Feed-forward and Convolutional Neural Networks

Introduction and Usecase

Nathan Danneman

May 20, 2020

Acknowledgements

Big thanks to:

- Charlotte Data Science, Machine Learning and AI Meetup
- Bojana and Maithili
- Data Machines (www.datamachines.io)
- DARPA

Outline

1. Just enough theory/background
2. Feed forward neural networks
3. Convolutional neural networks
4. Use case: Mask fusion for image manipulation localization

About This Talk

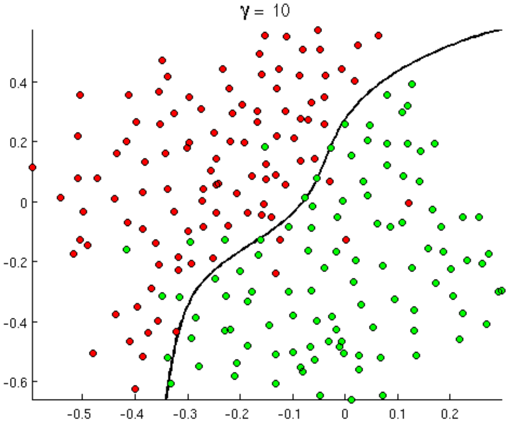
Goals:

- Give you nodding familiarity and ample pointers
- Leave you a starter code-base

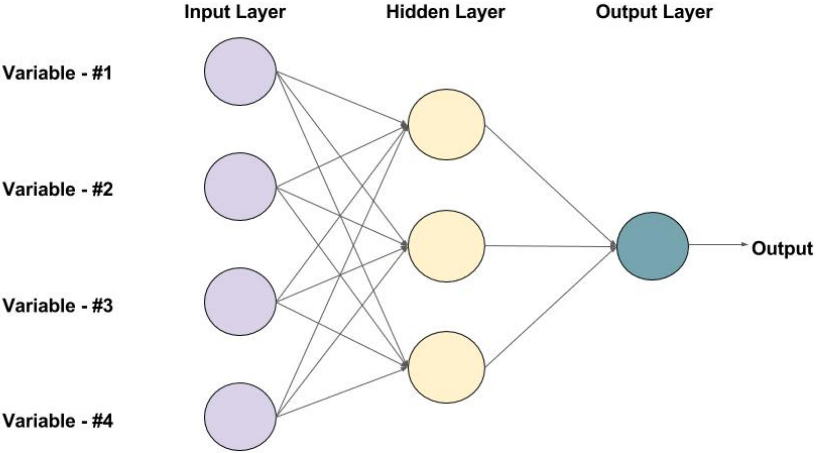
This talk lightly assumes:

- some familiarity with Python
- knowledge of general data science concepts

Supervised Models Are Function Approximators



The Feed Forward Neural Network



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

Neurons

Neurons take in the outputs of upstream neurons (or raw data).

They multiply that by a matrix of **weights**.

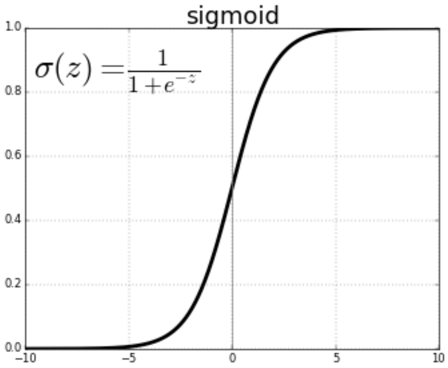
Add a **bias** term.

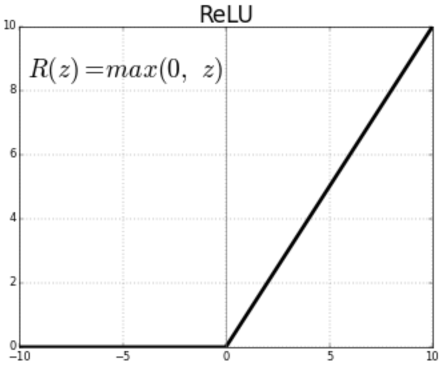
And pass that entire sum through an **activation function**.

For inputs X, weight matrix W, bias b, and activation function f:

Neuron output $= f(WX + b)$

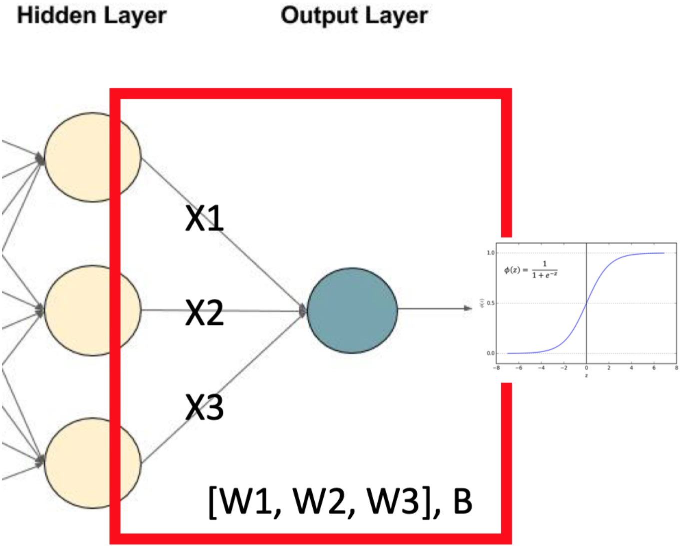
Activation Functions





datamachines

Understanding a Neuron



Learning/Estimation/Function Approximation

Remember: The point is (flexible) function approximation.

In this case, approximation is optimization with respect to a **loss function**.

We **learn** the weights and biases via stochastic gradient descent.

What is Keras?

Light wrapper over TensorFlow.

https://keras.io

APIs allow simplicity and still power.

Great for **using** neural networks, even very complex ones.

Poor for research **on** neural networks.

Setup

pip install keras

– it brings along TensorFlow, numpy, etc

(that was easy)

Setup for GPU usage

Code Example: Iris Classification

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
7.0	3.2	4.7	1.4	versicolor
...	...	...	...	...
...	...	...	...	...
5.9	3.0	5.1	1.8	virginica

A Simple Model

```
model = Sequential()

model.add(Dense(3, input_dim=4))
model.add(Dense(5))
model.add(Dense(3, activation='sigmoid'))

optimizer_details = SGD(lr=0.01, decay=1e-6)
model.compile(loss='binary_crossentropy',
              optimizer=optimizer_details)

model.fit(X_train, Y_train, epochs=20)
```

Hyperparameters and Overfitting

Hyperparameters:

- Number of layers
- Number of neurons in each layer

Strategies to avoid overfitting:

- Premature halting
- Dropout
- Regularization

Visit http://playground.tensorflow.org to see how hyperparameters affect outcomes in real models of toy data.

Pragmatics

- Start with an overly expressive model
- Track measure(s) of interest on test data; terminate upon plateau
- Match the problem type, output layer, and loss function

Problem Type	Output Layer	Loss Function
Regression	1-unit Linear	Mean Squared Error
Binary Classification	1-unit Sigmoid (softmax)	Binary Crossentropy
k-Class Classification	k-unit Sigmoid (softmax)	Categorical Crossentropy

Applicability

Q: When might you use a feed-forward neural network for regression or classification?

A: Probably never.

Image Classification with Neural Networks

Where **do** people get value out of neural nets?

Image Classification with Neural Networks

Where **do** people get value out of neural nets?

Images are arrays: Height by Width by 3 (RGB)

Why not "unwrap" them then use a set of fully connected layers?

- Tons of parameters
- Pixel location is relative

Convolutional Neural Network (CNN)

Conv Nets solve those problems by:

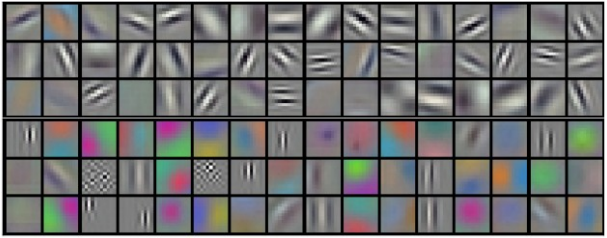
- Parameter sharing
- Looking at an image hierarchically

Idea:

- Find a set of micro-pictures that roughly capture the content across patches of all images (convolution)
- Then, let a single pixel represent its neighborhood, dropping the rest of the information (pooling)
- Repeat, generating a hierarchical view

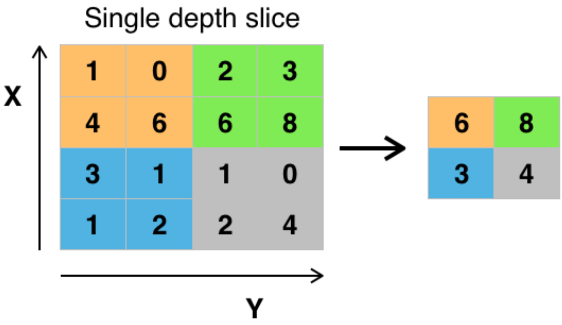
Convolutional Filters

"Find a set of micro-pictures that roughly capture content in every patch of all images"

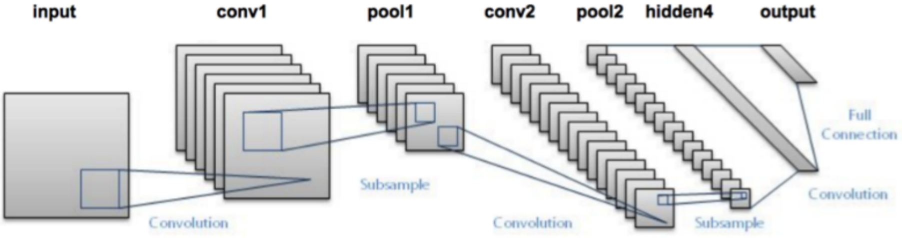


Pooling Operation

"Let a single pixel represent how much each filter matches the surrounding patch"



Repeat for Hierarchical Image Description



To learn:

- filters (which are just weights and biases with structure)
- weights and biases for hidden/output layers

A Tiny Example: Model Setup

```
from keras import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense
from keras.layers import Flatten
from keras.optimizers import SGD

model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5),
input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

A Tiny Example: Model Fitting

```
model.compile(loss=categorical_crossentropy,
              optimizer=SGD(lr=0.01),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1)
```

Some Pragmatics

- Are your images special?
  - This is unlikely; consider fine-tuning an existing model.
- Are your images all the same size?
  - Either resize or consider pyramidal layers.
- Are your images large? Or many of them?
  - You likely need a GPU

A Challenging Use Case

Program Challenge: Write software that identifies altered images

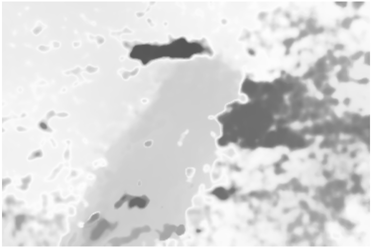
Researchers write analytics that output scores and masks

My team tries to **fuse** the masks into a single, optimal mask.

Example Data

Example Data

Example Data



Mask Fusion: Some Intuition

- Spatial smoothing
- Handle missing data
- Lightly parameterized – modest data
- Output is same shape as input image

Approach

- Large filters for spatial smoothing
- Not too deep a model (fewer paramters)
- Sized all images to 128x128, resize output back to original shape
- Missing data placeholder to -1

Eventually: Added information from an image segmentation model.

Example Autoencoding Setup

```
mod = Sequential()
# Encoder Layers
mod.add(Conv2D(18, (4,4), input_shape=(256, 256, 27)))
mod.add(BatchNormalization())
mod.add(MaxPooling2D((4,4), padding='same'))
# Decoder Layers
mod.add(UpSampling2D((4,4)))
mod.add(Conv2D(1, (4,4), activation='sigmoid'))
```

Results

Short Version: Better than the best single analytic by a large margin.

Resources

- Keras: https://keras.io
- Keras tutorial: https://elitedatascience.com/keras-tutorial-deep-learning-in-python
- Intro to CNNs: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
- Mathy intro to CNNs: https://arxiv.org/pdf/1511.08458.pdf
- Advanced CNN architectures: https://slazebni.cs.illinois.edu/spring17/lec04_advanced_cnn.pdf
- Hyperparameter tuning: http://playground.tensorflow.org

datamachines

Questions?

Code and slides: https://github.com/nhdanneman/intro_to_dl

Contact: nathandanneman [at] datamachines [dot] io