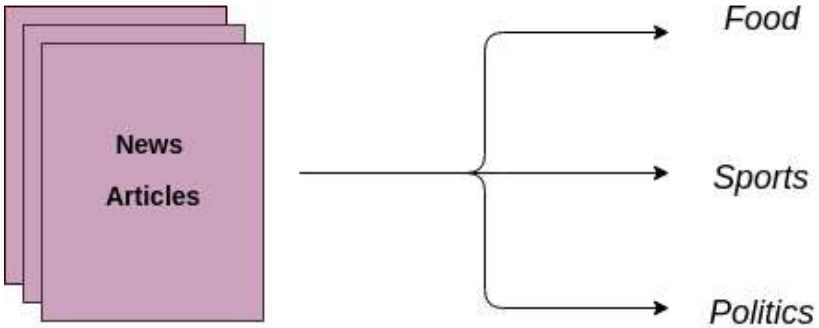
Model Evaluation

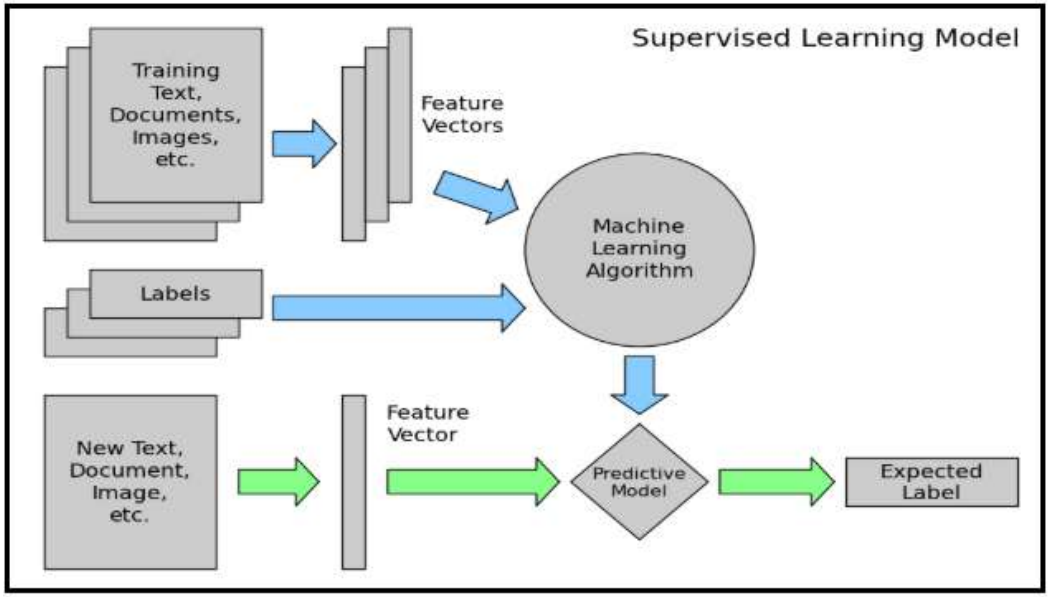
Outline

- Evaluation
- Training Issues
  - Measuring model quality
  - Over-fitting
  - Cross-validation

Example Text Classification

- Text Classification is an example of supervised machine learning task since a labelled dataset containing text documents and their labels is used for train a classifier



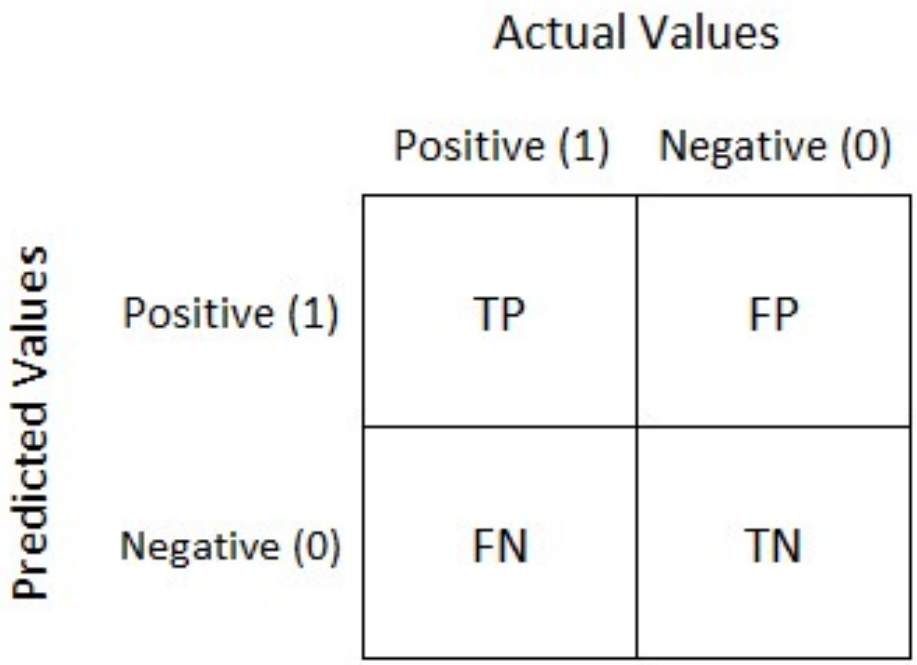


Evaluation

- The training dataset trains the model to predict the unknown labels of population data.
- There are multiple algorithms, namely, Logistic regression, K-nearest neighbor, Decision tree, Naive Bayes etc. All these algorithms have their own style of execution and different techniques of prediction.
- But, at the end, we need to find the effectiveness of an algorithm.
- To find the most suitable algorithm for a particular business problem, there are few model evaluation techniques.

Evaluation

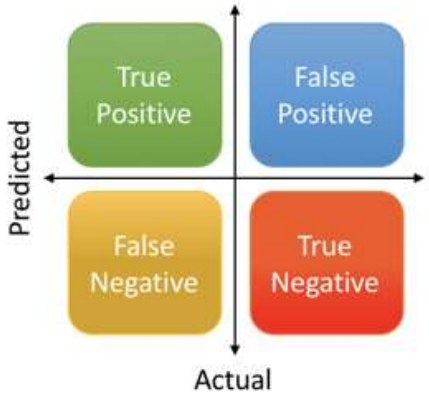
- For a classification task, *positive* means that an instance is labeled as belonging to the class of interest: we may want to automatically gather all news articles about Microsoft out of a news feed, or identify fraudulent credit card transactions, classify emails as spam or not e.t.c.
- A *false positive* is concluding that something is positive when it is not. False positives are sometimes called *Type I errors*.
- A *false negative* is concluding that something is negative when it is not. False negatives are sometimes called *Type II errors*.
- True negative is concluding that something is negative when it is actually negative
- True positive is concluding that something is positive when it is actually positive


Actual Values
Positive (1)
Negative (0)
Predicted Values
Positive (1)
Negative (0)
TP
FP
FN
TN

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

$$\textbf{Accuracy} = \frac{\text{True Positive + True Negative}}{\text{Total}}$$



Evaluation

- precision, recall
  - Precision means the percentage of your results which are relevant.
  - recall refers to the percentage of total relevant results correctly classified by your algorithm
- f-score
  - the harmonic mean of precision and recall:
  - near one when both precision and recall are high, and near zero when they are both low.
  - It is a convenient single score to characterize overall accuracy, especially for comparing the performance of different classifiers.

$$F_1 = \frac{2}{(1/\mathrm{precision} + 1/\mathrm{recall})} = 2 \times \frac{\mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Example

- Classify email as either spam or not spam

Classify email as either spam or not spam

		Actual	
		Spam	**Not spam**
System (Predicted)	spam	12	8
	Not spam	3	77

- True Positive: **12 (You have predicted the positive case correctly!), system predicted spam and the are truly spam.**
- True Negative: **77 (You have predicted negative case correctly!), system predicted not spam and the email are truly not spam.**
- False Positive: **8 (Oh! You have predicted these emails are spam, but in actual they are not spam. This is type-II error in this case.)**
- False Negative: **3 (Oh ho! You have predicted that these three emails are not spam. But actually are spam. This is dangerous! Be careful! This is type-I error in this case.)**

Classify email as either spam or not spam

- Accuracy the ratio of the accurately predicted number and the total number of people which is *(12+77)/100 = 0.89*.
- Precision - the ratio, ***12/(12+8) = 0.6*** is the measure of the accuracy of your model in detecting a person to have the disease.
- Recall - the ratio, 12/(12+3) = 0.8 is the measure of the accuracy of your model to detect a person having disease out of all the people who are having the disease in actual.

		Actual	
		Spam	**Not spam**
System (Predicted)	spam	12	8
	Not spam	3	77

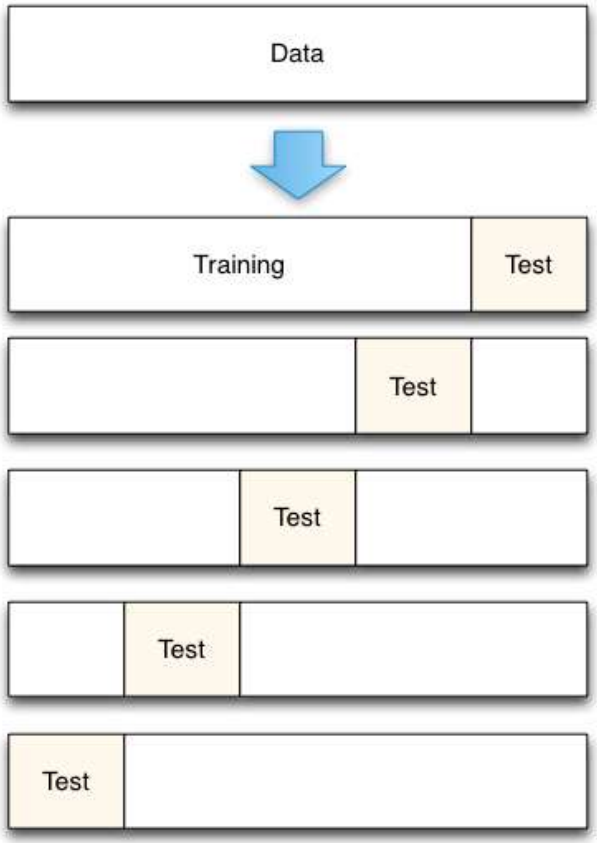
Cross-Validation

- Cross-validation involves **partitioning** your data into distinct **training** and **test** subsets.
- The test set **should never** be used to **train** the model.
- The test set is then used to **evaluate** the model after training.

K-fold Cross-Validation

- To get more accurate estimates of performance you can do this k times.
- Break the data into k equal-sized subsets A_i
- For each i in 1,…,k do:
  - Train a model on all the other folds A_1,…, A_{i-1}, A_{i+1},…, A_k
  - Test the model on A_i
- Compute the **average performance** of the k runs

5-fold Cross-Validation



Model Selection

Model Quality

Almost every model **optimizes some quality criterion**:

- For linear regression it was the **Residual Sum-of-Squares**
- …

The quality criterion is chosen often because of its good properties:

- **Convexity:** so that there is a unique, best solution
- **Closed form** for the optimum (linear regression) or at least for the gradient (for SGD).
- An algorithm that **provably converges**.

Model Quality

There are typically other criteria used to measure the quality of models. e.g.

For regression models:

- Stability of the model (sensitivity to small changes)
- Compactness (sparseness or many zero coefficients)

Model selection and generalization

- Machine learning problems (classification, regression and others) are typically ill-posed : the observed data is finite and does not uniquely determine the classification or regression function.
- In order to find a unique solution, and learn something useful, we must make assumptions (= inductive bias of the learning algorithm).

•Ex: the use of a hypothesis class H; the use of the largest margin; the use of the least-squares objective function.

- How to choose the right inductive bias, in particular the right hypothesis class H? This is the model selection problem.

Generalization

- Generalization: not memorization.

–minimizing error on the training set in general does not guarantee good generalization.

–too complex hypotheses could overfit training sample.

–how much complexity vs. training sample size?

Model selection and generalization

- The goal of ML is not to replicate the training data, but to predict unseen data well, i.e., to generalize well.
- For best generalization, we should match the complexity of the hypothesis class H with the complexity of the function underlying the data:

•If H is less complex: underfitting. Ex: fitting a line to data generated from a cubic polynomial.

•the excessively simple model fails to “Learn” the intricate patterns and underlying trends of the given dataset.

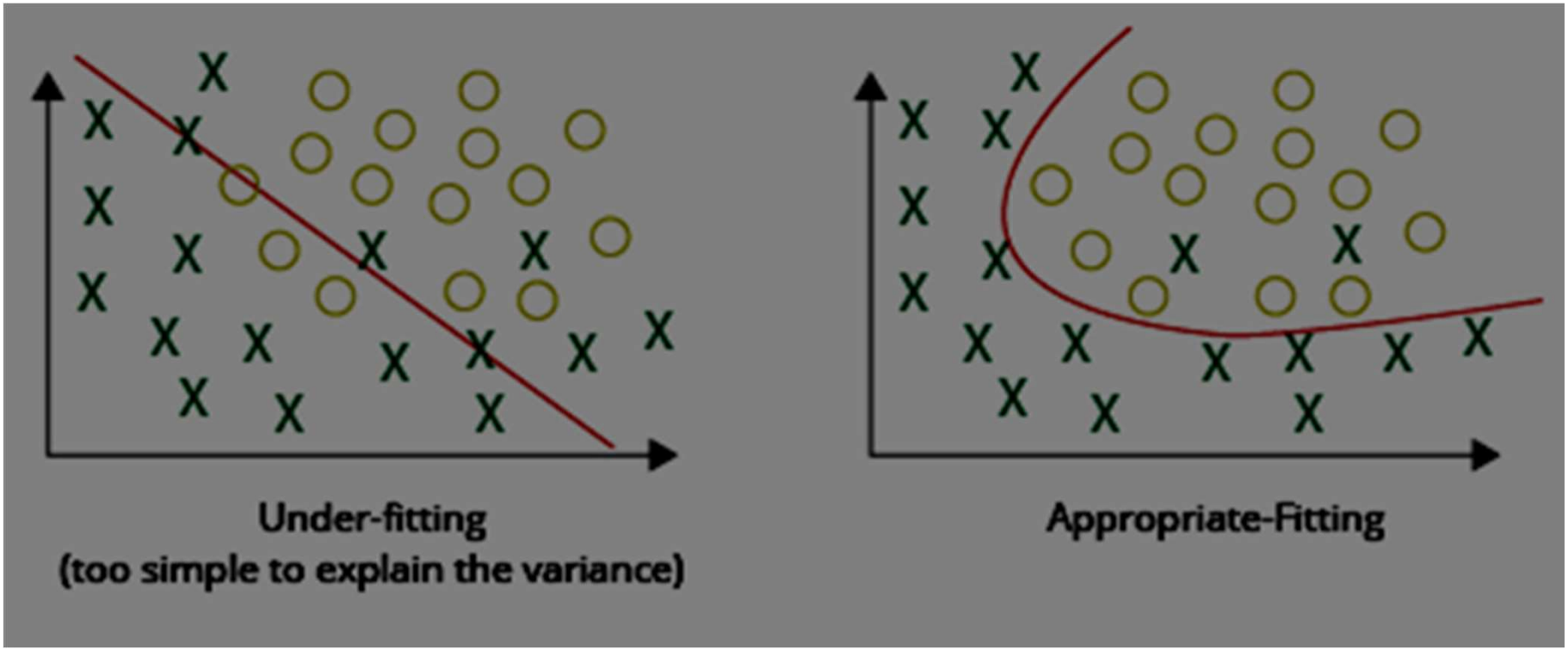
•If H is more complex: overfitting. Ex: fitting a cubic polynomial to data generated from a line.

• increasing model complexity, the model tends to fit the Noise present in data

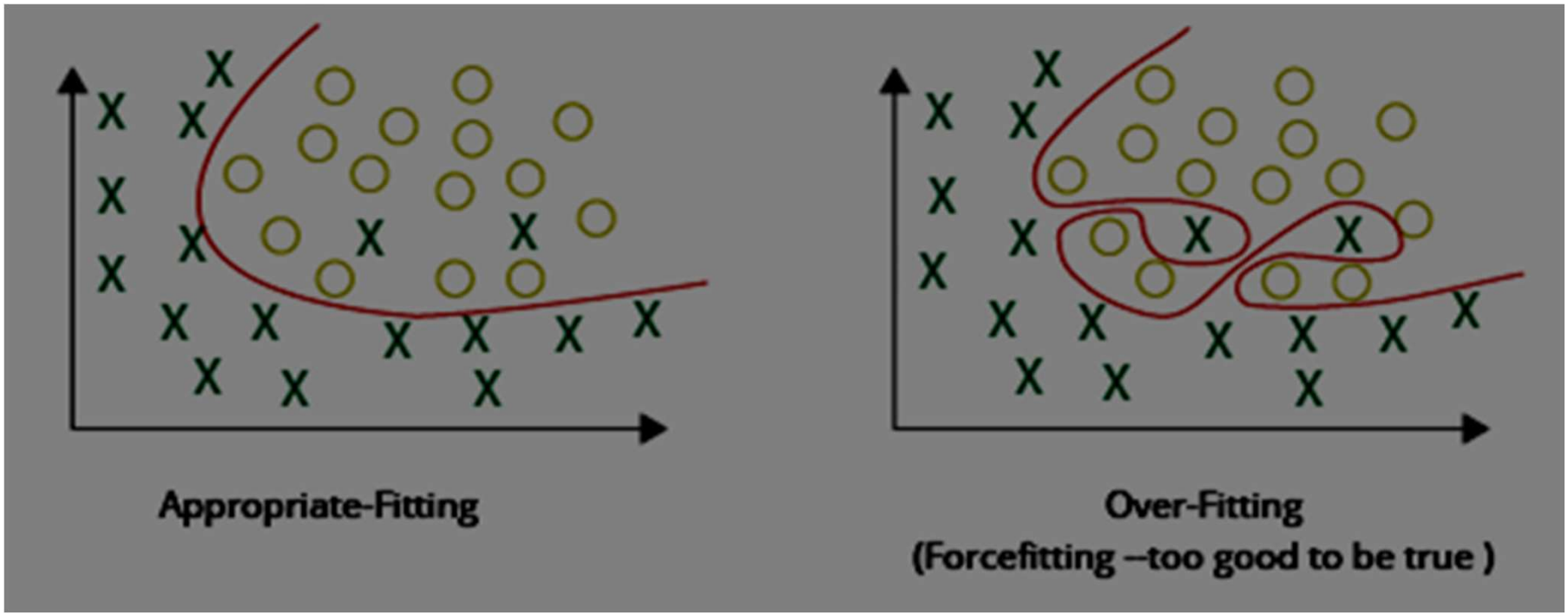
Over-fitting

- Your model should ideally fit an **infinite sample** of the type of data you're interested in.
- In reality, you only have a **finite set** to train on. A good model for this subset is a good model for the infinite set, up to a point.
- Beyond that point, the model quality (measured on new data) starts to **decrease**.
- Beyond that point, the model is **over-fitting** the data.

Model selection and generalization



Model selection and generalization



Bias-Variance Tradeoff

- Bias is the amount of error introduced by approximating real-world phenomena with a simplified model.
- Variance is how much your model's test error changes based on variation in the training data. It reflects the model's sensitivity to the idiosyncrasies of the data set it was trained on.
- As a model increases in complexity and it becomes more wiggly (flexible), its bias decreases (it does a good job of explaining the training data), but variance increases (it doesn't generalize as well). Ultimately, in order to have a good model, you need one with low bias and low variance.

Bias-Variance Tradeoff

- There are two major sources of error in machine learning: bias and variance. Understanding them will help you decide whether adding data, as well as other tactics to improve performance, are a good use of time.
- Suppose you hope to build a cat recognizer that has 5% error. Right now, your training set has an error rate of 15%, and your dev set has an error rate of 16%. In this case, adding training data probably won't help much.
- First, the algorithm's error rate on the training set. In this example, it is 15%. We think of this informally as the algorithm's bias.
- • Second, how much worse the algorithm does on the dev (or test) set than the training set. We think of this informally as the algorithm's variance.

Bias-Variance Tradeoff

- Consider our cat classification task. An “ideal” classifier (such as a human) might achieve nearly perfect performance in this task.
- Suppose your algorithm performs as follows:
- • Training error = 1%
- • Dev error = 11%
- What problem does it have? Applying the definitions from the previous chapter, we estimate the bias as 1%, and the variance as 10% (=11%-1%). Thus, it has high variance. The classifier has very low training error, but it is failing to generalize to the dev set. This is also called **overfitting**.

Bias-Variance Tradeoff

- Now consider this:
- • Training error = 15%
- • Dev error = 16%
- We estimate the bias as 15%, and variance as 1%. This classifier is fitting the training set poorly with 15% error, but its error on the dev set is barely higher than the training error.
- This classifier therefore has high bias, but low variance. We say that this algorithm is **underfitting**.

Bias-Variance Tradeoff

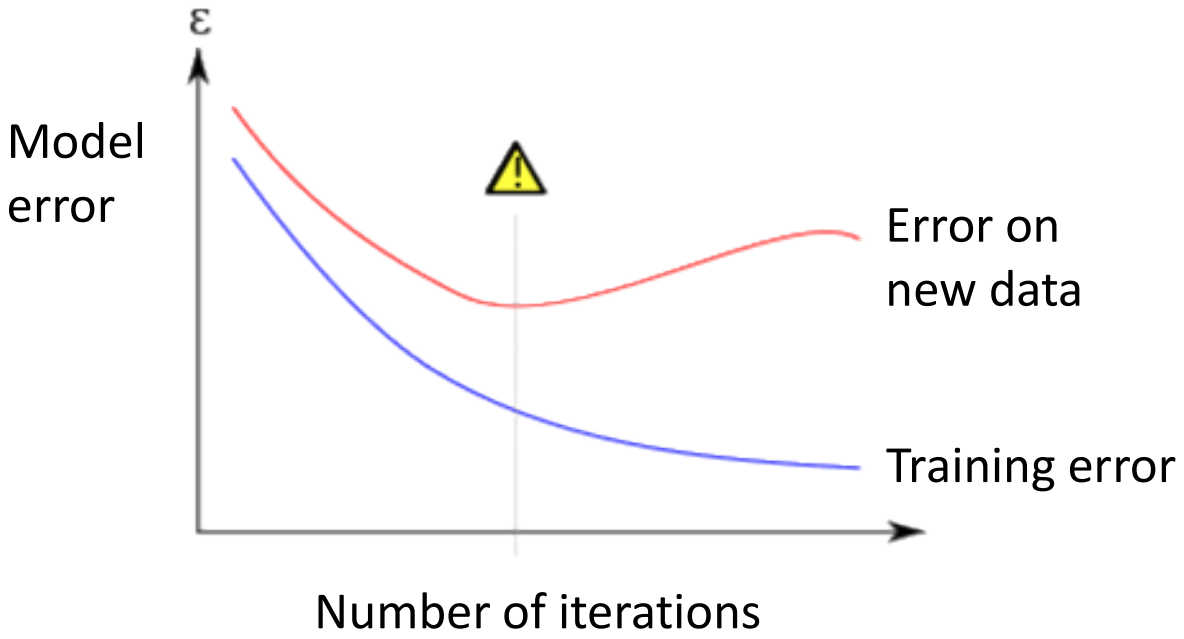
- Now, consider this:
- • Training error = 15%
- • Dev error = 30%
- We estimate the bias as 15%, and variance as 15%. This classifier has high bias and high variance: It is doing poorly on the training set, and therefore has high bias, and its performance on the dev set is even worse, so it also has high variance.
- The overfitting/underfitting terminology is hard to apply here since the classifier is simultaneously overfitting and underfitting.

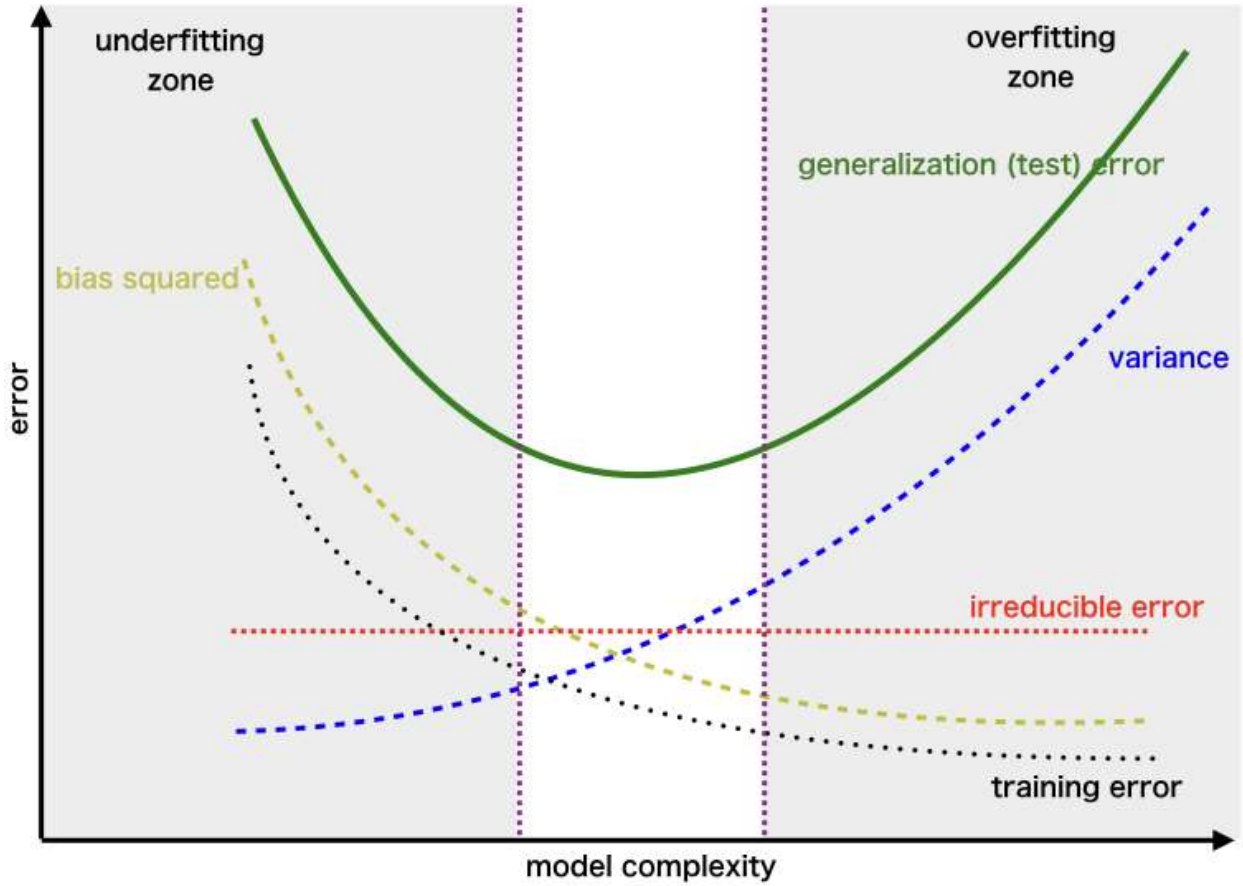
Bias-Variance Tradeoff

- Finally, consider this:
- • Training error = 0.5%
- • Dev error = 1%
- This classifier is doing well, as it has low bias and low variance. **Congratulations on achieving this great performance!**

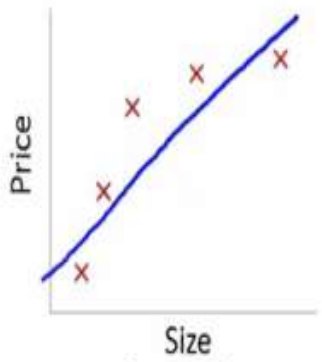
Over-fitting

Over-fitting during training



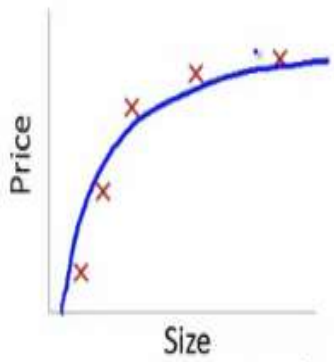

underfitting
zone
overfitting
zone
generalization (test) error
bias squared
variance
irreducible error
training error
error
model complexity

Bias-Variance Tradeoff



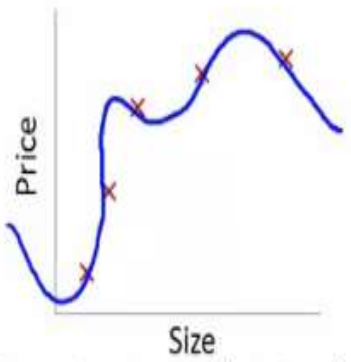
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

Remember that the only thing we care about is how the model performs on test data.

Classification example - Complex model

- Should we keep the hypothesis class simple rather than complex?
- Easier to use and to train (fewer parameters, faster).
- Easier to explain or interpret.
- Less variance in the learned model than for a complex model (less affected by single instances), but also higher bias.
- Given comparable empirical error, a simple model will generalize better than a complex one. (**Occam's razor** : simpler explanations are more plausible; eliminate unnecessary complexity.)

Model selection and generalization

- In summary, in ML algorithms there is a tradeoff between 3 factors:
- the complexity c(H) of the hypothesis class
- the amount of training data N
- the generalization error E

so that

- as $N \uparrow$, $E \downarrow$
- as $c(\mathcal{H}) \uparrow$, first $E \downarrow$ and then $E \uparrow$

Techniques for reducing variance

- **Add more training data:** This is the simplest and most reliable way to address variance, so long as you have access to significantly more data and enough computational power to process the data.
- **Add regularization(L2 regularization, L1 regularization, dropout):** This technique reduces variance but increases bias.
- **Add early stopping(i.e**., stop gradient descent early, based on dev set error): This technique reduces variance but increases bias. Early stopping behaves a lot like regularization methods, and some authors call it a regularization technique.
- **Feature selection to decrease number/type of input features:** This technique might help with variance problems, but it might also increase bias. Reducing the number of features slightly (say going from 1,000 features to 900) is unlikely to have a huge effect on bias. Reducing it significantly (say going from 1,000 features to 100—a 10x reduction) is more likely to have a significant effect, so long as you are not excluding too many useful features.
- **Decrease the model size (such as number of neurons/layers):** Use with caution.This technique could decrease variance, while possibly increasing bias. However, I don'trecommend this technique for addressing variance. Adding regularization usually gives better classification performance. The advantage of reducing the model size is reducing your computational cost and thus speeding up how quickly you can train models.

Techniques for reducing avoidable bias

- **Increase the model size (such as number of neurons/layers):** This technique reduces bias, since it should allow you to fit the training set better. If you find that this increases variance, then use regularization, which will usually eliminate the increase in variance.
- **Modify input features based on insights from error analysis:** Say your error analysis inspires you to create additional features that help the algorithm eliminate a particular category of errors. These new features could help with both bias and variance. In theory, adding more features could increase the variance; but if you find this to be the case, then use regularization, which will usually eliminate the increase in variance.
- **Reduce or eliminate regularization(L2 regularization, L1 regularization, dropout):** This will reduce avoidable bias, but increase variance.
- **Modify model architecture** (such as neural network architecture) so that it is more suitable for your problem: This technique can affect both bias and variance.
- One method that is not helpful:
  - Add more training data: This technique helps with variance problems, but it usually has no significant effect on bias.

Noise

- Noise is any unwanted anomaly in the data. It can be due to:
- Imprecision in recording the input attributes: xn .
- Errors in labeling the input vectors: yn .
- Attributes not considered that affect the label (hidden or latent attributes, may be unobservable).
- Noise makes learning harder.