Deep Learning with MRI

Neurotech MTL

Installation

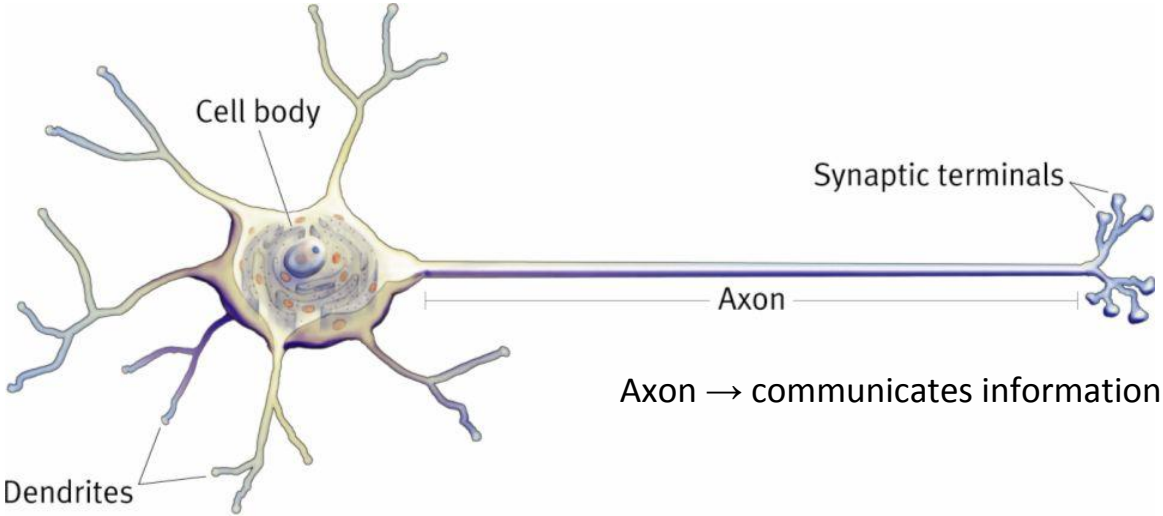
- Google Colab (super easy)
  - Create / Log-in to Google account
  - Open: https://drive.google.com/open?id=1dS3qoX52m2wsFWvCoBtr9lQhGextH8SO
- Docker (very easy):
  - Install docker on your OS
    - https://docs.docker.com/install/#cloud
  - docker pull tffunck/neurotech:latest
  - docker run -it --rm tffunck/neurotech:latest
- DIY (pretty easy):
  - wget https://bootstrap.pypa.io/get-pip.py
    - Or go to the link and download manually
  - python3 get-pip.py
  - pip3 install pandas numpy scipy h5py tensorflow keras
  - git clone https://github.com/tfunck/minc_keras
- Data
  - Unzip : minc_keras/data/ouput.gz.bz2

Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

Super simple neuroanatomy

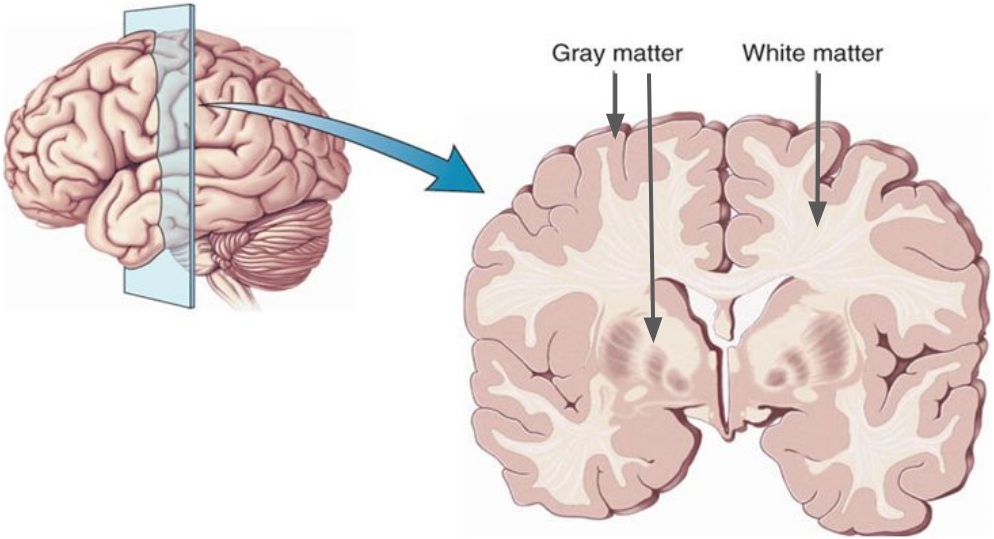
Cell body and dendrites → integrate information



Axon → communicates information downstream

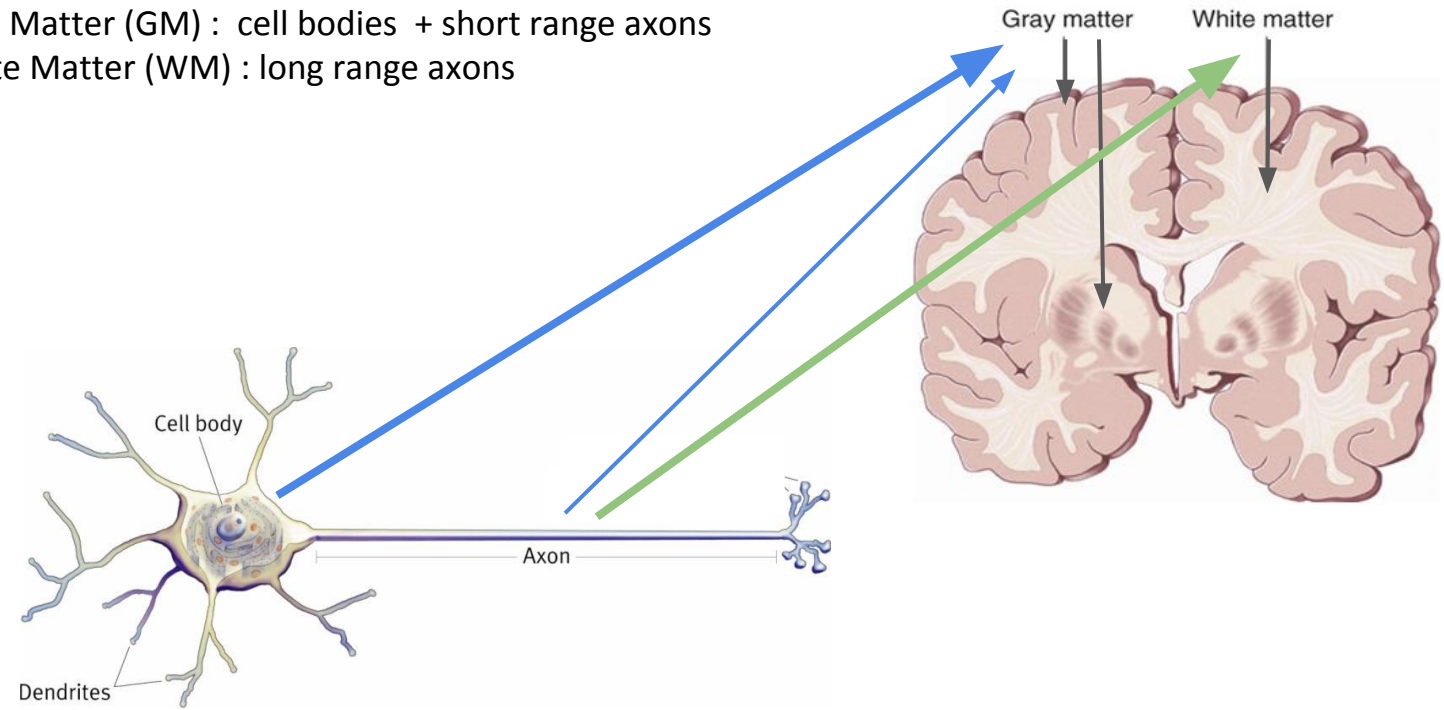

abp

Super simple neuroanatomy



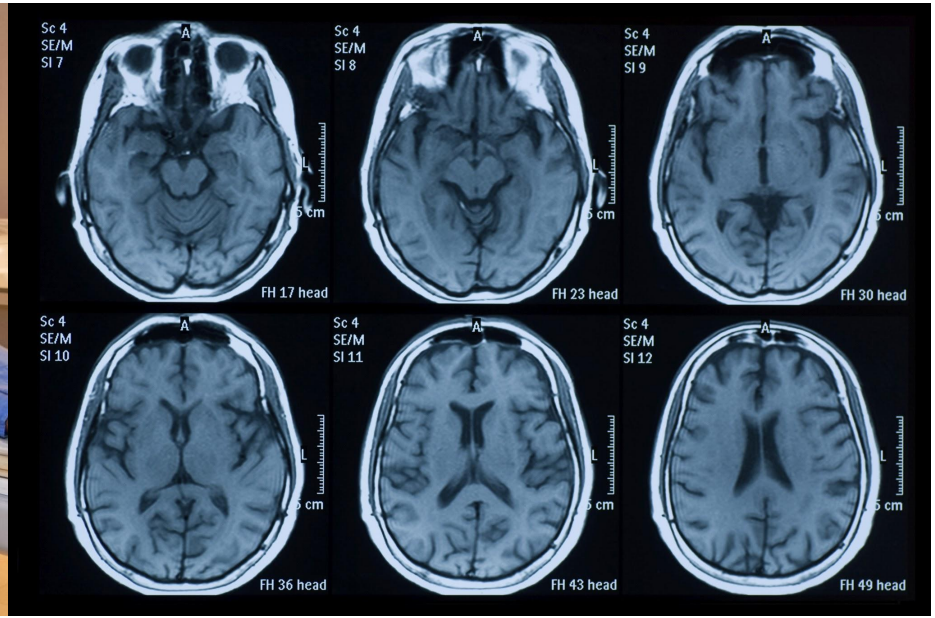
Super simple neuroanatomy

Grey Matter (GM) : cell bodies + short range axons
White Matter (WM) : long range axons



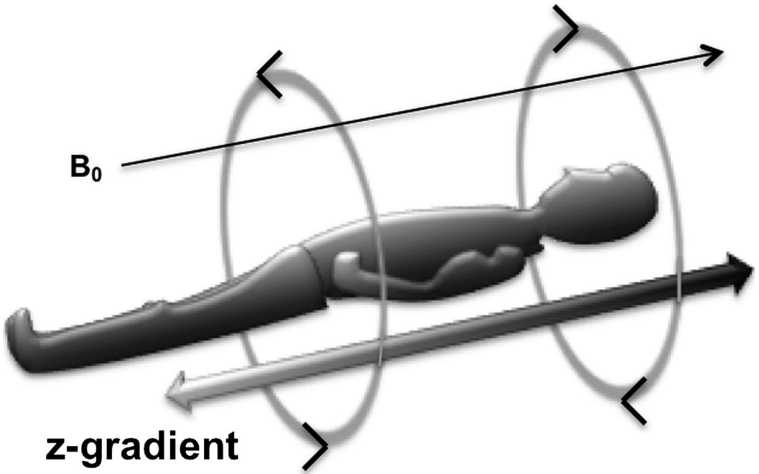
Magnetic Resonance Imaging (MRI)

- MRI is an imaging technique that uses powerful magnets to create images of biological tissue (e.g., brains)



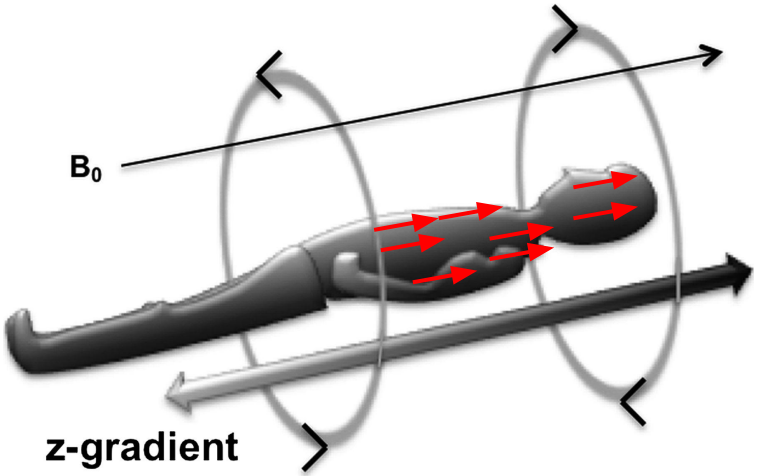
Magnetic Resonance Imaging (MRI)

1. MRI scanners first create a magnetic field along the axis of the scanner



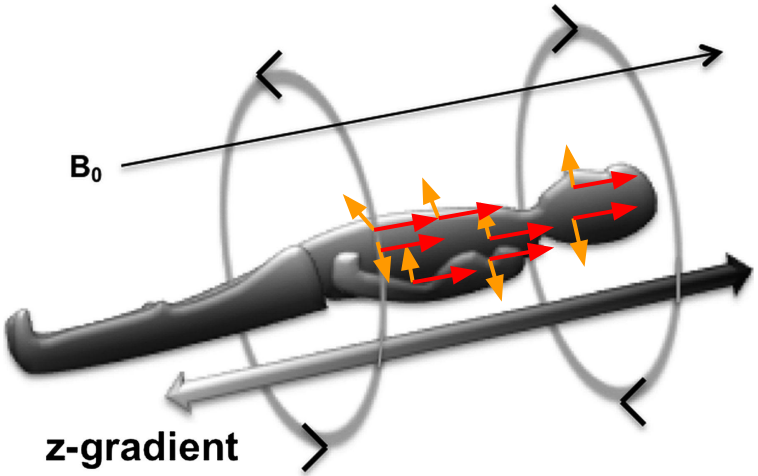
Magnetic Resonance Imaging (MRI)

1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field



Magnetic Resonance Imaging (MRI)

1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field
3. MRI emits magnetic pulse that knocks H atoms out of alignment

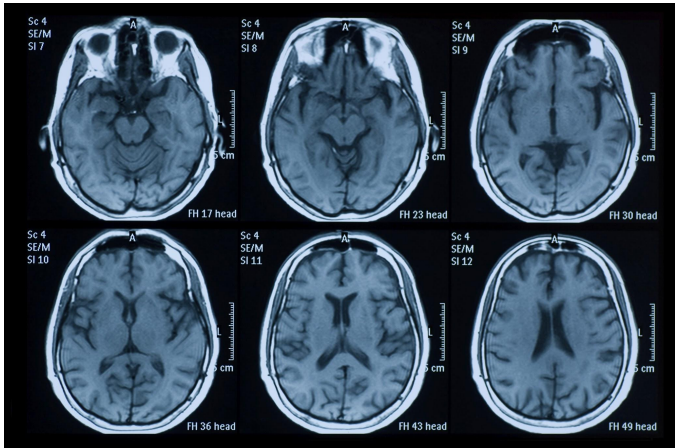


Magnetic Resonance Imaging (MRI)

1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field
3. MRI emits magnetic pulse that knocks H atoms out of alignment
4. Time it takes for H atoms to regain alignment depends on biological tissue

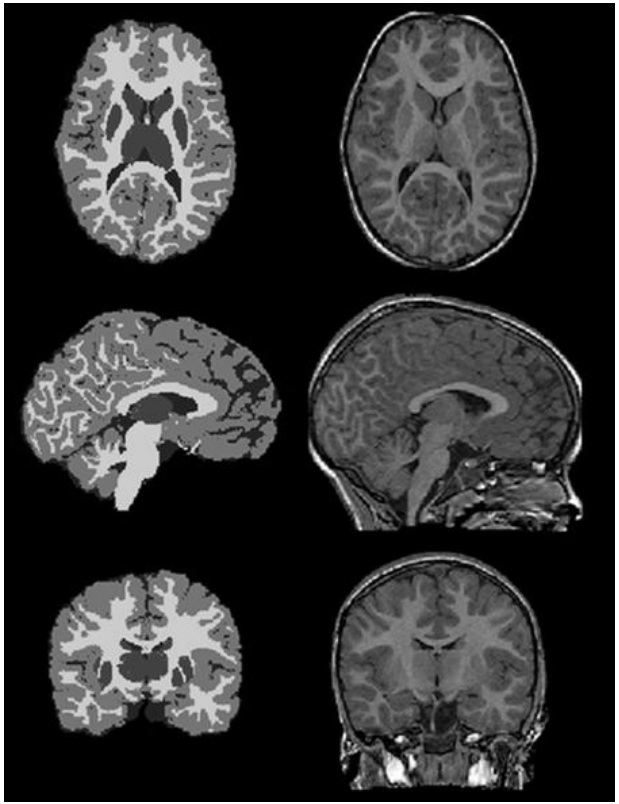
Magnetic Resonance Imaging (MRI)

1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field
3. MRI emits magnetic pulse that knocks H atoms out of alignment
4. Time it takes for H atoms to regain alignment depends on biological tissue
5. MRI image is based on this realignment time and this reflects type of tissue
    a. Realignment speed : WM (bright) > GM (medium) > CSF (dark)



MRI Segmentation

- Lots of ways to analyze MRI
  - Brain size
  - Cortical thickness
  - GM/WM intensity ratio
- Segmenting MRI very useful
  - Segmenting into GM and WM is common processing step
  - Helps to quantify brain metrics measured in these regions
- ML can be used to perform segmentation!



Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

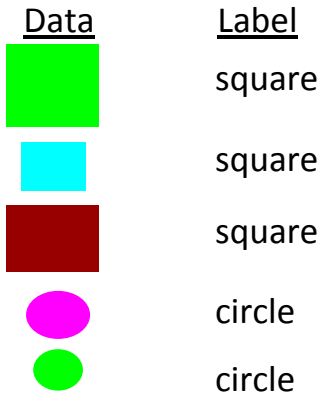
Machine Learning

- Finding patterns in data

Supervised vs Unsupervised ML

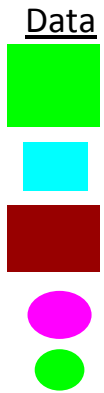
Supervised learning

- Learning from examples
- Requires data and “labels”
- Labels provide ground truth and serve to guide the learning process



Unsupervised learning

- Learning without examples
- Doesn’t require “labels”
- Generally more difficult
- Not the topic of today’s talk



Why use machine learning ?

- Find complex representations for data with limited prior knowledge
  - traditional statistical models make strong assumptions about distribution of data
  - difficult to create very complex representations with traditional models

- Many types of “representations”:
  - Support-vector machine (SVM)
  - Random-forest
  - Linear/logistic regression
  - **Artificial Neural Networks**

Components of a ML model

1. Model
   a. specified by user (still as much an art as a science)
2. Optimization method
   a. computer will try to find best fit given the model the user has specified
3. Loss function
   a. mathematically defines what we mean by “best”
4. Data
   a. Often need “big” data to capture variability found from population from which data is drawn

The loss function

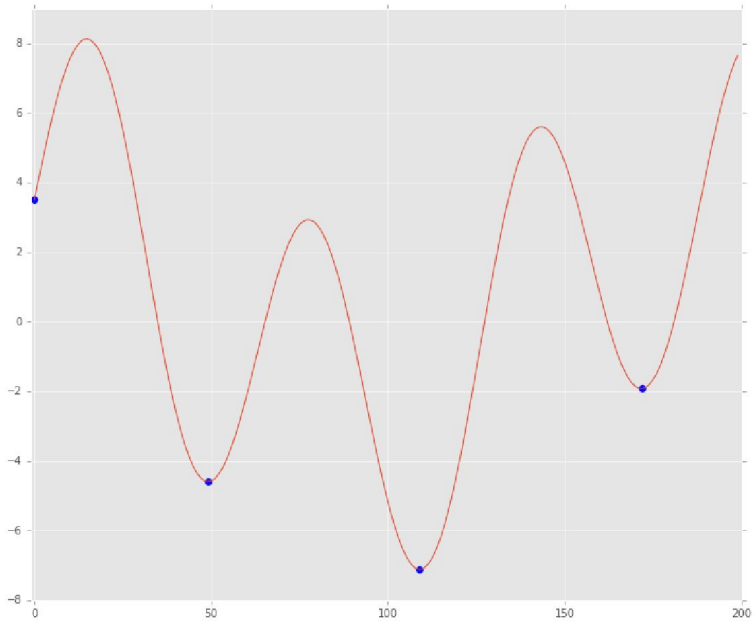
Best means we need to define a criterion (loss) that we will then minimize. It should take into account the discrepancy between the data and our predictions.

Part of the job consists in choosing the appropriate criterion for the task and the kind of data available.

Keywords: MSE, cross entropy, MLE, MAP

Global vs local minimum

- In Optimization, we are looking for **global minimums**.
- However in ML, we will sometime accept the best **local minimum** we can find in a given time.



Example of loss function

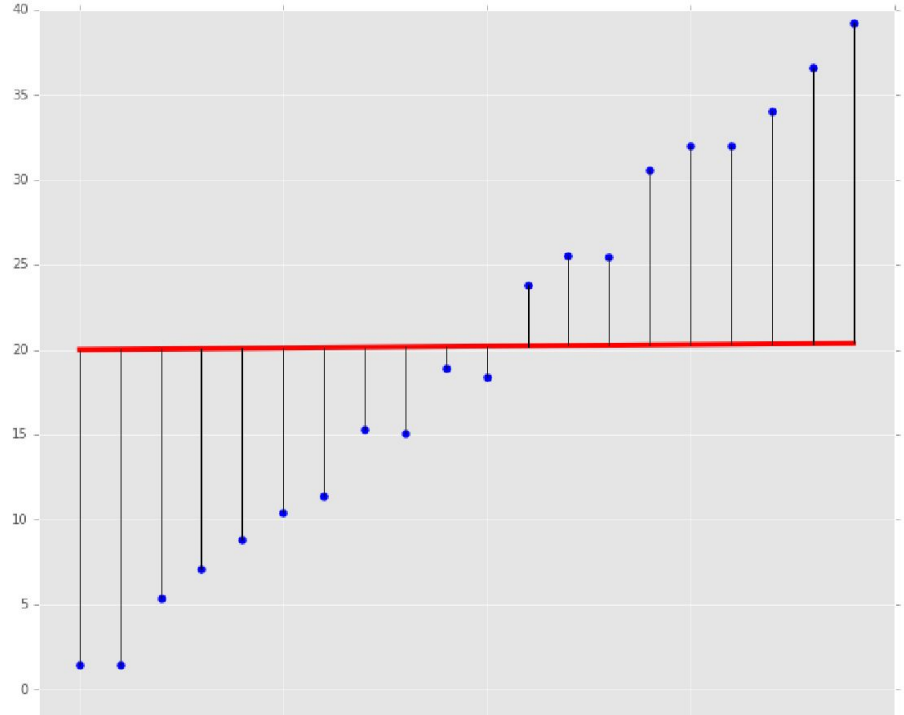
- Mean squared error is a simple loss function
  - squared-error between data (y_i) and predicted value ($\hat{y}_i$)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Example model

Model : simple linear regression

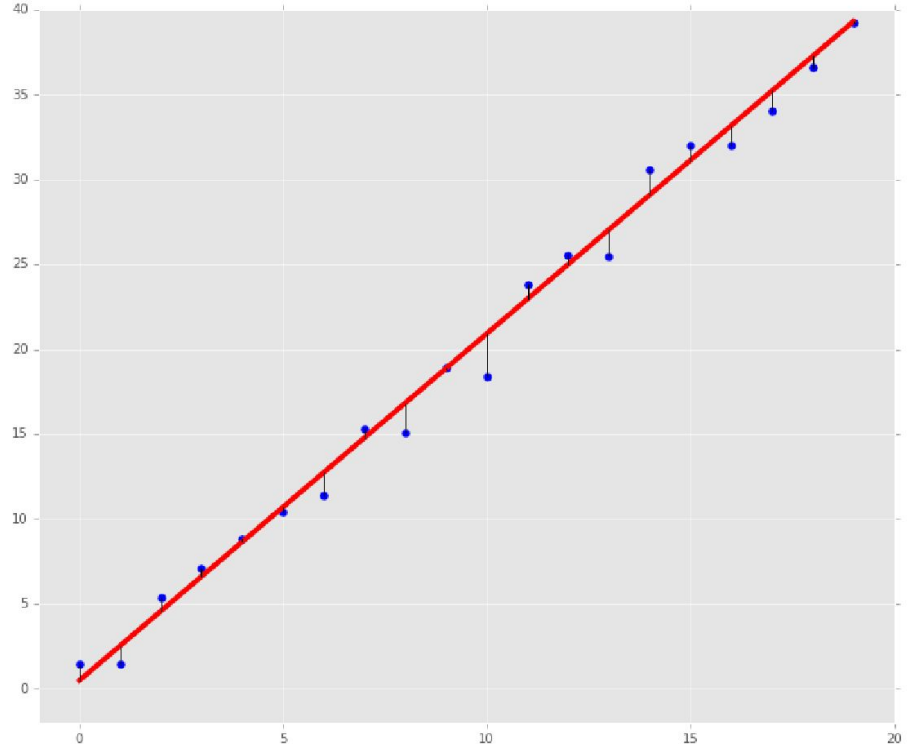
- First example: $MSE \approx 133$



Examples

- First example: $MSE \approx 133$
- Second example: $MSE \approx 1.2$

This second curve is the best linear model in the sense of the MSE.

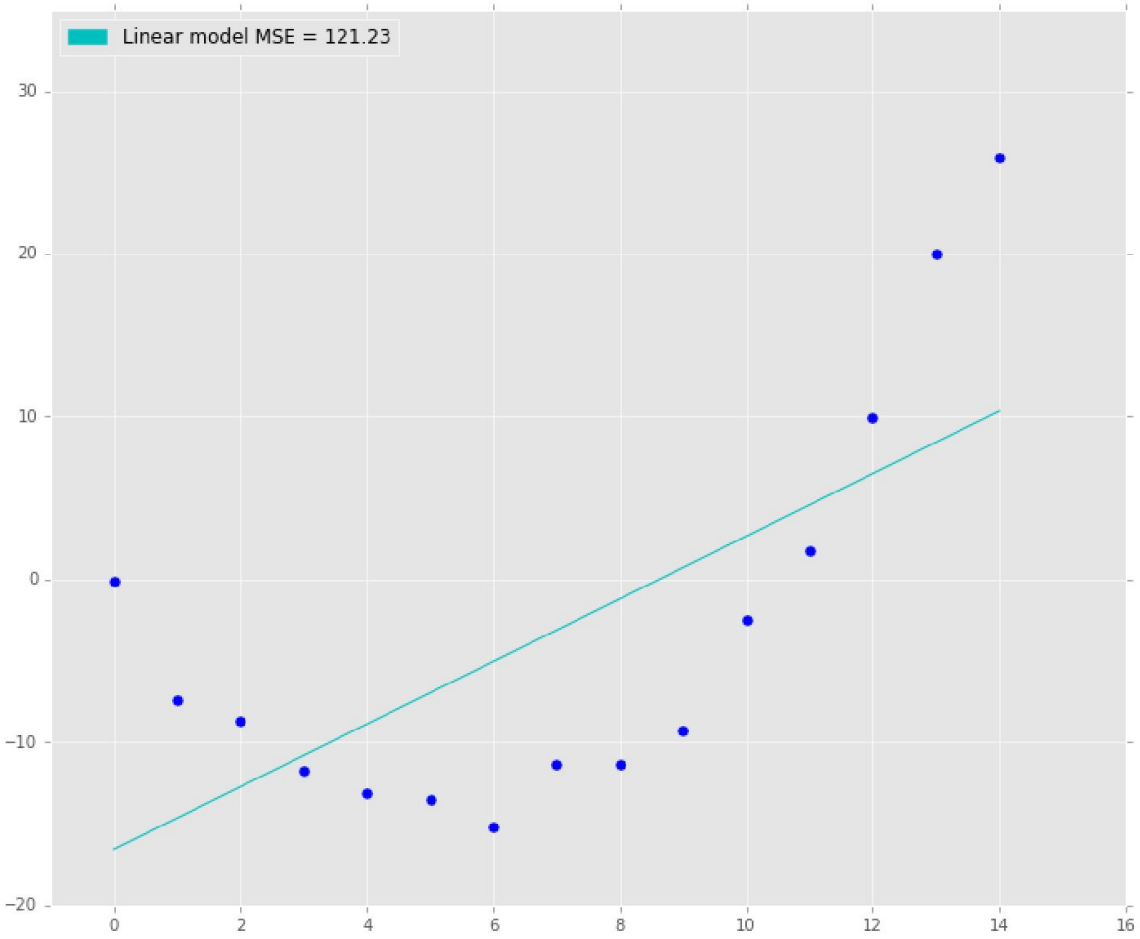


Model specification based on data

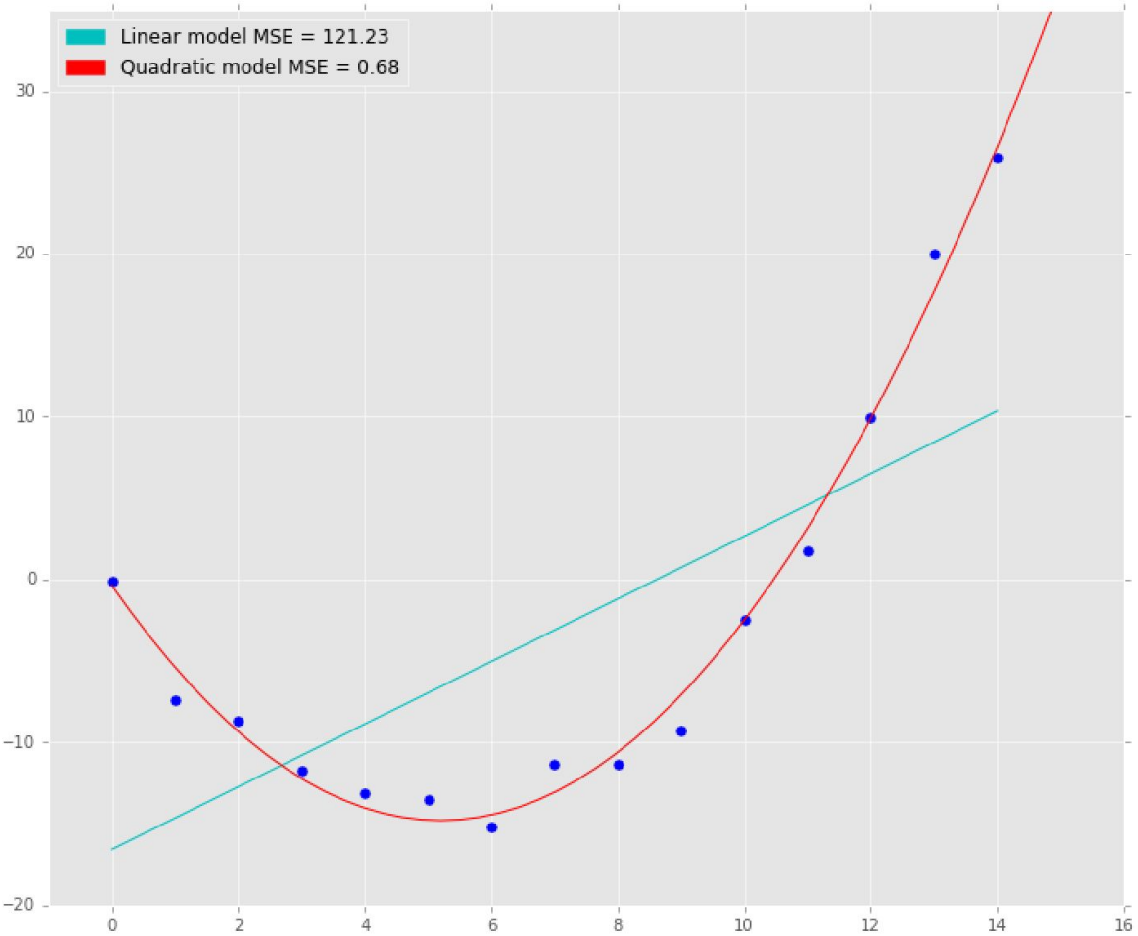
- Finding best model depends on your data

- It can be difficult to find the appropriate model for your data!

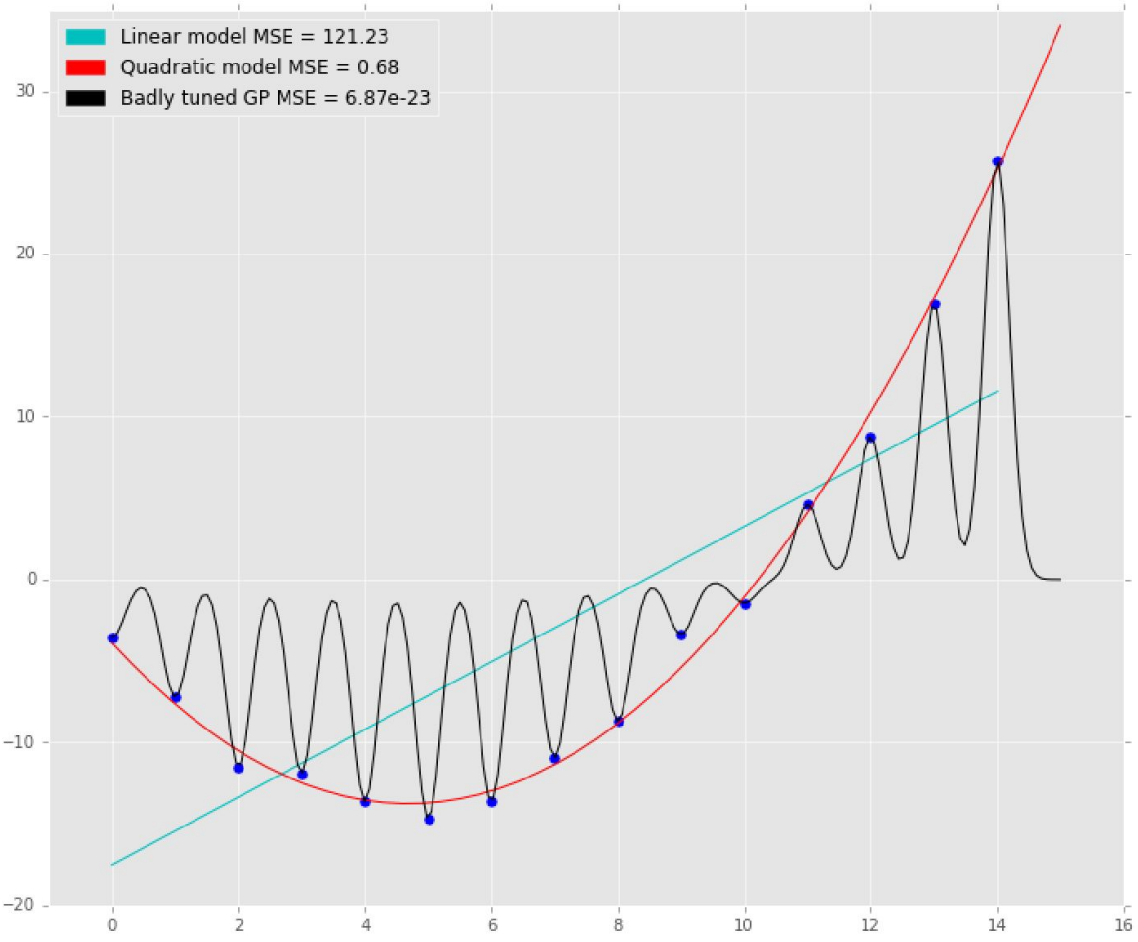
The curve II



The curve II


Linear model MSE = 121.23
Quadratic model MSE = 0.68
30
20
10
0
−10
−20
0
2
4
6
8
10
12
14
16

The curve II


Linear model MSE = 121.23
Quadratic model MSE = 0.68
Badly tuned GP MSE = 6.87e-23
30
20
10
0
−10
−20
0
2
4
6
8
10
12
14
16

Generalization

- Highly accurate model may not ***generalize*** to new data points

- Generalization is often more important than accuracy

- Split data into training set and validation set
  - ***training set*** → used to fit model
  - ***validation set*** → used to test accuracy of model on new data

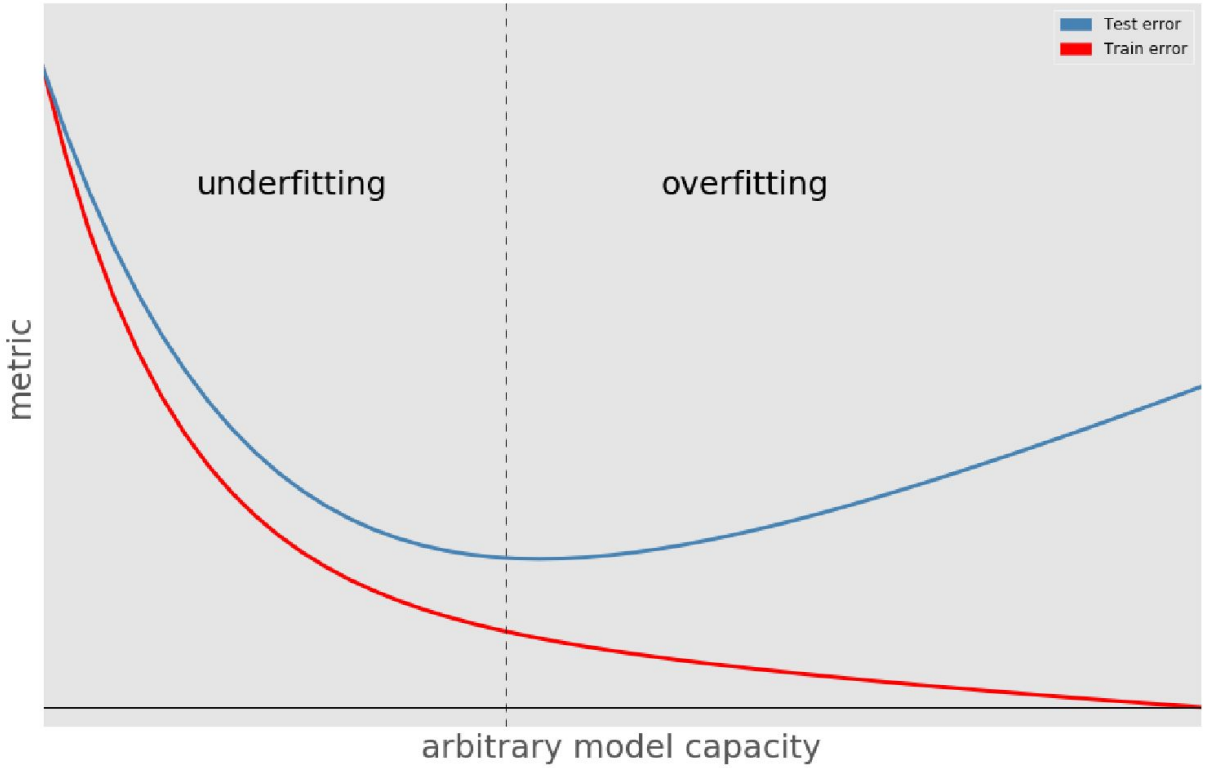
Model Capacity

- Informally, ***capacity***: complexity of the representations used to fit the data
  - Very loosely measured by the number of parameters in a model
  - More parameters → more complex representation

- A model with a capacity too high will generalize badly (overfitting).

- A model with a capacity too low will fit the train badly (underfitting).

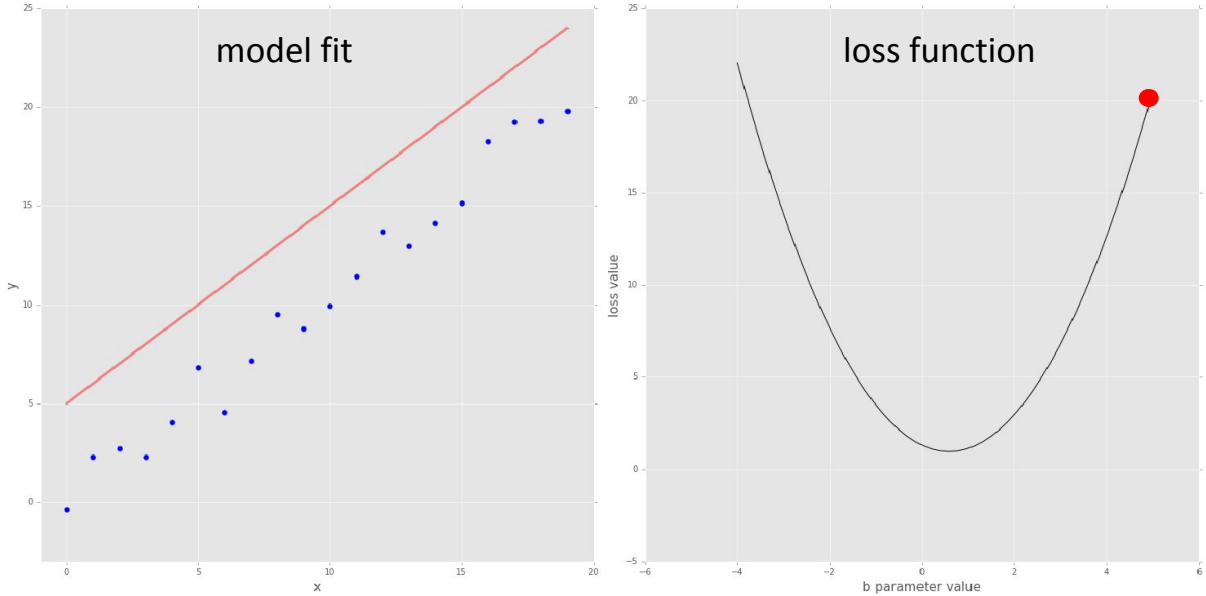
Capacity, overfitting and underfitting II



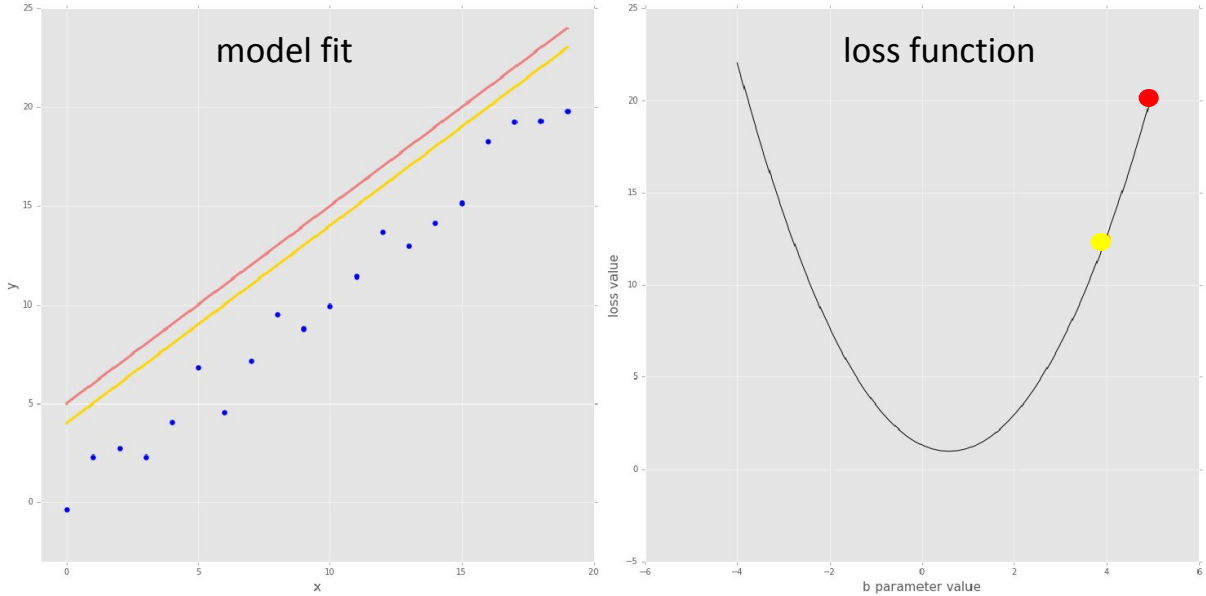
Optimization

- Find a way to minimize the loss function to find optimal model fit
  - Analytic solution : closed-form mathematical solution to minimize loss function
  - $$y_i = ax_i + b + \epsilon_i$$
    $$\hat{a} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$
    $$\hat{b} = \bar{\mathbf{y}} - \hat{a}\bar{\mathbf{X}}$$
  - Iterative solution : find incrementally better solutions to minimize loss functions
- Gradient descent
  - Iterative solution
  - Imagine loss function as a bowl
    - bottom of the bowl represents best possible solution to fit the model to the data
    - we start the optimization process at some point on bowl
    - find bottom by iteratively going in the steepest direction of the bowl

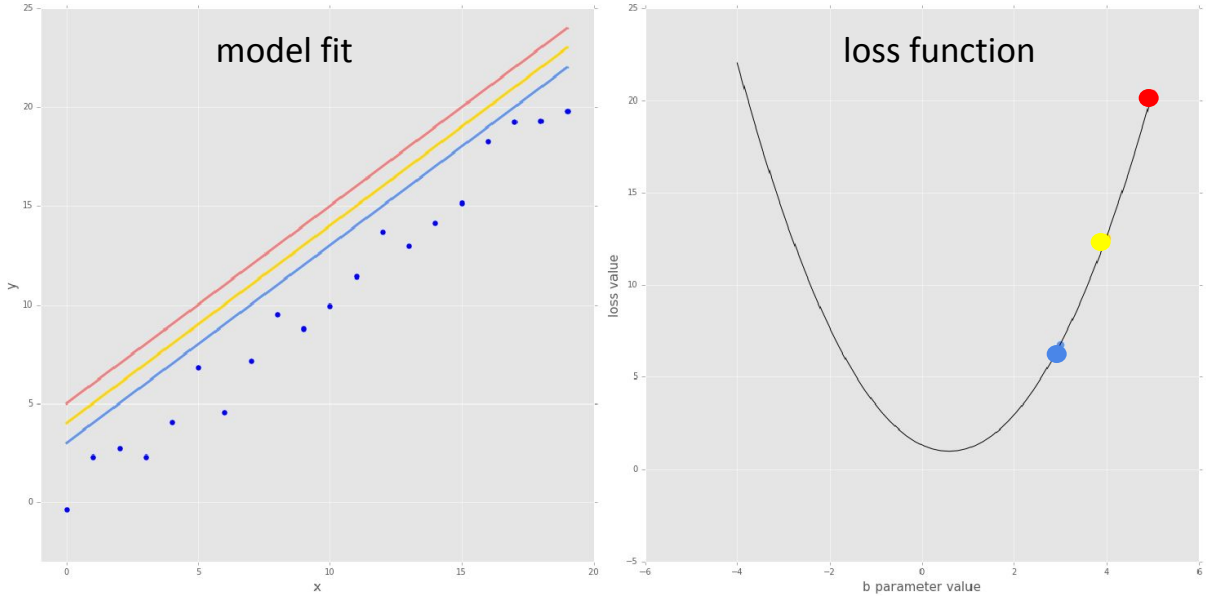
Iterative gradient descent, 1 dimensional



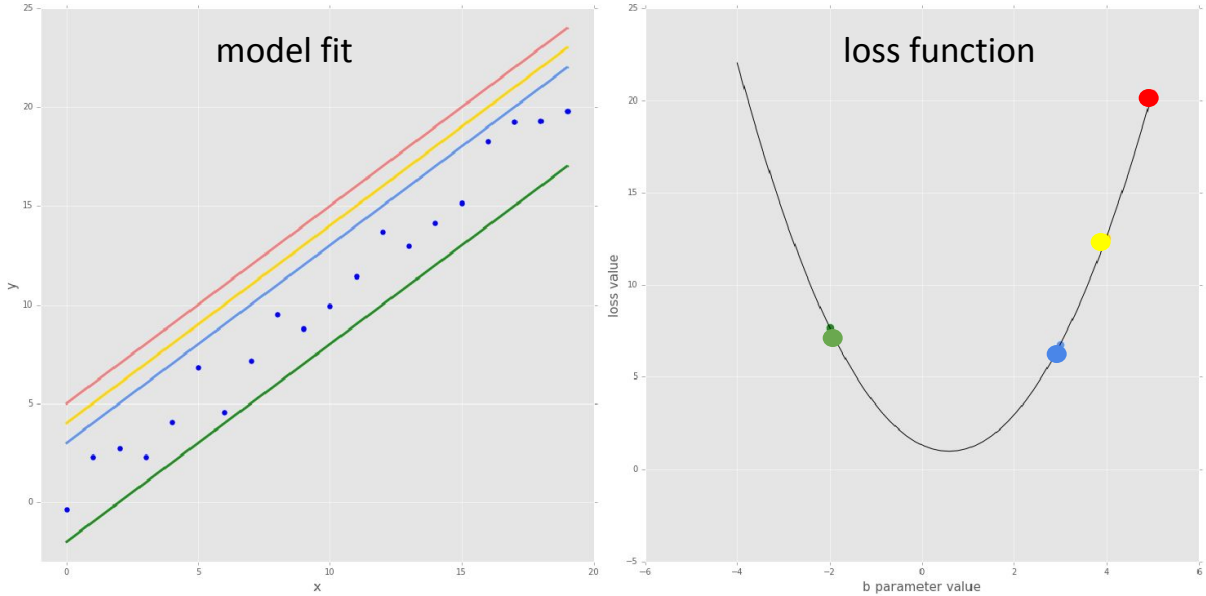
Iterative gradient descent, 1 dimensional



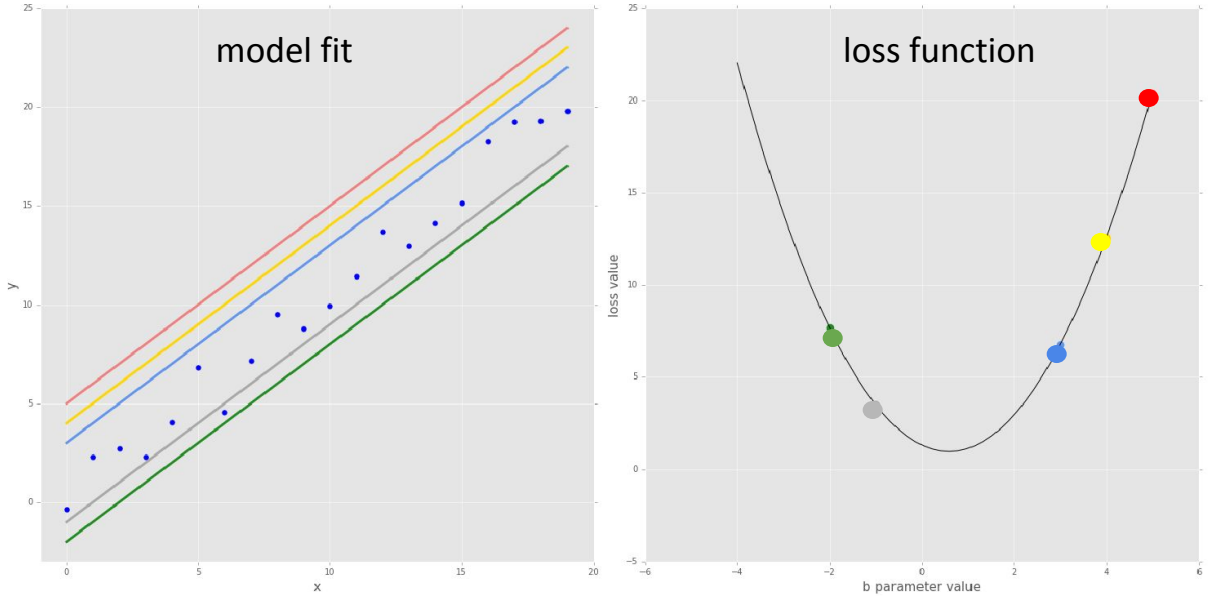
Iterative gradient descent, 1 dimensional



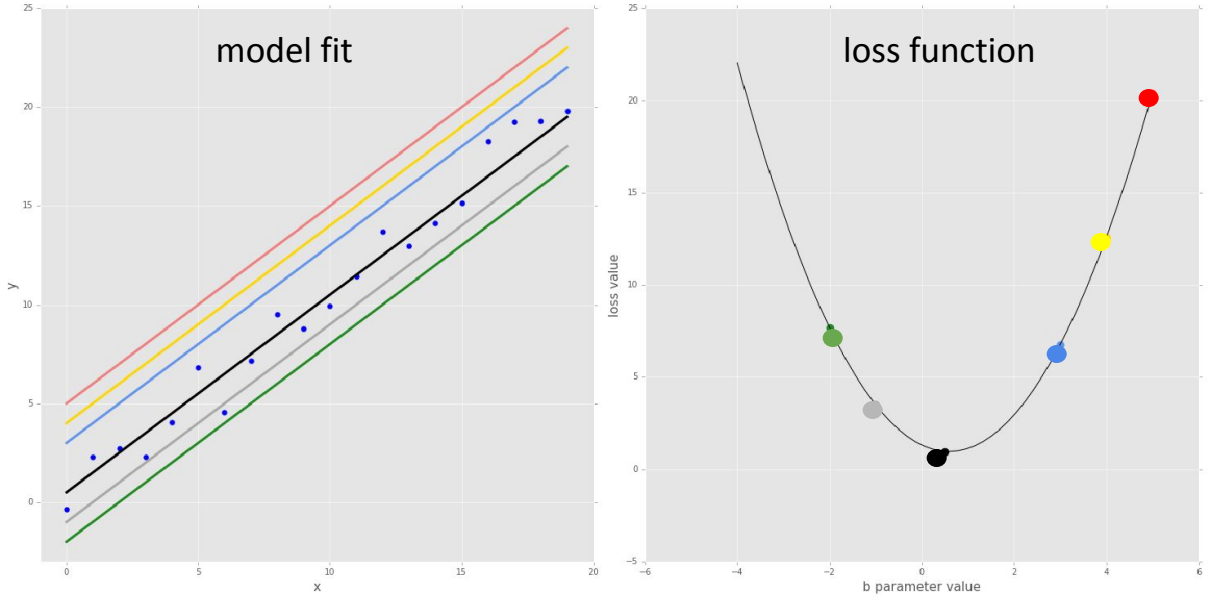
Iterative gradient descent, 1 dimensional



Iterative gradient descent, 1 dimensional



Iterative gradient descent, 1 dimensional



Two important considerations

- Interpretability
  - ML models can produce amazing results...
  - But it can be difficult to understand how they got to this result
  - This is especially a problem for neural networks
  - Ongoing topic of research in AI and ML
- Bias
  - If you’re data is biased, then your model will be biased!
  - Be very careful that your training data does not contain implicit biases
  - Garbage in => garbage out

Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

Introduction to Deep Learning

- Layers of artificial neurons learn increasingly abstract representations of your data
- Can capture very complex interactions between features using non-linear activations

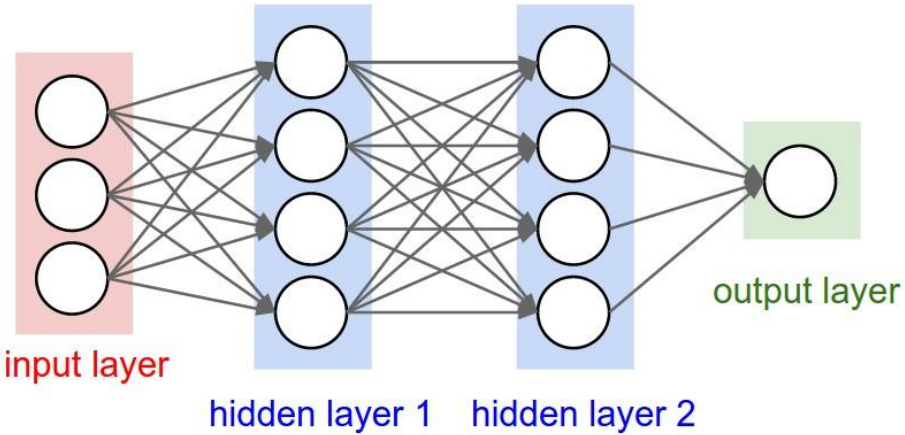


Image: http://cs231n.github.io/convolutional-networks/

Components of neural networks

- Each layer has a set off artificial neurons/nodes
- Traditionally activated with rectified linear units (i.e. ReLU)
- Regularization done through dropout

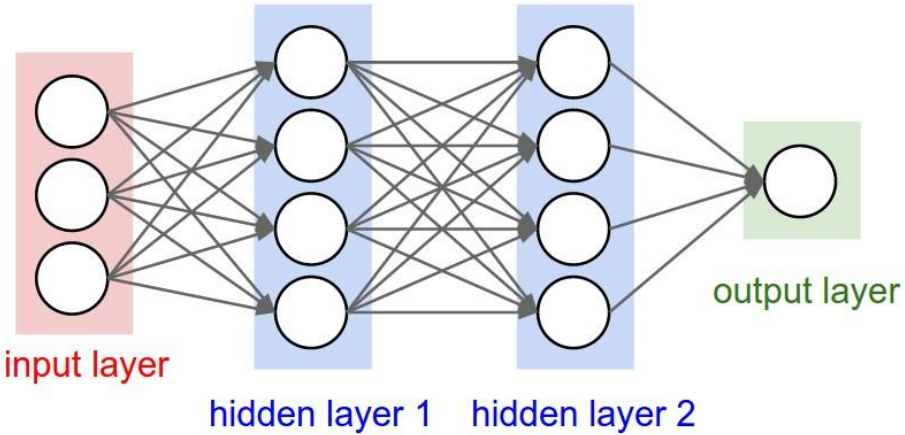
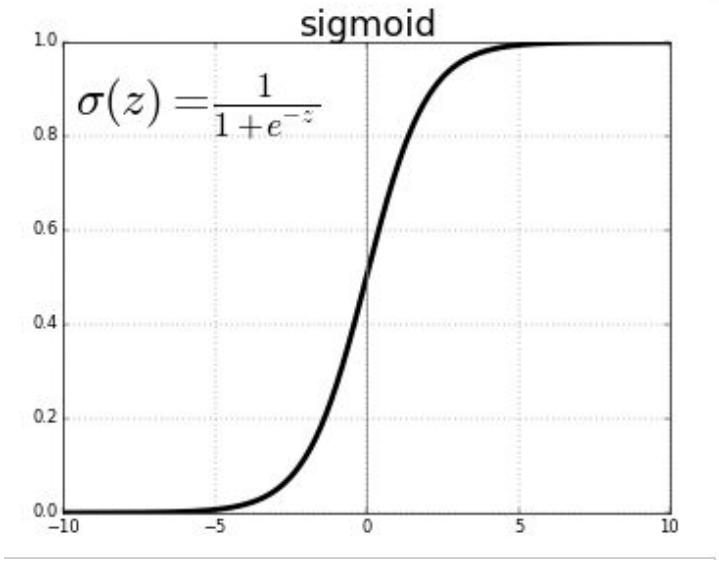
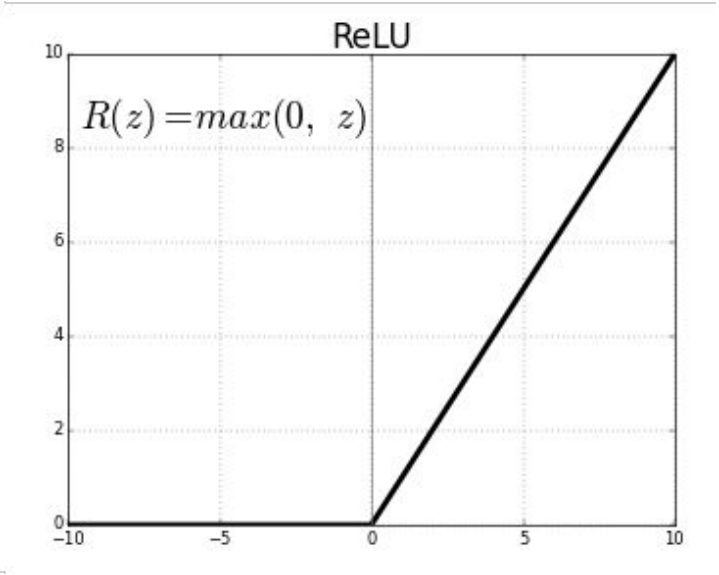


Image: http://cs231n.github.io/convolutional-networks/

Components of neural networks

- Each layer has a set off artificial neurons/nodes
- Traditionally activated with rectified linear units (i.e. ReLU)
- Regularization done through dropout

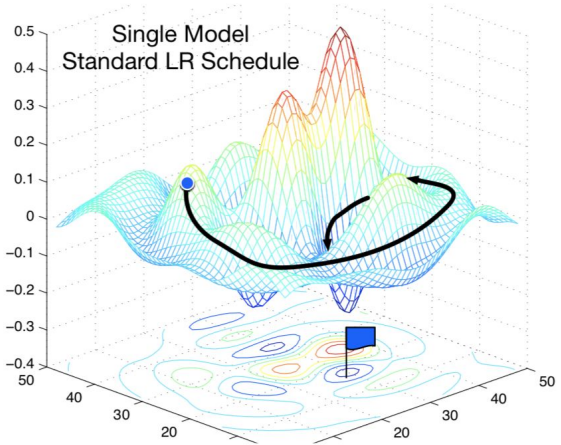




https://cdn-images-1.medium.com/max/1600/1*XxxiA0jJvPrHEJHD4z893g.png

Training neural networks

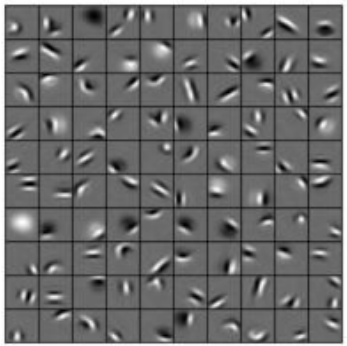
- Stochastic gradient descent
  - Pick a training sample
  - Feed it through the network
  - Determine the error of said prediction
  - Compute the share of correction for all hidden nodes and update the weights
- Many algorithms for updating the weights such as:
  - Gradient Descent
  - RMSProp
  - Adam

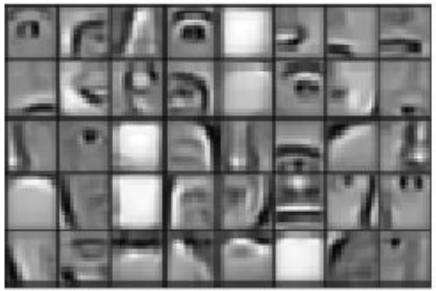


https://cdn-images-1.medium.com/max/2000/1*T5WWecP_EaQWk1yDX15h_w.pn

Convolutional neural networks

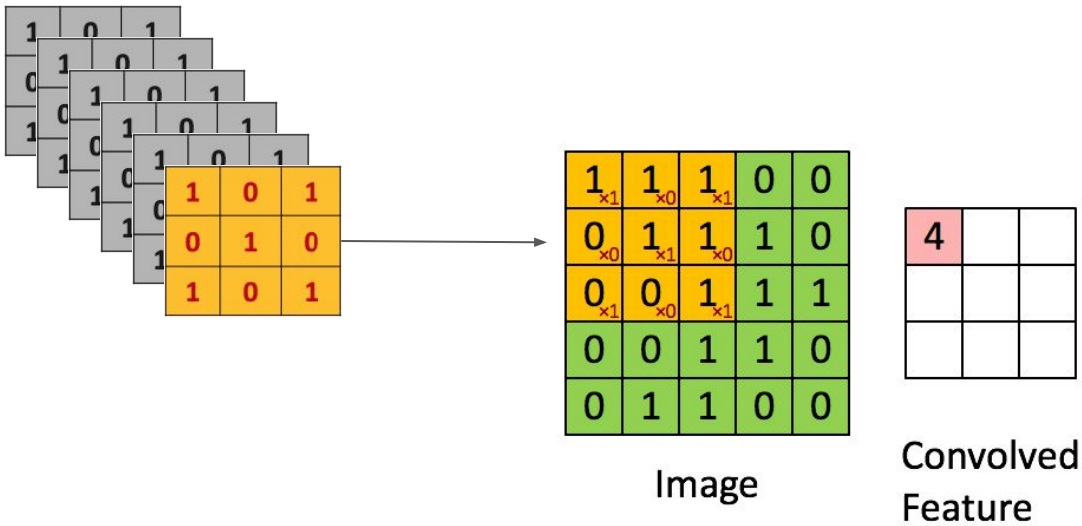
- Typically used in imaging data
- Many feature filters scan over the image to pick out specific features (e.g. edges, curves, eyes, nose, face)
- These filters learn to recognize features that make the performance of the prediction increase





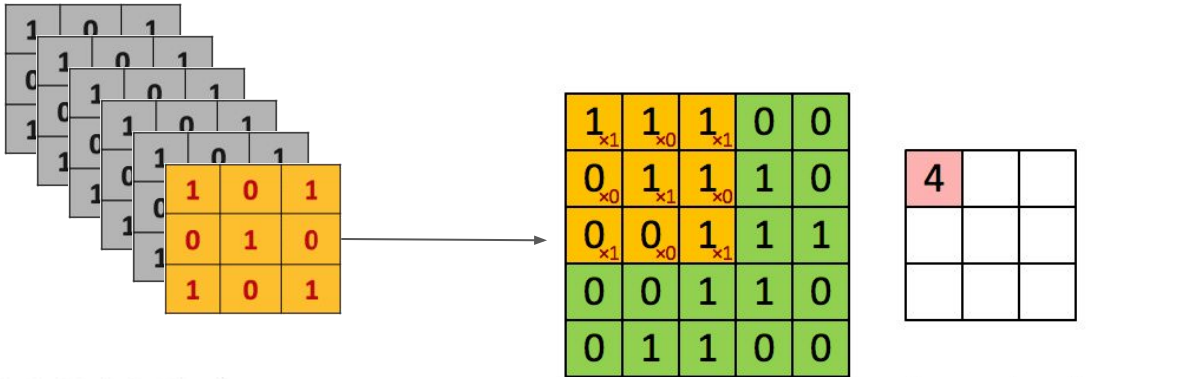
https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concep

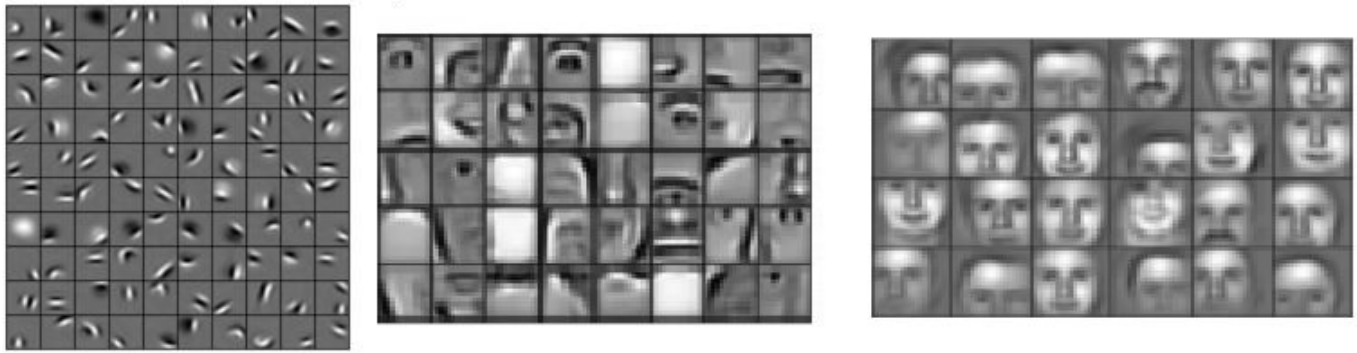
Learning features through kernels



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

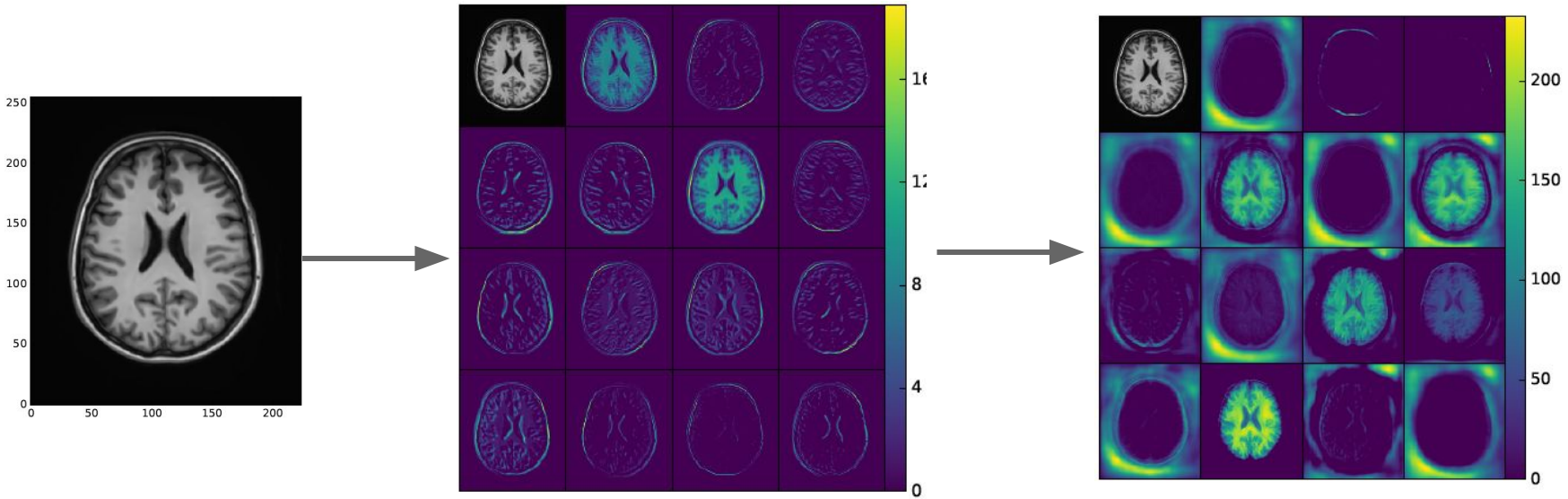
Learning features through kernels





http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Learning brain segmentation



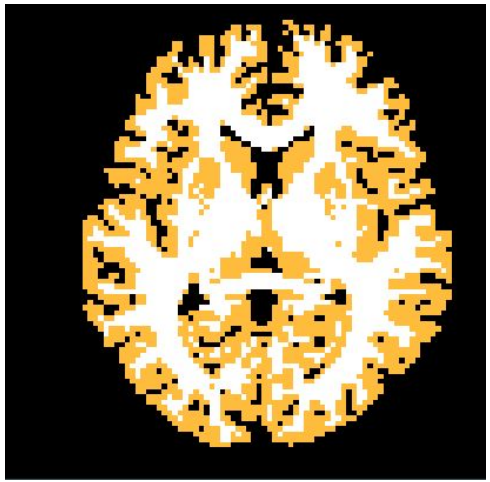
Fratila, R., Fonov, V., Collins, L.D., Arnold, D.L., Brown, R. (2017). DeepDiscovery: Rapid and Efficient Deep Learning for NeuroImaging. NeuroImage (submitted).

Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

1000 Functional Connectomes Project

- 261 MRI (skull stripped)
  - http://fcon_1000.projects.nitrc.org/fcpClassic/FcpTable.html
- GM/WM segmentation with FSL-5.0-fast



Build a Conv Net with Keras

```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,  kernel_size=[3,3],activation='relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(1, kernel_size=1,  padding='same', activation='sigmoid')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

Setup & Fit Model

make_and_run_model.py

```
def compile_and_run(X_train, Y_train, X_validate, Y_validate, epochs):
     #setup optimizer
     ada = keras.optimizers.Adam(0.0001)

     #setup define loss function
     loss function = 'binary_crossentropy'

     #compile the model
     model.compile(loss = loss_function, optimizer=ada,metrics=[dice_metric] )

     #fit model
     model.fit([X_train],Y_train, batch_size, validation_data=([X_validate], Y_validate),
epochs = epochs])

     #save model
     model.save(model_name)
```

Base Model

models/neurotech_models.py

```
def base_model(image_dim, nK, kernel_size, drop_out):
     # nK = number of kernels per layer
     # kernel_size = size of the kernels
     # dropout = dropout rate for each layer

     #Setup the input (IN) and output (OUT) layers based on image dimensions
     IN = OUT = Input(shape=(image_dim[1], image_dim[2],1))

     n_layers=int(len(nK)) # number of layer
     kDim=[kernel_size] * n_layers #list of kernel size equal in length to n_layers

     for i in range(n_layers): # for each layer...
          OUT=Conv2D(nK[i],kernel_size=[kDim[i],kDim[i]],activation='relu',padding='same')(OUT)
          OUT = Dropout(drop_out)(OUT)

     OUT = Conv2D(1, kernel_size=1,  padding='same', activation='sigmoid')(OUT)
     model = keras.models.Model(inputs=[IN], outputs=OUT)
     return(model)
```

model_0_0

models/neurotech_models.py

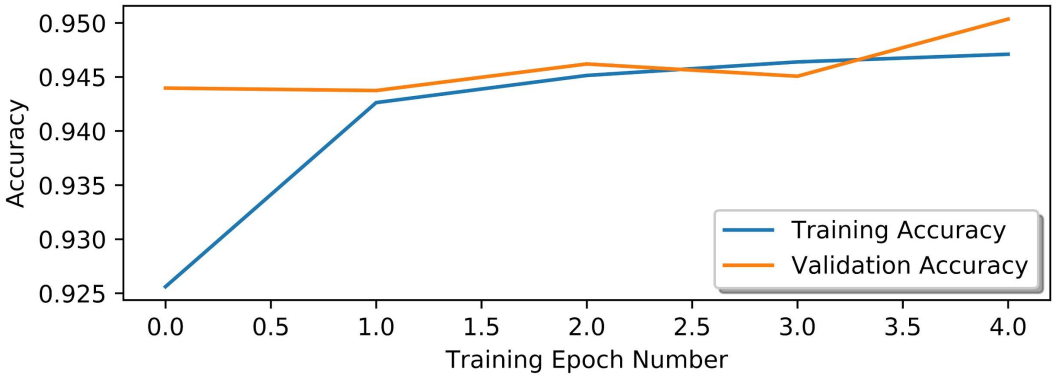
```
def model_0_0(image_dim):
      # Create a convolutional network with
      # [3,3] x 16
      # [3,3] x 16
      # [3,3] x 32
      # [3,3] x 32
      # [3,3] x 64
      # [3,3] x 64
      nK=[16,16,32,32,64,64]
      kernel_size = 3
      drop_out=0
      return base_model( image_dim, nK, kernel_size, drop_out)
```

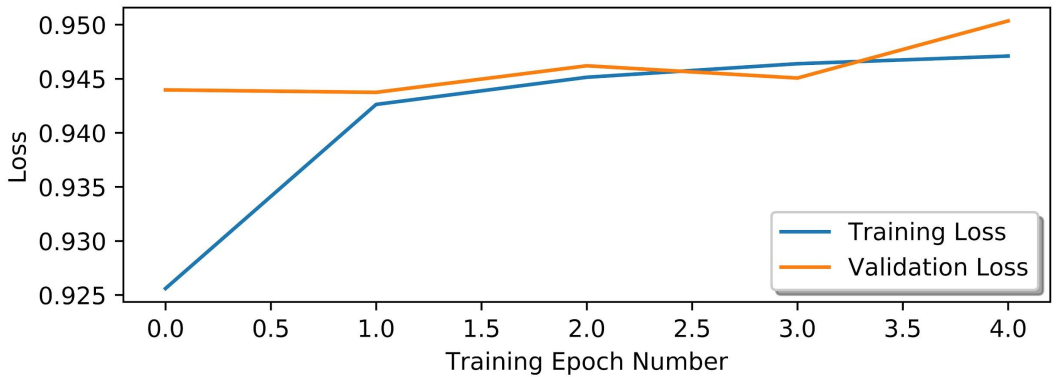
How to run a model with *minc_keras.py*

```
python3 minc_keras/minc_keras.py
          --source output/          # source dir where input data is stored
          --target .                # output dir where results will be placed
          --epochs 5                # number of epochs to use to fit model
          --model-type "model_0_0”  # name of model you want to use
          --input-str "*T1w_anat*"  # string that identifies the input MRI images
          --label-str "*seg*"       # string that identifies the labeled GM/WM images
          --predict 1               # comma-separated list of subjects for prediction
          --ratios 0.1 0.1          # ratio to use in training/validation/test splits
```

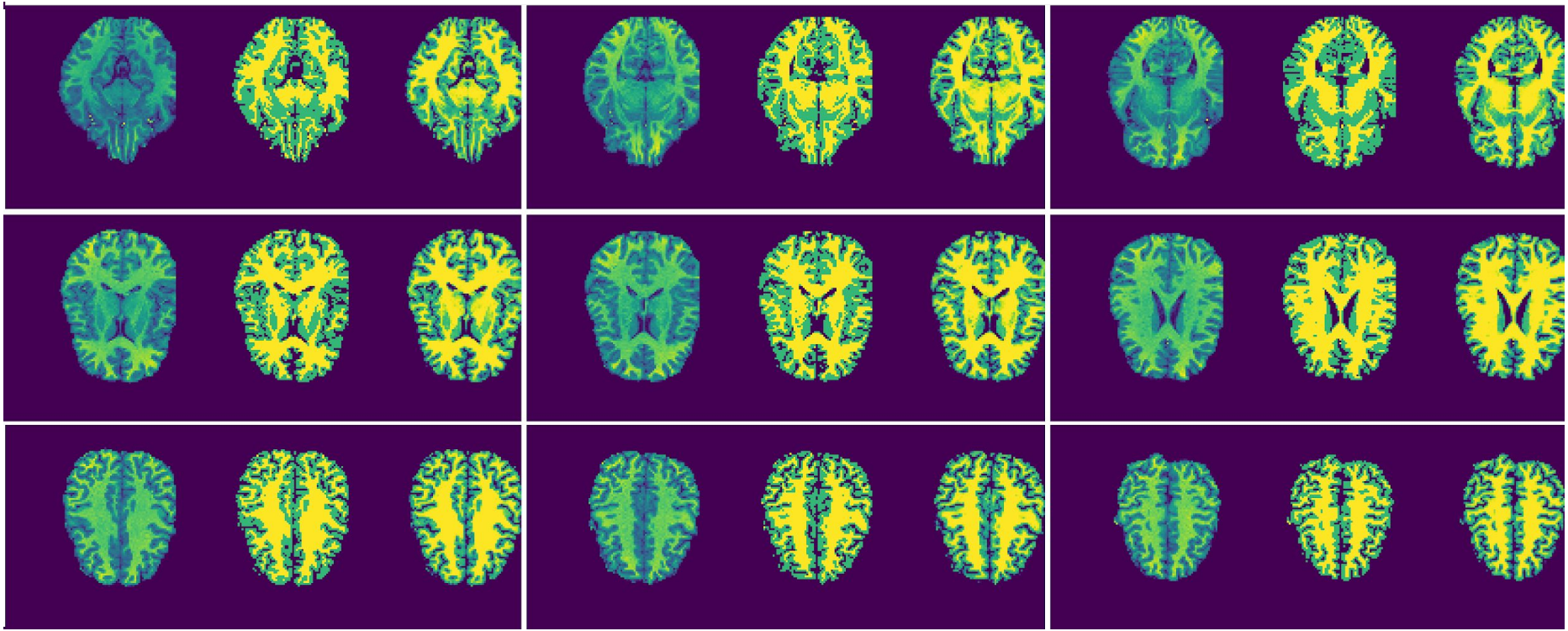
Results for model_0_0

- Parameters =





Results for model_0_0



model_1_0

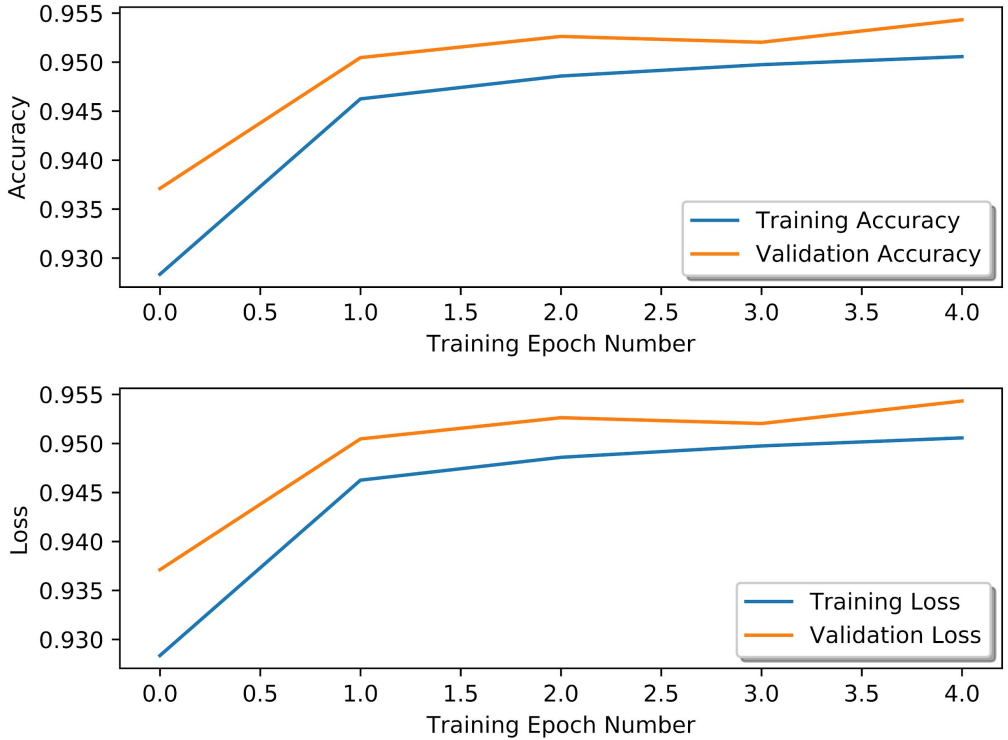
models/neurotech_models.py

```
def model_1_0(image_dim):
      # Create a convolutional network with
      # [3,3] x 16
      # [3,3] x 16
      # [3,3] x 16
      # [3,3] x 32
      # [3,3] x 32
      # [3,3] x 64
      # [3,3] x 64
      # [3,3] x 64
      nK=[16,16,16,32,32,64,64,64]
      kernel_size = 3
      drop_out=0
      return base_model( image_dim, nK, kernel_size, drop_out)
```

```
python3 minc_keras/minc_keras.py --source output/ --target . --epochs 5 --model-type "model_1_0"
--input-str "*T1w_anat*" --label-str "*seg*" --predict 1 --ratios .1 .1
```

Results for model_1_0

- Parameters =
- Slightly higher accuracy
- Validation accuracy still increasing after 5 epochs



model_2_0

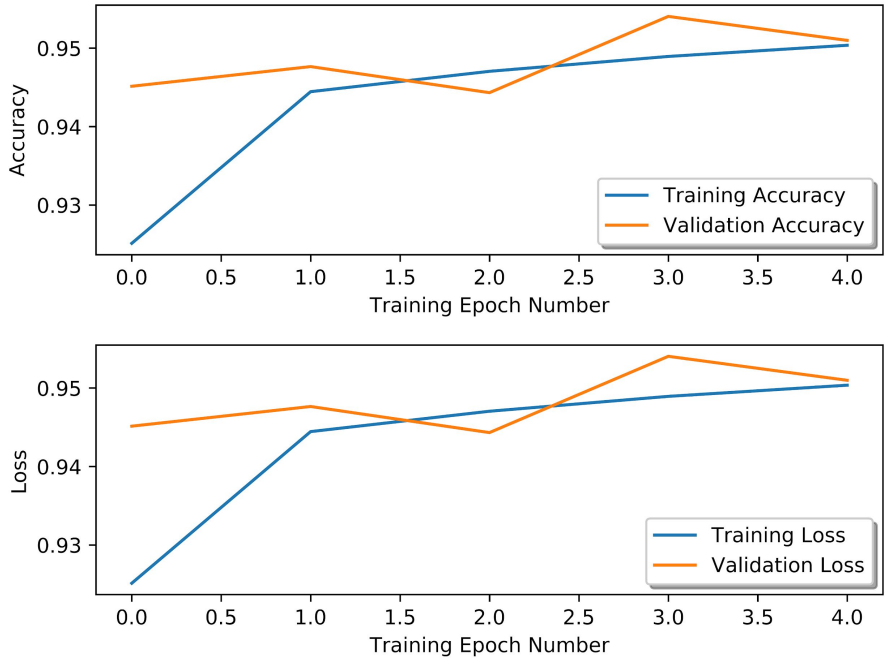
models/neurotech_models.py

```
def model_2_0(image_dim):
      # Create a convolutional network with
      # [5,5] x 16
      # [5,5] x 16
      # [5,5] x 32
      # [5,5] x 32
      # [5,5] x 64
      # [5,5] x 64
      nK=[16,16,32,32,64,64]
      kernel_size = 5
      drop_out=0
      return base_model( image_dim, nK, kernel_size, drop_out)
```

```
python3 minc_keras/minc_keras.py --source output/ --target . --epochs 5 --model-type model_2_0”
--input-str "*T1w_anat*” --label-str "*seg*" --predict 1 --ratios .1 .1
```

Results for model_2_0

- Parameters =
- Slightly higher accuracy
- Validation accuracy still increasing after 5 epochs



model_4_0

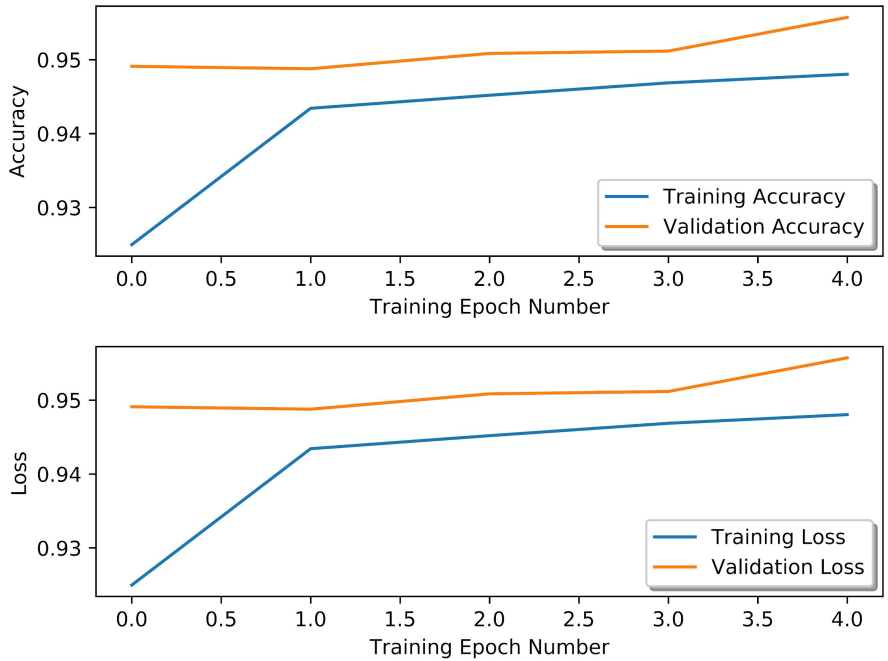
models/neurotech_models.py

```
def model_1_0(image_dim):
      # Create a convolutional network with
      # [5,5] x 16
      # [5,5] x 16
      # [5,5] x 32
      # [5,5] x 32
      # [5,5] x 64
      # [5,5] x 64
      nK=[16,16,32,32,64,64]
      kernel_size = 5
      drop_out=0.25
      return base_model( image_dim, nK, kernel_size, drop_out)
```

```
python3 minc_keras/minc_keras.py --source output/ --target . --epochs 5 --model-type model_4_0”
--input-str "*T1w_anat*” --label-str "*seg*" --predict 1 --ratios .1 .1
```

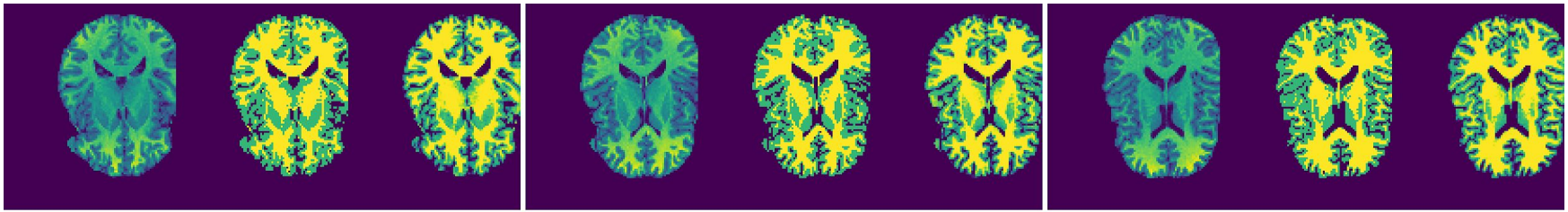
Results for model_4_0

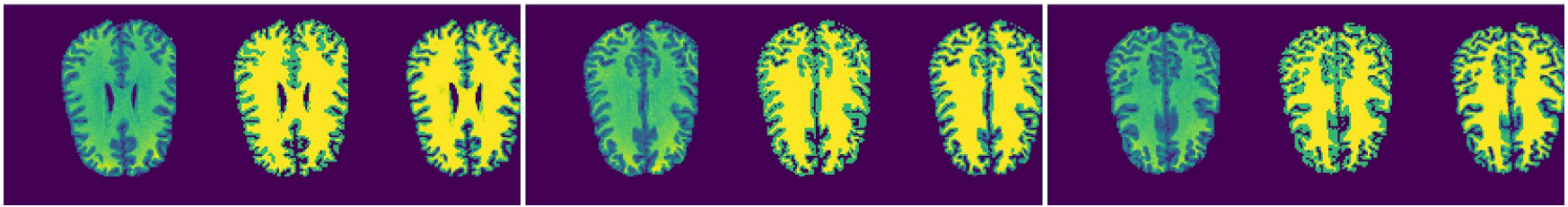
- Parameters =
- Slightly higher validation accuracy than model_0_0
- Better ratio of training to validation accuracy



Results for model_4_0

- Parameters =
- Slightly higher validation accuracy than model_0_0
- Better ratio of training to validation accuracy





Free coding

1. Can you find a model that does better than 95% without overfitting?

2. You can change the training/validation/test ratio to increase samples for training
   a. e.g., --train 0.3 0.3 (30% training, 30% validation, 60% split)
   b. a large training set will slow down the training, but reduce overfitting

3. Ask volunteers and organizers for help!

Acknowledgements