Applied Deep Learning, Fall 2019

Today's agenda

Administrative stuff

- New TAs / office hours posted
- HW2 walkthrough (posted early)

Bag of topics

- Famous incidents
- Linear classifiers
- Break / work on HW1 / or start on HW2
- Softmax, loss
- Numerical stability (prob next week)

Office hours

Benedikt Schifferer	Monday TBD	Mudd CS TA room
Su Ji Park	Tues 1:30pm - 3pm	Mudd CS TA room
Pratik Dubal	Wed 10am - 11:30am	Mudd CS TA room
Kunyan Han	Thurs 1:30pm - 3pm	Mudd CS TA Room
Josh Gordon	Thurs 5:30pm - 6:30pm	Mudd 417
Pengyu Chen	Fri 11am - 12:30pm	Mudd CS TA room

Questions from last time?

HW2 Walkthrough

Demo

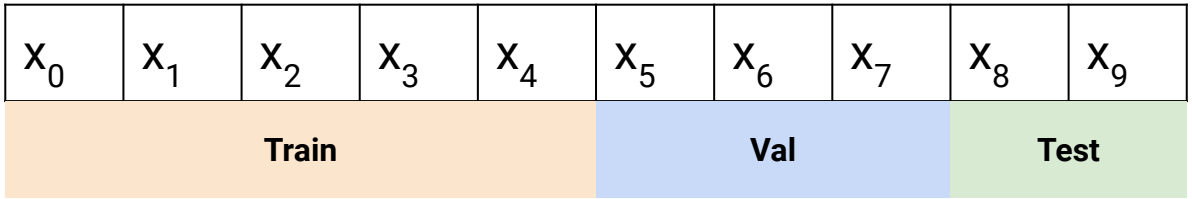
Optional, donate your dataset

here

Famous incidents

Warm up

Working w/ a complete dataset (e.g. from Kaggle)

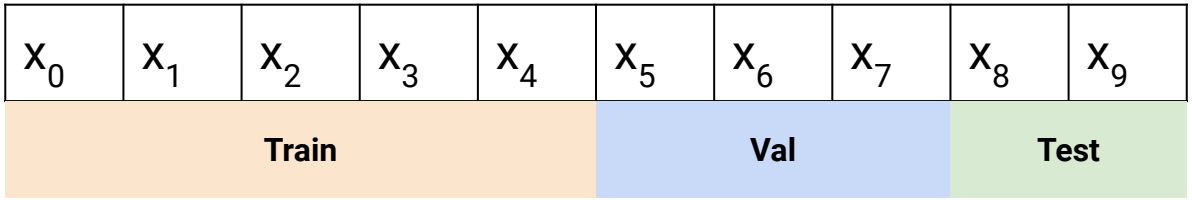


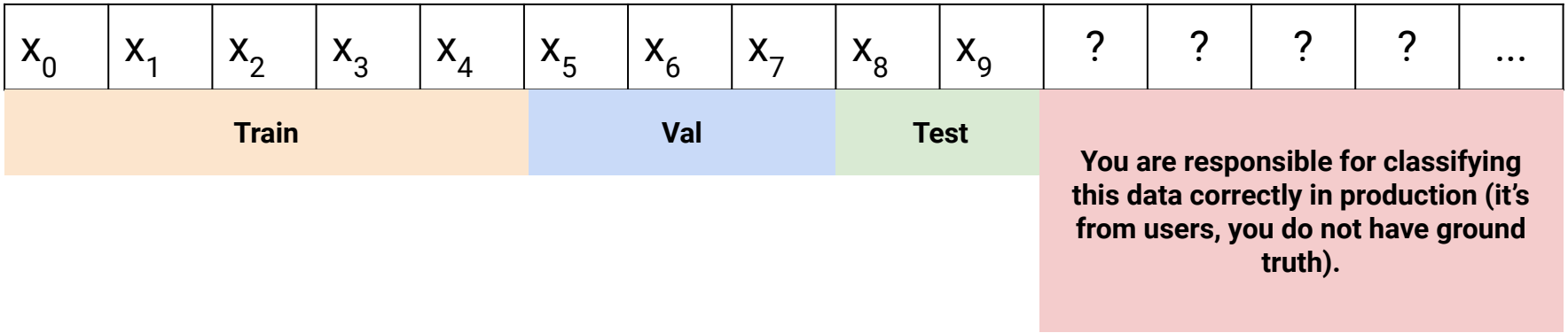
You have access to all the data your system will ever see.

...including a test set you're not supposed to look at more than once

… but probably will end up using a bunch of times.

Deploying a model

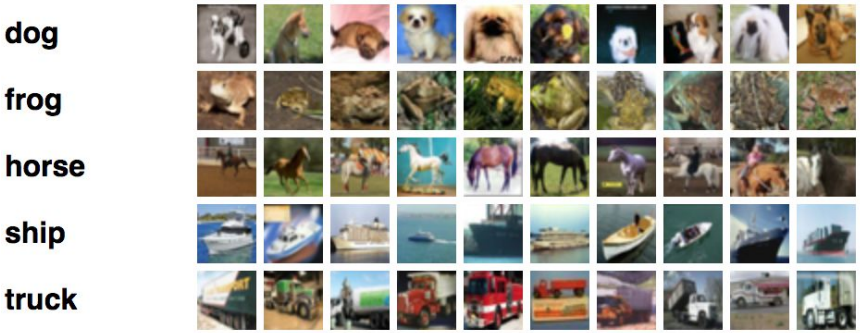




Quick discussion: how do you know if your system is working well?

Look at your data

Train / test / val accuracy



Dogs: 95%

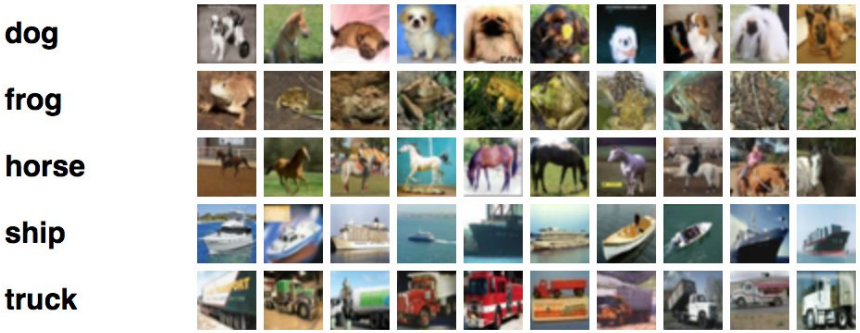
Frogs: 95%

Horse: 95%

Ship: 95%

Truck: 95%

Accuracy as reported by users



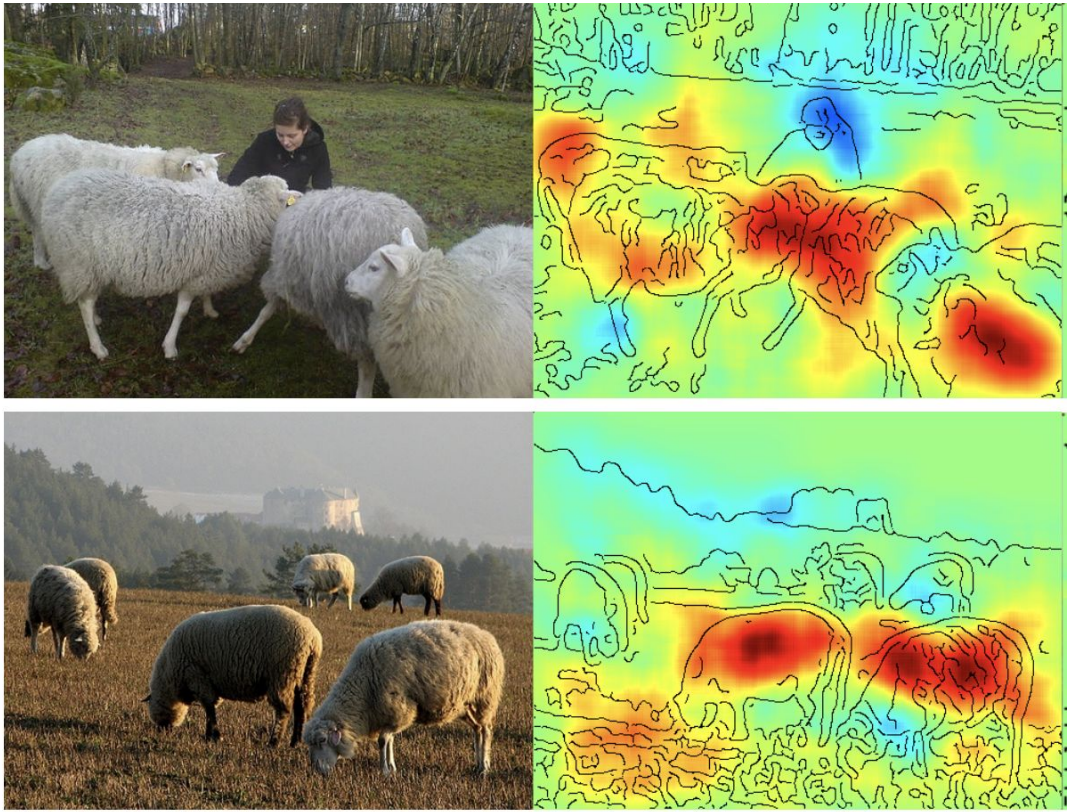
Dogs: 95%

Frogs: 95%

Horse: 0%

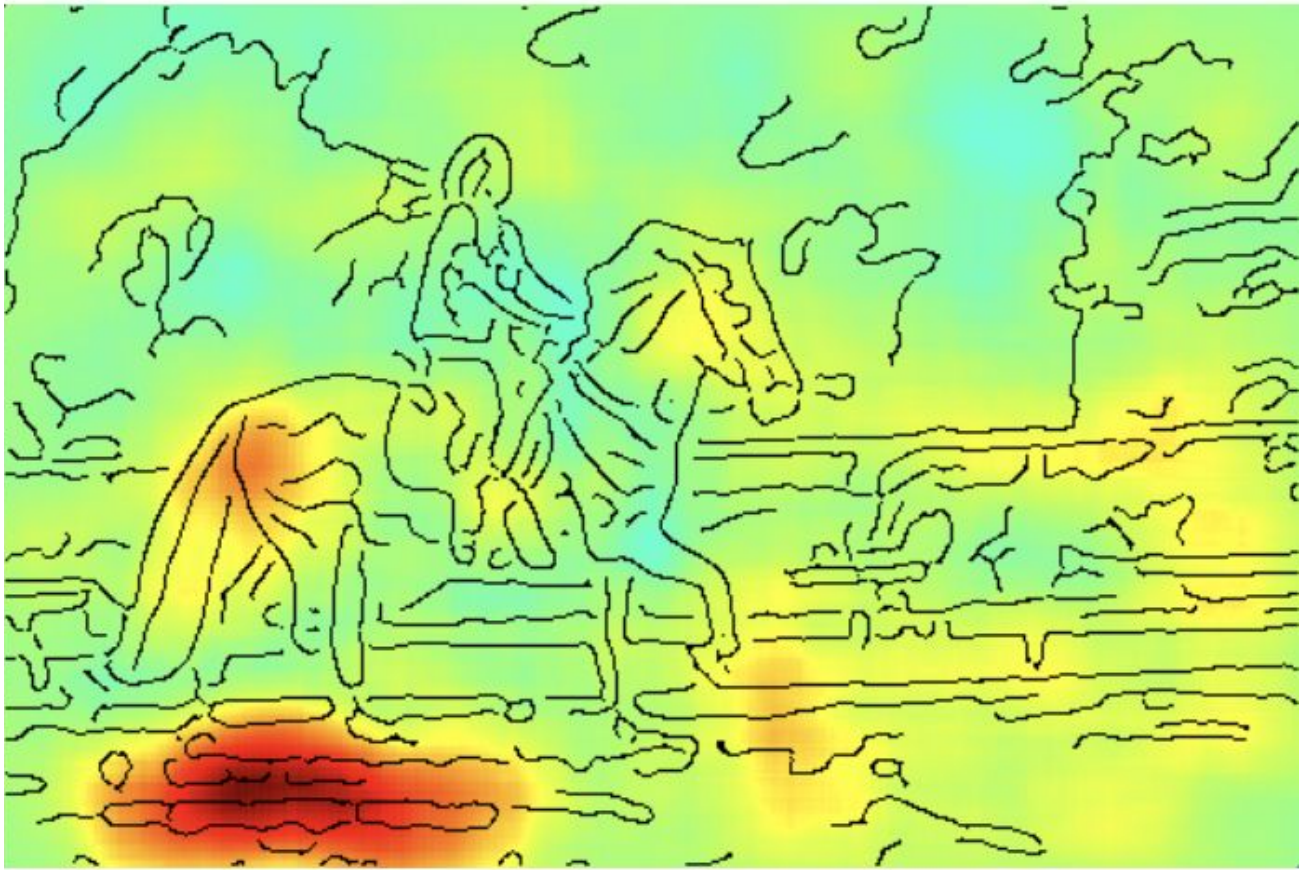
Ship: 95%

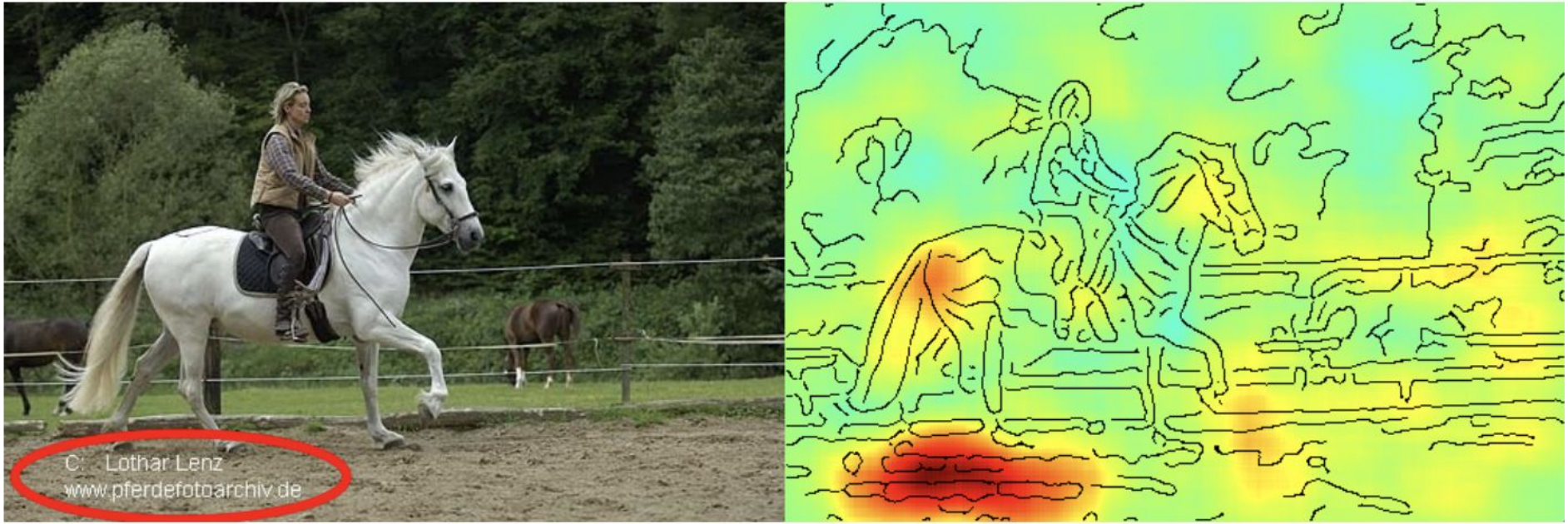
Truck: 95%



Analyzing Classifiers: Fisher Vectors and Deep Neural Networks (not suggesting you read this one right now, just an FYI with a great example).

Why might this region be important?





All horse images from the same photographer → the model conditions on the copyright.

Is your training set diverse?

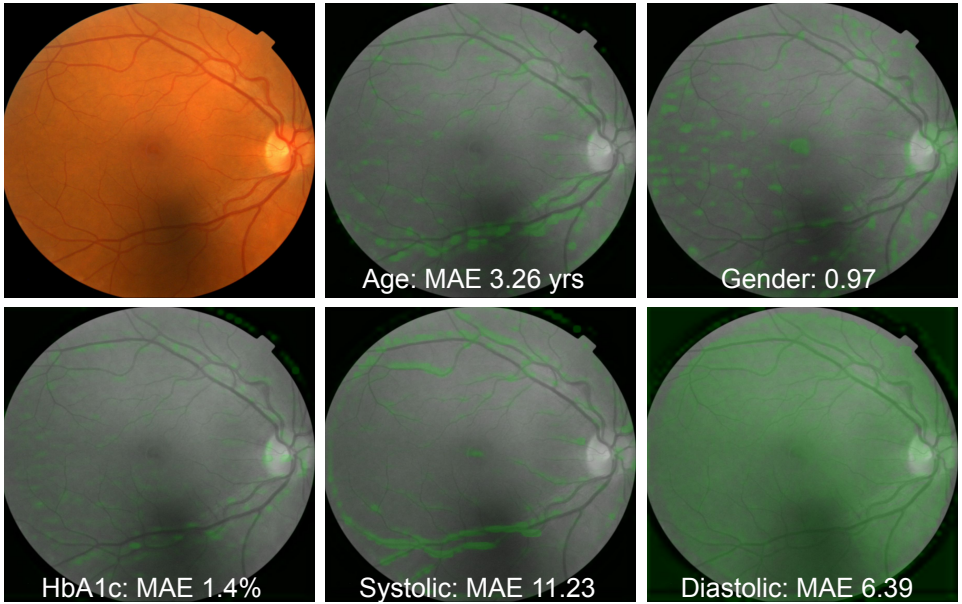
Why?

Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%

Quick discussion: why might this be happening?

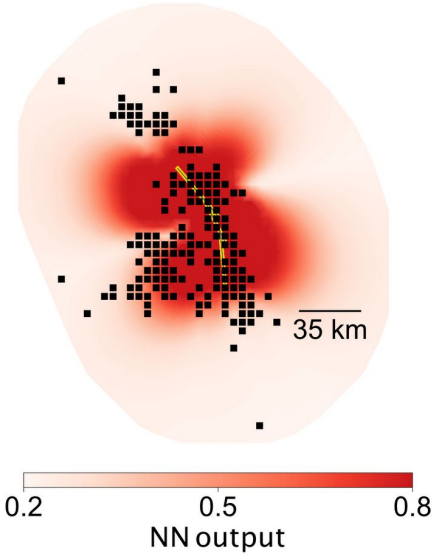
Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification

Last time: Supporting basic science



[Prediction of cardiovascular risk factors from retinal fundus photographs]

[Forecasting earthquake aftershock locations with AI-assisted science]

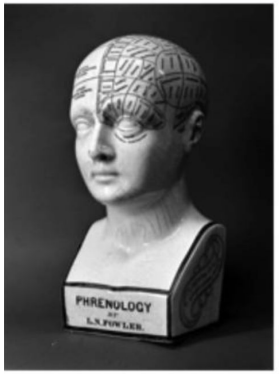


Forecasted distribution of aftershock location probabilities for the Landers earthquake. Dark red colors indicate regions predicted to experience aftershocks. Black dots are the locations of observed aftershocks. Yellow line shows the faults that ruptured during the mainshock.

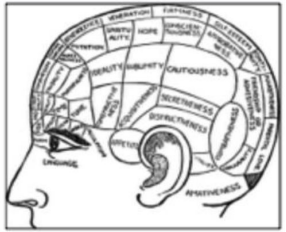
Echos of phrenology and physiognomy are sadly back.

Cover of the economist from **Sep 9th 2017**

Testing 19th century claims… with 21st century method

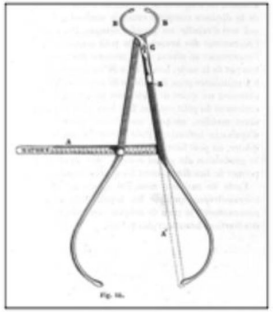


Fowler head



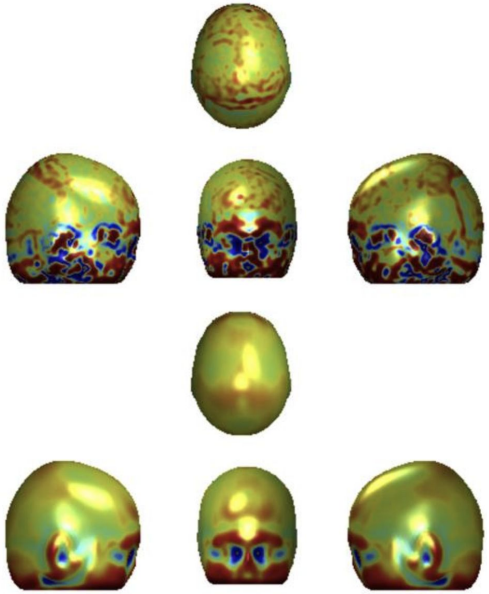
Phrenology chart

Measuring tape



Calipers

Palpation



Subject

Database average

Scalp curvature

[An empirical, 21st century evaluation of phrenology.]

#	Faculty	Biobank lifestyle measure
I	Impulse to propagation (*Amativeness*)	Lifetime number of sexual partners
II	Tenderness for the offspring or parental love (*Philoprogenitiveness*)	People in the house related to participant (son/daughter/mother/father)
III	Friendly attachment or fidelity (*Adhesiveness*)	People in the house not related to participant (husband/wife/partner/other)
IV	Valour, self-defence (*Combativeness*)	Solicitor, lawyer, barrister, judge (job)
V	Murder, carnivorousness (*Destructiveness*)	Beef intake
VI	Sense of cunning (*Cunning*)	Scientist (job)
VII	Larceny, sense of property (*Acquisitiveness*)	Number of vehicles in household
VIII	Pride, arrogance, love of authority (*Self-Esteem*)	Banker (job)
IX	Ambition and vanity (*Love of Approbation*)	Financial situation satisfaction
X	Circumspection (*Cautiousness*)	Alcohol intake frequency
XI	Aptness to receive an education or the memoria realis (*Eventuality and Individuality*)	Age completed full time education
XII	Sense of locality (*Locality*)	Time spent doing light physical activity
XII	Sense of locality (*Locality*)	Time spent doing light physical activity
XIV	Words, verbal memory (*Words*)	Letter fluency
XV	Faculty of language (*Language*)	Authors, writers (job)
XVI	Disposition for colouring, delighting in colours (*Colouring*)	Photographers, painter (job)
XVII	Sense for sounds, musical talent (*Tune*)	Music profession (job)
XVIII	Arithmetic, counting, time (*Number*)	Mathematician (job)
XIX	Mechanical skill (*Constructiveness*)	Hand grip strength (right)
XX	Comparative perspicuity, sagacity (*Comparison*)	Concept interpolation
XXI	Metaphysical perspicuity (*Causality*)	Clergy (job)
XXII	Causality, sense of inference (*Mirthfulness*)	Writer, actor, comedian (job)
XXIII	Poetic talent (*Ideality*)	Poet (job)
XXIV	Good nature, compassion, moral sense (*Benevolence*)	Charity (job)

Mapping 19th century traits to data in the Biobank

An empirical, 21st century evaluation of phrenology.

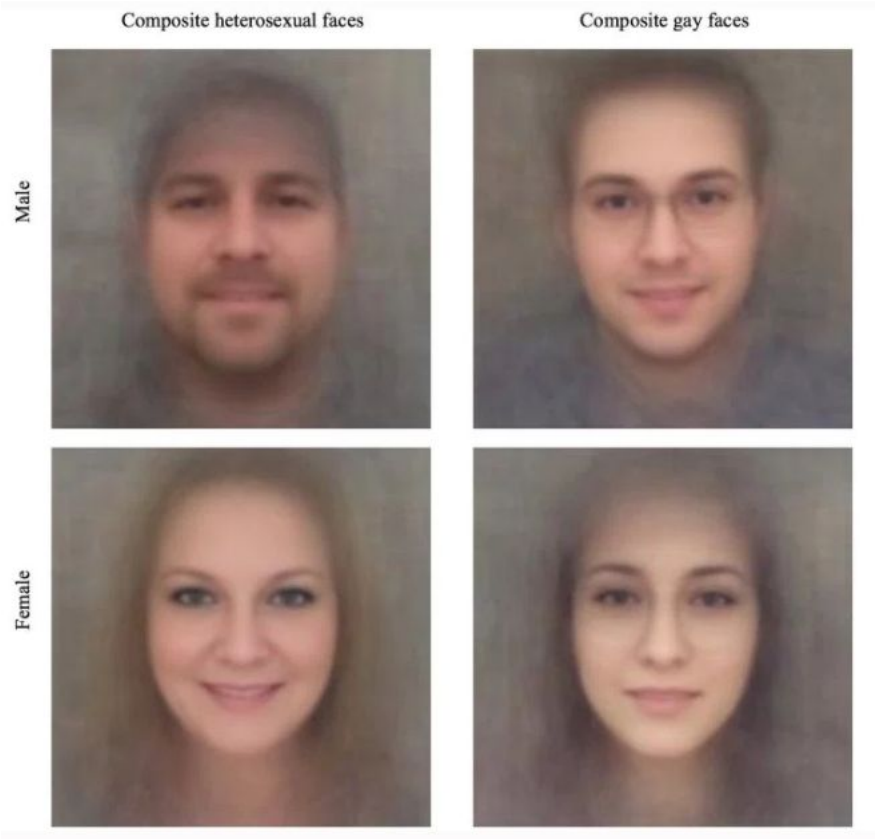
From the paper

“We show that faces contain much more information about sexual orientation than can be perceived and interpreted by the human brain.

We used deep neural networks to extract features from 35,326 facial images [...] these features were entered into a logistic regression aimed at classifying sexual orientation [...]

Those findings **advance our understanding of the origins of sexual orientation** and the limits of human perception.”

Deep neural networks are more accurate than humans at detecting sexual orientation from facial images.



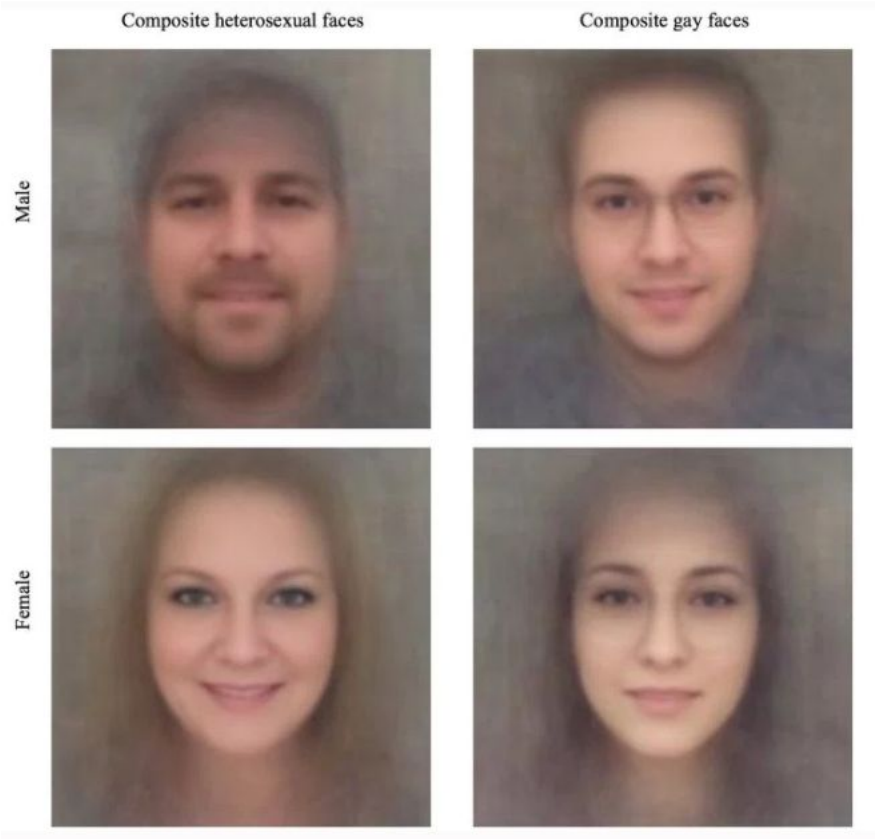
From a rebuttal

"The authors assert that the key differences are in physiognomy, 地貌 meaning that a sexual orientation tends to go along with a characteristic facial structure.

However, we can immediately see that some of these differences are more superficial…"

Quick discussion: What do you see?

Do algorithms reveal sexual orientation or just expose our stereotypes?



Important variables:

- **Makeup**
- **Eyeshadow**
- **Facial hair**
- **Glasses**
- **Selfie angle**
- **Amount of sun exposure**

Yes/no questions about these do nearly as well at guessing orientation as the supposedly sophisticated software.

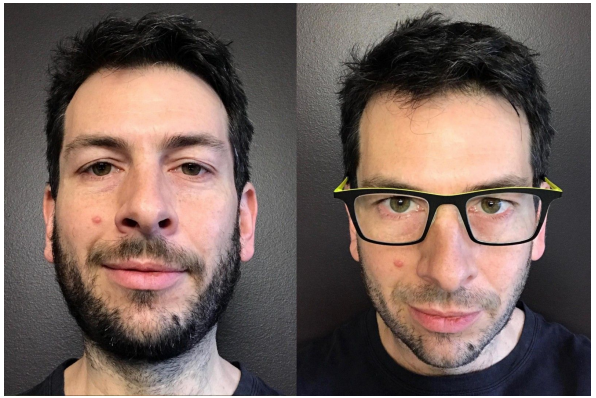
[Do algorithms reveal sexual orientation or just expose our stereotypes?]

Those findings ~~advance our understanding of the origins of sexual orientation~~ ~~and the limits of human perception.~~”

Or maybe they just tell us if someone is wearing glasses...

[Do algorithms reveal sexual orientation or just expose our stereotypes?]
[Deep neural networks are more accurate than humans at detecting sexual orientation from facial images.]





[Do algorithms reveal sexual orientation or just expose our stereotypes?]

Fig. 3 summarises the distribution of responses obtained for the lifestyle measures for the available subjects. The numbers of subjects sampled for each category were rather large, except for the "job"-based lifestyle measures (see Table 1).We ultimately left these faculties out of the final phrenological analysis in which scalp morphology was correlated against lifestyle measures. For those interested, there is a paper exploring the lifestyle measures in greater detail (Miller et al., 2016).

2.4. Relating local scalp morphology to personal measures

Starting with phrenology's claim that bumps on the head relate to individual traits, we used multivariate regression to search for associations between local scalp curvature and lifestyle measures extracted from the Biobank. More concretely, we modelled vertex-wise scalp curvature against lifestyle measures including gender and age as nuisance regressors. For the binary measures, such as whether or not someone was a banker, the regression model could be set up as an unpaired *t*-test. For illustration purposes, all resulting t-statistics were converted to z-statistics (see Discussion). In total, we note that there were 14 binary lifestyle measures (faculties II, III, IV, VI, VIII, XV, XVI, XVII, XVIII, XX, XXI, XXII, XXIII and XXIV) and nine non-binary measures (faculties I, V, VII, IX, X, XI, XII, XIV, and XIX) all shown in Fig. 3. However, there were low numbers in some of the binary measures (faculties IV, VI, VIII, XV, XVI,

underlying brain gyrification given that phrenology assumes a relationship between head and brain morphology. We found that brain gyrification explains very little of the variance in local scalp curvature (Fig. 4). Second, we correlated local scalp curvature with a set of lifestyle measures interpreted as Victorian "faculties" (e.g. "lifetime number of sexual partners" was used as a proxy for the faculty of "Amativeness", or the "impulse to propagation"). Despite the size of our sample and automation of our methods, we found no evidence to support phrenology's fundamental claim. The regions depicted on phrenological busts (Fig. 1) therefore should not be trusted. According to our results, a more accurate phrenological bust should be left *blank* since no regions on the head correlate with any of the faculties that we tested. But even below the level of statistical significance, we found historic phrenological predictions to be uninsightful. For example, Fig. 5 shows the unthresholded z-statistic map for correlations between local head curvature and lifetime number of sexual partners ("Amativeness"). Unsurprisingly, the "frontal horn" area that we point out does not correspond to ROIs proposed by phrenologists, which included areas at the *back* of the skull (Fowler & Fowler, 1859). For the reckless, zealous or simply curious reader, we include the remaining unthresholded z-statistic maps (none statistically significant) in the Supplementary Materials (Supplementary Figure 3 to Supplementary Figure 24). We did not analyse the relationship between lifestyle measures and brain morphology, since many such relationships are known and uncontroversial within 21st century neuroscience (Grogan et al., 2012, Maguire et al., 2000, Mechelli

3. Results

The phrenological analyses produced no statistically significant or meaningful effects.

In order to test the second claim of phrenology, that bumps on the head should reflect the underlying shape of the cerebral cortex, we correlated each subject's local scalp curvature (described above) with a local index of brain gyrification (projected onto the scalp). This gyrification index was quantified using a surface ratio, corresponding to the amount of cortical surface packed within a limited spherical volume at every point on the cortex (Toro et al., 2008). If the phrenological hypothesis is true, we would expect a large negative correlation between scalp curvature and the gyrification index. We assume that the amount of cortical surface correlates with pressure under the skull but note that 19th century phrenologists imagined pressure to come from individual gyri instead. For data we extracted the cortical (pial) surface from each subject's T1-weighted scan using FreeSurfer (Dale, Fischl, & Sereno, 1999). In order to summarise the surface ratio of the cortex underlying each scalp vertex, we used the average surface ratio within a 20 mm sphere that was centred around the nearest cortical vertex. Once both measures (scalp curvature and cortical convolution) were mapped onto the scalp surface, we were able to correlate the two measures and answer the question of whether scalp morphology looks like a consistent proxy for underlying brain morphology.

3. Results

The phrenological analyses produced no statistically significant or meaningful effects.

4. Discussion

Fowler heads (Fig. 1) were the results of underpowered studies based on anecdotal evidence (Eling et al., 2017).

Set against the strengths of our study, an apparent weakness is our use of 19th century "faculty psychology" with its description of human nature in idiosyncratic terms like "Amativeness" and "Philoprogenitiveness" (Poldrack, 2010), and grouping together of attributes like "eats meat" and "likes to kill", which may strike some today as odd (Eling et al., 2017). Therefore it might be objected that we should have used a more recent *ontology*. However, phrenology's "faculty psychology" is not as different from current ontologies as it might first seem. One can readily find examples of 19th century faculties in the neuroimaging literature, albeit under different names (Table 2). We were also interested in grounding our study in Victorian concepts, despite an emphasis on 21st century methods. The lifestyle features that we selected also ranged over a wide number of behavioural and cognitive domains (e.g. motor skills, language, spatial awareness, decision making, etc.). So regardless of ontology, we hope to have covered many topics of interest.

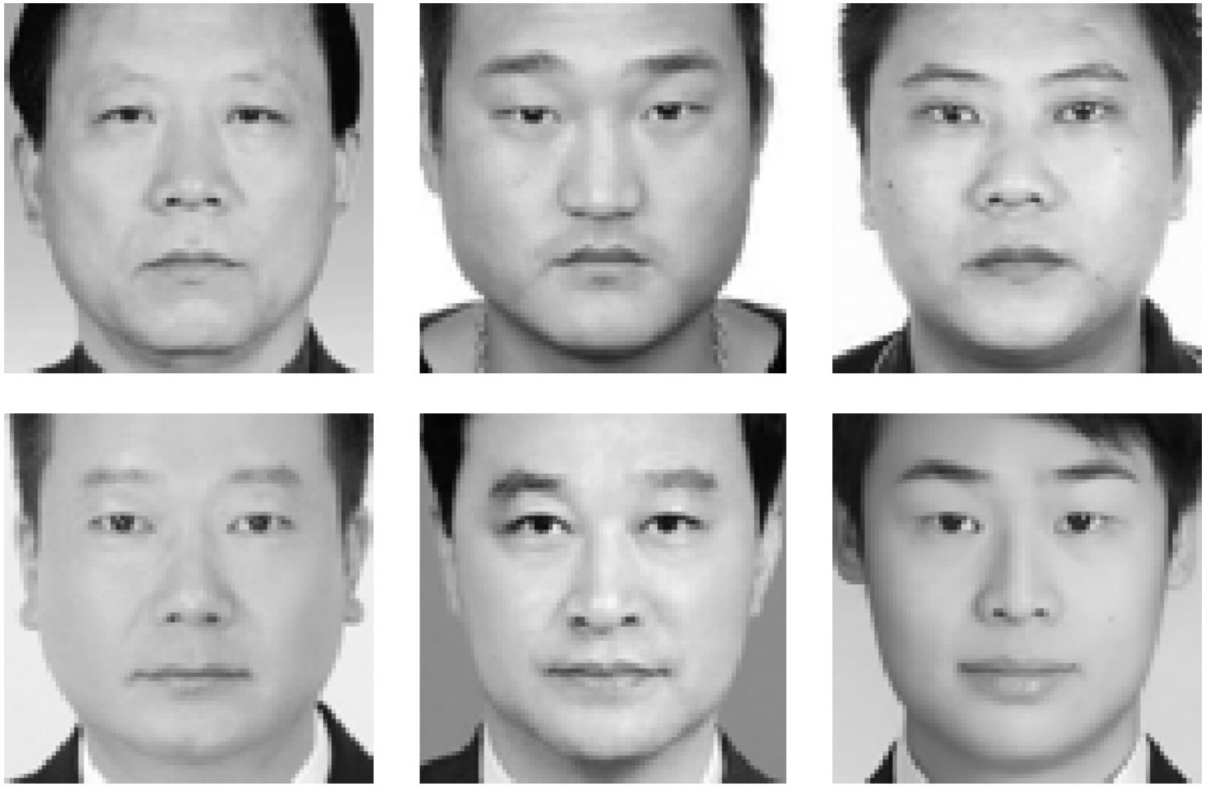
As to the objection that phrenology was already a known dead-end scientifically, and that its claims did not need to be tested rigorously, it is indeed hard to find a time in history when phrenology was not seriously criticised. Even in 1815, the year that Spurzheim published his influential book on Gall's method, phrenology was dismissed by one reviewer as "a piece of thorough quackery from beginning to end" (Gordon, 1815). Not only did the reviewer take issue with the use of palpation as an indirect method for measuring the brain and its mental faculties, but he also objected to the idea the brain might be composed of multiple specialised components, writing the following (Gordon, 1815: 243):

When all you have is a hammer, everything looks like a nail...

The authors only shared six images (always a bad sign...)

Do you see anything they may not have controlled for?

Criminals on the top row.

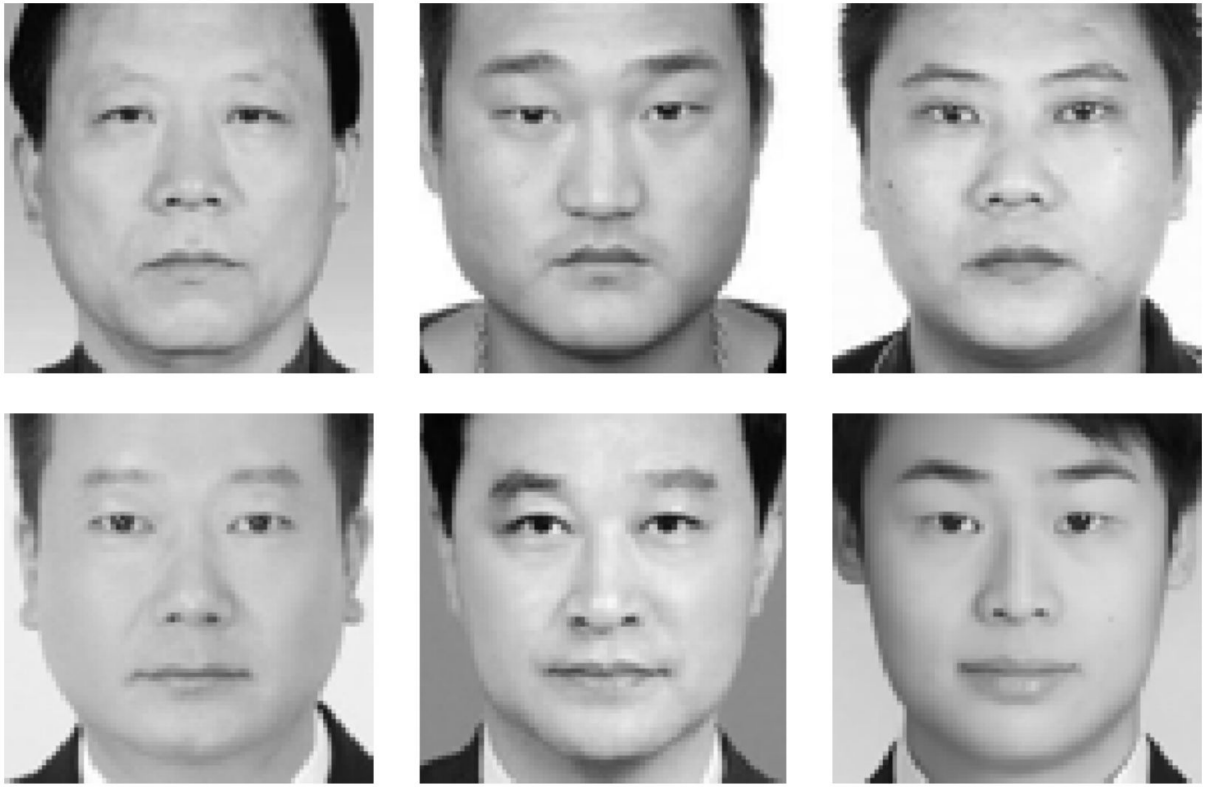


[Automated Inference on Criminality using Face Images]
[Physiognomy's New Clothes]

The authors only shared six images (always a bad sign...)

Do you see anything they may not have controlled for?

Criminals on the top row.



Frowning

Smiling

[Automated Inference on Criminality using Face Images]
[Physiognomy's New Clothes]

Quick discussion: what could go wrong? How long might it take?

Tae

Google photos incident

Concretely, can you suggest reasons why this may have happened?

Outline a course of action to **prevent this in the future**.

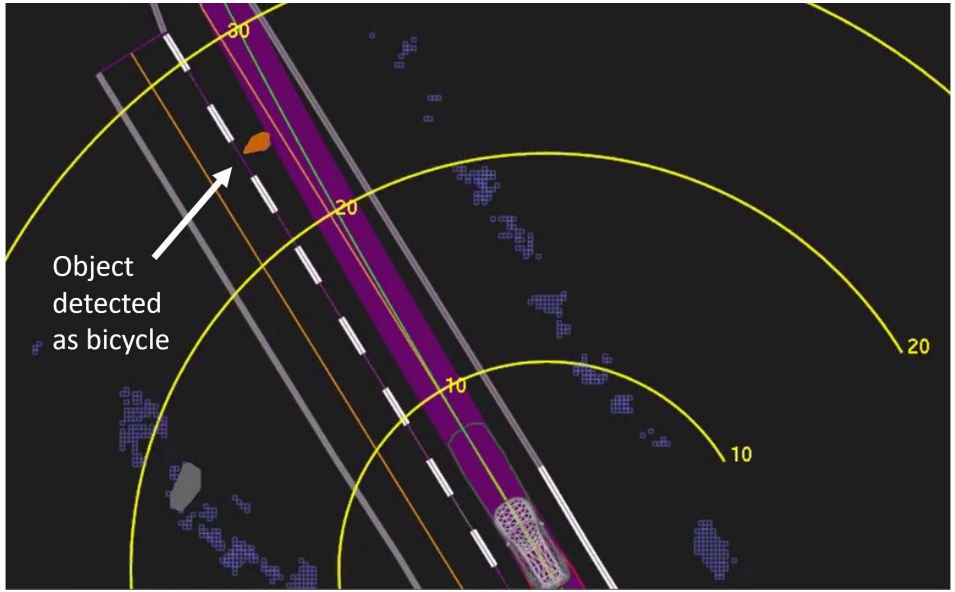
- What are the costs involved?
- How likely is it to work?

Google apologises for Photos app's racist blunder (2015)

Sadly, the first self-driving car fatality

NTSB Preliminary Report

- System determined emergency braking was needed 1.3 seconds before impact.
- Emergency braking maneuvers were disabled while vehicle was under computer control (to reduce potential for erratic behavior).
- Instead, vehicle operator was relied on to take action.



Data driven errors vs programming bugs

Data driven errors vs **programming bugs**

Knight Capital (2013)

$460M loss + $12M fine

Bug in an automated system caused trades losing $472 million in 45 minutes.

- An obsolete code path designed to be used in an experimental environment was activated in production (causing unexpected behavior).

Quick discussion

- In retrospect, how would you fix the above?

https://www.sec.gov/news/press-release/2013-222

Data driven errors vs programming bugs

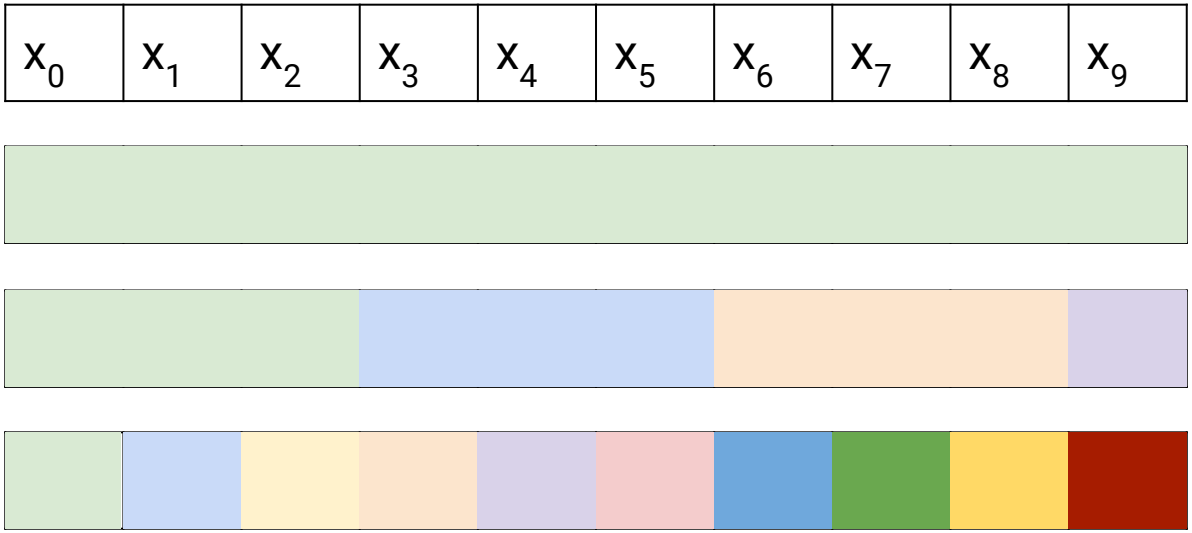
Testing ML systems is much harder

- We're using ML **exactly when** it's difficult or impossible to write software logic to produce the desired behavior.
- **Data influences behavior**. Testing components of ML systems in isolation is difficult: a small change in data can have large downstream effects.
- **Changing Anything Changes Everything**.

- Break (10 mins)
- Work on HW1 or HW2 (20 mins)
- We will resume at [ ]
- Questions? Please come up

Batch size and epochs

Epochs and batches



Say this is all of our training data

Batch gradient descent (batch size = len(training_data)

Mini batch (batch size of three)

Stochastic (batch size of one)

Batch size and epochs

An epoch

One iteration over the training data (every example is used once to update the weights).

- The longer you train your model, the more tightly it will fit the training data (but may also overfit the test data).
- To find the right number of epochs, monitor the loss on the validation data when training. Stop training when the validation loss begins increasing (**rule of thumb**).

Batch size and epochs

Batch size

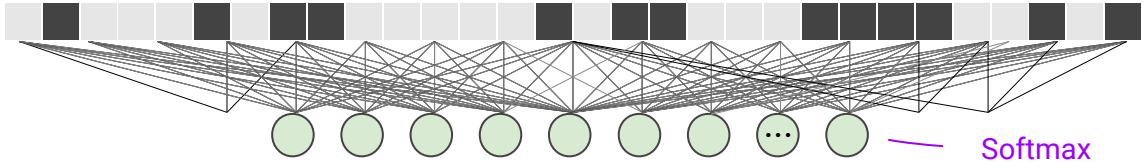
The number of examples used per gradient update.

- A batch size of 1 is *stochastic gradient descent*. Many updates per epoch, but each is inaccurate.
- A batch size of len(training_set) is *batch gradient descent*. One accurate update per epoch, but slow to compute.
- Intermediate sizes (the default in Keras is 32) are called *mini-batch gradient descent*.

In general, you will not need to change this parameter unless your examples are very small (structured data, in which case you can use a larger batch size).

https://keras.io/models/model/#fit

Dense layers




```
model = Sequential()
model.add(Dense(10, activation='softmax'), input_shape=(784,)
```

Linear model

$$f(x) = softmax(W_1 x)$$

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

Choose the right loss and activation function

Start with the output layer and loss function.

Classification (with a single correct class)

- Output layer: *softmax activation, neurons equal to number of classes.*
- Loss: *categorical_crossentropy* (if your labels are in one-hot format), or *sparse_categorical_crossentropy (*if your labels are numeric).

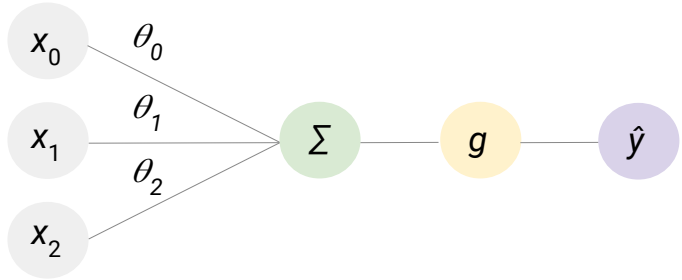
Regression (for a probability-like number between 0-1)

- Output layer: sigmoid activation, a single neuron.
- Loss: *binary_crossentropy.*

Regression (for an unbounded number, like the price of an item)

- Output layer: linear activation ('None'), a single neuron.
- Loss: *squared error.*

A gotcha / FYI
With unusual inputs (with very large features), this model may produce very positive or negative outputs unlike any in your training set.



Inputs **weights** **sum** **activation** **output**

Linear combination of inputs and weights

$$\hat{y} = g\left(\sum x_i \theta_i\right)$$

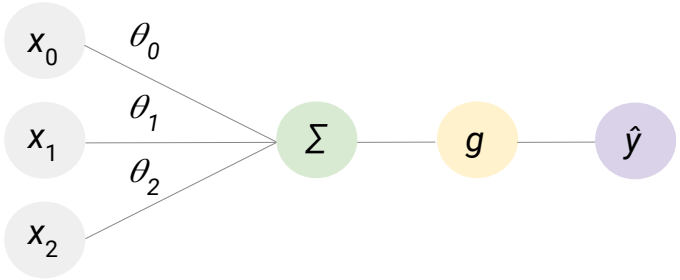
Can rewrite as a dot product

$$\hat{y} = g\left(x^T \theta\right)$$

Bias not drawn (you could set x_1 to be a constant input of 1).

Three views of a neuron

```
layer = Dense(units=1, kernel_initializer='ones', use_bias=False, input_shape=(3,))
data = tf.constant([[1.0, 2.0, 3.0]]) # Note: a batch of data
sum = layer(data) # tf.Tensor([[6.]], shape=(1, 1), dtype=float32)
y = sigmoid(sum) # tf.Tensor([[0.9975274]], shape=(1, 1), dtype=float32)
```



Inputs **weights** **sum** **activation** **output**

Linear combination of inputs and weights

$$\hat{y} = g\left(\sum x_i \theta_i\right)$$

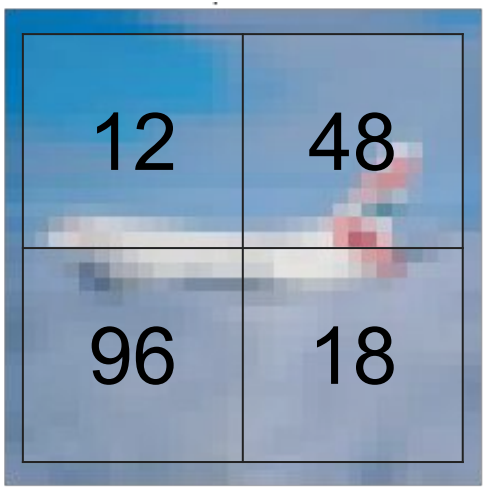
Can rewrite as a dot product

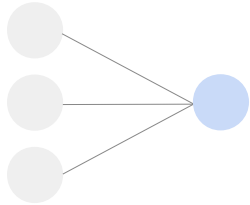
$$\hat{y} = g\left(x^T \theta\right)$$

Bias not drawn (for simplicity, you could set x_1 to be a constant input of 1).

One image and one class

Interpret as “how strongly do you think this image is a plane?”





Multiple inputs; one output

1.4	0.5	0.7	1.2

12
48
96
18

+

0.5

=

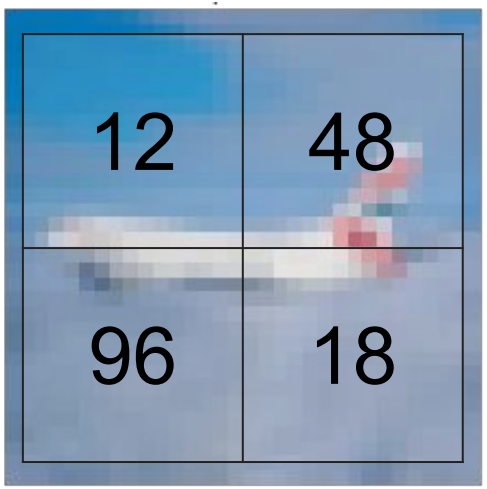
130.1	Plane

I realize I occasionally use Wx and xW in these slides (I haven’t had a chance to standardize them yet, core concepts are the same).

w	x	b	Output
Weights	Inputs	Bias	Scores

```
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 10)                7850
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
```

One image and two classes



1.4	0.5	0.7	1.2
-2.0	0.1	0.2	-0.7

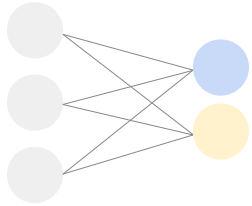
12
48
96
18

+

0.5
1.2

=

130.1	Plane
-11.4	Car



Multiple inputs; multiple outputs

W is now a matrix

W — Weights

x — Inputs

b — Bias

Output — Scores

Two images and three classes

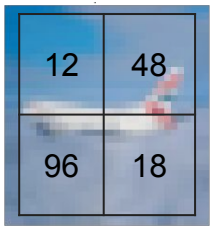


Image 1

Image 2

N x D

1.4	0.5	0.7	1.2
-2.0	0.1	0.2	-0.7
0.2	0.9	-0.2	0.5

W

Weights

D x batch_size

12	4
48	18
96	2
18	96

x

Inputs

+

N x 1

0.5
1.2
0.2

b

Bias

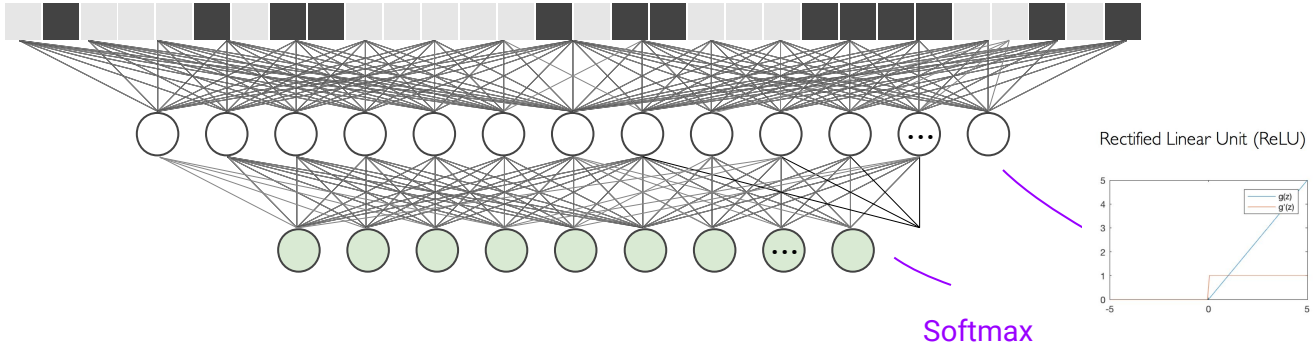
=

N x batch_size

Image 1	Image 2	
130.1	131.7	Plane
-11.4	-71.7	Car
12.8	64.8	Truck

Output

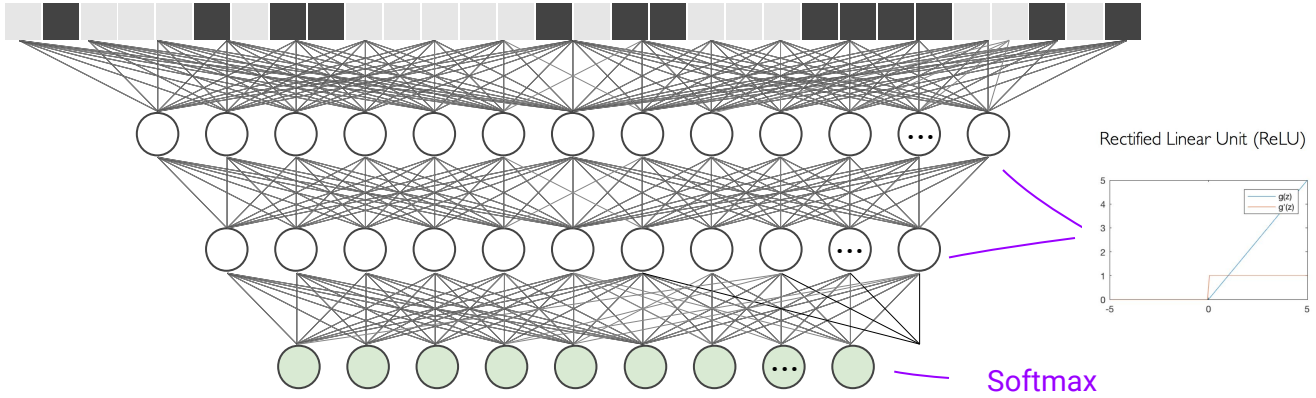
Scores




```
model = Sequential()
model.add(Dense(256, activation='relu',input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
```

Linear model $f(x) = softmax(W_1 x)$

Neural network $f(x) = softmax(W_2(g(W_1 x)))$




```
model = Sequential()
model.add(Dense(256, activation='relu',input_shape=(784,)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Linear model $f(x) = softmax(W_1 x)$

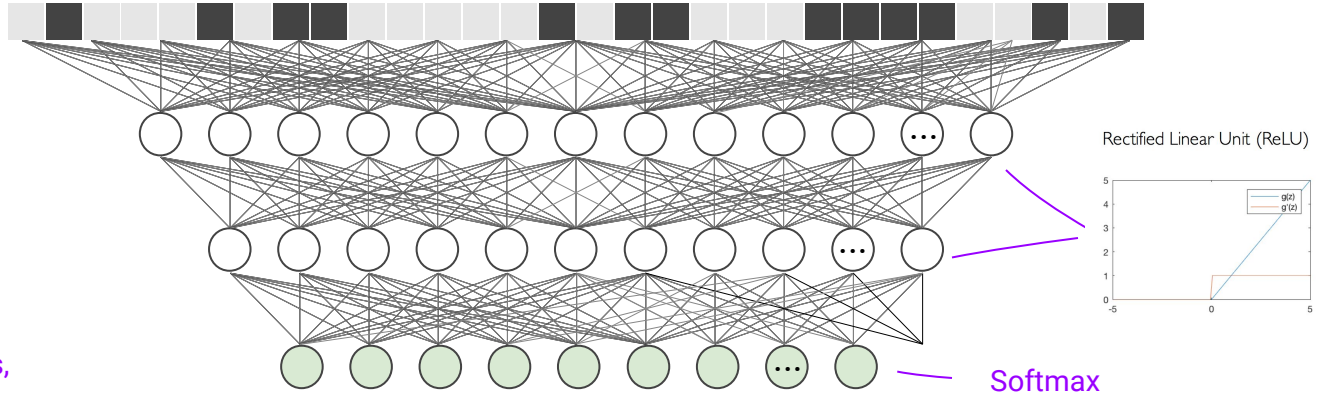
Neural network $f(x) = softmax(W_2(g(W_1 x)))$

Deep neural network $f(x) = softmax(W_3(g(W_2(g(W_1 x)))))$

Terminology

Depth of the network (three layers)

Width of a layer (number of units, or neurons)




```
model = Sequential()
model.add(Dense(256, activation='relu',input_shape=(784,)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Linear model $f(x) = softmax(W_1 x)$

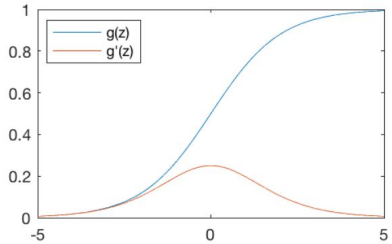
Neural network $f(x) = softmax(W_2(g(W_1 x)))$

Deep neural network $f(x) = softmax(W_3(g(W_2(g(W_1 x)))))$

Activation functions

```
from tensorflow.nn import relu, sigmoid, tanh
```

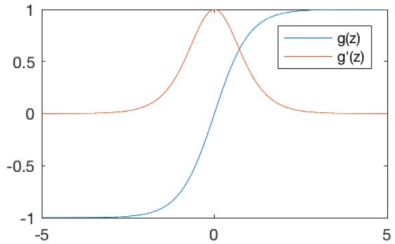
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

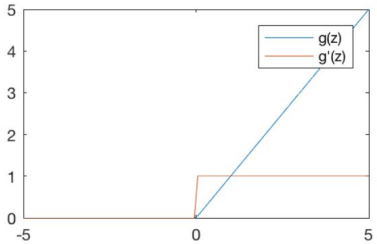
Hyperbolic Tangent



$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



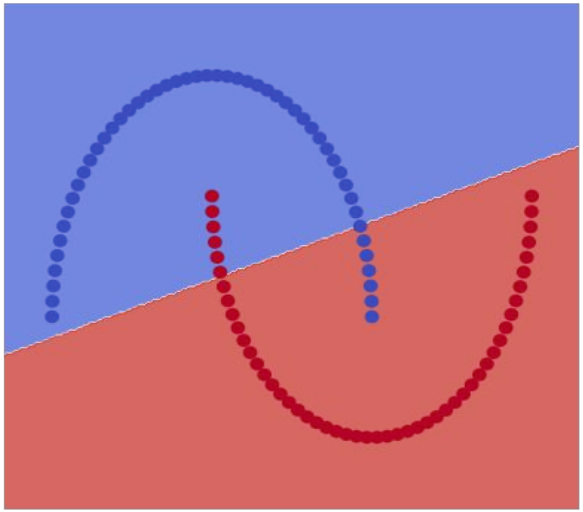
$$g(z) = \max(0, z)$$

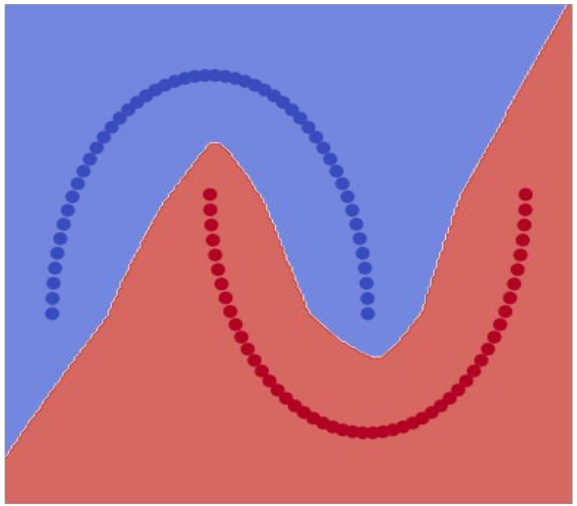
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

A good default: use ReLU for all of your layers except for the last.

Figure from friends at MIT.

Activation functions introduce non-linearities





Notes
- You can make similar plots (and more) with this example. Note: from an older version of TF, but should work out of the box in Colab.
- Each of our convolutional layers used an activation as well (not shown in previous slides).

Without activation, many layers are equivalent to one

```
# If you replace 'relu' with 'None', this model ...
model = Sequential([
  Dense(256, activation='relu', input_shape=(2,)),
  Dense(256, activation='relu'),
  Dense(256, activation='relu'),
  Dense(1, activation='sigmoid')
])
```

```
# ... has the same representation power as this one
model = Sequential([Dense(1, activation='sigmoid', input_shape=(2,))])
```

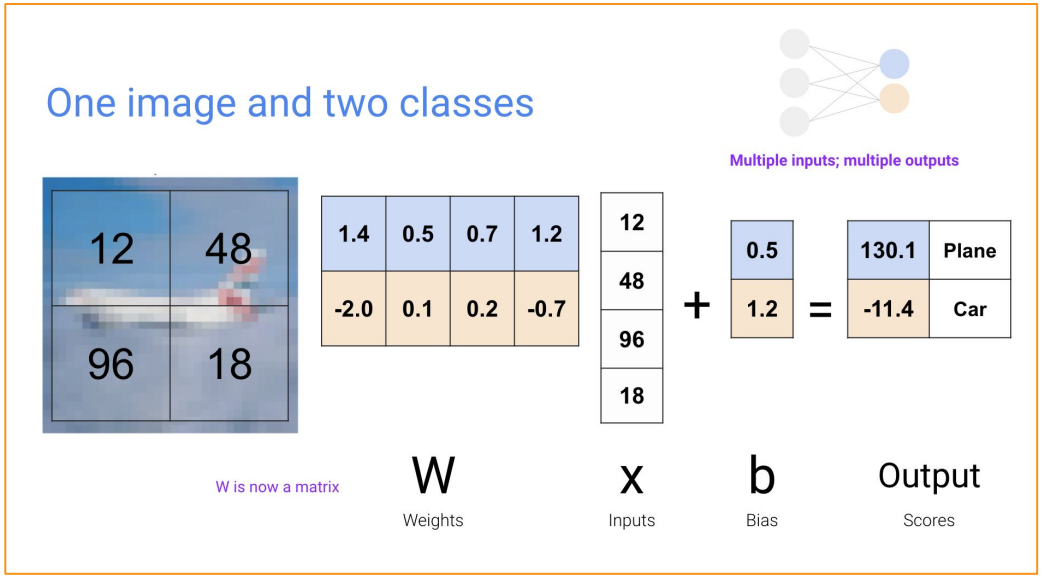
Notes

- Linear activation functions result in linear decision boundaries regardless of the depth of the network.
- You can demo this live in TensorFlow Playground.

Dense layer

$$f = W_2 g(Wx)$$

After we apply the nonlinearity, the result becomes the input to the next layer.

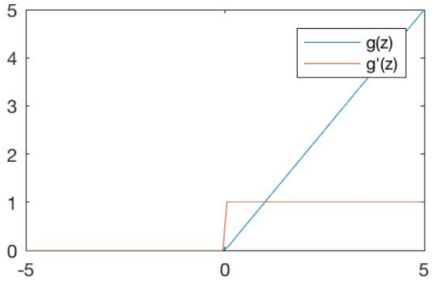


Most are applied piecewise.

Some (like softmax) compute statistics about the distribution of their inputs, first.

130.1	Plane
-11.4	Car
12.8	Truck

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

g(130.1)	Plane
g(-11.4)	Car
g(12.8)	Truck

=

?	Plane
?	Car
?	Truck

$$f = W_2 g(Wx)$$

Output

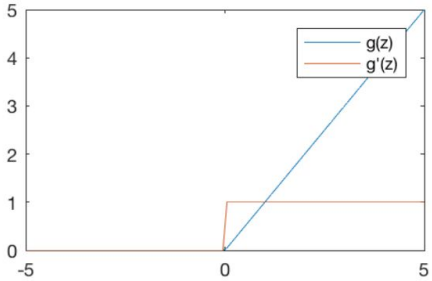
Scores

Most are applied piecewise.

Some (like softmax) compute statistics about the distribution of their inputs, first.

130.1	Plane
-11.4	Car
12.8	Truck

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

g(130.1)	Plane
g(-11.4)	Car
g(12.8)	Truck

=

130.1	Plane
0	Car
12.8	Truck

$$f = W_2 g(Wx)$$

Output

Scores

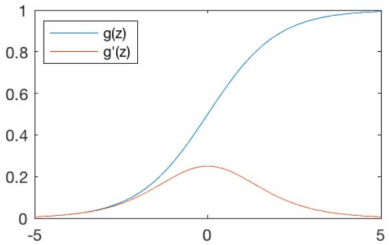
Why ReLU?

If ReLU feels unsatisfying to you (as compared to using a Sigmoid activation) you are not alone. Why do we use them?

Sigmoid has a shortcoming. The gradient saturates (the derivative approaches zero) near the extremes. This can stall learning, especially in deep networks.

ReLU is much faster to compute, but that's *not* why the model converges faster.

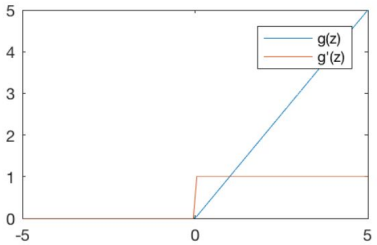
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

Rectified Linear Unit (ReLU)

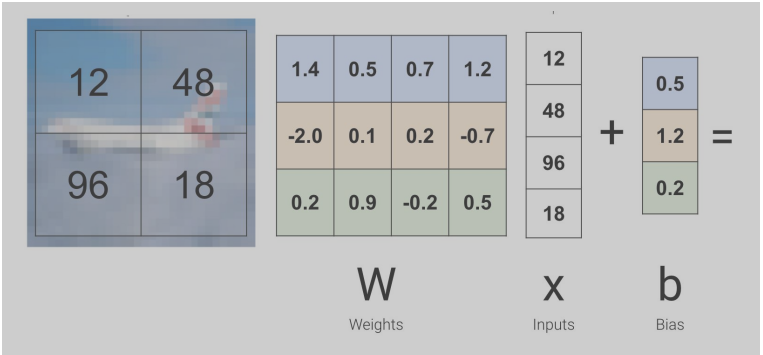


$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Softmax

At this point we have scores. What we want are probabilities.

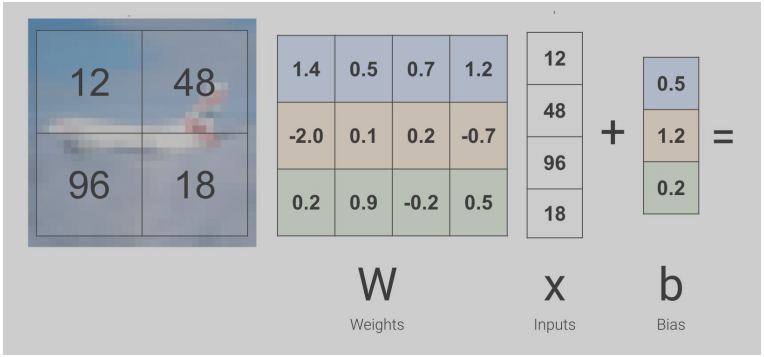


130.1	Plane
-11.4	Car
12.8	Truck

Scores

Softmax converts scores to “probabilities”

Only input to the softmax function is the class scores. No learned parameters.



130.1	Plane
-11.4	Car
12.8	Truck

```
softmax([130.1, -11.4, 12.8])
>>> 0.999, 0.001, 0.001
```

Scores

Probabilities

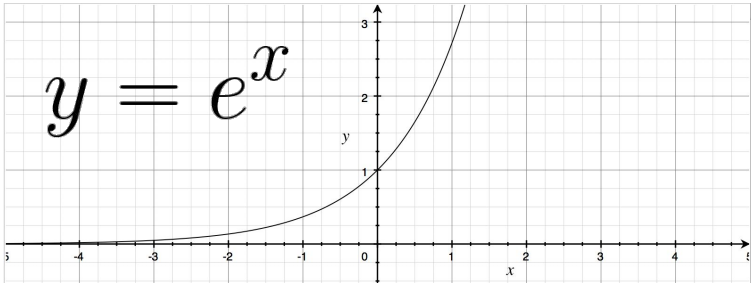
Note: these are ‘probability like’ numbers (do not go to vegas and bet in this ratio).

Math view

The last activation function in a network used for classification. Normalizes each output to 0 < x < 1, and such that they sum to 1.

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

Why exp? No prediction will have zero or negative probability.



Reminder

Code view and a few examples (not numerically stable!)

```
import numpy as np
softmax = np.exp(scores) / np.sum(np.exp(scores))
```

Could be written more efficiently, we're computing exp(scores) twice.

```
>>> softmax([1,2,3])
# 0.09, 0.24, 0.66
```

Higher scores increase output multiplicatively

```
>>> softmax([0, 0, 10])
# 4.53e-05 4.53-05 9.99e-01
```

Outputs may approach 1 (but will always be less than 1, rounding errors aside)

```
>>> softmax([-10, -5, -8])
# 0.006 0.946 0.047
```

No output ever has zero or negative probability

Note: this needs to be numerically stabilized (may overflow otherwise).

Cross entropy

Cross entropy loss

The most common (and standard) loss for classification problems. Compares two distributions.

- The probabilities you predicted
- The probabilities you wanted.

One-hot encodings

In CIFAR-10 we have 10 classes, airplane, automobile, etc

Each example has a label in a one-hot format

This is a bird

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

Interpret as the probability distribution we want our model to predict

Each example has a label in a one-hot format

This is a bird

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

True probabilities

airplane 

automobile 

bird 

cat 

deer 

dog 

frog 

horse 

ship 

truck 

Softmax output is predicted distribution

Each example has a label in a one-hot format

This is a bird

0	1	2	3	4	5	6	7	8	9	
0	0	1	0	0	0	0	0	0	0	True probabilities
0.1	0.2	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0	Predicted probabilities

Recall: Softmax output 0 < x < 1

Each example has a label in a one-hot format

This is a bird

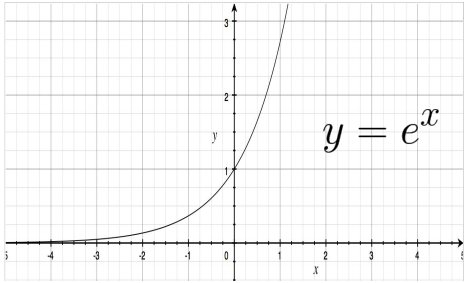
0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	0

True probabilities

0.1	0.2	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0

Predicted probabilities

Rounded! Softmax output is always 0 < x < 1



$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$$

CE compares these distributions

Each example has a label in a one-hot format

This is a bird

0	1	2	3	4	5	6	7	8	9	
0	0	1	0	0	0	0	0	0	0	True probabilities
0.1	0.2	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0	Predicted probabilities

Rounded! Softmax output is always 0 < x < 1

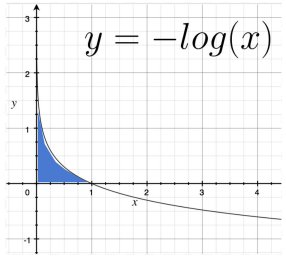
Cross entropy loss for a batch of examples

True prob (either 1 or 0) in our case!

$$L = -\sum \hat{y}\, ln(y_i)$$

Sum over all examples

Predicted prob (between 0-1)



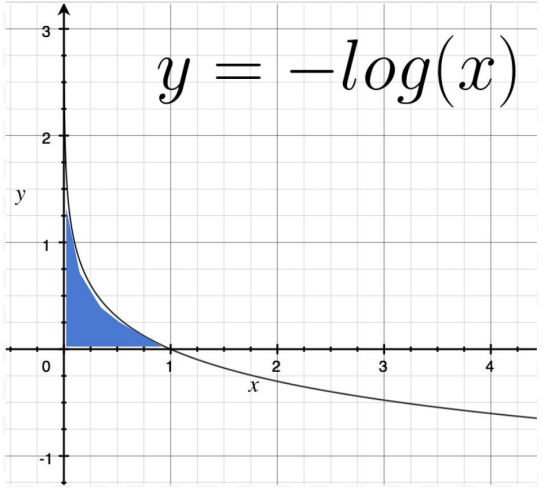
Minimum and maximum possible loss?

When does loss reach a **minimum**?

-

When does loss reach a **maximum**?

-



CE loss will be in this region

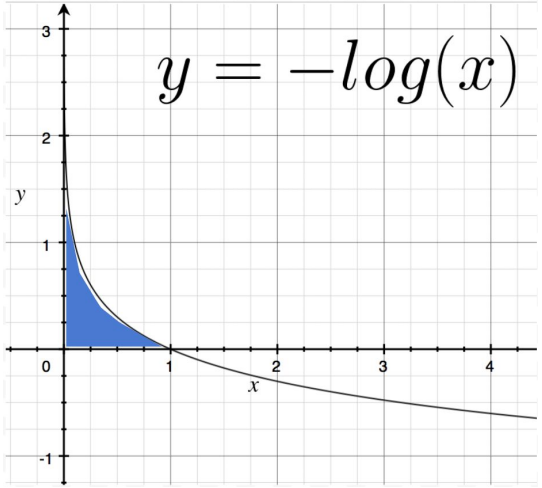
Minimum and maximum possible loss?

When does loss reach a **minimum**?

- Softmax output on the correct example approaches 1, loss approaches 0.

When does loss reach a **maximum**?

- Softmax output on the correct example approaches 0, loss approaches +inf.



CE loss will be in this region

What initial loss do we expect?

Imagine we have 10 classes, weights initialized randomly. What's the average loss over a large balanced training set?

Predicted probabilities

?	?	?	?	?	?	?	?	?	?

What initial loss do we expect?

Imagine we have 10 classes, weights initialized randomly. What's the average loss over a large balanced training set?

loss = ?

Predicted probabilities

0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

What initial loss do we expect?

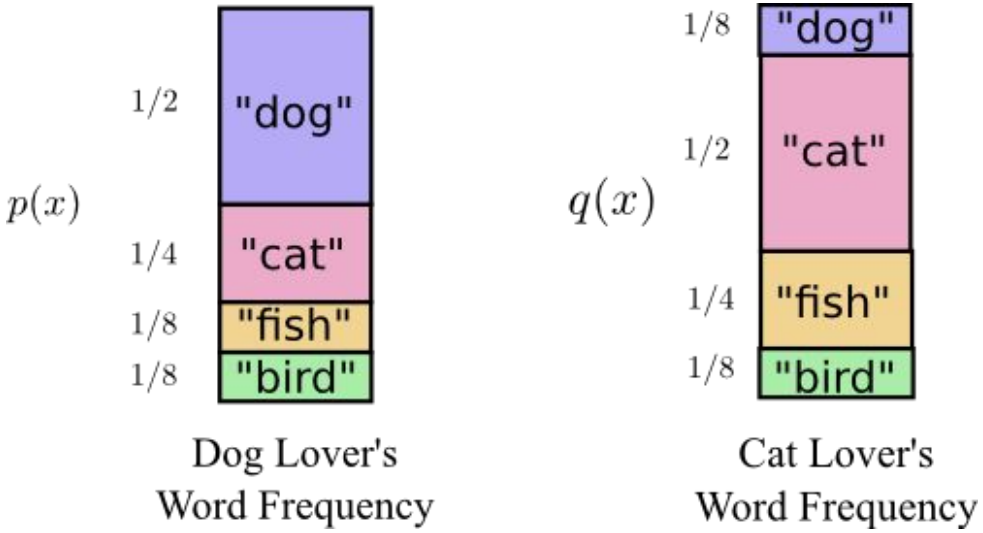
Imagine we have 10 classes, weights initialized randomly. What's the average loss over a large balanced training set?

loss = -ln(1 / n_classes) = ~2.3

Predicted probabilities

0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

If you'd like to learn much more about CE



colah.github.io/posts/2015-09-Visual-Information/

Why not classification error?

Targets (true label)			Predicted class probabilities			Correct?
Car	Plane	Truck	Car	Plane	Truck	
1	0	0	0.4	0.3	0.3	Yes
0	1	0	0.3	0.4	0.3	Yes
0	0	1	0.1	0.8	0.1	No

Confidently incorrect predictions should be penalized more

Targets (true label)			Predicted class probabilities			Correct?
Car	Plane	Truck	Car	Plane	Truck	
1	0	0	0.4	0.3	0.3	Yes
0	1	0	0.3	0.4	0.3	Yes
0	0	1	0.1	0.8	0.1	No

CE takes this in to account

A helpful code reference

The keras source code is a good learning resource for the details of various loss functions, especially the Numpy backend:

- github.com/keras-team/keras/blob/master/keras/losses.py
- github.com/keras-team/keras/blob/master/keras/backend/numpy_backend.py

For next time

Reading

- Deep Learning with Python: 3.
- Deep Learning: 4.1, 4.2, 4.3, 5.1, 5.2
- Essence of Calculus (helpful videos if you need a refresher)