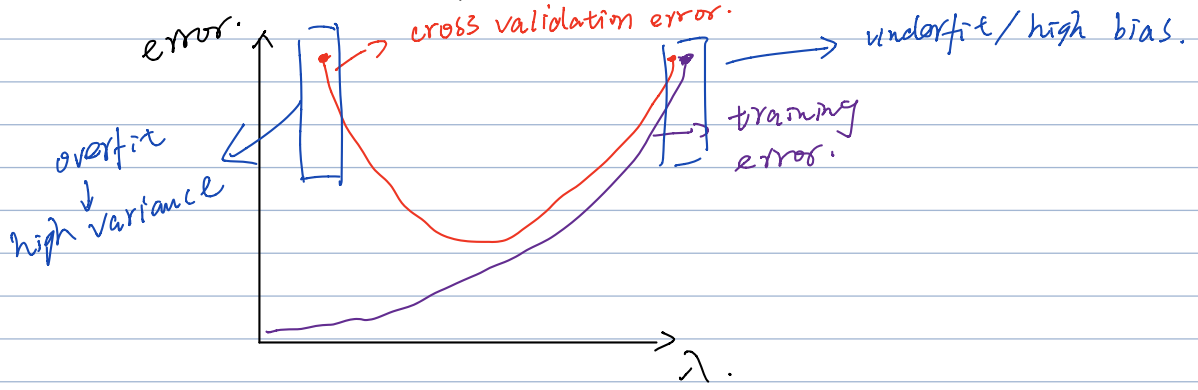
define $J_{train}(\theta)$ without regularization

training error 不包含正則化.

$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m_{train}}(h_\theta(x_{train}^{(i)}) - y_{train}^{(i)})^2 = J_{cv}(\theta) = J_{test}(\theta).$$

1. $\lambda = 0$ → $\min J(\theta) \to \theta^{(1)} \to J_{cv}(\theta^{(1)})$
2. $\lambda = 0.01$ → $\min J(\theta) \to \theta^{(2)} \to J_{cv}(\theta^{(2)})$
3. $\lambda = 0.02$ → $\to \theta^{(3)}$ $J_{cv}(\theta^{(3)})$
4. $\lambda = 0.04$
5. $\lambda = 0.08$
⋮
12. $\lambda = 10$ → $\min J(\theta) \to \theta^{(12)} \to J_{cv}(\theta^{3})$

pick $\theta^{(k)}$ with lowest $J_{cv}(\theta^{(k)})$.
⇓
$J_{test}(\theta^{(k)})$.

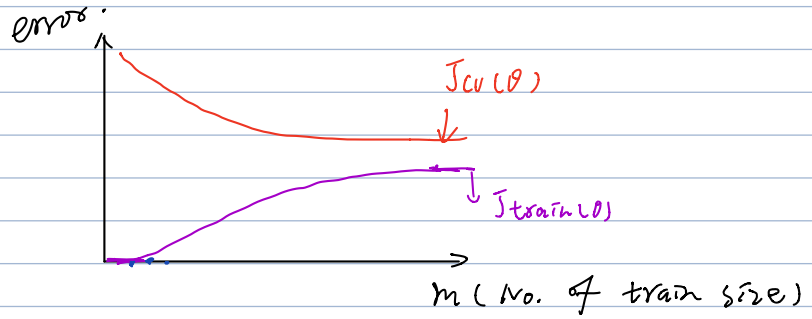


Learning curves.

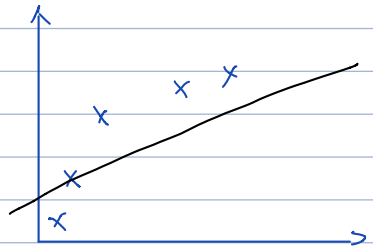
suppose $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

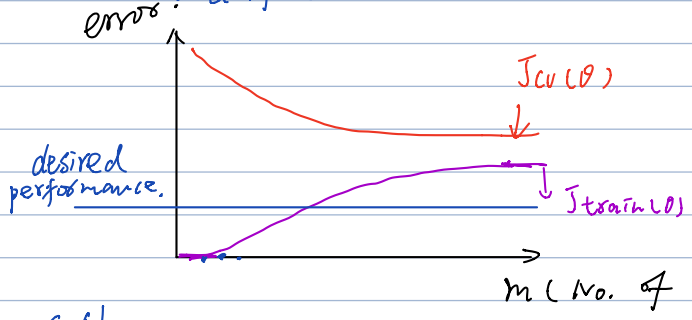
$$J_{train}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J_{cv}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$



High bias: $h_\theta(x) = \theta_0 + \theta_1 x$. → underfit.

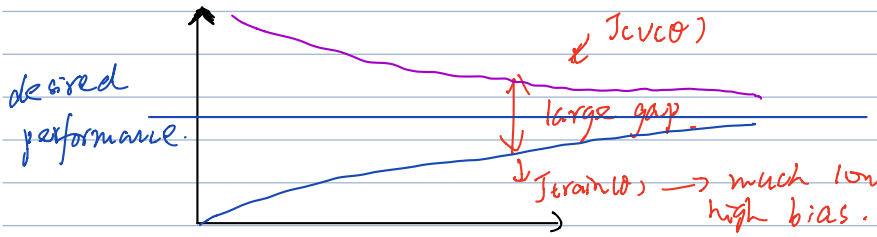




⇒ high $J_{cv}(\theta)$ & $J_{train}(\theta)$.

If a learning algorithm is suffering from high bias, getting more training data will not help much.

High variance: $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_{100} x^{100}$



增加训练集来改进高模型效果.

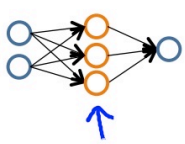
$J_{train}(\theta) < J_{cv}(\theta)$ but the difference between them remains significant.

① Get more training examples. → fix high variance
② Try smaller sets of features → fix high variance
③ Try getting additional features → fix high bias.
④ Try getting polynomial features (x_1^2, x_2^2, x_1x_2) ← fix high bias.
⑤ Try decreasing / increasing λ.

↓ fix high bias ↓ fix high variance.

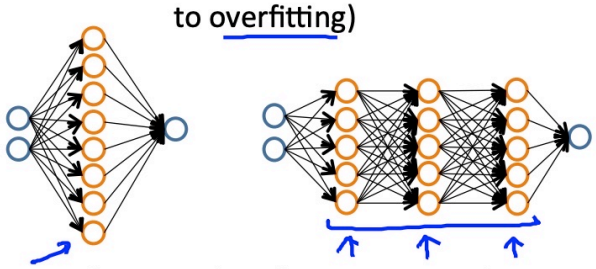
Neural networks and overfitting

→ "Small" neural network (fewer parameters; more prone to underfitting)



Computationally cheaper

→ "Large" neural network (more parameters; more prone to overfitting)



Computationally more expensive.

Use regularization (λ) to address overfitting.

$J_{cv}(\theta)$ ↑

通常选择较大的神经网络并使用正则化比使用较小的 NN 的效果好.

prediction class \ True class	1	0
1	True positive (TP)	False positive (FP)
0	False negative (FN)	True negative (TN)

$$\text{precision} = \frac{TP}{TP + FP}$$

在我们预测有 tumor 的病人中，真正有 tumor 的病人

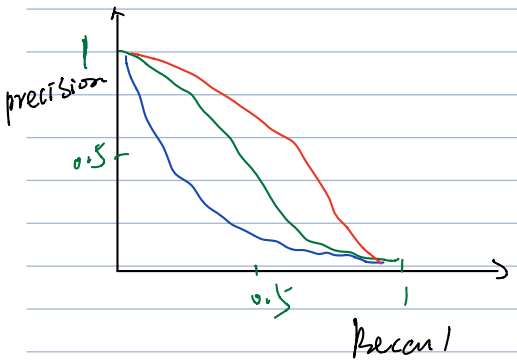
$$\text{Recall} = \frac{TP}{TP + FN}$$

在实际有 tumor 的人中，我们预测的人数。

(查全率，召回率)

越高越好！

Trade off precision and recall.



F1 score (F score)

$$2 \cdot \frac{PR}{P + R}$$