MFE230T-1 - Introduction to Deep Learning

Vinicio De Sola, MFE, MIDS

Data Science Projects - What to expect?

1. Define the goal
2. Get the Data
3. Clean the Data
4. Enrich the Data
5. Find Insights and Visualize (Exploration)
6. Deploy your models (Classic ML / Deep Learning)
7. Iterate

NLP - Revisited

Language

Why is language understanding hard?

Boy paralysed after tumour fights back to gain a black belt

A BOY left paralysed after a brain tumour has baffled doctors by making an unexpected recovery – and gaining a black belt in Taekwondo.

Daniel Kimmins, now 14, underwent emergency surgery days before his seventh birthday when doctors discovered the tumour.

After the operation at Bristol's Frenchay Hospital, the youngster was left paralysed down the left side of his body and doctors warned him that it could be permanent.

But now Daniel, from Bath, Somerset, has fought against the odds to make a full recovery.

He first took up the martial art when he was six – and within a year had earned his yellow belt.

By **Mark Reynolds**

ability to speak. We were never given any hope and got the impression he would not recover from the tumour.

"But he has learnt to walk and talk all over again and he is able to do all of the things he did before – which makes me unbelievably proud of him."

At first, Daniel found the Taekwondo a lot harder than he used to.

So he was over the moon to be rewarded with the black belt a few weeks ago.

He said: "It was just absolutely amazing."

Advisers to tackle unruly pupils

The government is setting up an expert group of a dozen teachers and head teachers to advise it on improving classroom behaviour in England.

Discipline is a cross-party issue

Education Secretary Ruth Kelly said all schools must have a culture of respect.

Ms Kelly said her plans we
the panel could identify th
be replicated everywhere.

The Tories - who made dis
more than "a talking shop"

'Clear sanctions'

Paltrow gives birth to baby Apple

US actress Gwyneth Paltrow, 31, has given birth to her first child, a girl called Apple.

Paltrow and Martin married in secret in December

The Hollywood star underwent a long labour before delivering her first born at a London hospital on Friday.

Gwyneth and her husband Chris Martin, front man in the band

Information Theory

Information Entropy

Intuition:

- Really high entropy?
- Really low entropy?
- What about 0 entropy?

Exercise

A language has two symbols:

“A” with probability 0.5

“B” with probability 0.5

How would you encode them in binary?

Exercise

A language has two symbols:

“A” with probability 0.5

“B” with probability 0.5

How would you encode them in binary?

A-> 0; B-> 1

Exercise

"A" with probability 0.5 (0)

"B" with probability 0.5 (1)

Expected length per symbol?

Exercise

“A” with probability 0.5 (0)

“B” with probability 0.5 (1)

Expected encoding length per symbol?

E[symbol_encoding_length] = ∑ p(symbol) * symbol_length

= 0.5 * 1 + 0.5 * 1 = 1

= ∑ p(symbol) * (-lg(p(symbol)))

Exercise

A language has three symbols:

“A” with probability 0.5

“B” with probability 0.25

“C” with probability 0.25

How would you encode them in binary?

Exercise

A language has three symbols:

“A” with probability 0.5

“B” with probability 0.25

“C” with probability 0.25

How would you encode them in binary?

Observations...

- Encode the symbols that happen often with really short bit strings
- Notice that the # of bits you need for these symbols is -lg(p(symbol))

Exercise

“A” with probability 0.5 (0)

“B” with probability 0.25 (10)

“C” with probability 0.25 (11)

E[symbol_length] = ∑ p(symbol_i) * symbol_length_i

= ∑ p(symbol_i) * -lg(p(symbol_i)) = 0.5 * 1 + 0.25 * 2 + 0.25 * 2 = 1.5.

Aside 1: Huffman Coding

The name of the algorithm you’ve already been doing.

The paper, for the incredibly curious:

http://compression.ru/download/articles/huff/huffman_1952_minimum-redundancy-codes.pdf

Information Entropy

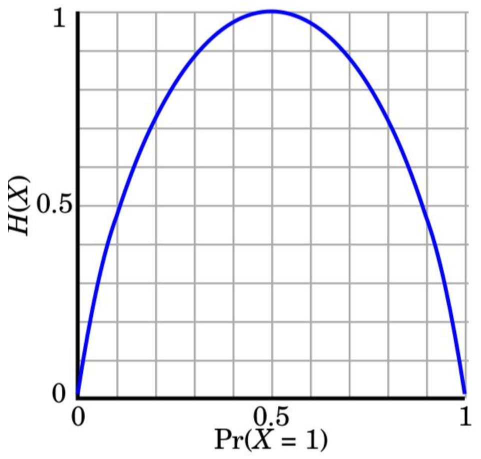
Abstract idea of "information" - usually measured in bits

- Similar to digital bits: one bit = {0,1}, two bits = {0,1,2,3}, etc.

Entropy: information needed to specify a random variable

- *In expectation: can be fractional*

$$\mathrm{H}(X) = \sum_{i=1}^{n} \mathrm{P}(x_i)\,\mathrm{I}(x_i) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i),$$



What if your probability distribution is wrong?

1. I give you a distribution of symbols
2. You assign them encodings
3. It turns out that my distribution is wrong!
4. We start sending symbols with your encoding.

What happens?

What if your probability distribution is wrong?

1. I give you a distribution of symbols (q(symbol))
2. You assign them encodings
3. It turns out that my distribution is wrong! (really p(symbol))
4. We start sending symbols with your encoding.

What happens?

Expected encoding length = $\sum p(x)$ * length of x = $-\sum p(x) \lg q(x)$

Cross-Entropy

Cross-entropy: measure of information between two samples

P(x) = “true” distribution
Q(x) = predicted/estimated distribution

Given optimal encoding of for Q, how many bits to encode p ~ P?

$$H(p,q) = -\sum_x p(x)\,\log q(x).$$

$$H(p,q) = \mathrm{E}_p[-\log q] = H(p) + D_{\mathrm{KL}}(p\|q)$$

KL Divergence

[KL-divergence explained]

KL Divergence: measure of "distance*" *between* two distributions

P(x) = “true” distribution
Q(x) = predicted/estimated distribution

Given Q, how many bits (on average) to specify P?

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}.$$

* KL Divergence is not an actual distance metric (not symmetric).

Aside: Minimizing Cross-Entropy

- Cross entropy is the general concept for probability distributions with multiple classes
- Applied to logistic regression reduces into familiar loss

$p(x)$ = real distribution, usually 1-hot

$q(x)$ = model estimated distribution

Since $p(x)$ is (very) sharp, you get no loss if $q(x)$ matches it exactly.

What if your probability distribution is wrong (2)?

p = real; q = original guess

Expected encoding length = $-\sum p(x) \lg q(x)$

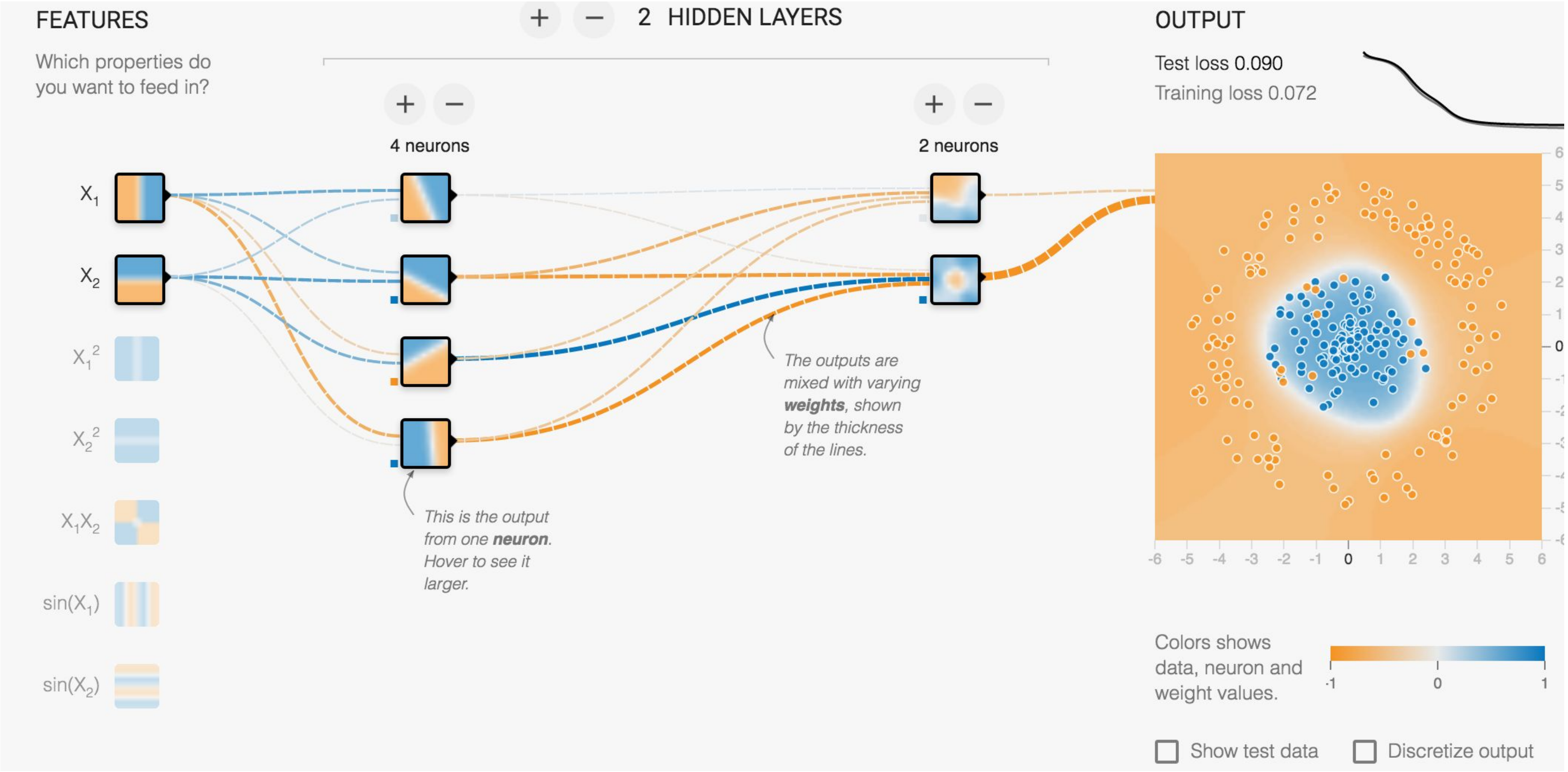
$= -\sum p(x) \lg p(x) + [\sum p(x) \lg p(x) - \sum p(x) \lg q(x)]$

$= - \sum p(x) \lg p(x) + \sum p(x) \lg (p(x)/q(x))$

= [Optimal encoding for p, had we known] + [extra because we were wrong]

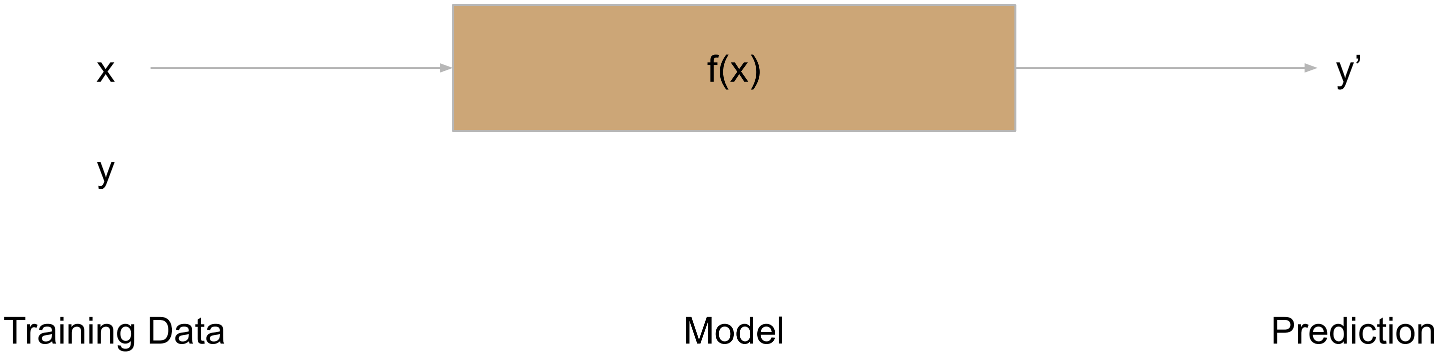
= entropy(p) + KL Divergence (P || Q)

Machine Learning and Simple Neural Nets



TensorFlow playground: http://playground.tensorflow.org/

Supervised Learning



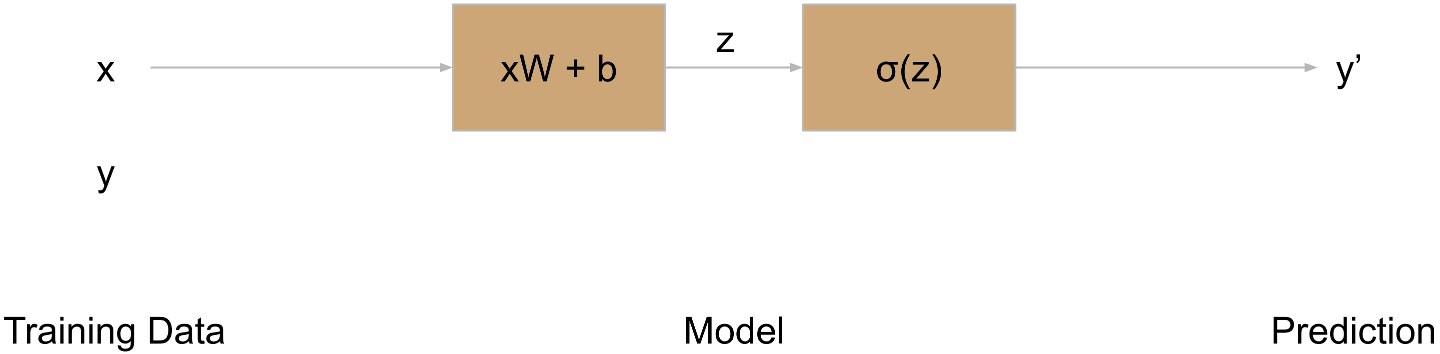
Supervised Learning

x = (x1, x2) → f(x) → y’

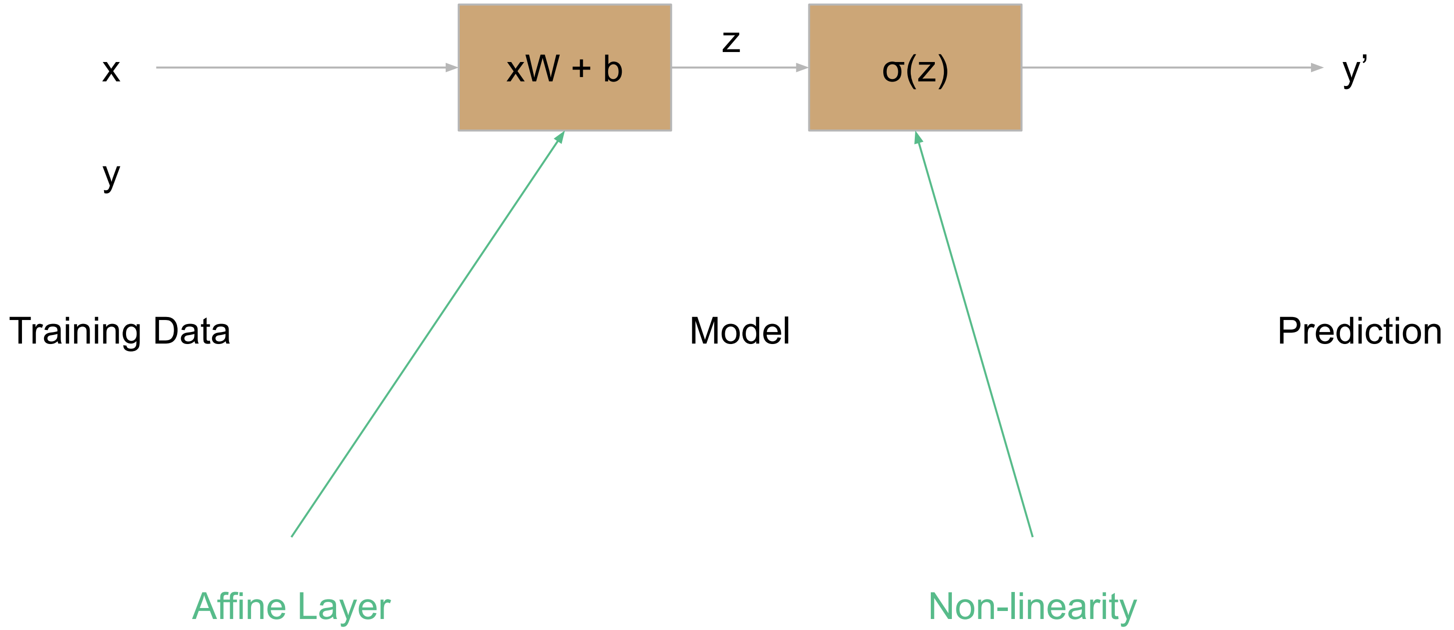
y = {“blue”, “orange”}

Training Data | Model | Prediction

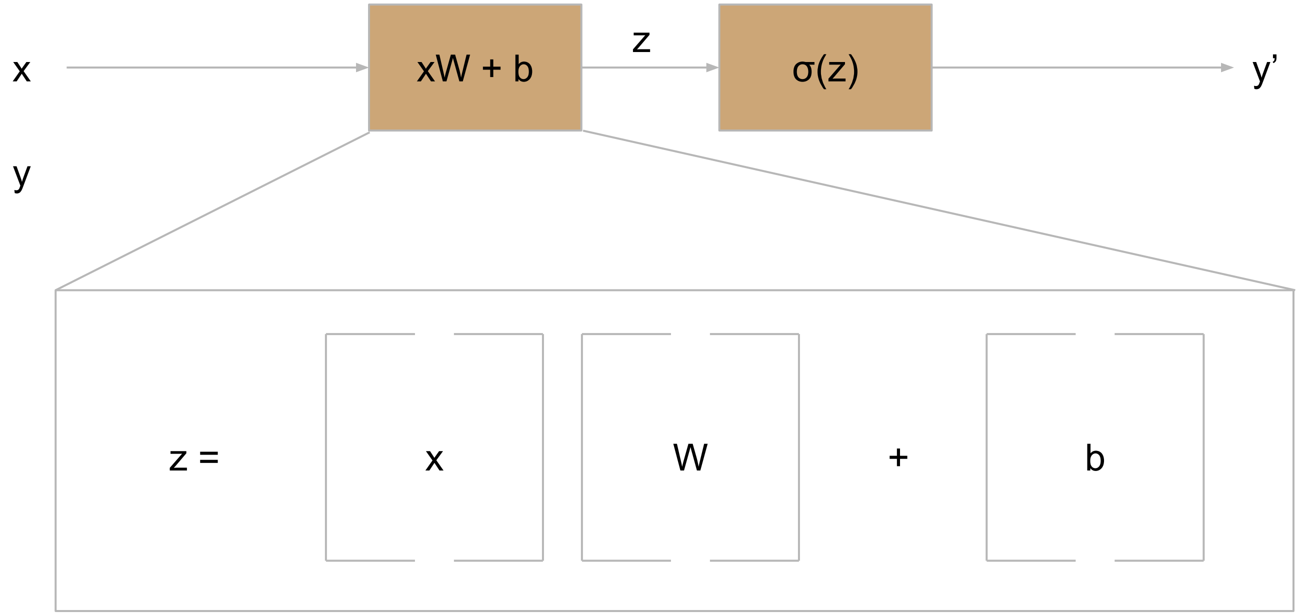
e.g. logistic regression: $y' = \sigma(x_1 w_1 + x_2 w_2 + b)$



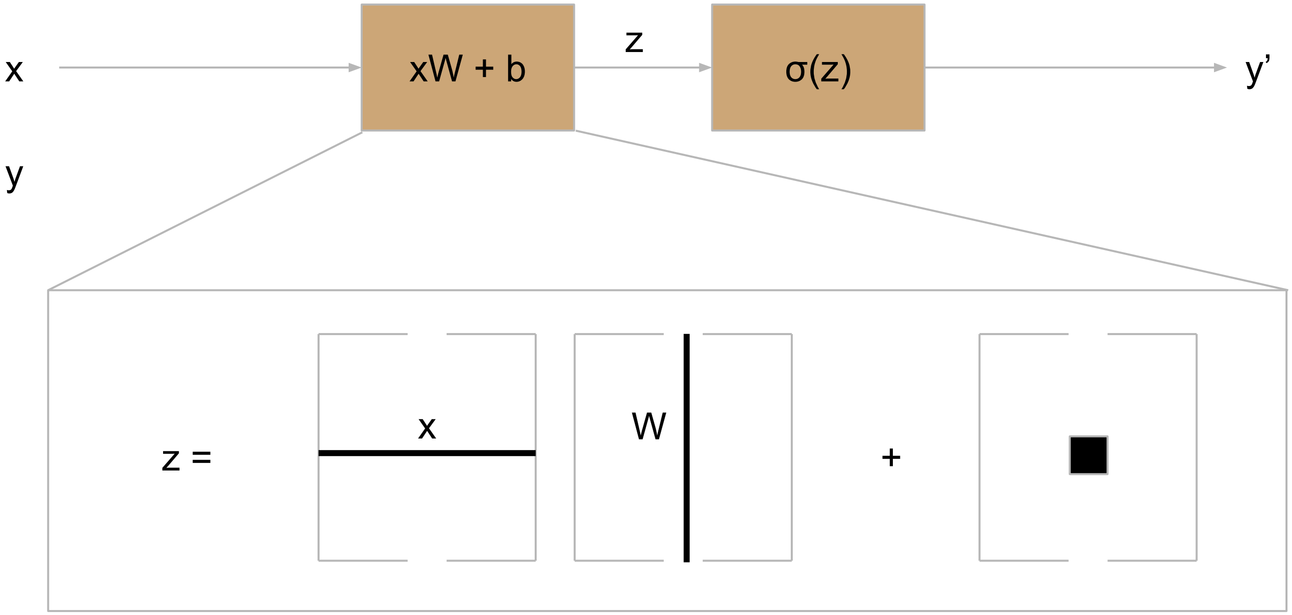
Logistic Regression



Logistic Regression



Dimensions?



Dimensions?

Fully-connected "Affine" Layer

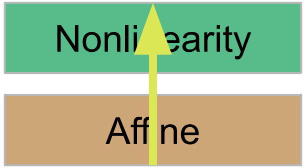
$$h = f(x W + b)$$

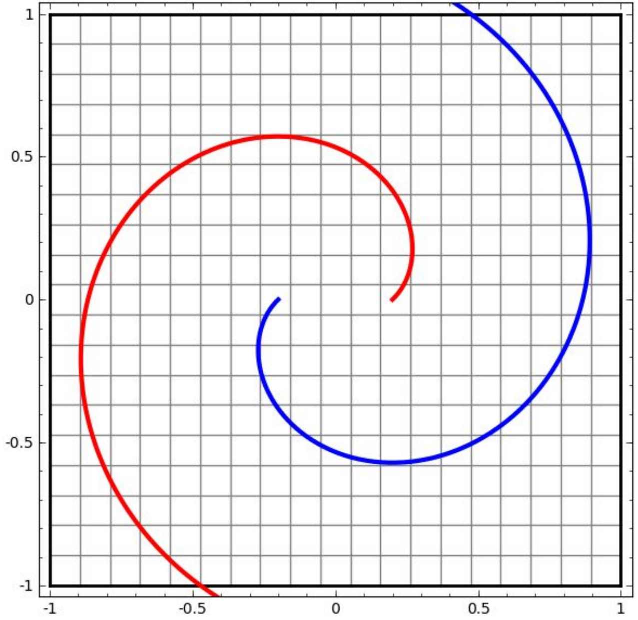
Affine layer:

- Matrix multiply **W** (rotate & scale)
- Bias term **b** (translate in space)

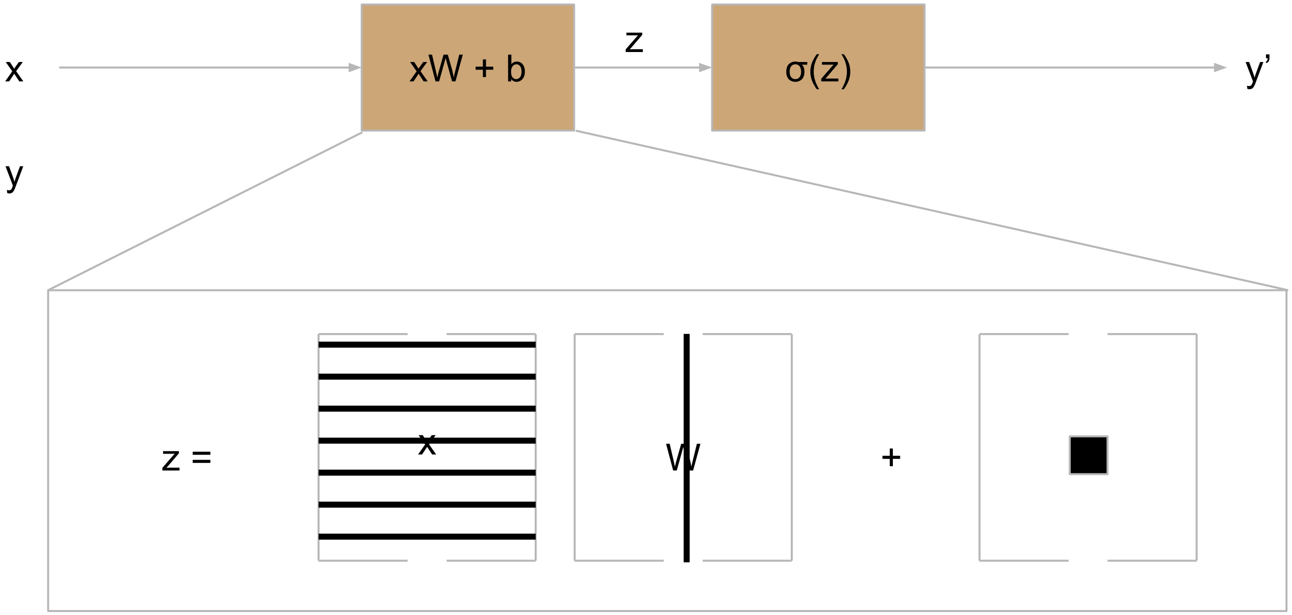
Then:

- Nonlinearity **f** (squish like dough)





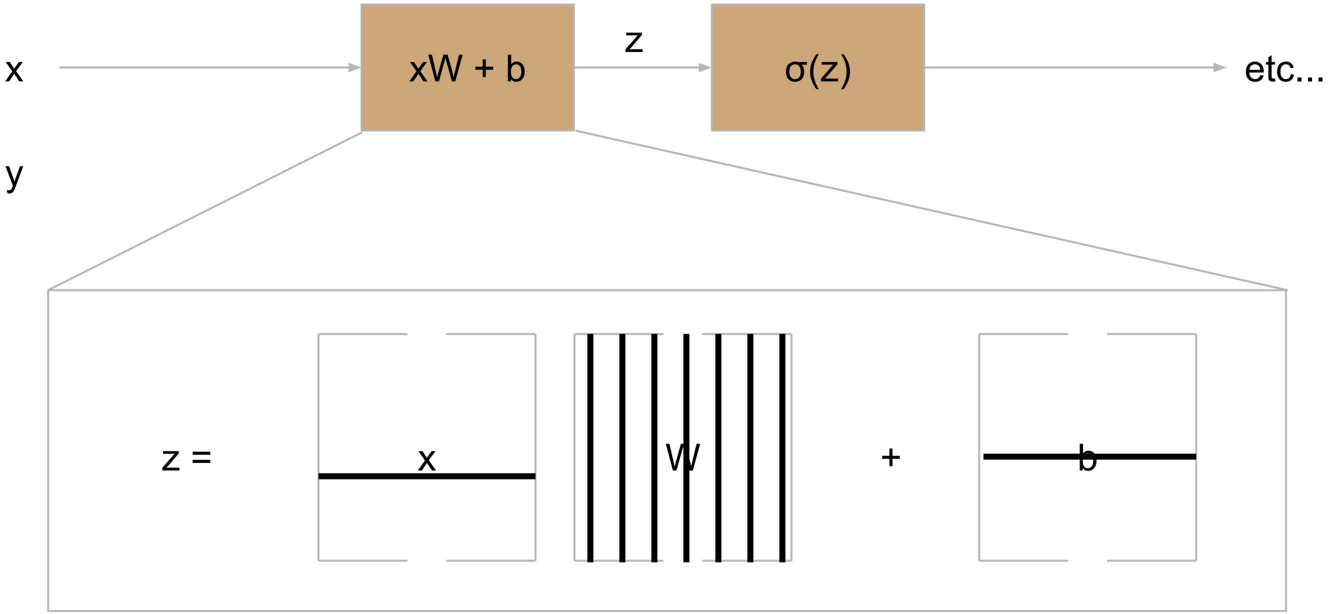
From "Neural Networks, Manifolds, and Topology" (Chris Olah, 2014)



Batching: more than one x at a time

$$\sum_i X_{ai} \; W_i + b \; = z_a$$

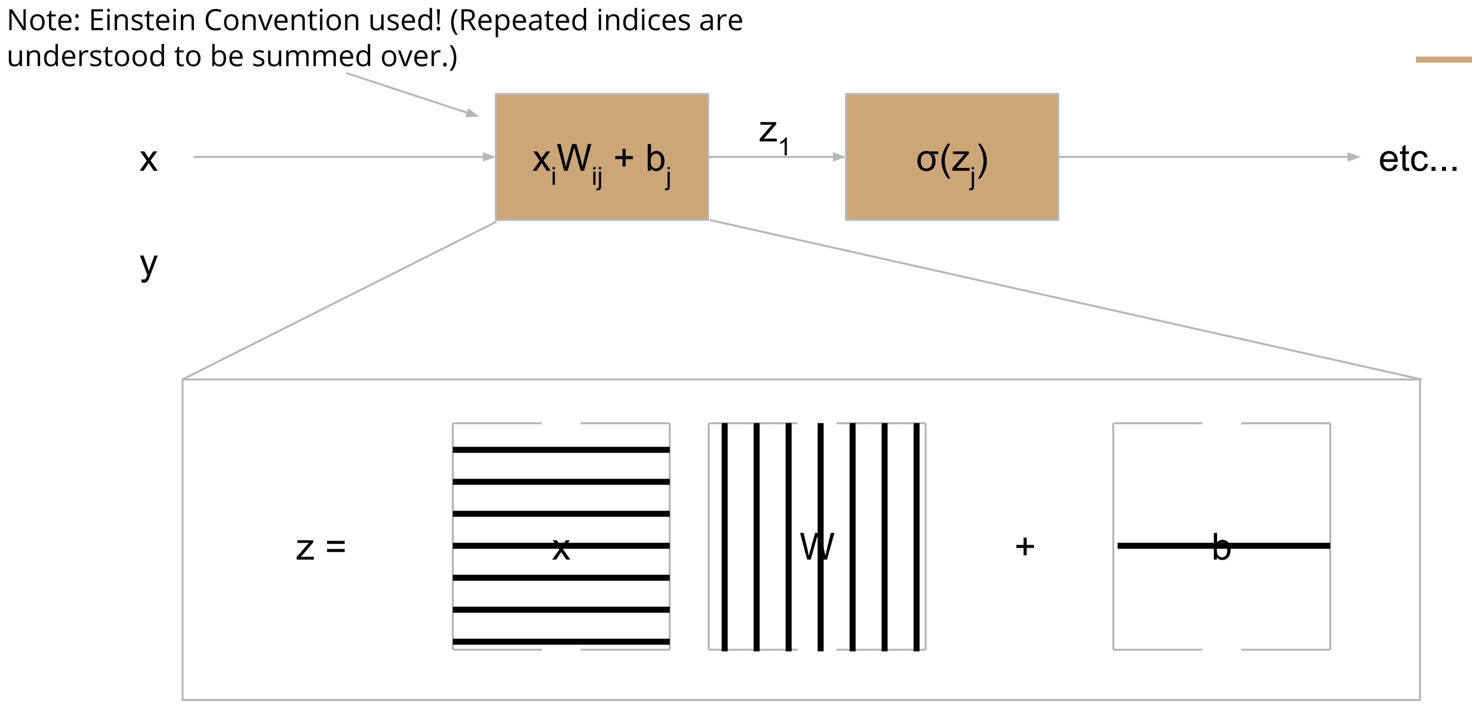
a: batch index, i: 'incoming' layer index



What if W represents multiple affine transformations?
Each transformation leads to a logistic regression…
...with its own weights and bias parameter!

$$X\ W_k + b_k = z_k$$

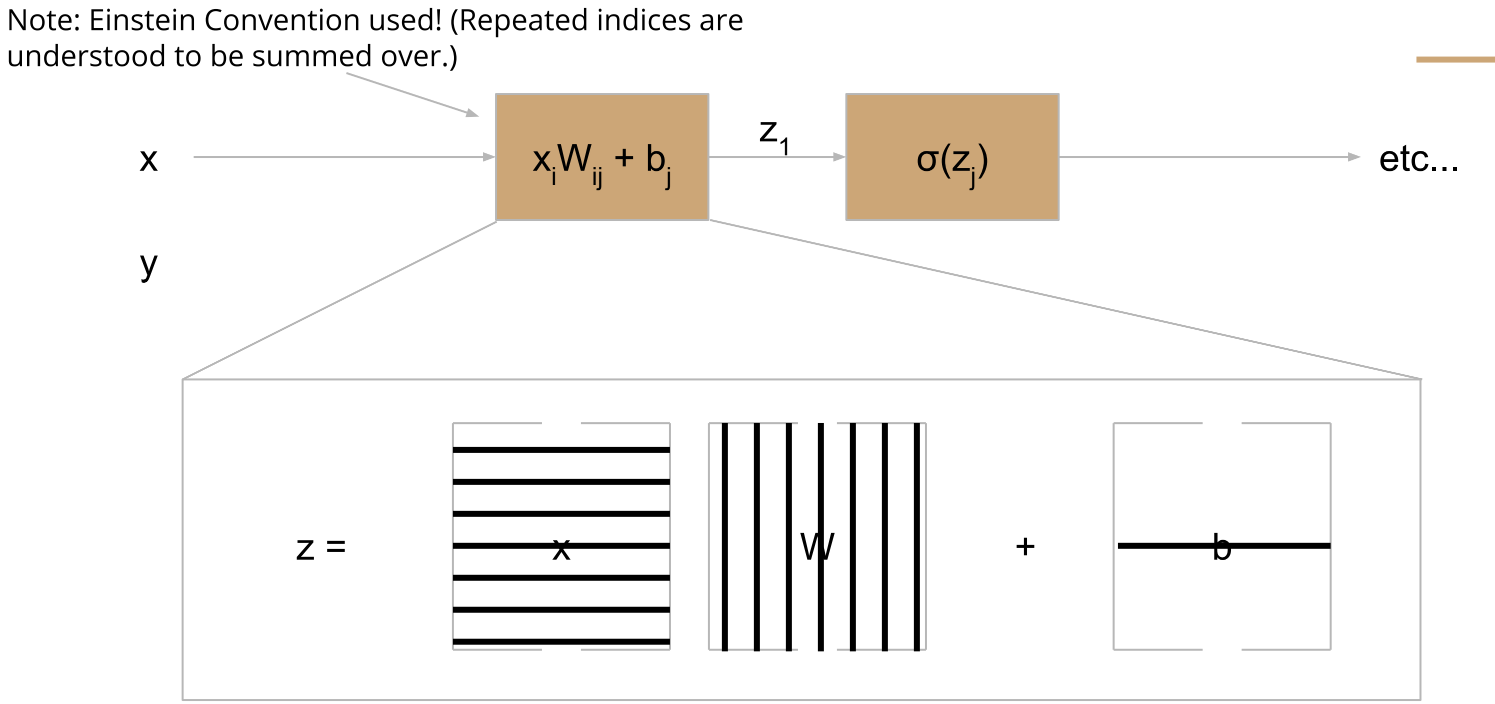
k: target layer index



Batching and multiple neurons at the same time

$$\sum_i X_{ai} \; W_{ik} + b_k \; = z_{bk}$$

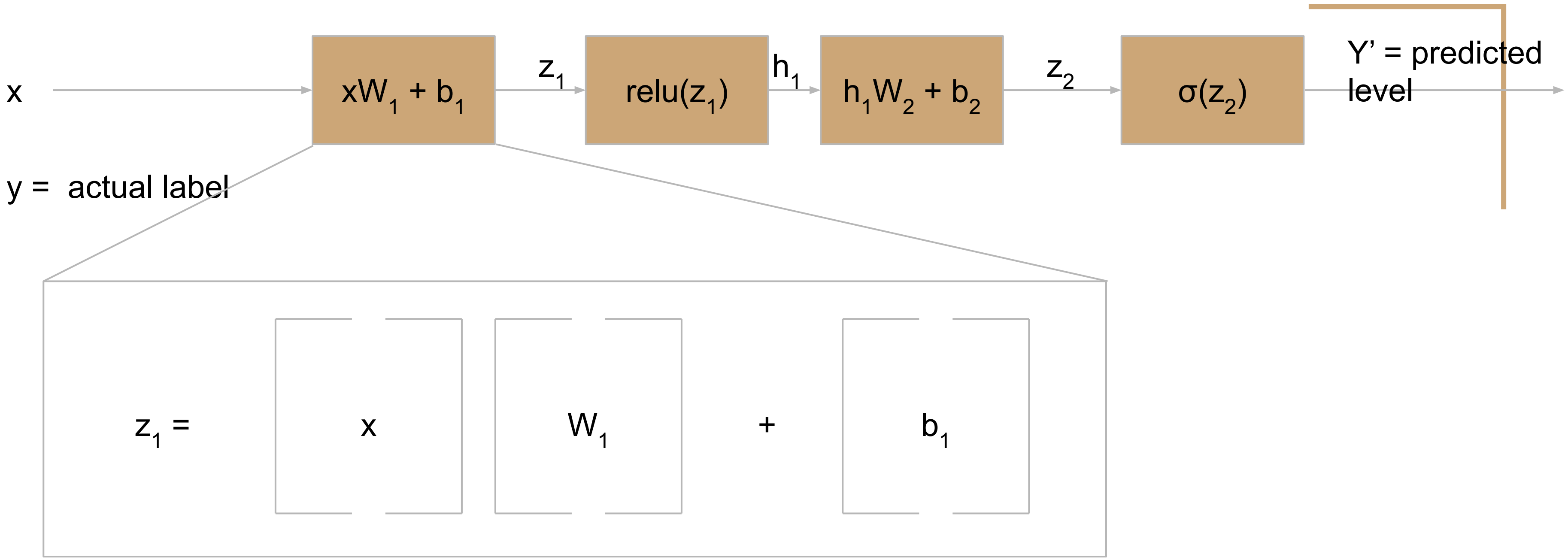
a: batch index, i: ‘incoming’ layer index, k: target layer index



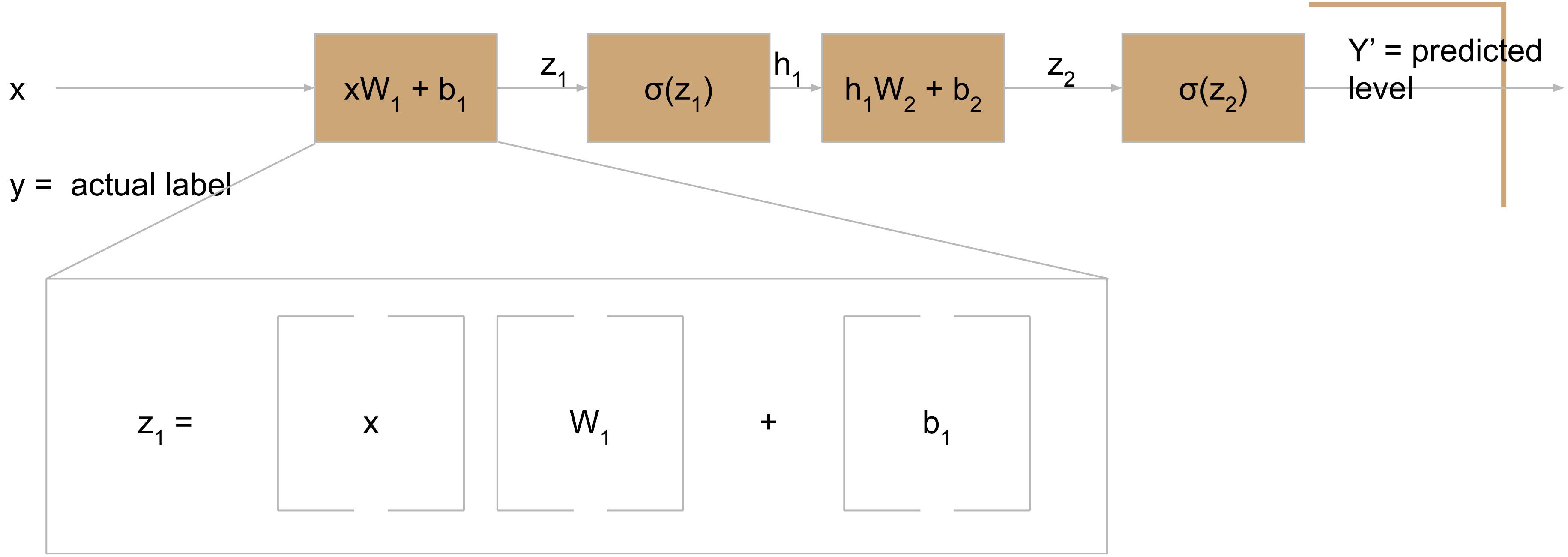
From Logistic Regression to Neural Nets:

Each neuron is doing a logistic regression… with its own weights and bias parameter!

(Note: non-linearity does not have to be sigmoid… more later!)



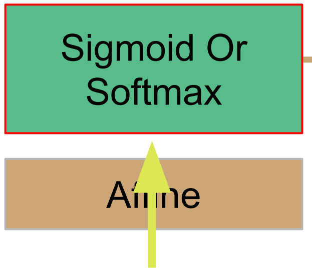
From one to more Hidden Layers:
Output of previous hidden layer is input for next layer



Question:

- 100-dimensional feature vectors x
- 10,000 examples
- 200 dimensional first hidden layer….

… ***What are the dimensions of W_1 and b_1?***



How do we actually classify?
Output Layer

Output Layer - Approach

If *k* classes, we need: ***k* probabilities** for each input

We have: a last Hidden Layer of (say) dimension d

How do we get the probabilities (k positive numbers summing to 1)?

1. Use one more affine layer to map from *d* to *k* dimensional vectors
2. Use Softmax:
   a. Exponentiate all elements of vector (now we have k-dim vector with all components positive)
   b. Divide all components by the sum of all components

Softmax

$h_{final} W + b$

e^z

$e^{zi} / sum(e^z)$

C_1	C_2	C_3
-7	-7	-7
54.5	403.4	0.0009
0.12	0.88	0.0

Softmax: the Wikipedia version!

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

Questions:

- *"If you have 100k classes... could there be a problem here?"*

Sigmoid Or Softmax

Affine

2 Classes: Just Sigmoid

Affine

$$h_{final} W + b$$

$$e^{z}$$

Sigmoid

$$1 / (1 + e^{-z})$$

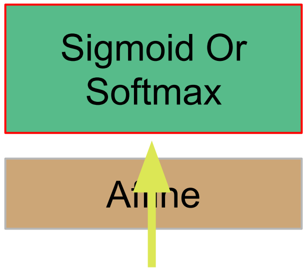
4

54.5

0.98

Question: *"How does this relate to Softmax?"*

2 Classes: Just Sigmoid



Affine

$h_{final} W + b$

e^z

4

54.5

0.98

Sigmoid

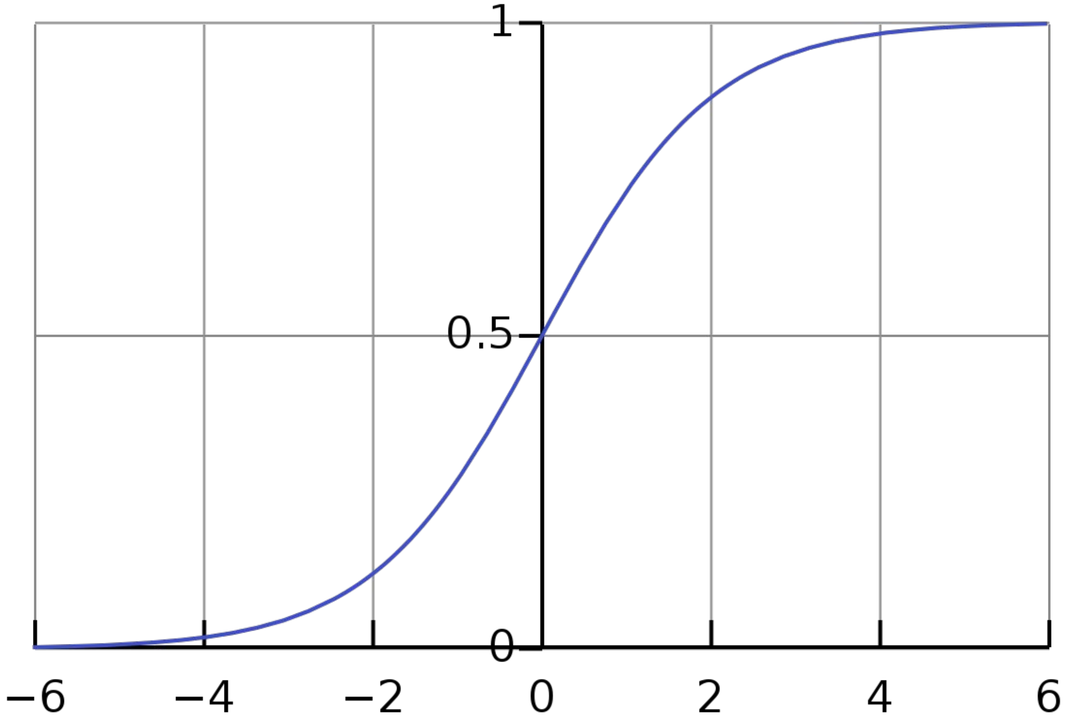
$$1 / (1 + e^{-z})$$

$$= e^z / (e^z + 1)$$

(**...** *Like comparing to 2nd class that had affine output = 0)*

Sigmoid: Intuition

Sigmoid Or
Softmax



Sigmoid is binary softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

$$P(potato|Z) = \sigma(Z) = \frac{1}{1 + e^{-Z}}$$

$$P(potato|Z) = softmax(Z)_{potato} = \frac{e^{Z_{potato}}}{e^{Z_{potato}} + e^{Z_{tomato}}}$$

Sigmoid is binary softmax

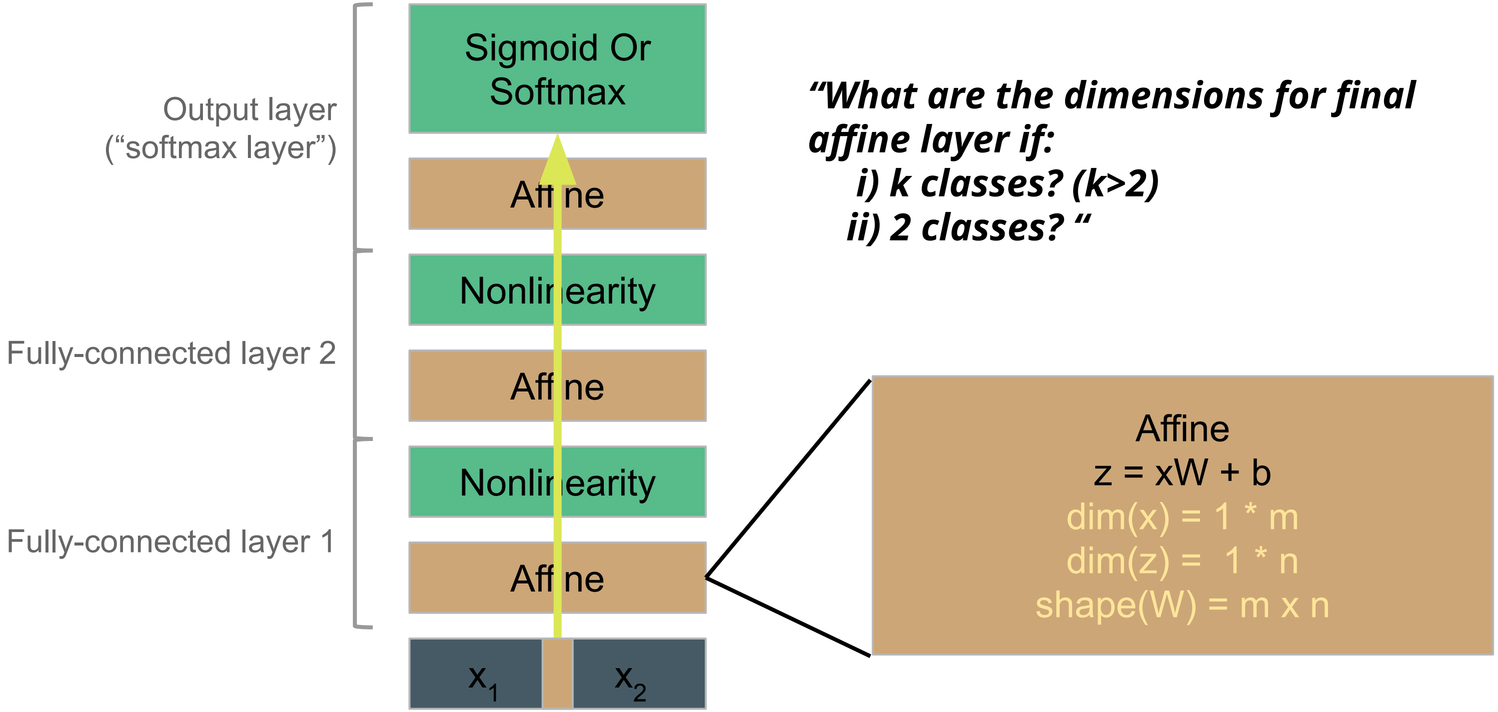
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

$$P(potato|Z) = \sigma(Z) = \frac{1}{1 + e^{-Z}}$$

$$P(potato|Z) = softmax(Z)_{potato} = \frac{e^{Z_{potato}}}{e^{Z_{potato}} + e^{Z_{tomato}}}$$

Putting it All Together

Fully Connected Network for Classification



"What are the dimensions for final affine layer if:
i) k classes? (k>2)
ii) 2 classes? "

Why Neural Nets?

Deep computation (multiple layers)

- Complex decision boundary
- Fewer parameters than big shallow network

Learned representations

- Word embeddings
- Learn jointly with objective

Key: avoid sparsity problem by computing in dense space

- Without sacrificing representation power

Hyperparameters & Other Comments

Hyperparameters:

- Number of layers
- Dimensions of layers
- Regularization parameters (*“what? I can overtrain?” “For sure!”*)
- Choice of non-linearities (sigmoid, tanh, relu,...)
- ....

Other Comments:

- Optimizers (SGD, adam, RMSProp)
- Overfitting is a real concern
- Cost function is not convex! But it still works...

FinBERT

Pre-trained model on SEC filings for financial natural language tasks

Justification

Sample Text Fragment from Training Corpus

Our total debt at December 31, 2018 was $5,960.1 million, compared to $5,957.1 million at December 31, 2017, net of the ***unamortized discount and issuance costs of notes issued under par*** of $91.1 million and $94.1 million at December 31, 2018 and 2017, respectively. This ***debt is all denominated in dollars at fixed interest rates***, weighed at 5.89%. The ratio of total debt to total capitalization was 47.4% at December 31, 2018, compared to 49.2% at December 31, 2017.

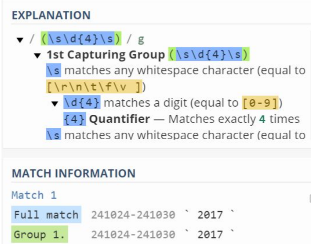
Mostly Dry Technical Language

Primary Goals

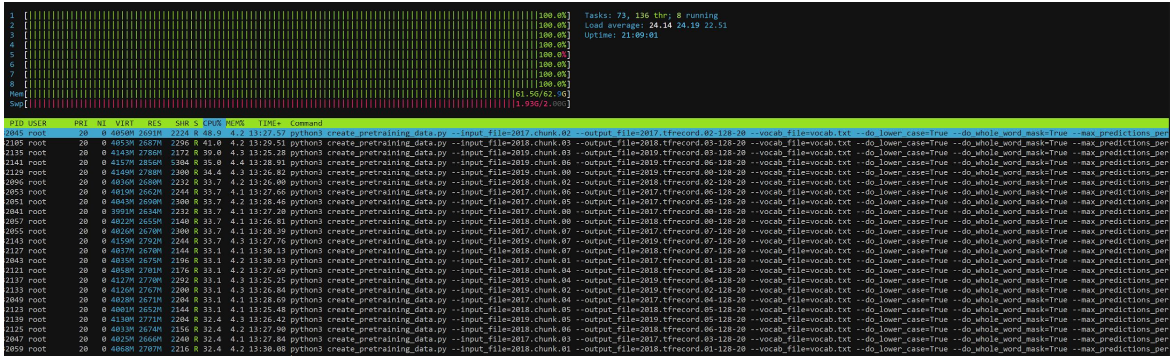
- Create new embeddings trained on annual 10-K financial filings
- Demonstrate that these domain-specific embedding understood financial context better than the generalized BERT embeddings
- Research the changes in financial language over the last 20 years

Data Acquisition and Cleansing

- **900 Gigabytes from SEC's EDGAR (XBRL)**
  - 131,153 10-Ks
  - 11,494 Corporations
- **30 Gigabytes of Training Data (TFRecords)**
  - Sharded 16 ways



30+ Regex Rules

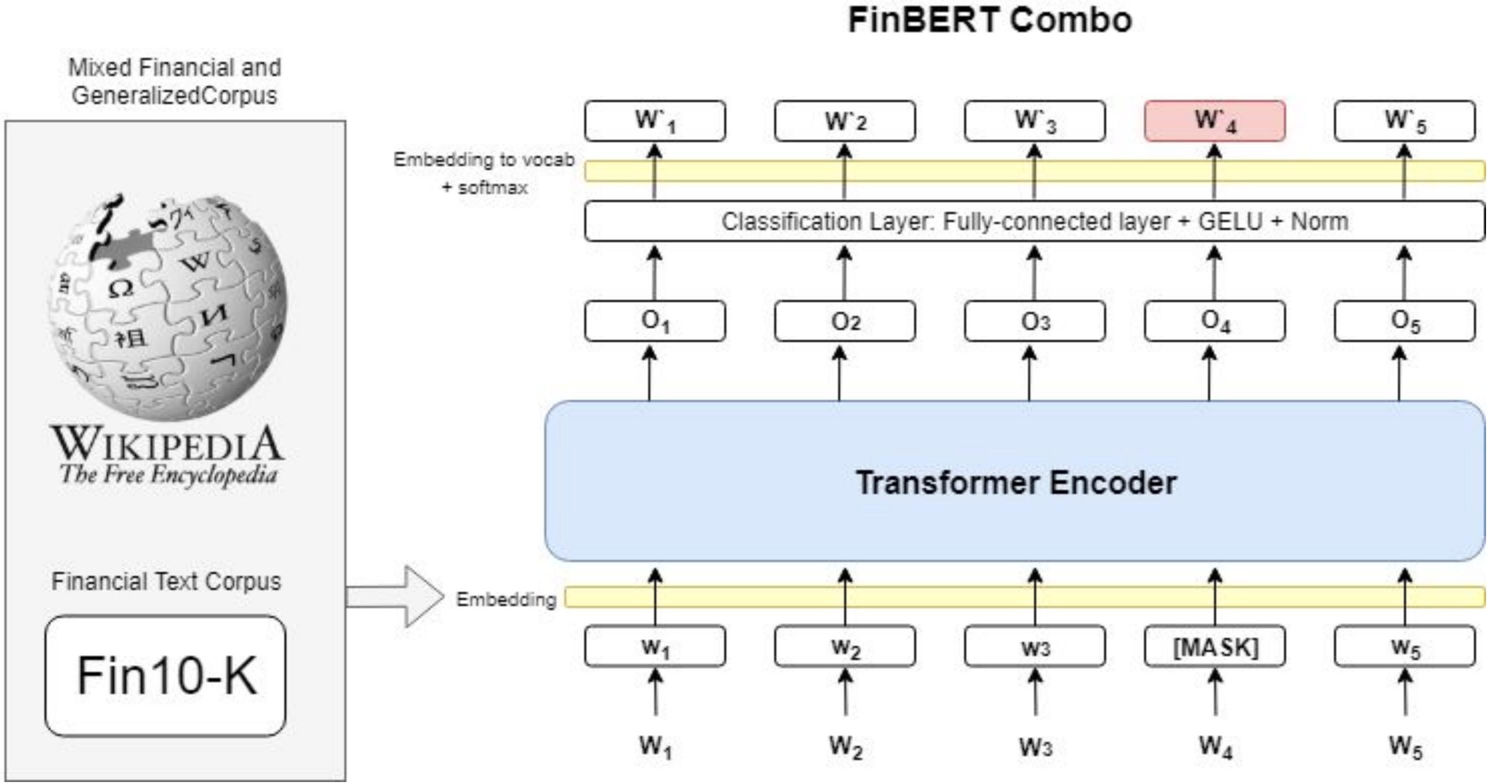


Parallelized Data Generation

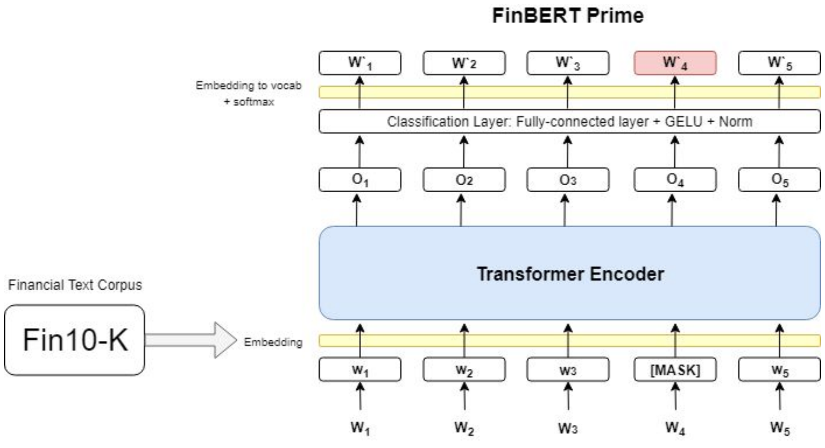
lxml

Significant Effort Spent on Building Dataset

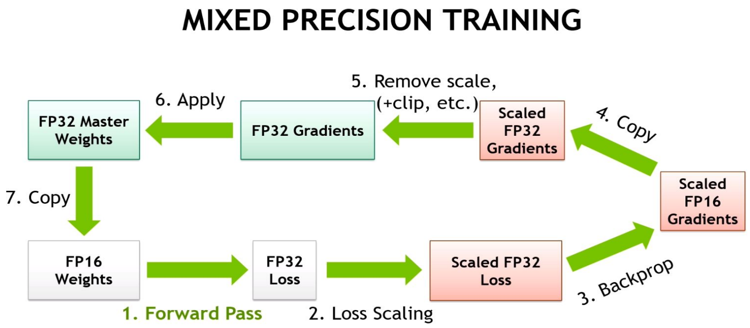
BERT Explanation

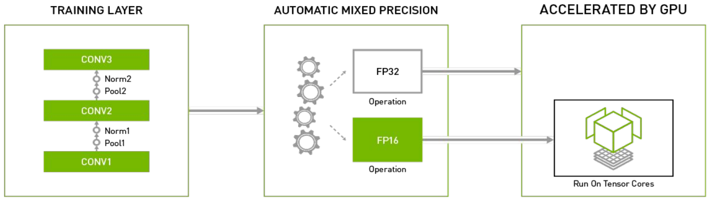


Pre-Training




```
root@v100x2: ~/w266-final/data/prep
v100x2                   Mon Jul 29 08:26:38 2019  418.67
[0] Tesla V100-PCIE-16GB | 40'C,  ?? %,  99 % | 15442 / 16130 MB | root:python3/123016(15427M)
[1] Tesla V100-PCIE-16GB | 42'C,  ?? %,  93 % | 15446 / 16130 MB | root:python3/123017(15427M)
```





Distributed Across Multiple GPUs

Pre-Training Results

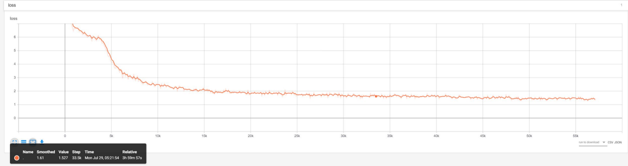
Pre-Trained Model	128MSL		512MSL	
	250K	500K	100K	200K
FinBERT-Prime	78.07%	81.30%	79.37%	76.14%
	97.88%	99.13%	98.38%	97.50%
FinBERT-Pre2K	84.67%			
	100.00%			
FinBERT-Combo	83.16%	87.16%	80.42%	
	98.88%	100.00%	98.13%	
Global Step	250K	500K	600K	700K

Accuracy of Masked Language Model

Accuracy of Next Sentence Prediction

Intermediate Checkpoints Used for Hyperparameter Evaluation

```
FinBERT-Combo_128MSL-100K/
FinBERT-Combo_128MSL-250K/
FinBERT-Combo_128MSL-500K/
FinBERT-Combo_128MSL-500K_512MSL-100K/
FinBERT-Pre2K_128MSL-250K/
FinBERT-Prime_128MSL-250K/
FinBERT-Prime_128MSL-500K/
FinBERT-Prime_128MSL-500K_512MSL-050K/
FinBERT-Prime_128MSL-500K_512MSL-100K/
FinBERT-Prime_128MSL-500K_512MSL-200K/
GooBERT/
```



Progress Monitored through Tensorboard

Result Driven Hyperparameter Adjustments

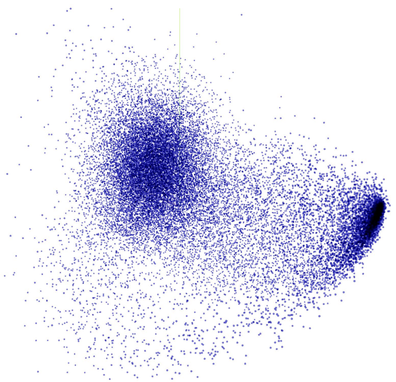
Performance on Validation Data

[CLS] the [company] has a fiduciary duty to its [shareholders]. [SEP]

Sample Predictions

Model	MLM	NSP	Loss
FinBERT-Prime	80.17%	98.50%	0.87
FinBERT-Pre2K	77.20%	91.88%	2.06
FinBERT-Combo	77.20%	90.63%	1.35
BERT	51.16%	62.38%	5.379

Table 5. Pre-Trained Evaluation on 2019



Interesting Clustering of Word Embeddings

```
import torch
from pytorch_transformers import BertTokenizer, BertForMaskedLM
```

```
S1 = '[CLS] the company has a fiduciary duty to its shareholders . [SEP]'
S2 = 'one of its many regulatory requirements . [SEP]'

MI = [2,12,19,20,]
```

```
tokenizer      = BertTokenizer.from_pretrained('bert-base-uncased')
text           = f'{S1} {S2}'
tokenized_text = tokenizer.tokenize(text)

for i in MI :
    tokenized_text[i] = '[MASK]'

print(tokenized_text)

indexed_tokens    = tokenizer.convert_tokens_to_ids(tokenized_text)
segments_ids      = [0] * len(tokenizer.tokenize(S1)) + [1] * len(tokenizer.tokenize(S2))

tokens_tensor     = torch.tensor([indexed_tokens])
segments_tensors = torch.tensor([segments_ids])
```

```
['[CLS]', 'the', '[MASK]', 'has', 'a', 'fi', '##du', '##cia', '##ry', 'duty', 'to', 'its', '[MASK]', '.', '[SEP]', 'one', 'of', 'its', 'many', '[MASK]', '[MASK]', '.', '[SEP]']
```

```
model            = {}
model['GooBERT'] = BertForMaskedLM.from_pretrained('GooBERT')
model['FinBERT'] = BertForMaskedLM.from_pretrained('FinBERT-Prime_128MSL-250K')
model['PreBERT'] = BertForMaskedLM.from_pretrained('FinBERT-Pre2K_128MSL-250K')
model['ComBERT'] = BertForMaskedLM.from_pretrained('FinBERT-Combo_128MSL-250K')
```

```
model['FinBERT-Prime_128MSL-500K_512MSL-100K'] = BertForMaskedLM.from_pretrained('FinBERT-Prime_128MSL-500K_512MSL-100K')
```

```
preds  = {}
for m in model:
    with torch.no_grad():
        preds[m] = model[m](tokens_tensor, token_type_ids = segments_tensors)[0]
```

```
d = ' | '
for m in preds:
    tokens = []
    for i in MI:
        predicted_index = torch.argmax(preds[m][0, i]).item()
        predicted_token = tokenizer.convert_ids_to_tokens([predicted_index])[0]
        tokens.append(f'{predicted_token:<12} [{predicted_index:>5}]')

    print(f'{m :<37} : {d.join(tokens)}')
```

```
GooBERT                               : state        [ 2110] | members      [ 2372] | important    [ 2590] | duties       [ 5704]
FinBERT                               : company      [ 2194] | shareholders [15337] | stock        [ 4518] | ##holders    [17794]
PreBERT                               : bank         [ 2924] | directors    [ 5501] | are          [ 2024] | ##rs         [ 2869]
ComBERT                               : company      [ 2194] | shareholders [15337] | directors    [ 5501] | is           [ 2003]
```

```
tokenizer.convert_ids_to_tokens([1998])[0]
```

```
'and'
```

MLM : 80% vs 50%, NSP : 99% vs 62%

Performance on New Data

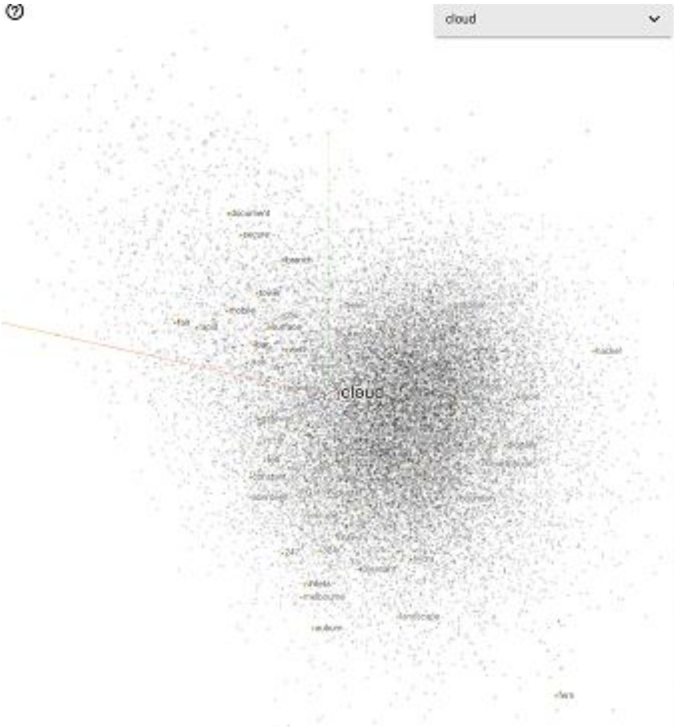
- 2 Datasets tested: 10Q and Earning Calls
  - 10Q are quarterly filings
  - Earning Calls are Analysts discussing 10K with management.

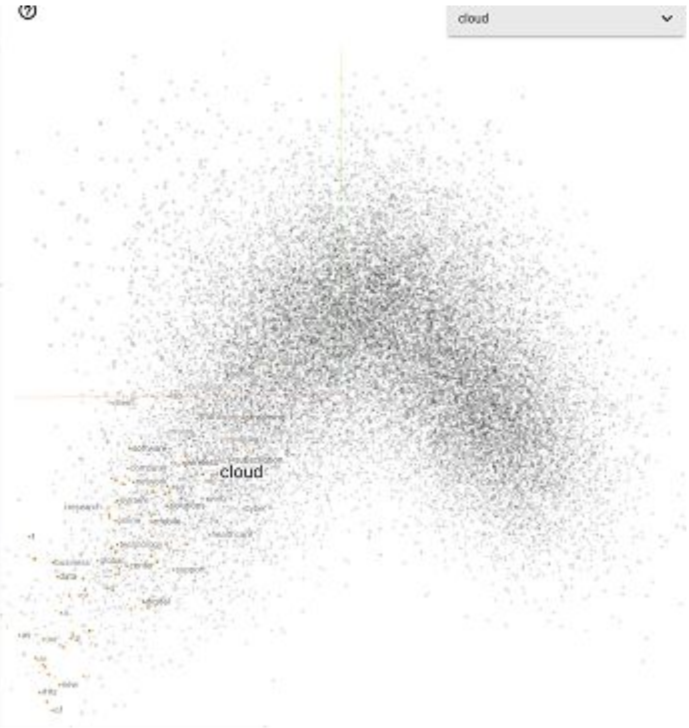
FinBERT understands!

Dataset	Model	MLM	NSP
10-Q	FinBERT-Prime	77.52%	94.50%
	FinBERT-Pre2K	70.33%	93.00%
	FinBERT-Combo	75.33%	94.38%
	BERT	51.18%	60.88%
Earnings Call	FinBERT-Prime	42.81%	53.13%
	FinBERT-Pre2K	38.44%	51.63%
	FinBERT-Combo	45.81%	56.38%
	BERT	46.87%	29.88%

Table 6. Test Results on 10-Q's and Earning Calls

Change in Financial Language over 20 Years





1999

2019

Change in Financial Language over 20 Years

Cloud

tower	0.830	subscription	0.669
victor	0.832	center	0.670
digitally	0.832	systems	0.670
surf	0.832	wireless	0.672
##eta	0.833	future	0.675
aberdeen	0.833	server	0.677
wastewater	0.834	hosted	0.678

1999

2019

Change in Financial Language over 20 Years

Taxes

interest	0.719
losses	0.722
laws	0.725
distributions	0.728
items	0.728
rates	0.729
properties	0.729

revenues	0.642
##s	0.642
,	0.645
amounts	0.647
the	0.647
for	0.648
obligations	0.649

1999

2019

Conclusion and Future Work

- Build a sentiment analysis by Fine-Tuning Earning Calls on top of FinBERT - Interpret the sentiment of questions posted by the analysts and their score
  - Stock Prediction shouldn't be the goal - many variables are involved. But changes in analysts mind can be trained.
- Get a new dataset for Question and Answers, so people can ask financial questions to FinBERT (FiQA?)

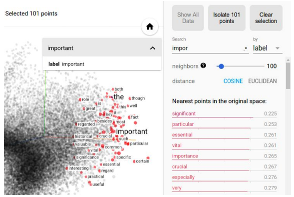
Artifacts

Repository https://github.com/psnonis/FinBERT

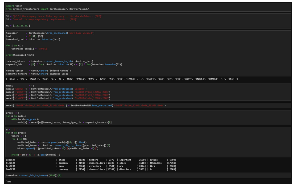
Dataset http://people.ischool.berkeley.edu/~khanna/fin10-K/

Paper https://github.com/psnonis/.../FinBERT - DeSola, Hanna, Nonis.pdf

Plus



Tensorboards



Notebooks

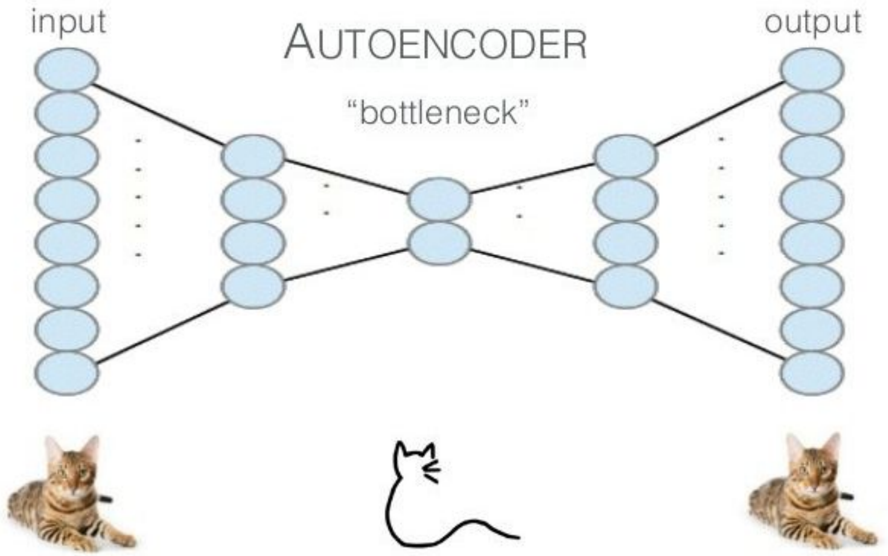
Dockerfiles

All Code Openly Available on GitHub

Autoencoders - Unsupervised Deep Learning

Vinicio De Sola

Theory Refresher - Primer



- We can use CNN, RNN, or LSTM for the architecture
- We will produce one output that's reconstructed from the original
- Left NN it's called Encoder, Right NN Decoder

Time Series - Options

- Options are a derivative product that depends on a underlying Stock or Index
- We care about three possible acts of manipulation
  - Best Execution
  - Mini Manipulation
  - Spoofing / Layering
- We create “images of time”
  - 10 min windows
  - 5 series: Price / Best Ask Option / Best Bid Option / Best Ask Stock / Best Bid Stock

Outlier Detection

- We use a CNN encoder
- Dimensions of our tensor: (5, 601)
- Number of tensors and trades: 132000
- We have to standardize the vector - Z-scores
- Once trained, we reconstruct all the trades
    - We use MSE to score the difference between shapes
    - Any trade with a MSE higher than 10 is considered an outlier

MSE: 35.08