CS 232: Artificial Intelligence

Fall 2023

Prof. Carolyn Anderson
Wellesley College

YLLATAILY Discussion

Pick one of these concepts to discuss in your small group:

- Catastrophic Forgetting
- Adversarial Attacks
- Bias Amplification

Are these concepts relevant for today's state-of-the-art models like ChatGPT? Why or why not?

Recap:
Logistic Regression Classifiers

Idea of logistic regression

weighted sum

weights ↘ ↙ features

wx+b ↲

- Compute w·x+b ← bias term
- Pass it through the sigmoid function: σ(w · x+b) so that we can treat it as a probability

$$P(y=1) \;=\; \sigma(w \cdot x + b)$$

$$P(y=0) \;=\; 1 - \sigma(w \cdot x + b)$$

The two phases of logistic regression

Training: we learn weights *w* and *b* using **stochastic gradient descent** and **cross-entropy loss**.

Test: Given a test example *x* we compute *p*(*y*|*x*) using learned weights *w* and *b*, and return whichever label (*y* = 1 or *y* = 0) is higher probability

Logistic Regression Example: Text Classification

Slides borrowed from Jurafsky & Martin Edition 3

Sentiment example: does y=1 or y=0?

It's hokey . There are virtually no surprises , and the writing is second-rate . So why was it so enjoyable ?

For one thing , the cast is great .

Another nice touch is the music . I was overcome with the urge to get off the couch and start dancing . It sucked me in , and it'll do the same to you .

Features

	Definition	Value
x_1	count of positive words [great, fun... enjoyable]	3
x_2	count of negative words [bad, terrible ... boring]	2
x_3	$\begin{cases} 1 & \text{if "no" occurs} \\ 0 & \text{otherwise} \end{cases}$	1

Weights

$W = [4, -3, -1]$

$b = -0.5$

Classifying sentiment for input x

Weighted sum: $wx + b$

$$\overset{w_1 x_1}{=\ 4 \cdot 3} + \overset{w_2 x_2}{-3 \cdot 2} + \overset{w_3 x_3}{-1 \cdot 1} + \overset{b}{-0.5}$$

$$= 4.5$$

$$p(+|x) = \sigma(\text{weighted sum})$$

$$= \sigma(4.5)$$

$$= 0.989$$

$$p(-|x) = 1 - p(+|x)$$

$$= 0.011$$

98.9 % that the review is positive

Classification in (binary) logistic regression: summary

Given:

- a set of classes: (+ sentiment,- sentiment)
- a vector **x** of features `[x1, x2, …, xn]`
  - x1= ~~count("awesome")~~ count (positive)
  - x2 = ~~log(number of words in review)~~ count (negative)
- A vector **w** of weights `[w1, w2, …, wn]`
  - w_i for each feature f_i

$$P(y=1) = \sigma(w \cdot x + b) = \frac{1}{1+\exp(-(w \cdot x + b))}$$

Multi-class Regression

Slides borrowed from Jurafsky & Martin Edition 3

Multinomial Logistic Regression

Often we need more than 2 classes

- Positive/negative/neutral
- Parts of speech (noun, verb, adjective, adverb, preposition, etc.)
- Classify emergency SMSs into different actionable classes

If >2 classes we use **multinomial logistic regression**

= Softmax regression

= Multinomial logit

= Maximum entropy modeling or MaxEnt

So "logistic regression" means binary (2 classes)

Multinomial Logistic Regression

The probability of everything must still sum to 1

```
P(positive|doc) + P(negative|doc) +
P(neutral|doc) = 1
```

Need a generalization of the sigmoid called **softmax**

- Takes a vector $z = [z1, z2, ..., zk]$ of k arbitrary values
- Outputs a probability distribution

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)} \qquad 1 \leq i \leq k$$

The softmax function

Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of k arbitrary values into probabilities :

$$z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$$

$$\text{softmax}(z) = \left[\frac{\exp(z_1)}{\sum_{i=1}^{k} \exp(z_i)}, \frac{\exp(z_2)}{\sum_{i=1}^{k} \exp(z_i)}, \ldots, \frac{\exp(z_k)}{\sum_{i=1}^{k} \exp(z_i)} \right]$$

$$[0.055, 0.090, 0.006, 0.099, 0.74, 0.010]$$

Softmax in multinomial logistic regression

$$p(y=\overset{\text{action}}{\overset{\uparrow}{c}}|x) \;=\; \frac{\exp\left(\overset{\text{action}}{w_c}\cdot x+\overset{\text{action}}{b_c}\right)}{\displaystyle\sum_{j=1}^{K}\exp\left(\underset{\text{genre}}{\underset{\uparrow}{w_j}}\cdot x+\underset{\text{genre}}{\underset{\uparrow}{b_j}}\right)}$$

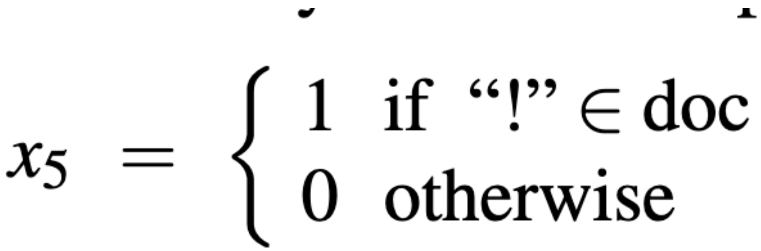
Input is still the dot product between weight vector *w* and input vector *x*, but now we need separate weight vectors for each of the *K* classes.

action = [-2, 3, 0]   romcoms = [-1, 5, 2]

anime = [7, -1, 4]   . . .

Features in binary versus multinomial logistic regression

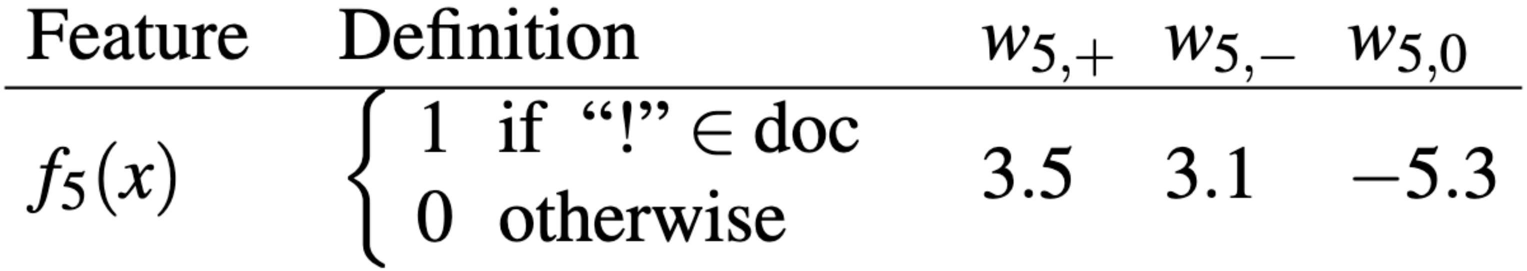
Binary: positive weight



$$x_5 = \begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$$

$$w_5 = 3.0$$

Multinominal: separate weights for each class:



Feature	Definition	$w_{5,+}$	$w_{5,-}$	$w_{5,0}$
$f_5(x)$	$\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$	3.5	3.1	−5.3

Feature Representations

Slides borrowed from Jurafsky & Martin Edition 3

What does recent English borrowing *ongchoi* mean?

Suppose you see these sentences:

- Ong choi is delicious **sautéed with garlic**.
- Ong choi is superb **over rice**
- Ong choi **leaves** with salty sauces

And you've also seen these:

- …spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

Conclusion:

- Ongchoi is a leafy green like spinach, chard, or collard greens
  - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

What kinds of contexts does the word occur in?

	garlic	**rice**	**green**	**leaves**
spinach	75	10	339	290
salmon	81	102	3	5
dinosaur	2	1	418	113
ongchoi	91	113	381	287

Ongchoi: *Ipomoea aquatica "Water Spinach'*

空心菜
kangkong
rau muống
…

Yamaguchi, Wikimedia Commons, public domain

Text Features

We represent text using word vectors.

Idea: a word meaning is based on its **distance** from other word meanings.

Each word = a vector (not just "good" or "w_{45}")

Similar words are "**nearby in semantic space**"

We build this space automatically by seeing which words are **nearby in text**

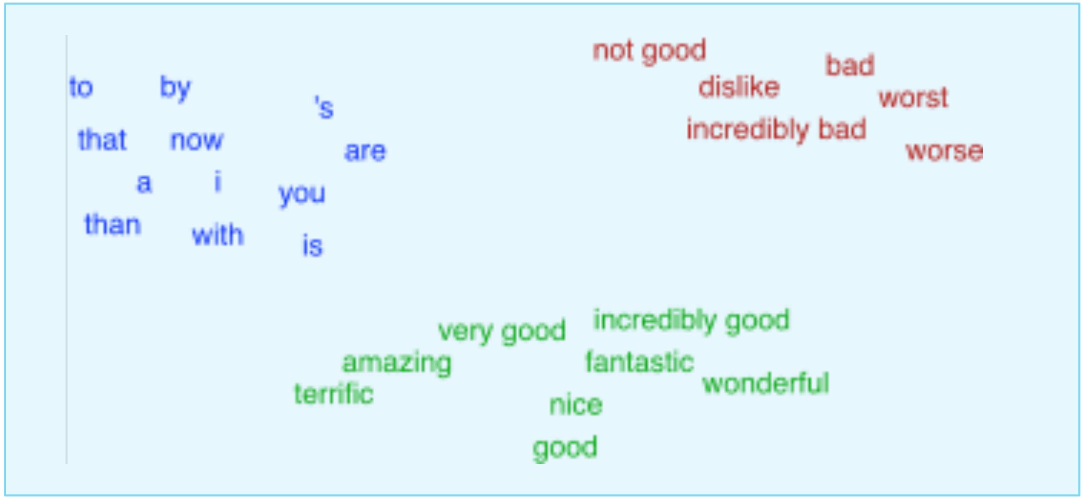


Image Features

For computer vision applications, we need a way of describing images. We represent images as matrices of pixel values.

Grayscale images can be represented with a single matrix.

Color images need to be represented with a **3D tensor** (3rd dimension encodes color channel).

Why matrices for images and vectors for text? Language is **sequential**, which makes it more useful to concatenate vectors lengthwise rather than stack them.

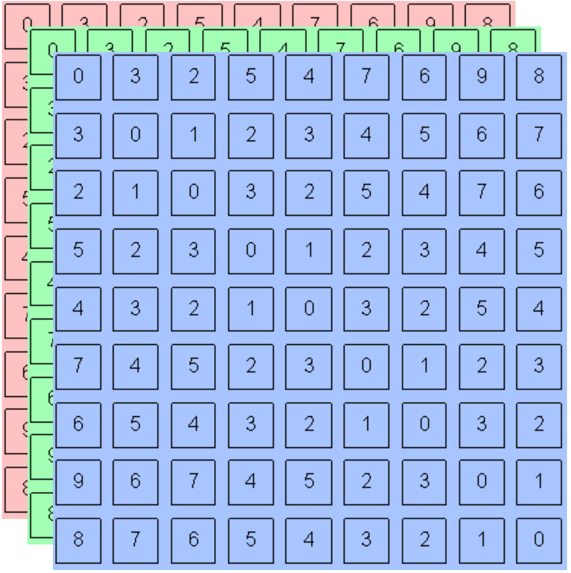
grayscale images are matrices

0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

what range of values can each pixel take?

Slides adapted from Mohit Iyyer

color images are tensors



0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

channel x height x width

Channels are usually RGB: Red, Green, and Blue

Other color spaces: HSV, HSL, LUV, XYZ, Lab, CMYK, etc

Slides adapted from Mohit Iyyer

Logistic Regression Example: Pet Picture Classification

Goal: Classify Pet Pictures

- Dataset: cat + dog pictures
- Goal: classify a picture as either a cat or a dog
- Input: grayscale images

Building a Model

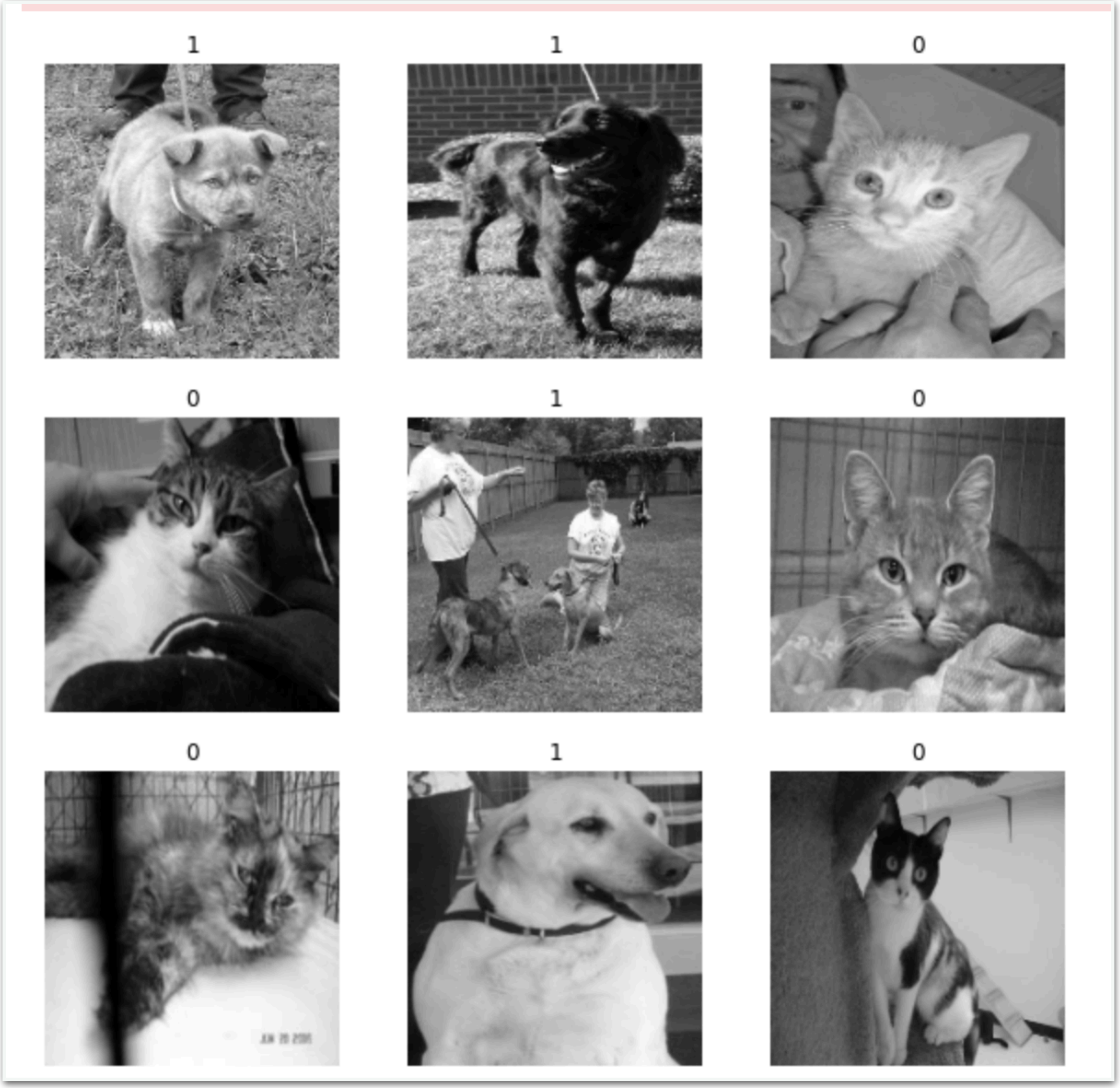
We'll build our model using a machine learning library called **Tensorflow**.

Tensorflow is a Python library, but most functions are implemented in C (so they are fast!).

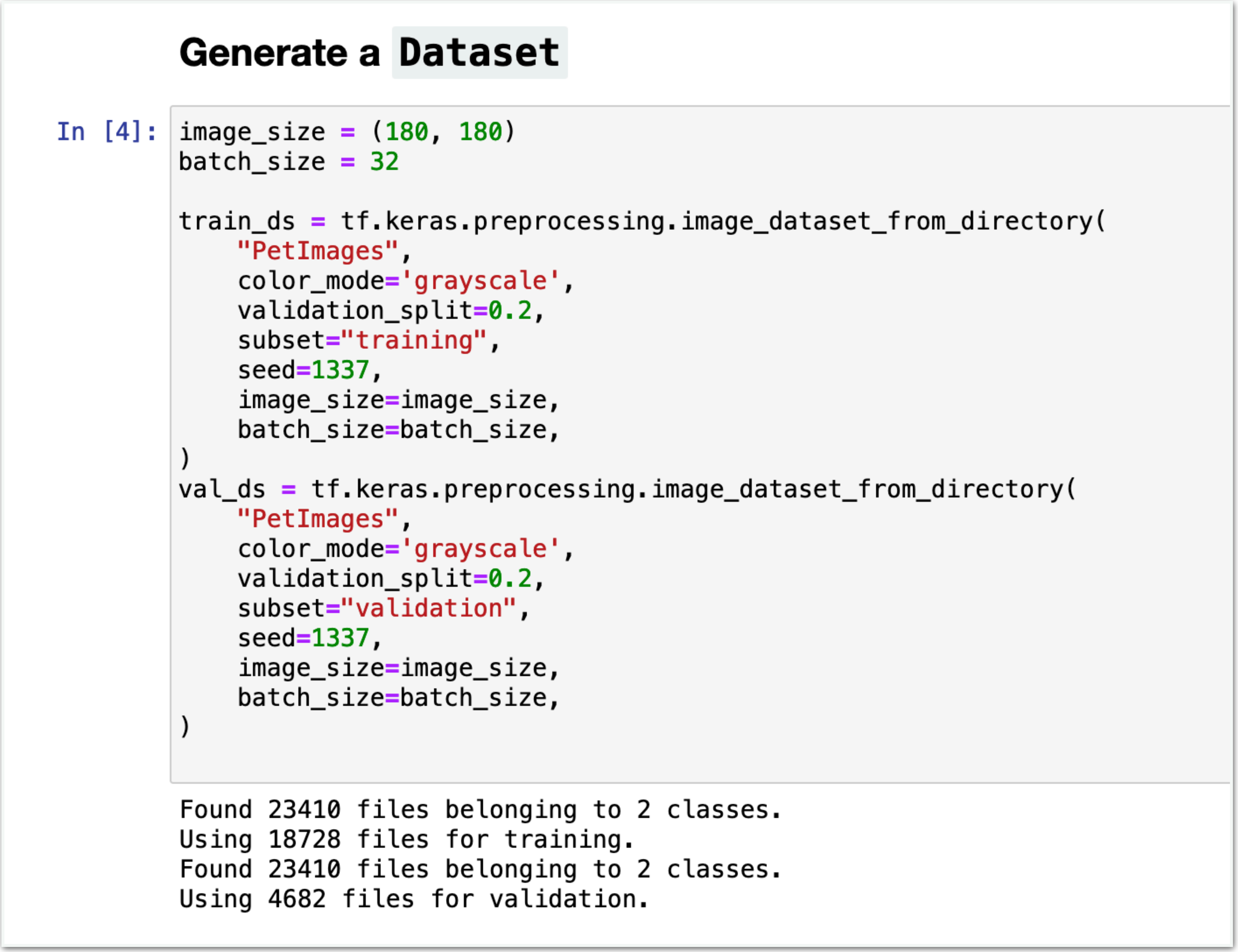
Tensorflow provides useful abstractions for models:

- **tensor**: n-dimensional container for data
- **layer**: apply functions to an input tensor of n dimensions to produce an output tensor of m dimensions.
- **model**: consist of layers connected together

Example Data



Splitting Our Data



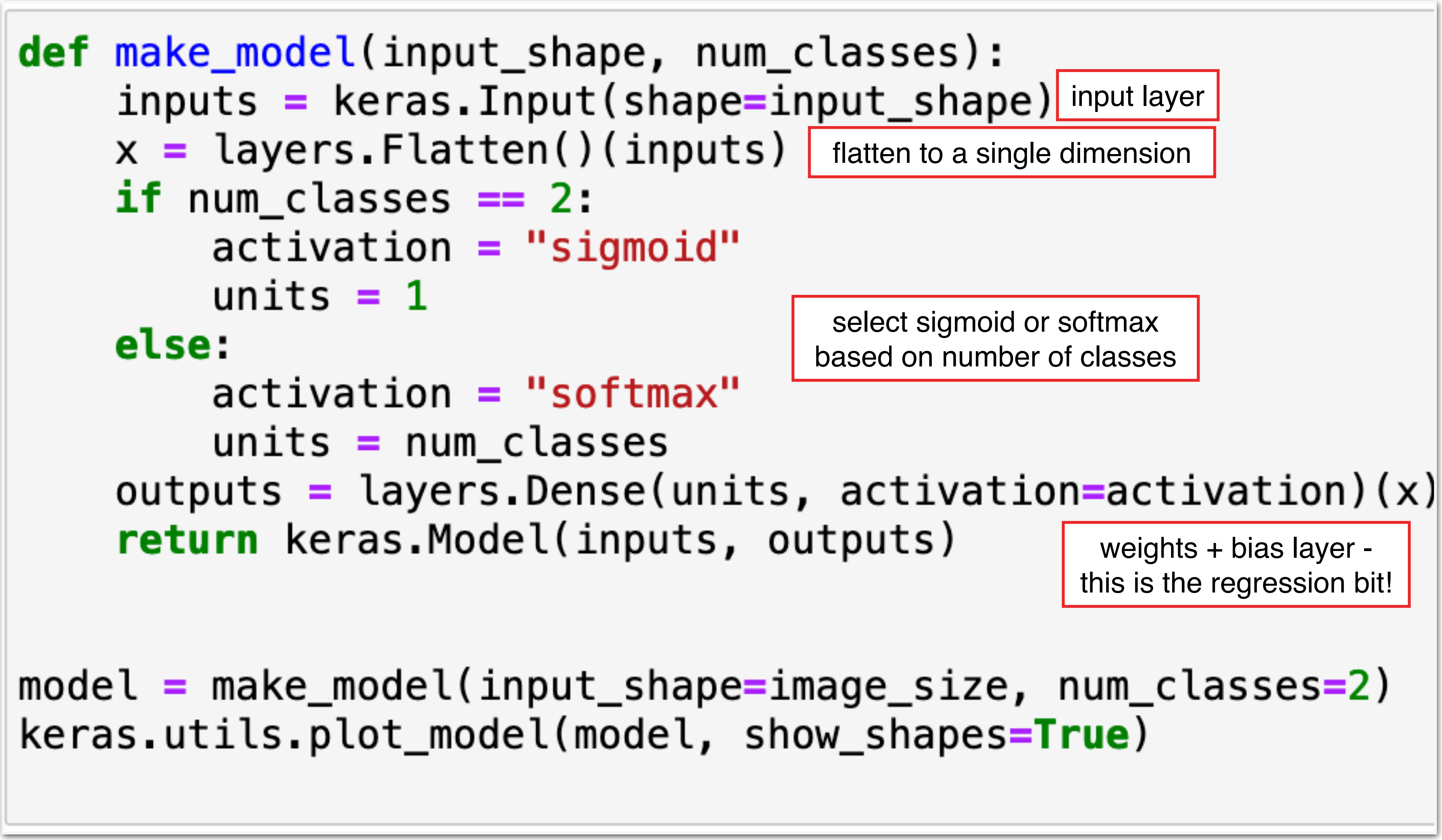
Generate a `Dataset`

In [4]:
```
image_size = (180, 180)
batch_size = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "PetImages",
    color_mode='grayscale',
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "PetImages",
    color_mode='grayscale',
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
)
```

```
Found 23410 files belonging to 2 classes.
Using 18728 files for training.
Found 23410 files belonging to 2 classes.
Using 4682 files for validation.
```

Creating Our Model Architecture



```
def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)
    x = layers.Flatten()(inputs)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)


model = make_model(input_shape=image_size, num_classes=2)
keras.utils.plot_model(model, show_shapes=True)
```

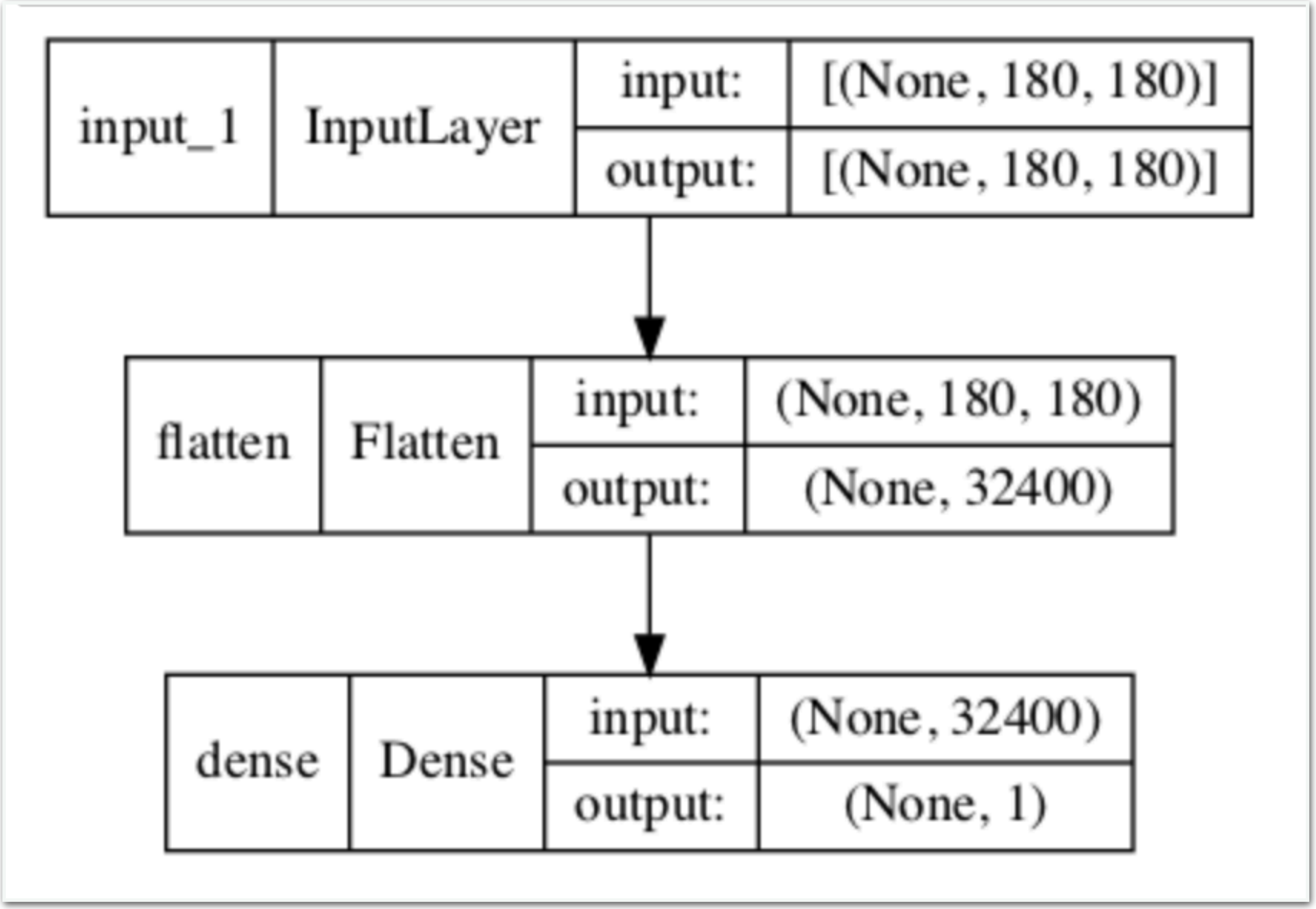
input layer

flatten to a single dimension

select sigmoid or softmax based on number of classes

weights + bias layer - this is the regression bit!

Creating Our Model Architecture

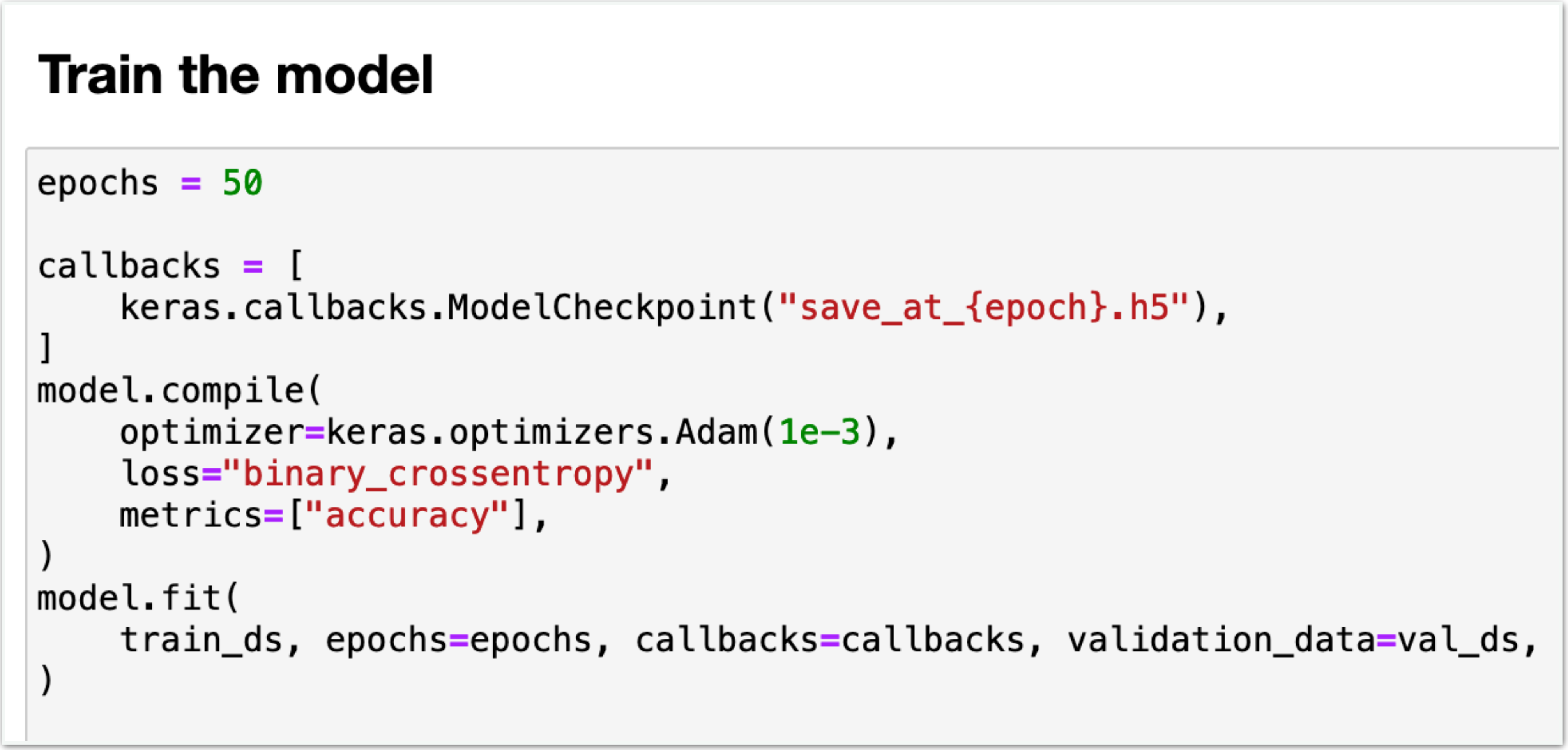


input_1	InputLayer	input:	[(None, 180, 180)]
		output:	[(None, 180, 180)]

flatten	Flatten	input:	(None, 180, 180)
		output:	(None, 32400)

dense	Dense	input:	(None, 32400)
		output:	(None, 1)

Training



Train the model

```
epochs = 50

callbacks = [
    keras.callbacks.ModelCheckpoint("save_at_{epoch}.h5"),
]
model.compile(
    optimizer=keras.optimizers.Adam(1e-3),
    loss="binary_crossentropy",
    metrics=["accuracy"],
)
model.fit(
    train_ds, epochs=epochs, callbacks=callbacks, validation_data=val_ds,
)
```

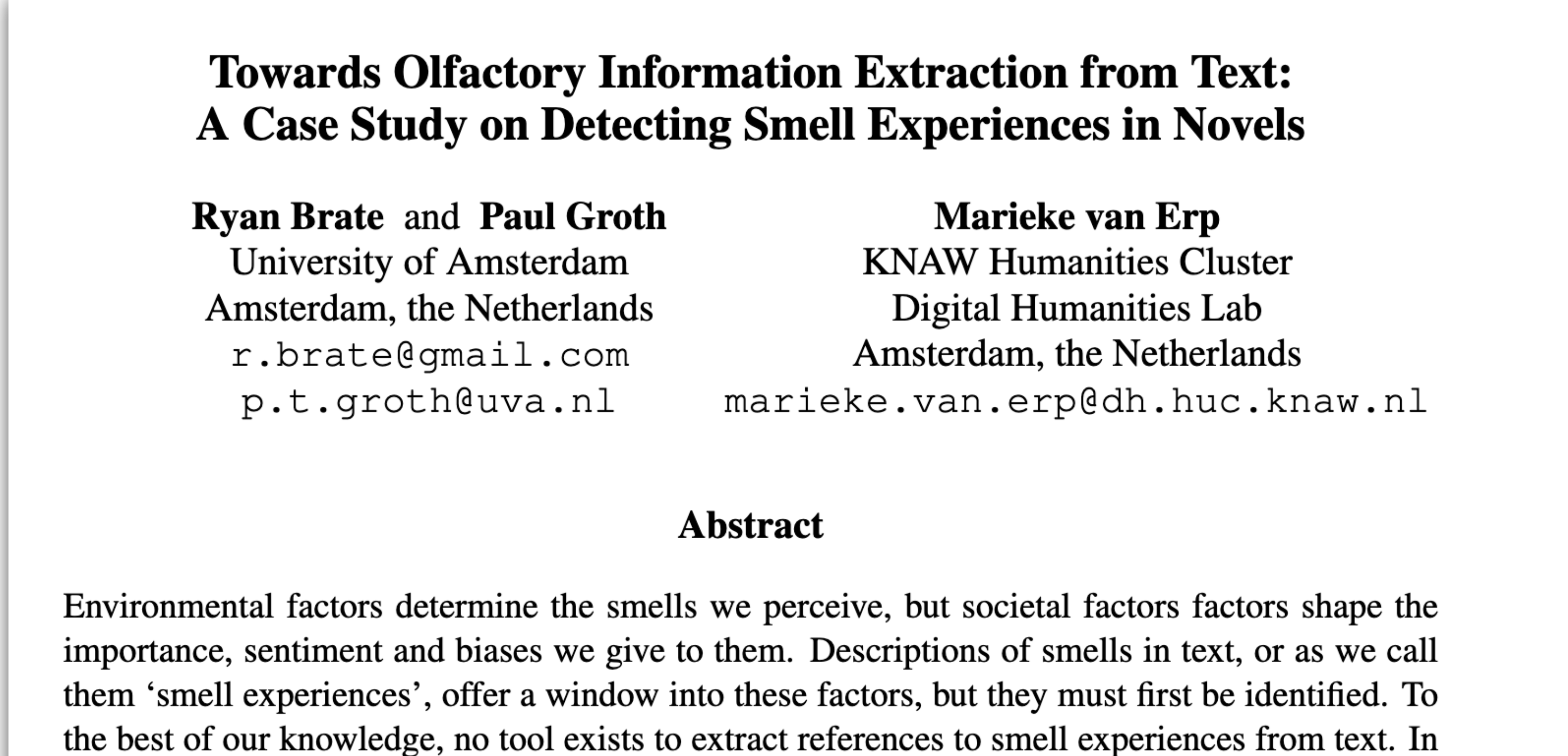
Evaluating Classifiers

Slides borrowed from Jurafsky & Martin Edition 3

Evaluation

Consider a binary text classification task:

Is this passage from a book a "smell experience" or not?



Towards Olfactory Information Extraction from Text: A Case Study on Detecting Smell Experiences in Novels

Ryan Brate and **Paul Groth**
University of Amsterdam
Amsterdam, the Netherlands
r.brate@gmail.com
p.t.groth@uva.nl

Marieke van Erp
KNAW Humanities Cluster
Digital Humanities Lab
Amsterdam, the Netherlands
marieke.van.erp@dh.huc.knaw.nl

Abstract

Environmental factors determine the smells we perceive, but societal factors factors shape the importance, sentiment and biases we give to them. Descriptions of smells in text, or as we call them 'smell experiences', offer a window into these factors, but they must first be identified. To the best of our knowledge, no tool exists to extract references to smell experiences from text. In

Evaluation

Consider a binary text classification task:

Is this passage from a book a "smell experience" or not?

You build a "smell" detector

- Positive class: paragraph that involves a smell experience
- Negative class: all other paragraphs

The 2-by-2 confusion matrix

Truth

Prediction

	🟩	🟥
	🟥	🟩

Evaluation: Accuracy

Why don't we use **accuracy** as our metric?

Imagine we saw 1 million paragraphs

- 100 of them mention smells
- 999,900 talk about something else

We could build a classifier that labels every paragraph "not about smell"

Evaluation: Accuracy

Why don't we use **accuracy** as our metric?

Imagine we saw 1 million paragraphs

- 100 of them mention smells
- 999,900 talk about something else

We could build a classifier that labels every paragraph "not about smell"

- It would get 99.99% accuracy!
- But the whole point of the classifier is to help literary scholars find passages about smell to study--- so this is useless!
- That's why we use **precision** and **recall** instead

Evaluation: Precision

% of items the system detected (i.e., items the system labeled as positive) that are in fact positive (according to the human gold labels)

PRECISION =

Evaluation: Recall

% of items actually present in the input that were correctly identified by the system.

RECALL =

Why Precision and recall

Our no-smells classifier

- Labels nothing as "about smell"

Accuracy =

Recall =

Precision =

Numerical Underflow

So far we've been working with relatively small sample spaces. This means our probabilities have been decently large.

As we go on in this class, our sample spaces are going to get much larger. We want to be able to reason about the probabilities of things like:

- All words in English
- All pixels in a photo
- All possible game states for Pacman

Solution: make the numbers bigger

- Intuition: we care about how big probabilities are relative to the other probabilities in our distribution, not the actual value.

Probabilities:
p(heart) = 0.1
p(rainbow) = 0.2
p(letter) = 0.7

Interpretation: a letter is **7 times more likely** than a heart!

Solution: make the numbers bigger

- Intuition: we care about how big probabilities are relative to the other probabilities in our distribution, not the actual value.

Probabilities:
p(heart) = ~~0.1~~ 100
p(rainbow) = ~~0.2~~ 200
p(letter) = ~~0.7~~ 700

What if we just multiply all our probs by 100?

This preserves the ratio.

Solution: make the numbers bigger

- What if we just multiply all our probs by 100? This preserves the ratio.

<u>Probabilities:</u>

p(heart) = ~~0.1~~ 100

p(rainbow) = ~~0.2~~ 200

p(letter) = ~~0.7~~ 700

However, if we want to recover the probabilities later, we'll need to **renormalize** them. This means **remembering that we multiplied by 100**.

Solution: log-transform the numbers

- Instead, we use a log transformation. This changes the range from [0,1] to [-∞, 0].

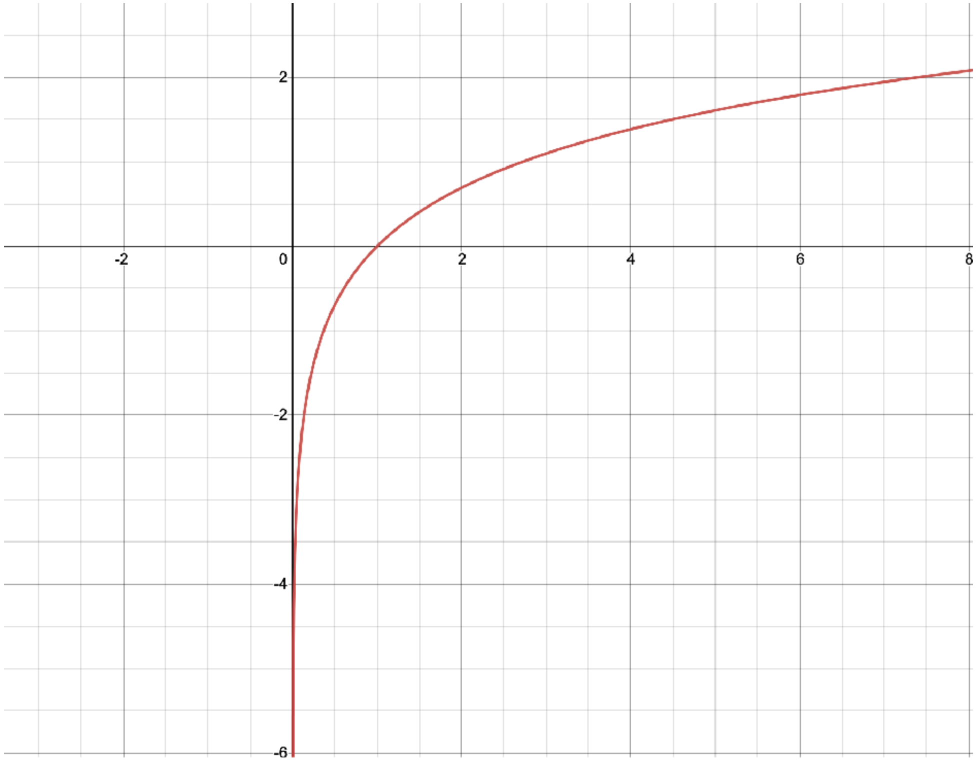
Log base doesn't matter much but we usually use natural log (base e):

<u>Probabilities:</u>

p(heart) = ~~0.1~~ -2.3

p(rainbow) = ~~0.2~~ -1.6

p(letter) = ~~0.7~~ -0.36



www.desmos.com/calculator/aczt76asao

Avoiding Harms in Classification

Slides borrowed from Jurafsky & Martin Edition 3

Harms in sentiment classifiers

Kiritchenko and Mohammad (2018) found that most sentiment classifiers assign lower sentiment and more negative emotion to sentences with African American names in them.

This perpetuates negative stereotypes that associate African Americans with negative emotions

Harms in toxicity classification

Toxicity detection is the task of detecting hate speech, abuse, harassment, or other kinds of toxic language

But some toxicity classifiers incorrectly flag as being toxic sentences that are non-toxic but simply mention identities like blind people, women, or gay people.

This makes it harder for members of these communities to connect, organize, and report mistreatment.

What causes these harms?

Can be caused by:

- Biases in training data (machine learning systems can amplify biases in their training data)
- Imbalance/lack of representation in training data
- Annotator bias
- Problems in the resources used (like lexicons)
- Problems in model architecture (like how the model's goal is defined)

Mitigation of these harms is an open research area