CS440/ECE448 Lecture 4: Evaluation & Generalization

Mark Hasegawa-Johnson

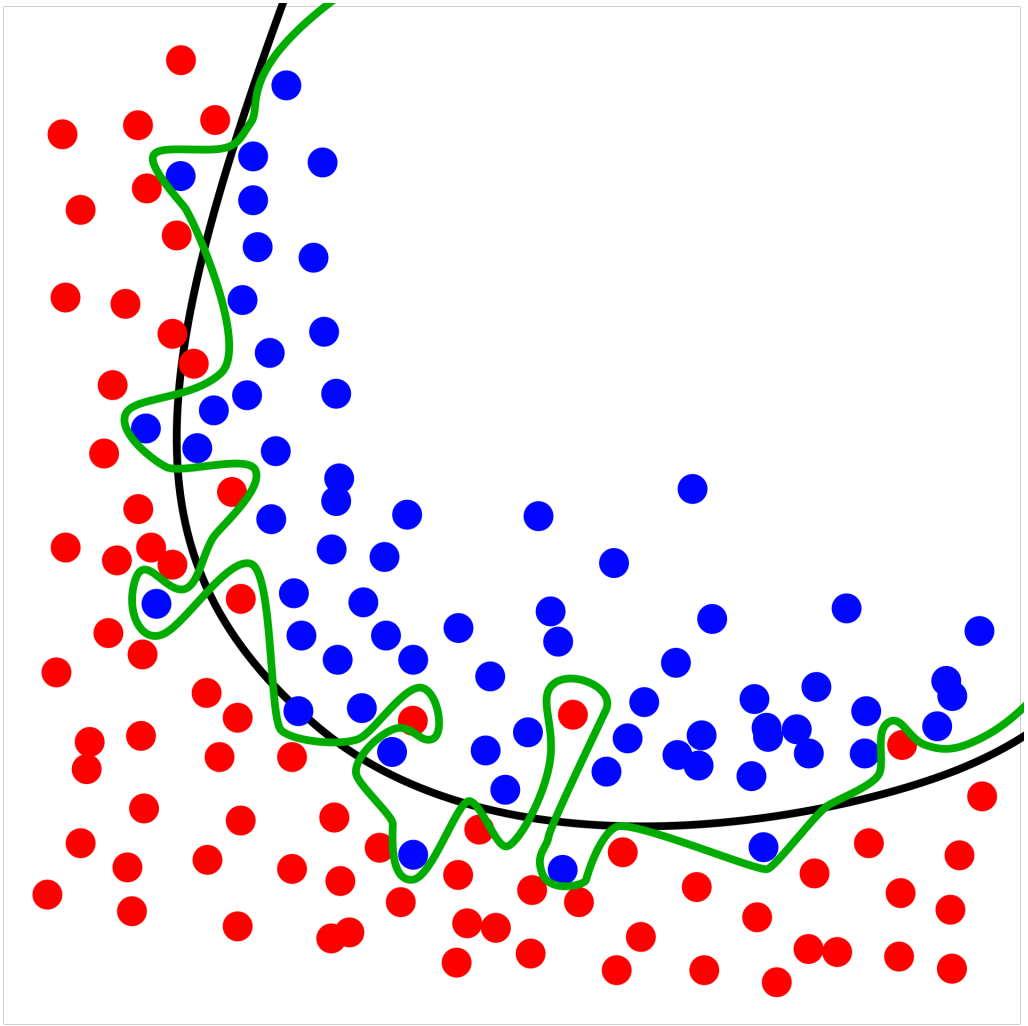


Diagram showing overfitting of a classifier. CC-BY 4.0, Ignacio Icke, 2008

Outline

- Accuracy, Error Rate, and the Bayes Error Rate
- Confusion Matrix, Precision & Recall
- Train, Dev, and Test Corpora
- Overfitting
- Laplace Smoothing

Accuracy

When we train a classifier, the metric that we usually report is "accuracy."

$$\text{Accuracy} = \frac{\#\text{ tokens correctly classified}}{\#\text{ tokens total}}$$

Error Rate

Equivalently, we could report error rate, which is just 1-accuracy:

$$\text{Error Rate} = \frac{\#\text{ tokens incorrectly classified}}{\#\text{ tokens total}}$$

Error Rate

Error rate is the probability that the classifier output, $f(x)$, is not equal to the correct label, y, averaged over all possible x:

$$\text{Error Rate} = \sum_{x} P(X = x)P(Y \neq f(x)|X = x)$$

Bayes Error Rate

The “Bayes Error Rate” is the smallest possible error rate of any classifier with labels y and features x:

$$\text{Error Rate} = \sum_{x} P(X = x) \min_{y} P(Y \neq y | X = x)$$

It’s called the “Bayes error rate” because it’s the error rate of the Bayesian classifier.

Outline

- Accuracy, Error Rate, and the Bayes Error Rate
- Confusion Matrix, Precision & Recall
- Train, Dev, and Test Corpora
- Overfitting
- Laplace Smoothing

The problem with accuracy

- In most real-world problems, there is one class label that is much more frequent than all others.
  - Words: most words are nouns
  - Animals: most animals are insects
  - Disease: most people are healthy
- It is therefore easy to get a very high accuracy. All you need to do is write a program that completely ignores its input, and always guesses the majority class. The accuracy of this classifier is called the "chance accuracy."
- It is sometimes very hard to beat the chance accuracy. If chance=90%, and your classifier gets 89% accuracy, is that good, or bad?

The solution: Confusion Matrix

<u>Confusion Matrix</u> =

- $(m, n)^{\text{th}}$ element is
- the number of tokens of the m^{th} class
- that were labeled, by the classifier, as belonging to the n^{th} class.

```
title: Consonant Confusions in CV utterances, for V=/a/, for S/N = +12db and
Phones involved: 16, namely p t k f T (th) s S (sh) b d g v D (dh) z Z (zh) m

    p   t   k   f   T   s   S   b   d   g   v   D   z   Z   m   n      Total
p   228 7   7   1   0   0   1   0   0   0   0   0   0   0   0   0    p 244
t   0   236 8   0   0   0   0   0   0   0   0   0   0   0   0   0    t 244
k   26  5   213 0   0   0   0   0   0   0   0   0   0   0   0   0    k 244
f   6   1   1   194 35  0   0   3   0   0   1   3   0   0   0   0    f 244
T   0   2   2   96  146 2   0   2   1   0   1   8   0   0   0   0    T 260
s   0   2   0   1   31  204 1   1   9   4   0   7   0   0   0   0    s 260
S   0   0   0   0   0   1   243 0   0   0   0   0   0   0   0   0    S 244
b   0   0   0   13  12  0   0   207 2   3   19  8   0   0   0   0    b 264
d   0   0   0   0   0   0   0   0   240 9   0   0   0   3   0   0    d 252
g   0   0   0   0   0   0   0   1   41  199 0   0   2   1   0   0    g 244
v   0   0   0   3   3   0   0   20  0   2   182 47  2   0   0   1    v 260
D   0   0   0   0   7   0   0   10  3   22  49  170 19  0   0   0    D 280
z   0   0   0   0   1   0   0   3   8   24  2   22  145 3   0   0    z 208
Z   0   0   0   0   0   0   1   0   2   0   0   0   13  264 0   0    Z 280
m   0   0   0   0   0   0   0   0   0   0   0   0   0   0   213 11   m 224
n   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   248  n 248
```

Plaintext versions of the Miller & Nicely matrices, posted by Dinoj Surendran, http://people.cs.uchicago.edu/~dinoj/research/nicely.html

Confusion matrix for a binary classifier

Suppose that the correct label is either 0 or 1. Then the confusion matrix is just 2x2.

For example, in this box, you would write the # tokens of class 1 that were misclassified as class 0

False Positives & False Negatives

- TP (True Positives) = tokens that were correctly labeled as “1”
- FN (False Negatives) = tokens that should have been “1”, but were mislabeled as “0”
- FP (False Positives) = tokens that should have been “0”, but were mislabeled as “1”
- TN (True Negative) = tokens that were correctly labeled as “0”

Classified As:

Correct Label:	0	1
0	TN	FP
1	FN	TP

Summaries of a Binary Confusion Matrix

The binary confusion matrix is standard in many fields, but different fields summarize its content in different ways.

- In medicine, it is summarized using Sensitivity and Specificity.
- In information retrieval (IR) and AI, we usually summarize it using Recall and Precision.

Classified As:

Correct Label:	0	1
0	TN	FP
1	FN	TP

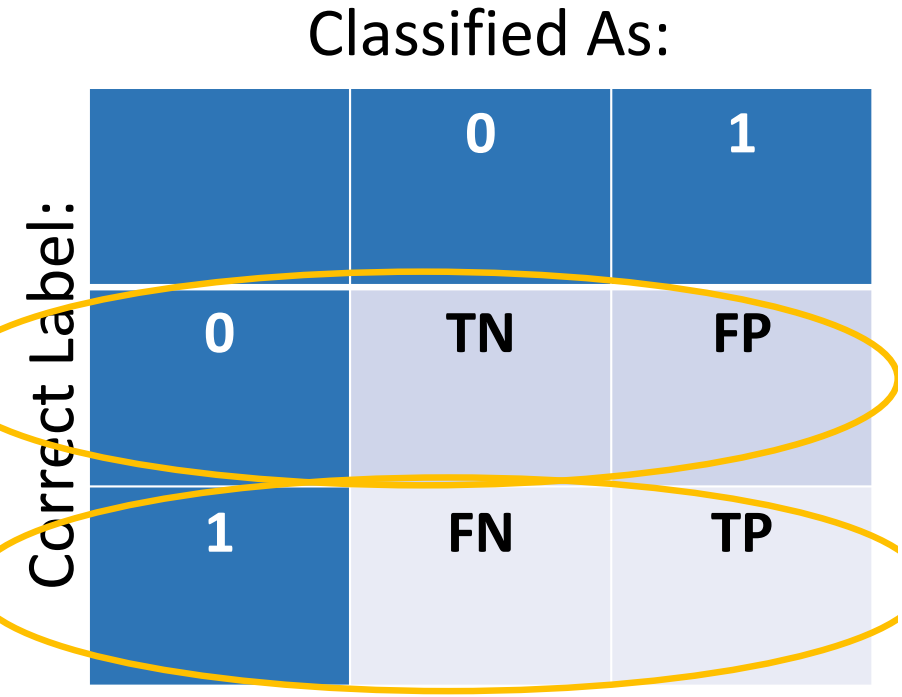
Specificity and Sensitivity

Specificity = True Negative Rate (TNR):

$$TNR = P(f(X) = 0|Y = 0) = \frac{TN}{TN + FP}$$

Sensitivity = True Positive Rate (TPR):

$$TPR = P(f(X) = 1|Y = 1) = \frac{TP}{TP + FN}$$



How a high-sensitivity, high-specifity test can interact with an *a priori* rare event:

1% of women at age forty who participate in routine screening have breast cancer. The test has sensitivity of 80%, and selectivity of 90.4%.

$$P(f(X) = 0, Y = 0) = P(f(X) = 0|Y = 0)P(Y = 0)$$
$$= (0.904)(0.99) \approx 0.895$$

$$P(f(X) = 1, Y = 0) = P(f(X) = 1|Y = 0)P(Y = 0)$$
$$= (0.096)(0.99) \approx 0.095$$

$$P(f(X) = 0, Y = 1) = P(f(X) = 0|Y = 1)P(Y = 1)$$
$$= (0.2)(0.01) = 0.002$$

$$P(f(X) = 1, Y = 1) = P(f(X) = 1|Y = 1)P(Y = 1)$$
$$= (0.8)(0.01) = 0.008$$

Classified As:

Correct Label:	0	1
0	0.895	0.095
1	0.002	0.008

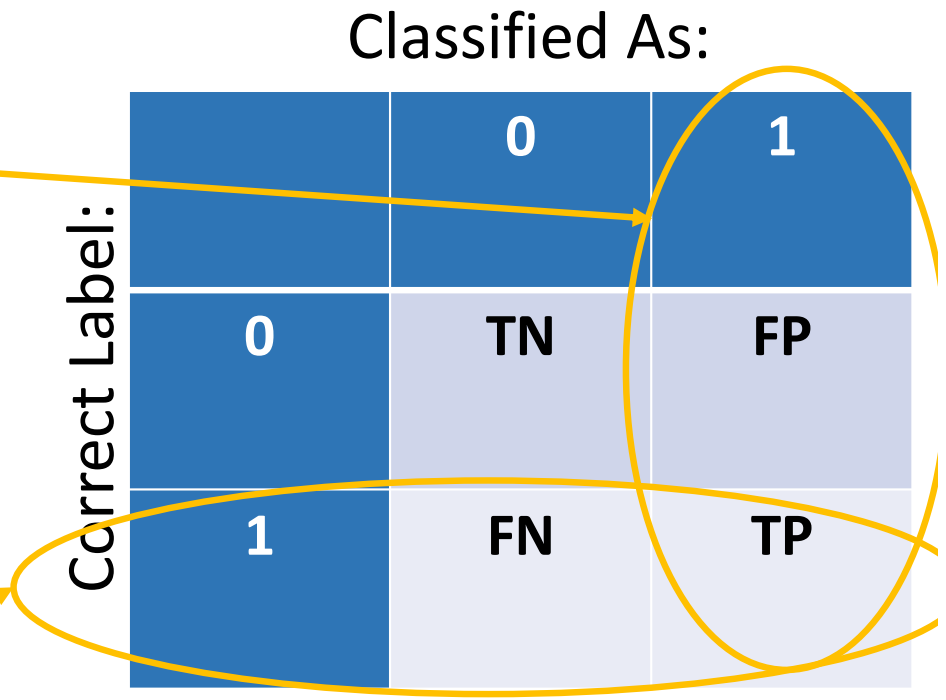
How IR and AI summarize confusions

Precision:

$$P = P(Y = 1 | f(X) = 1) = \frac{TP}{TP + FP}$$

Recall = Sensitivity = TPR:

$$R = P(f(X) = 1 | Y = 1) = \frac{TP}{TP + FN}$$



The Misdiagnosis Problem

1% of women at age forty who participate in routine screening have breast cancer. 80% of women with breast cancer will get positive mammographies. 9.6% of women without breast cancer will also get positive mammographies.

A woman in this age group had a positive mammography in a routine screening. What is the probability that she actually has breast cancer?

$$P(\text{Cancer} = T|\text{Test} = T) = \frac{P(\text{Test} = T|\text{Cancer} = T)P(\text{Cancer} = T)}{P(\text{Test} = T)}$$

$$= \frac{P(\text{Test} = T|\text{Cancer} = T)P(\text{Cancer} = T)}{P(\text{Test} = T|\text{Cancer} = T)P(\text{Cancer} = T) + P(\text{Test} = T|\text{Cancer} = F)P(\text{Cancer} = F)}$$

$$= \frac{(0.8)(0.01)}{(0.8)(0.01) + (0.096)(0.99)} = 0.0776$$

Recall:

$R = P(f(X) = 1|Y = 1)$

$= 0.8$

Precision:

$P = P(Y = 1|f(X) = 1)$

$= 0.0776$

Outline

- Accuracy, Error Rate, and the Bayes Error Rate
- Confusion Matrix, Precision & Recall
- Train, Dev, and Test Corpora
- Overfitting
- Laplace Smoothing

Accuracy on which corpus?

Consider the following experiment: among all of your friends' pets, there are 4 dogs and 4 cats.

1. Measure several attributes of each animal: weight, height, domesticity, color, number of letters in its name…
2. You discover that, among your friends' pets, all dogs have 1-syllable names, while the names of all cats have 2+ syllables.
3. Your classifier: an animal is a cat if its name has 2+ syllables.
4. Your accuracy: 100%

Is it correct to say that this classifier has 100%? Is it useful to say so?

Training vs. Test Corpora

Training Corpus = a set of data that you use in order to optimize the parameters of your classifier (for example, optimize which features you measure, what are the weights of those features, what are the thresholds, and so on).

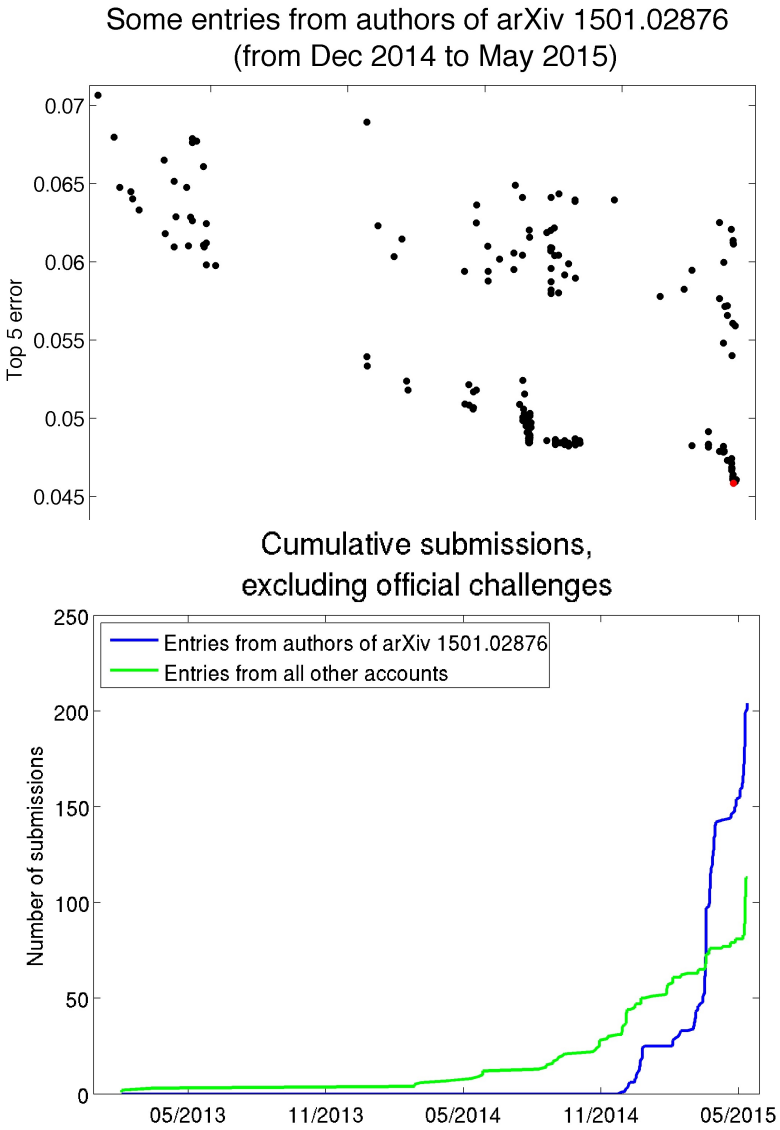
Test Corpus = a set of data that is non-overlapping with the training set (none of the test tokens are also in the training dataset) that you can use to measure the accuracy.

- Measuring the training corpus accuracy is useful for debugging: if your training algorithm is working, then training corpus accuracy should always go up.
- Measuring the test corpus accuracy is the only way to estimate how your classifier will work on new data (data that you've never yet seen).

Accuracy on which corpus?

This happened:

- Large Scale Visual Recognition Challenge 2015: Each competing institution was allowed to test up to 2 different fully-trained classifiers per week.
- One institution used 30 different e-mail addresses so that they could test a lot more classifiers (200, total). One of their systems achieved <46% error rate – the competition's best, at that time.
- Is it correct to say that that institution's algorithm was the best?



Training vs. development test vs. evaluation test corpora

Training Corpus = a set of data that you use in order to optimize the parameters of your classifier (for example, optimize which features you measure, what are the weights of those features, what are the thresholds, and so on).

Development Test (DevTest or Validation) Corpus = a dataset, separate from the training dataset, on which you test 200 different fully-trained classifiers (trained, e.g., using different training algorithms, or different features) to find the best.

Evaluation Test Corpus = a dataset that is used only to test the ONE classifier that does best on DevTest. From this corpus, you learn how well your classifier will perform in the real world.

Outline

- Accuracy, Error Rate, and the Bayes Error Rate
- Confusion Matrix, Precision & Recall
- Train, Dev, and Test Corpora
- Overfitting and underfitting
- Laplace Smoothing

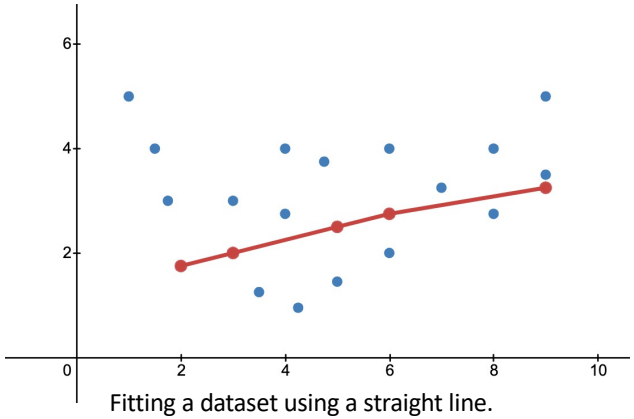
Overfitting

- Underfitting = your model has too few parameters. It is not able to get good accuracy on the training data.
- Overfitting = your model has too many parameters. It gets great accuracy on the training data, but very poor accuracy on the dev data.
- Overfitting is good! It proves that your code works.
- …but now that you've proven that your code works, now you need to do something to solve the overfitting.

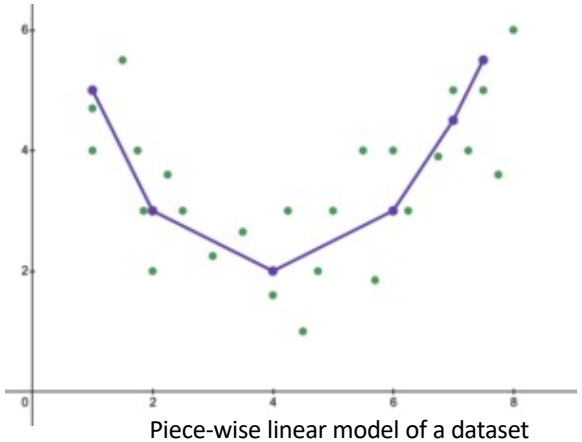
Example (from Wikipedia): curve fitting

- Suppose you have some examples of (x,y) training data points, where X and Y are both scalars.
- You want to try to find a polynomial, f(x) that
  1. Fits the training data points pretty well
  2. You think it is likely to also fit a test corpus pretty well

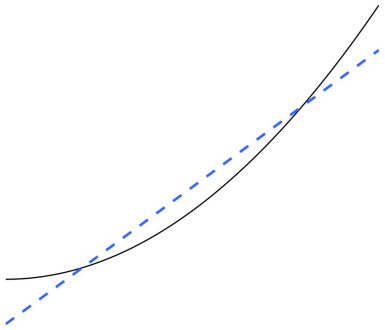
Which of these models of the data is best?



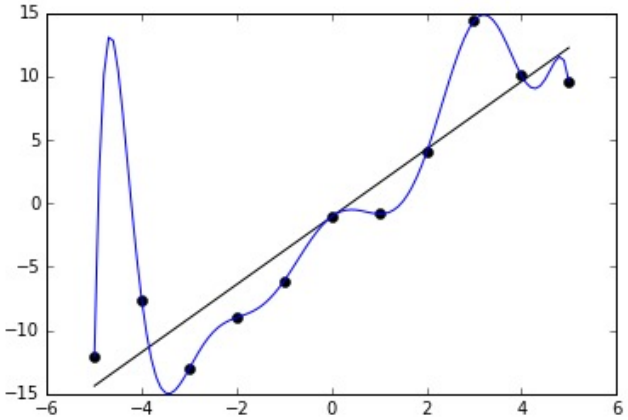
Fitting a dataset using a straight line.
Figure by AAStein, CC-BY-4.0, 2021



Piece-wise linear model of a dataset
Figure by AAStein, CC-BY-4.0, 2021



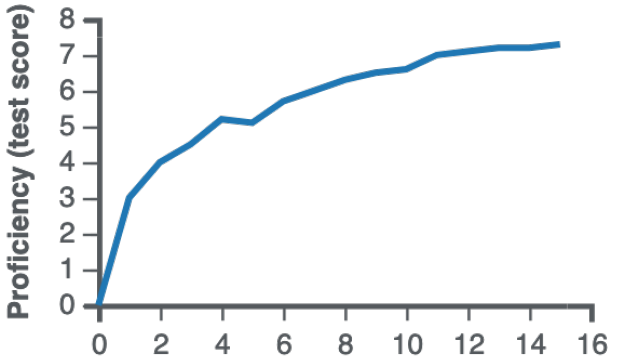
Fitting a dataset with a parabola.
Figure by Dfrankow, CC-BY-4.0, 2019



Fitting 11 data points using a 12^{th}-order polynomial.
Figure by Ghiles, CC-BY-4.0, 2016

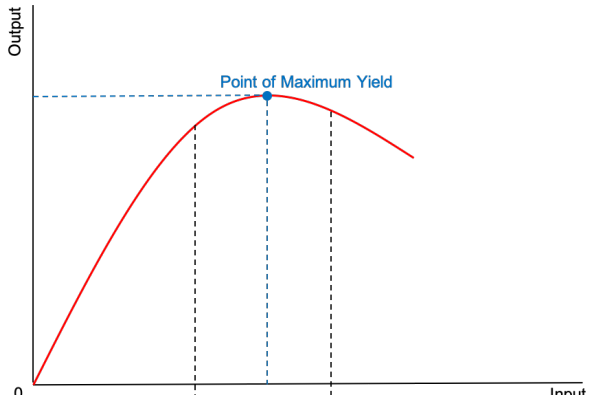
How to avoid both underfitting and overfitting

- Start with a model that has few trainable parameters
  - Train it on the training set.
  - Test it on the dev set.
  - Make sure you have similar accuracy on both the training set and dev set.
- Increase the number of trainable parameters in your model
  - Train on training data. As # parameters increases, accuracy on training set will always increase.
  - Test on dev data. As # parameters increases, accuracy on dev data will increase until it reaches a maximum, then start falling.
  - Best model = model that achieved lowest error rate on the dev set.



Accuracy on **training** data vs. # params.

https://en.wikipedia.org/wiki/Learning_curve, 2021



Accuracy on **dev** data vs. # params.

Happyavocado, CC-BY-4.0, 2021

Naïve Bayes spam detection example

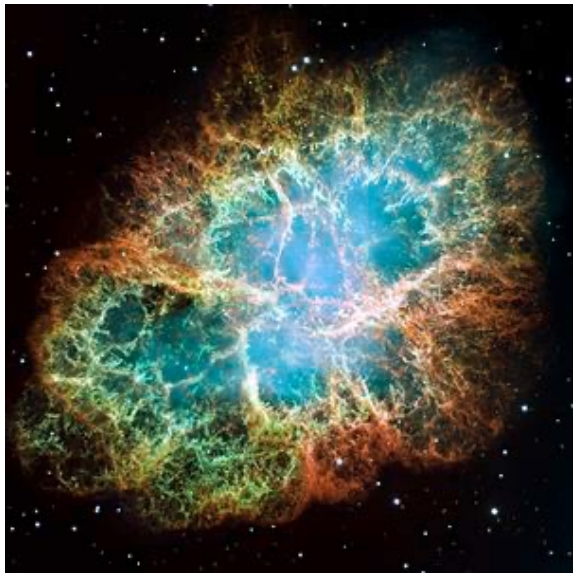
- How many trainable parameters does it have?
  - Unigram model: $2|X|$ trainable parameters (2 per unigram) – probably underfitted
  - Bigram model: $2|X|^2$ trainable parameters (2 per bigram) – probably overfitted
- The best classifier is probably somewhere in between the unigram and the bigram. Possible solutions include:
  - Limit the number of bigrams. For example, learn only the top 2000 bigrams. --- MP does not test this model, but it can work quite well.
  - Train a unigram model (which is probably underfitted) and a bigram model (which is probably overfitted), then try to find an optimum model by interpolating between them --- MP tests this model.

Outline

- Accuracy, Error Rate, and the Bayes Error Rate
- Confusion Matrix, Precision & Recall
- Train, Dev, and Test Corpora
- Overfitting
- Laplace Smoothing

What is the probability that the sun will fail to rise tomorrow?

- # times we have observed the sun to rise = 100,000,000
- # times we have observed the sun not to rise = 0
- Estimated probability the sun will not rise = $\frac{0}{0+100,000,000} = 0$



Oops….

Laplace Smoothing

- The basic idea: add k "unobserved observations" to every possible event
- # times the sun has risen or might have ever risen = 100,000,000+k
- # times the sun has failed to rise or might have ever failed to rise = 0+k
- Estimated probability the sun will rise tomorrow = $\frac{100{,}000{,}000+k}{100{,}000{,}000+2k}$
- Estimated probability the sun will not rise = $\frac{k}{100{,}000{,}000+2k}$
- Notice that, if you add these two probabilities together, you get 1.0.

Laplace Smoothing for Naïve Bayes

- The basic idea: add k "unobserved observations" to the count of every unigram
  - If a word occurs 2000 times in the training data, Count = 2000+k
  - If a word occur once in training data, Count = 1+k
  - If a word never occurs in the training data, then it gets a pseudo-Count of k
- Estimated probability of a word that occurred Count(w) times in the training data: =

$$P(w) = \frac{\text{Count}(w) + k}{\sum_w \text{Count}(w) + k(1 + \sum_w 1)}$$

- Estimated probability of a word that never occurred in the training data (an "out of vocabulary" or OOV word):

$$P(OOV) = \frac{0 + k}{\sum_w \text{Count}(w) + k(1 + \sum_w 1)}$$

- Notice that

$$P(OOV) + \sum_w \text{P}(w) = 1$$

Outline

- Bayes Error Rate:

$$\text{Bayes Error Rate} = \sum_{x} P(X = x) \min_{y} P(Y \neq y | X = x)$$

- Confusion Matrix, Precision & Recall

$$\text{Precision} = \frac{TP}{TP + FP}, \text{Recall} = \frac{TP}{TP + FN}$$

- Train, Dev, and Test Corpora
- Overfitting: increase # parameters in your model until you get the best accuracy on the dev data.
- Laplace Smoothing

$$P(w) = \frac{\text{Count}(w) + k}{\sum_w \text{Count}(w) + k(1 + \sum_w 1)}$$