## Logistic Regression

## Predicting Coin Flips?

* Imagine the problem of predicting probability of Heads for bent coins
* You might use features like angle of bend, coin mass, etc.
* What's the simplest model you could use?
* What could go wrong?



**Logistic Regression**

* Many problems require a probability estimate as output
* Enter **Logistic Regression**
* Handy because the probability estimates are **calibrated**
  + for example, p(house will sell) \* price = expected outcome
* Also useful for when we need a binary classification
  + spam or not spam? → p(Spam)

## Logistic Regression -- Predictions

y′=1/1+e^−(wTx+b)

Where: x: Provides the familiar linear model 1+e^−(...): Squish through a sigmoid

## LogLoss Defined

LogLoss=∑(x,y)∈D−ylog(y′)−(1−y)log(1−y′)

**Logistic Regression and Regularization**

* Regularization is super important for logistic regression.
  + Remember the asymptotes
  + It'll keep trying to drive loss to 0 in high dimensions
* Two strategies are especially useful:
  + **L2 regularization** (aka L2 weight decay) - penalizes huge weights.
  + **Early stopping** - limiting training steps or learning rate.

**Linear Logistic Regression**

* Linear logistic regression is extremely efficient.
  + Very fast training and prediction times.
  + Short / wide models use a lot of RAM.

# Logistic Regression: Calculating a Probability



**Estimated Time:** 10 minutes

Many problems require a probability estimate as output. Logistic regression is an extremely efficient mechanism for calculating probabilities. Practically speaking, you can use the returned probability in either of the following two ways:

* "As is"
* Converted to a binary category.

Let's consider how we might use the probability "as is." Suppose we create a logistic regression model to predict the probability that a dog will bark during the middle of the night. We'll call that probability:

p(bark|night)

If the logistic regression model predicts p(bark|night)=0.05, then over a year, the dog's owners should be startled awake approximately 18 times:

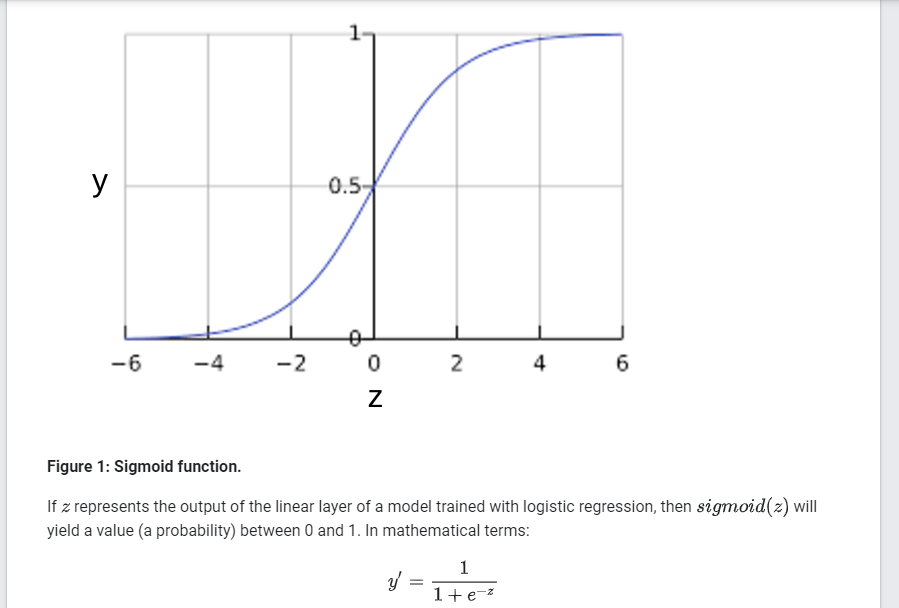
startled=p(bark|night)⋅nights=0.05⋅365 =18

In many cases, you'll map the logistic regression output into the solution to a binary classification problem, in which the goal is to correctly predict one of two possible labels (e.g., "spam" or "not spam"). A later [module](https://developers.google.com/machine-learning/crash-course/classification/video-lecture) focuses on that.

You might be wondering how a logistic regression model can ensure output that always falls between 0 and 1. As it happens, a **sigmoid function**, defined as follows, produces output having those same characteristics:

y=1/1+e^−z

The sigmoid function yields the following plot:



where:

* y′ is the output of the logistic regression model for a particular example.
* z=b+w1x1+w2x2+…+wNxN
  + The w values are the model's learned weights, and b is the bias.
  + The x values are the feature values for a particular example.

Note that z is also referred to as the *log-odds* because the inverse of the sigmoid states that z can be defined as the log of the probability of the 1 label (e.g., "dog barks") divided by the probability of the 0 label (e.g., "dog doesn't bark"):

z=log⁡(y1−y)

