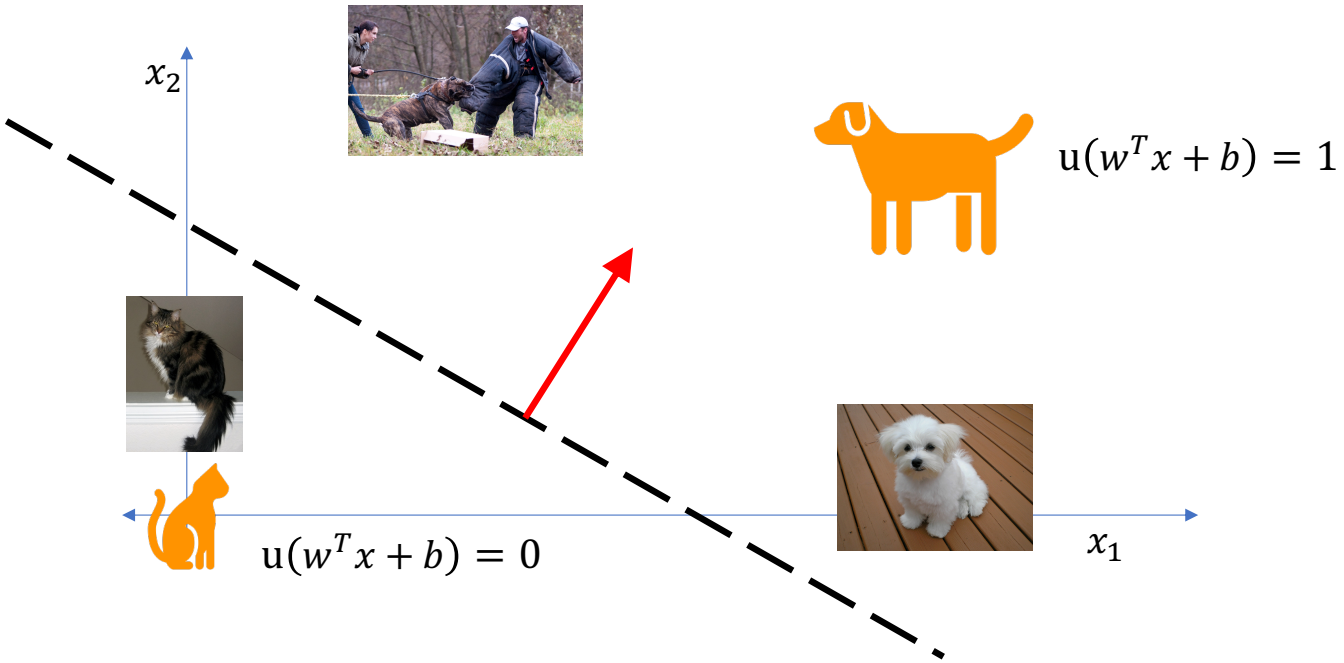
# CS440/ECE448 Lecture 7: Classifiers

Mark Hasegawa-Johnson, 2/2021

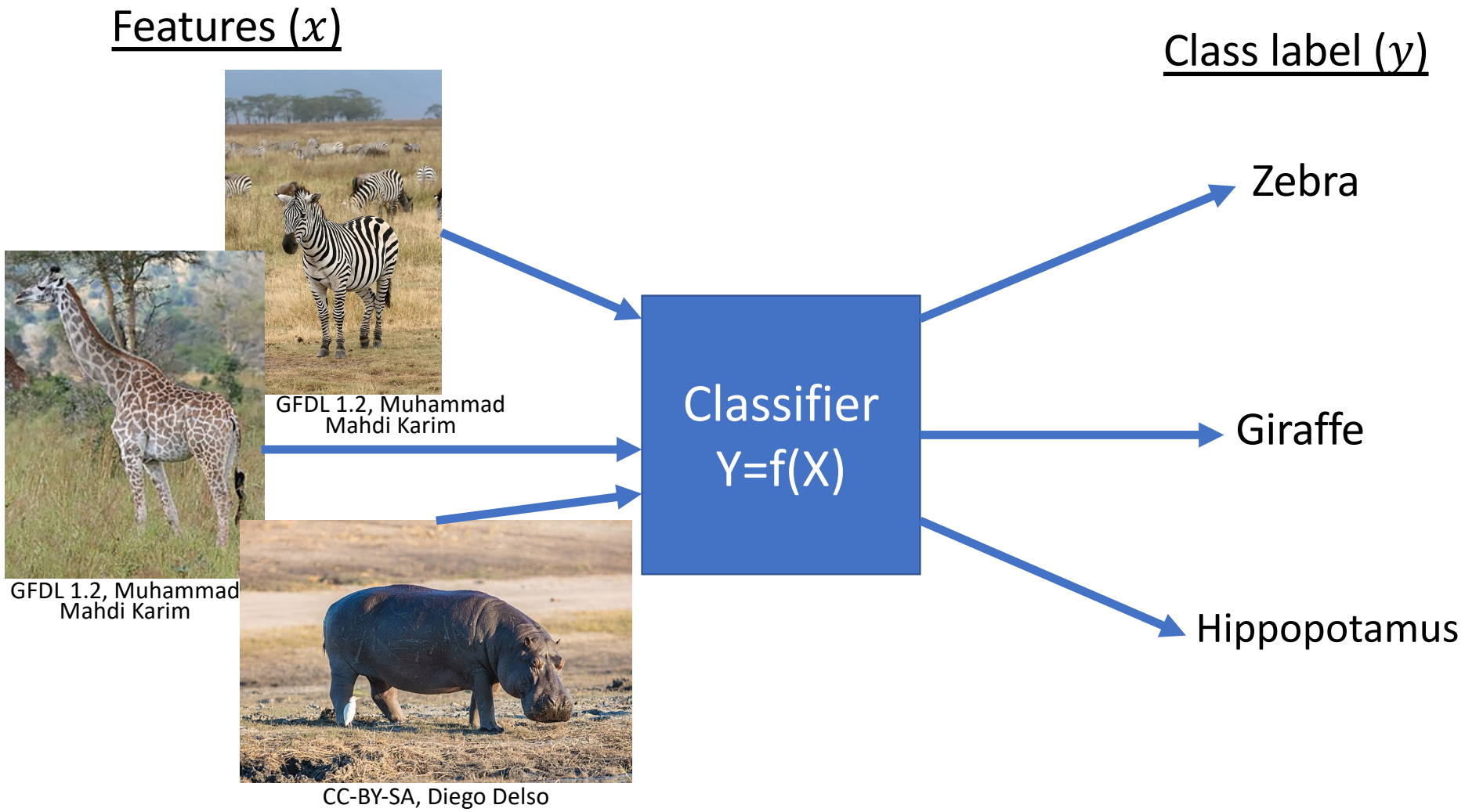




# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, development test, and evaluation test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

A classifier is a function $\hat{y} = f(x)$, where $x$=features, $y$=true label, $\hat{y}$=estimated label



# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, development test, and evaluation test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# Example: Naïve Bayes

- Class label = $Y$, drawn from some set of labels
- Features = $X_1, \ldots, X_D$
- Classification function: output the label, $Y = y$, which maximizes $P(Y = y, X_1, \ldots, X_D)$ under the following "naïve Bayes" assumption:

$$P(Y = y, X_1, \ldots, X_D) = P(Y = y) \prod_{d=1}^{D} P(X_d | Y = y)$$

# Naïve Bayes as a Function

Let's try to figure out how to make that a function, $\hat{y} = f(x)$.

1. A particular test token has the features $x = [x_1, \ldots, x_D]$.
2. For the classifier output, $\hat{y}$, we want to choose the value of $y$ that maximizes the probability:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(Y = y) \prod_{d=1}^{D} P(X_d = x_d | Y = y)$$

# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, development test, and evaluation test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# General Bayesian Classifier

- Class label $Y = y$, drawn from some set of labels
- Observation $X = x$, drawn from some set of features
- Bayesian classifier: choose the class label, $y$, that minimizes your probability of making a mistake:

$$\hat{y} = \underset{y}{\operatorname{argmin}}\, P(Y \neq y | X = x)$$

# MPE = MAP

- The minimum probability of error (MPE) classifier is the one that minimizes your probability of making a mistake:

$$\hat{y} = \underset{y}{\operatorname{argmin}}\, P(Y \neq y | X = x)$$

- The maximum a posteriori (MAP) classifier is the one that maximizes your probability of being correct:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(Y = y | X = x)$$

- Notice: they're the same!  This is called the MPE=MAP rule.

# 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, $\hat{y}$, is not equal to the correct label, $y$, averaged over all possible $x$:

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

# 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.

# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, development test, and evaluation test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# Accuracy

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

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

# 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 $m^{\text{th}}$ class
- that were labeled, by the classifier, as belonging to the $n^{\text{th}}$ 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 |

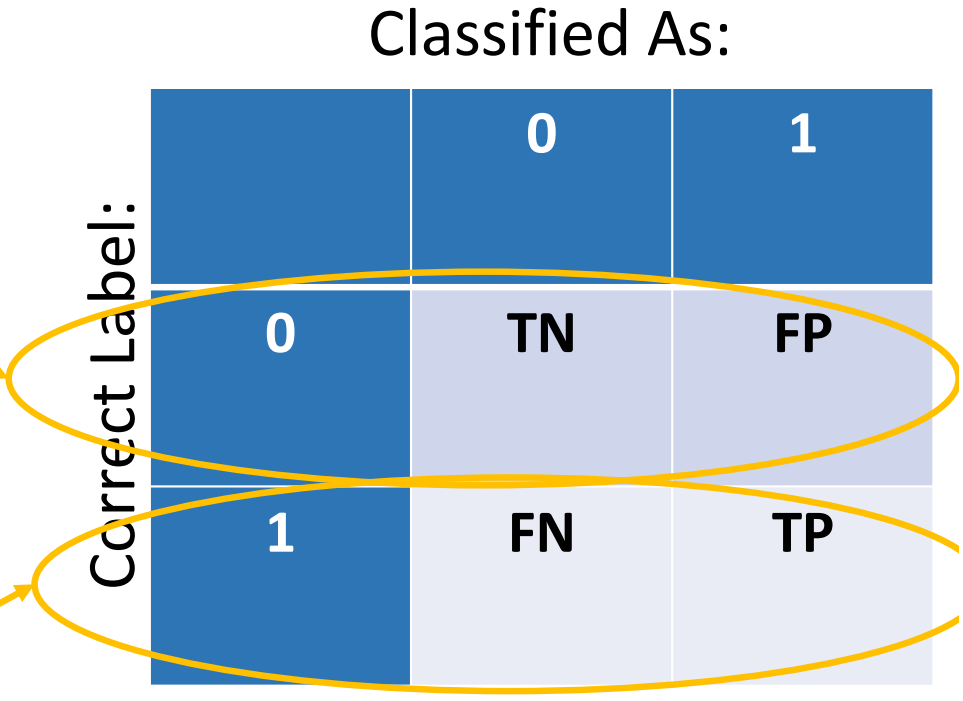
# False Alarms and Missed Detections

The **false alarm rate** of a classifier is the fraction of tokens that should have been 0, but were misclassified as 1:

$$\text{False Alarm Rate} = \frac{FP}{TN + FP}$$

The **missed detection rate** of a classifier is the fraction of tokens that should have been 1, but were misclassified as 0:

$$\text{Missed Detection Rate} = \frac{FN}{TP + FN}$$



# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, development test, and evaluation test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# 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).

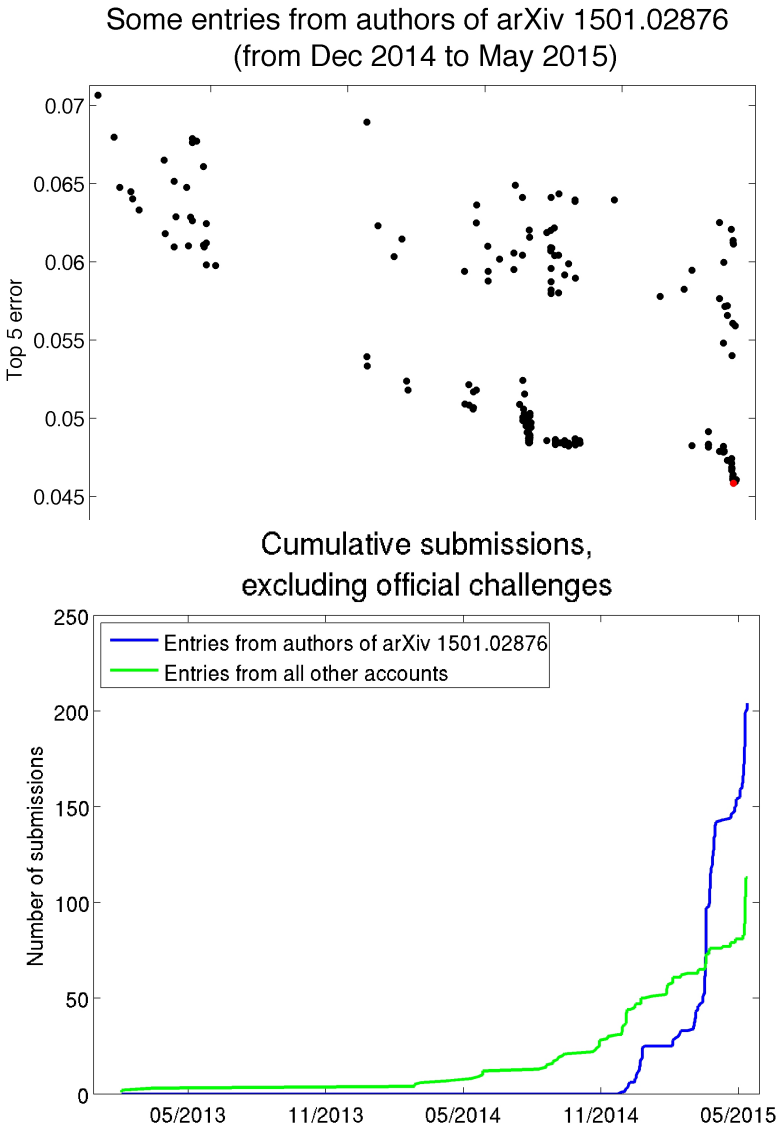
**Test Corpus** = a set of data that is non-overlapping with the training set (none of the test tokens are also in the training dataset) that you can use to measure the accuracy.

- 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.
- That institution was forbidden from participating in the ImageNet competitions for the following 12 months.



# 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), in order to see which one works 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.

# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, Development Test, and Evaluation Test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# Nearest Neighbors Classifier

- Given n different **training tokens**. Each one has a known class label.

- Input to the classifier: a **test token** $x$ whose correct label is unknown.
- Classification function:
  1. Find the training token, $x_i$, that is most similar to the test token.
  2. Find out the corresponding class label, $y_i$ =correct_label($x_i$).
  3. Output $y_i$ as the best guess for the label of test token $x$.

# Example of Nearest-Neighbor Classification

Test Token: Maltese



This is the most similar training token…

Therefore the Maltese is classified as a dog.

Training Tokens:



# K-Nearest Neighbors (KNN) Classifier

The nearest-neighbors classifier sometimes fails if one of the training tokens is unusual. In that case, a test token that is similar to the weird training token might get misclassified. Solution: K-Nearest Neighbors.

Test token:

Mandruss, CC BY-SA 4.0

Most similar training token:

DK1k, CC BY-SA 4.0

## K-Nearest Neighbors Classification Function

1. Find the K training tokens, $x_i$, that are most similar to the test token (K is a number chosen in advance by the system designer, e.g., $K = 3$).
2. Find out the corresponding class labels, $y_i$ =correct_label($x_i$).
3. Vote! Find the class label that is most frequent among the K-nearest neighbors, and output that as the label of the test token.

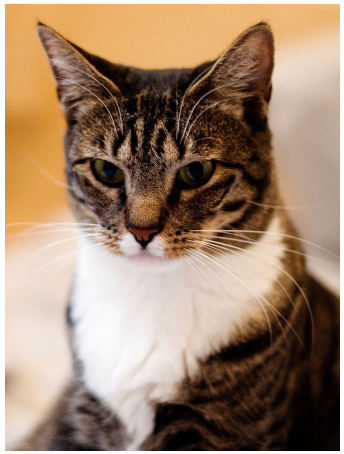
Test token:

Mandruss, CC BY-SA 4.0

3 most similar training tokens:

DK1k, CC BY-SA 4.0

Mike Powell, CC BY-SA 2.0



Dustin Warrington, CC BY-SA.

# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, Development Test, and Evaluation Test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

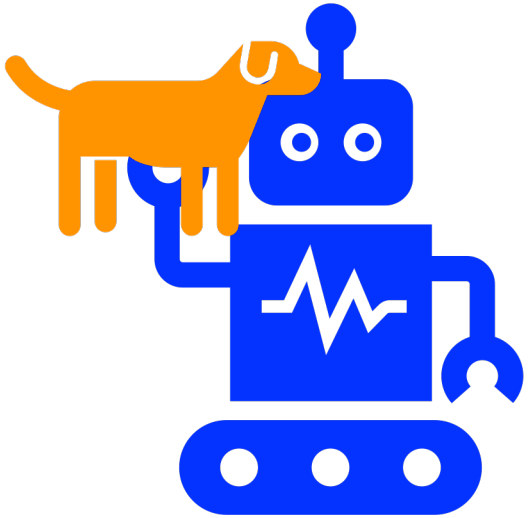
# Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

Idea #1:  Cats are smaller than dogs.

Our robot will pick up the animal and weigh it.

If it weighs more than 20 pounds, call it a dog.  Otherwise, call it a cat.



# Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

Oops.

# Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?

Idea #2: Dogs are tame, cats are wild.

We'll try the following experiment: 40 different people call the animal's name. Count how many times the animal comes when called.

If the animal comes when called, more than 20 times out of 40, it's a dog.

If not, it's a cat.

# Classifier example: dogs versus cats

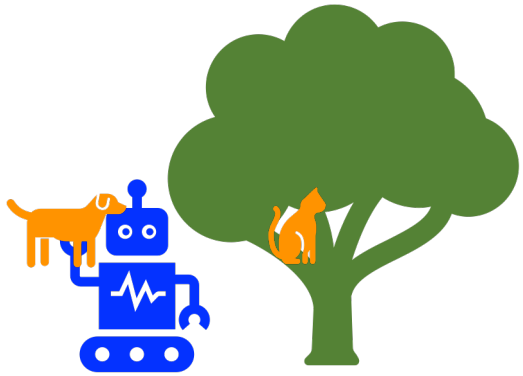
Can you write a program that can tell which ones are dogs, and which ones are cats?

Oops.

By Smok Bazyli - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=16864492

# Classifier example: dogs versus cats

Can you write a program that can tell which ones are dogs, and which ones are cats?



Idea #3:

$x_1$ = # times the animal comes when called (out of 40).

$x_2$ = weight of the animal, in pounds.

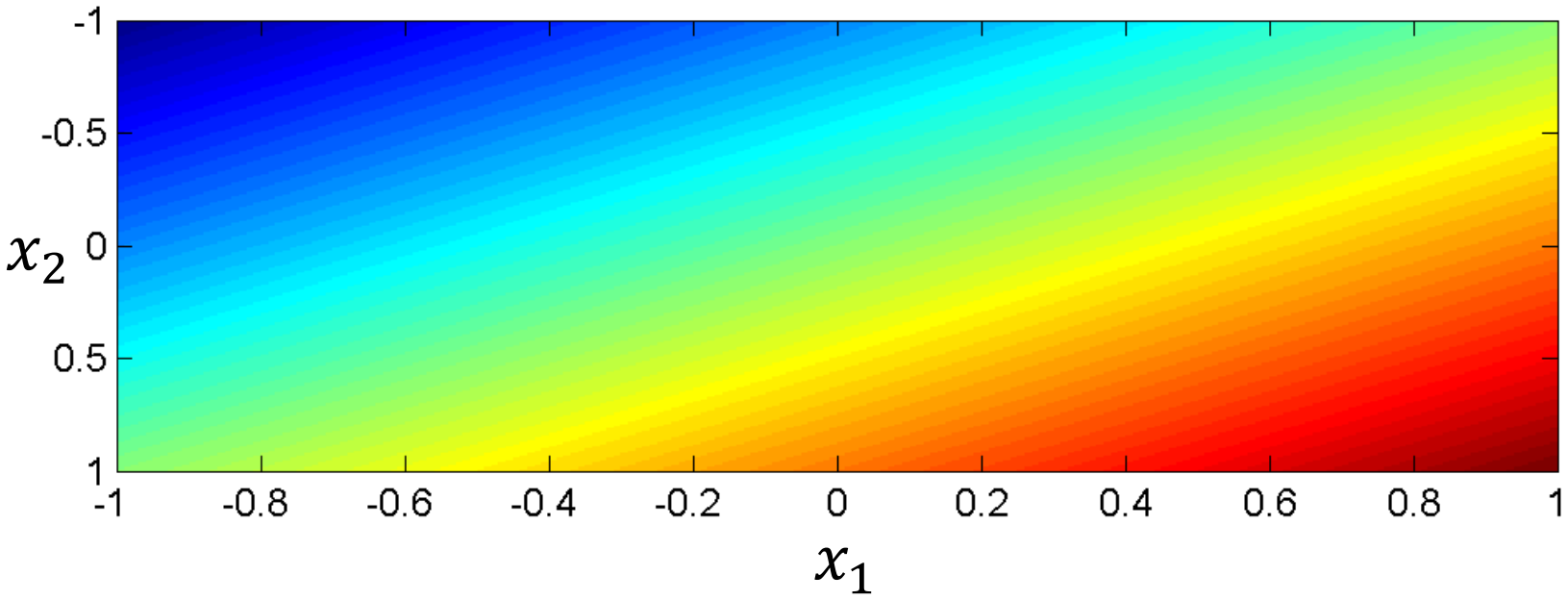
If $0.5x_1 + 0.5x_2 > 20$, call it a dog.

Otherwise, call it a cat.

This is called a "linear classifier" because $0.5x_1 + 0.5x_2 = 20$ is the equation for a straight line.

# Linear Classifiers in General

The function $b + \sum_{j=1}^{D} w_j x_j$ is an affine function of the features $x_j$. That means that its contours are all straight lines. Here is an example of such a function, plotted as variations of color in a two-dimensional space $x_1$ by $x_2$:

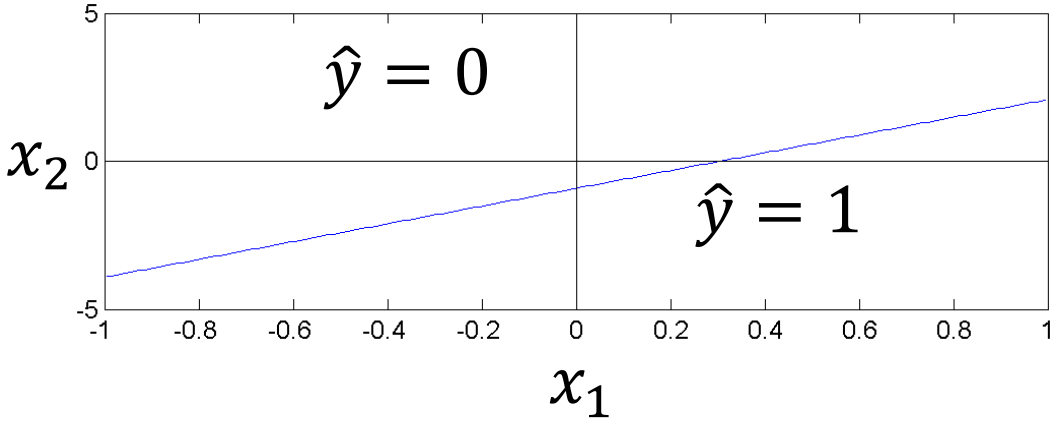


# Linear Classifiers in General

Consider the classifier

$$\hat{y} = 1 \quad \text{if} \quad b + \sum_{j=1}^{D} w_j x_j > 0, \qquad \text{otherwise } \hat{y} = 0$$

This is called a "linear classifier" because the boundary between the two classes is a line. Here is an example of such a classifier, with its boundary plotted as a line in the two-dimensional space $x_1$ by $x_2$:

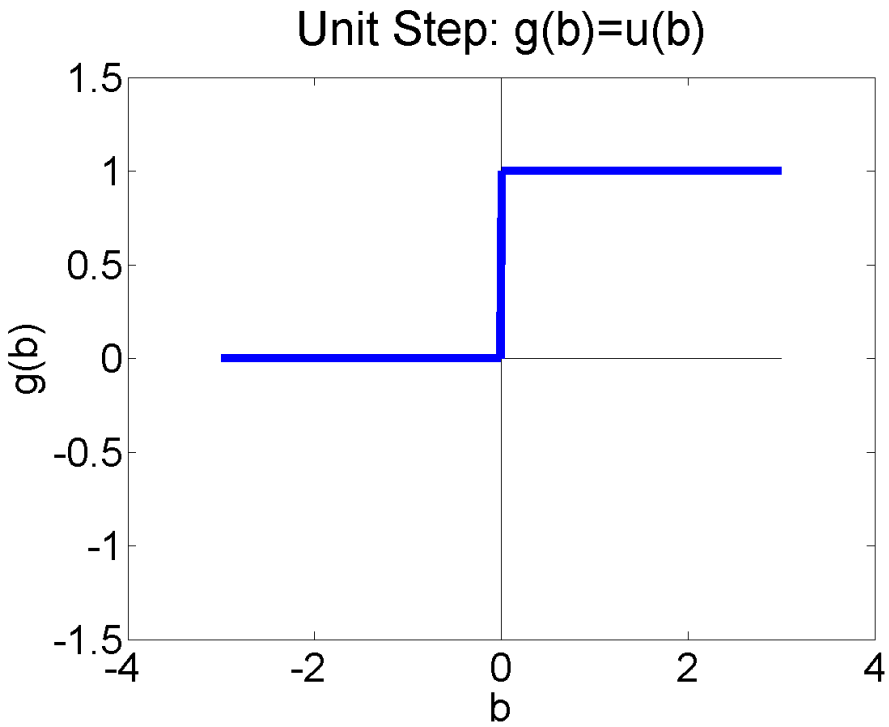


# The Unit Step Function

## Binary Classifier: Unit Step

The unit step function is defined as:

$$u(b) = \begin{cases} 1 & b \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

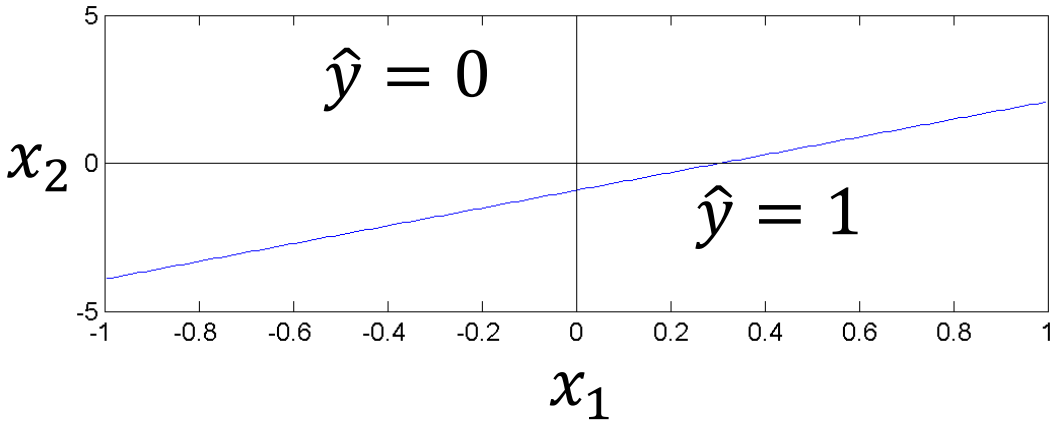


# Linear Classifiers in General

Consider the classifier

$$\hat{y} = u\left(b + \sum_{j=1}^{D} w_j x_j\right)$$

This is called a "linear classifier" because the boundary between the two classes is a line.



# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, Development Test, and Evaluation Test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

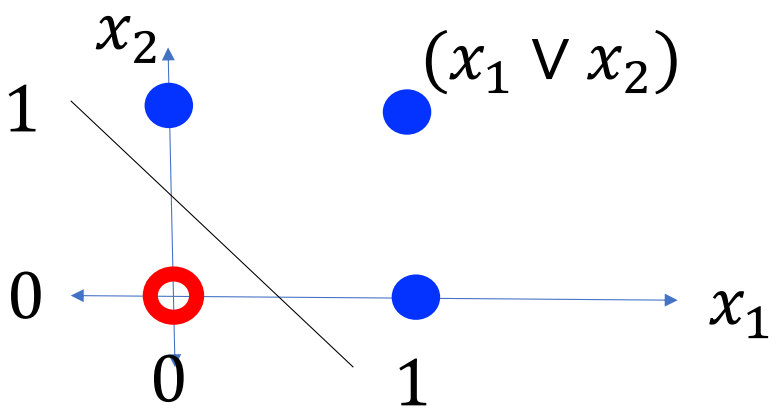
# Many logic functions are linear classifiers!

Many (but not all!) binary logic functions can be re-written as linear classifiers. For example, the function

$$\hat{y} = (x_1 \vee x_2)$$

can be re-written as

$$\hat{y} = u(x_1 + x_2 - 0.5)$$

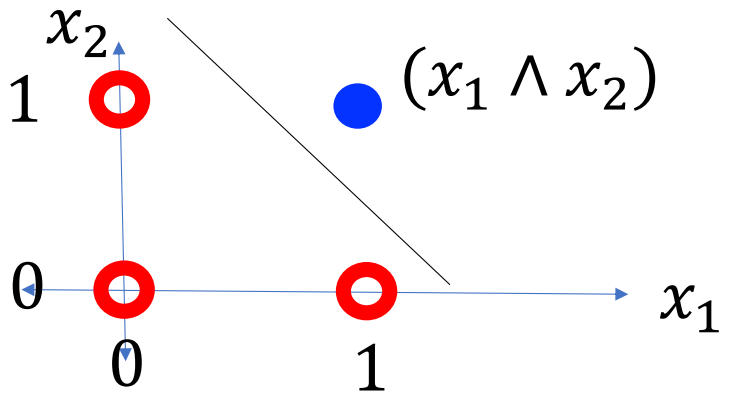



Similarly, the function

$$\hat{y} = (x_1 \wedge x_2)$$

can be re-written as

$$\hat{y} = u(x_1 + x_2 - 1.5)$$

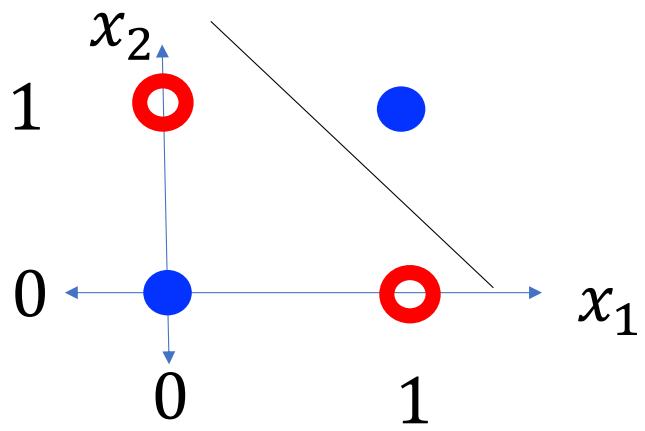


# … but not all logic functions are linear.

- Not all logical functions can be written as linear classifiers!
- Minsky and Papert wrote a book called *Perceptrons* in 1969. Although the book said many other things, the only thing most people remembered about the book was that:

**"A linear classifier cannot learn an XOR function."**

- Because of that statement, most people gave up working on neural networks from about 1969 to about 2006.
- Minsky and Papert also proved that a two-layer neural net can learn an XOR function. But most people didn't notice.



# Classifiers

- What is a classifier?
- The Naïve Bayes classifier
- General Bayesian classifier
- False alarms and missed detections
- Training, Development Test, and Evaluation Test datasets
- The nearest-neighbor classifier
- Linear classifiers
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!

# Naïve Bayes as a Classifier

Remember that Naïve Bayes can be written as a classifier, $\hat{y} = f(x)$.

1. A particular test token has the features $x = [x_1, \ldots, x_D]$.
2. For the classifier output, $\hat{y}$, we want to choose the value of $y$ that maximizes the probability:

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(Y = y) \prod_{d=1}^{D} P(X_d = x_d | Y = y)$$

# Example: Binary Naïve Bayes

For example, suppose all of the features are binary, $x_d \in \{0,1\}$. Then:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(Y = y) \prod_{d=1}^{D} \big(P(X_d = 1|Y = y)\big)^{x_d} \big(P(X_d = 0|Y = y)\big)^{1-x_d}$$

# Example: Binary Naïve Bayes

Suppose the class labels are also binary, $y \in \{0,1\}$. Then:

$\hat{y} = 1$ if

$$P(Y=1)\prod_{d=1}^{D}\left(P(X_d=1|Y=1)\right)^{x_d}\left(P(X_d=0|Y=1)\right)^{1-x_d}$$

$$\geq P(Y=0)\prod_{d=1}^{D}\left(P(X_d=1|Y=0)\right)^{x_d}\left(P(X_d=0|Y=0)\right)^{1-x_d}$$

Otherwise, $\hat{y} = 0$.

# Taking the logarithm

Multiplying together lots of terms can cause floating-point problems on a computer. Let's take the logarithms of both sides, in order to turn the multiplications into additions:

$\hat{y} = 1$ if

$$\log P(Y = 1) + \sum_{d=1}^{D} x_d \log P(X_d = 1|Y = 1) + (1 - x_d) \log P(X_d = 0|Y = 1)$$

$$\geq \log P(Y = 0) + \sum_{d=1}^{D} x_d \log P(X_d = 1|Y = 0) + (1 - x_d) \log P(X_d = 0|Y = 0)$$

Otherwise, $\hat{y} = 0$.

# Simplifying the equation

The equation is getting much too long. Let's re-arrange, to reduce the number of terms.

$\hat{y} = 1$ if

$$\log\left(\frac{P(Y=1)P(X_1=0|Y=1)\cdots P(X_D=0|Y=1)}{P(Y=0)P(X_1=0|Y=0)\cdots P(X_D=0|Y=0)}\right)$$
$$+\sum_{d=1}^{D} x_d \log\left(\frac{P(X_d=1|Y=1)P(X_d=0|Y=0)}{P(X_d=0|Y=1)P(X_d=1|Y=0)}\right)$$
$$\geq 0$$

Otherwise, $\hat{y} = 0$.

# New names for the model parameters

Now let's give new names to the model parameters.

$\hat{y} = 1$ if

$$b + \sum_{d=1}^{D} x_d w_d \geq 0$$

Otherwise, $\hat{y} = 0$.

# Binary Naïve Bayes is a Linear Classifier

A “linear classifier” is a classifier of the form:

$\hat{y} = 1$ if

$$b + \sum_{d=1}^{D} x_d w_d \geq 0$$

Otherwise, $\hat{y} = 0$.

Binary Naïve Bayes is a linear classifier.

# Binary Naïve Bayes is a Linear Classifier

Why did we go through this complicated derivation?

1. In order to use naïve Bayes in any practical situation (e.g., MP2), you need to know that logarithms turn multiplication into addition.
2. You need to know that naïve Bayes has two types of trainable parameters:
   a. The priors, $P(Y = y)$
   b. The likelihoods, $P(X_d = x_d | Y = y)$
3. You need to know that a linear classifier has two types of trainable parameters:
   a. The offset, $b$
   b. The weights, $w_d$

# How to learn a linear classifier

So far, you've learned three different ways to find the parameters of a linear classifier:

1. Plot scatter plots of $x$, for $y = 0$ and $y = 1$, and draw a line, or
2. Plot the outcomes of a binary logic function and draw a line, or
3. Convert naïve Bayes parameters into linear classifier parameters.

Next week, you'll learn two new methods:

4. Perceptron algorithm, and
5. Logistic regression, a.k.a. one-layer neural network.

# Classifiers

- What is a classifier? Answer: $\hat{y} = f(x)$
- The Naïve Bayes classifier: $\hat{y} = \underset{y}{\operatorname{argmax}} P(y) \prod_{d=1}^{D} P(x_d|y)$
- General Bayesian classifier: $\hat{y} = \underset{y}{\operatorname{argmax}} P(y|x)$
- False alarm rate $= \frac{FP}{TN+FP}$, missed detection rate $= \frac{FN}{TP+FN}$
- Training, development test, and evaluation test datasets
- The nearest-neighbor classifier
- Linear classifier: $\hat{y} = u\left(b + \sum_{d=1}^{D} w_d x_d\right)$
- Many logic functions are linear classifiers!
- Binary naïve Bayes is a linear classifier!