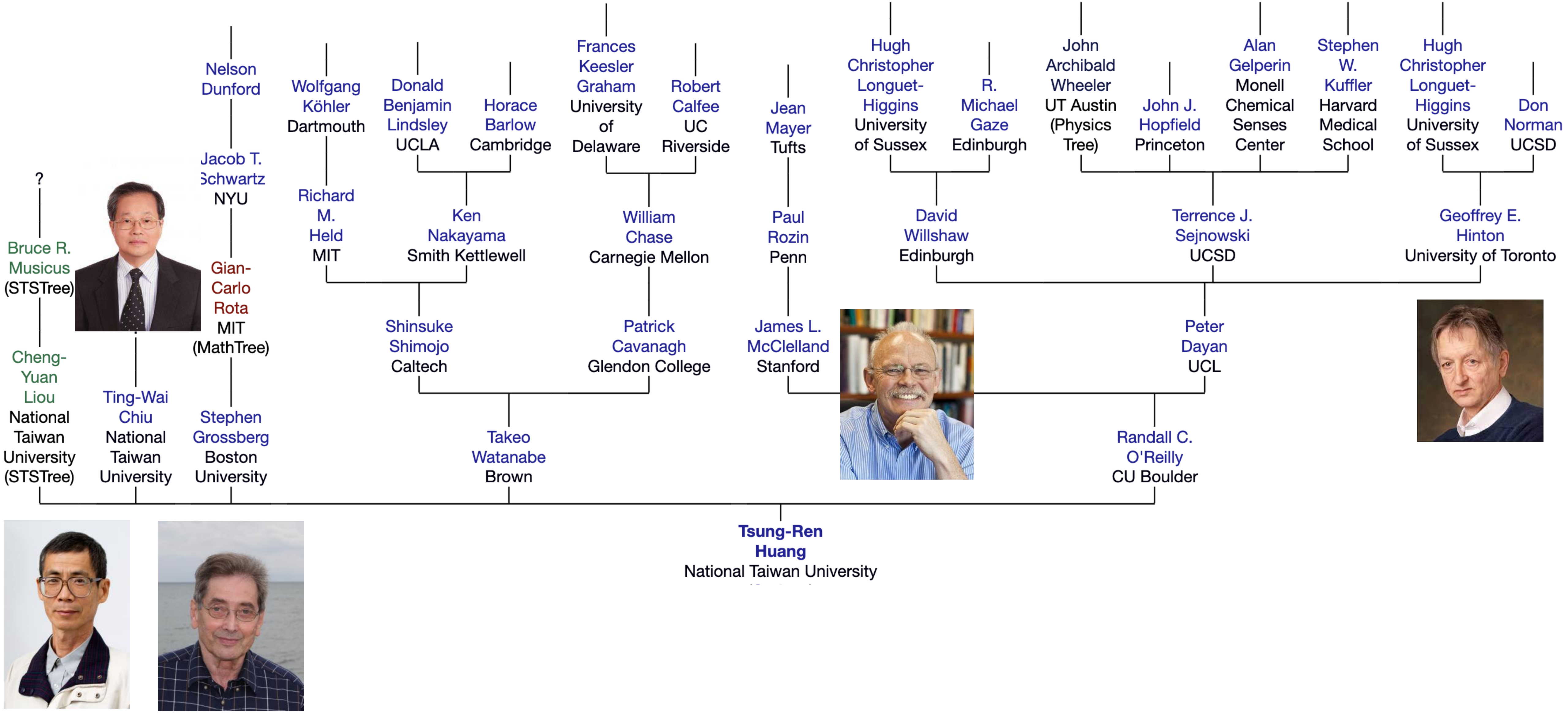
Psychoinformatics & Neuroinformatics

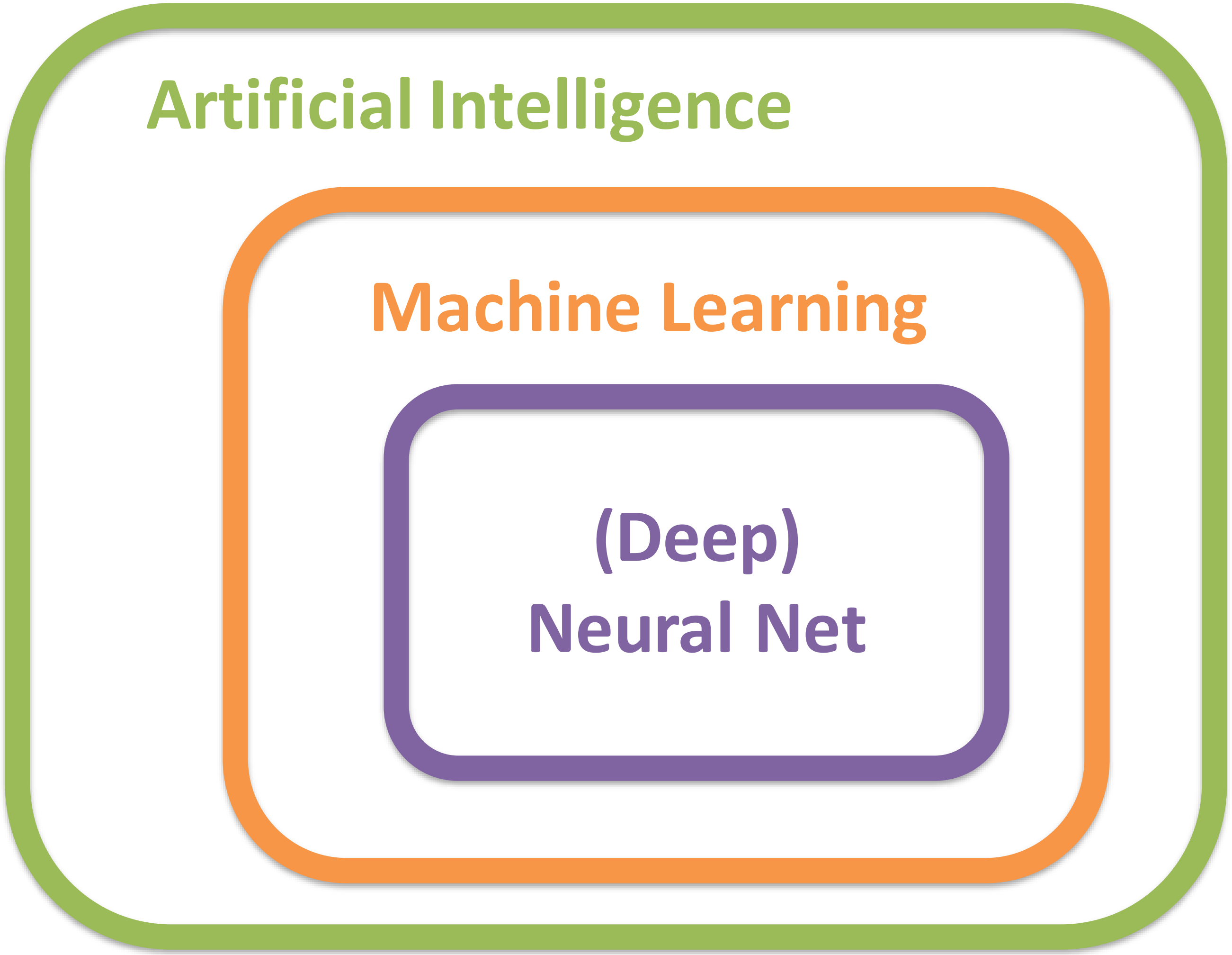
Week 11

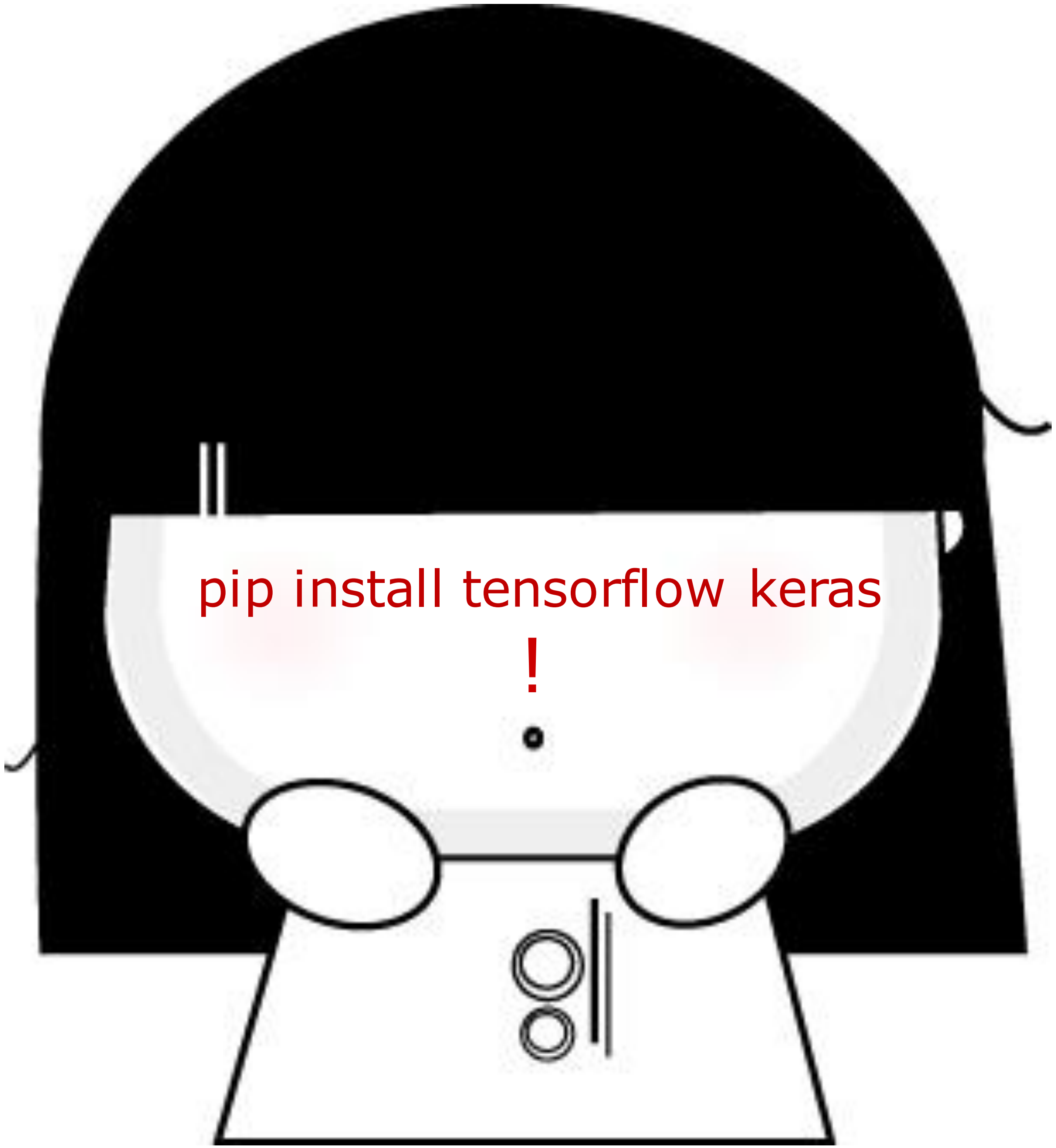
Machine Learning (3/3)

by Tsung-Ren (Tren) Huang 黃從仁

Tren's Academic Tree




Artificial Intelligence
Machine Learning
(Deep)
Neural Net

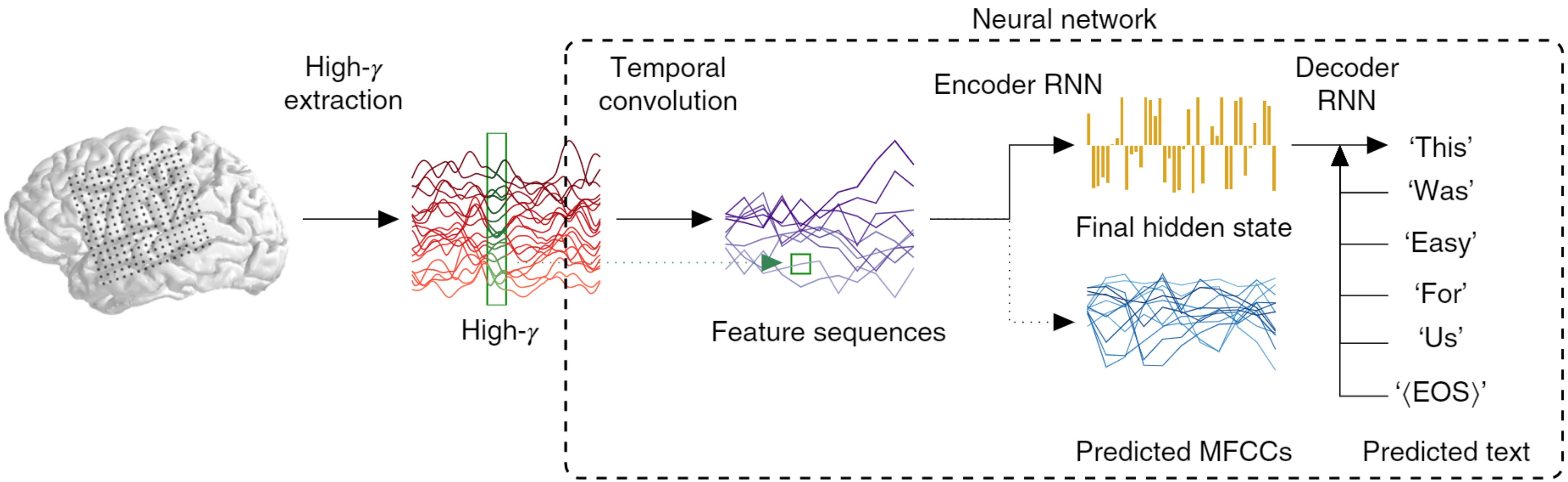

pip install tensorflow keras
!

Application of Deep Neural Nets (1/2)

Brain decoding using recurrent neural net

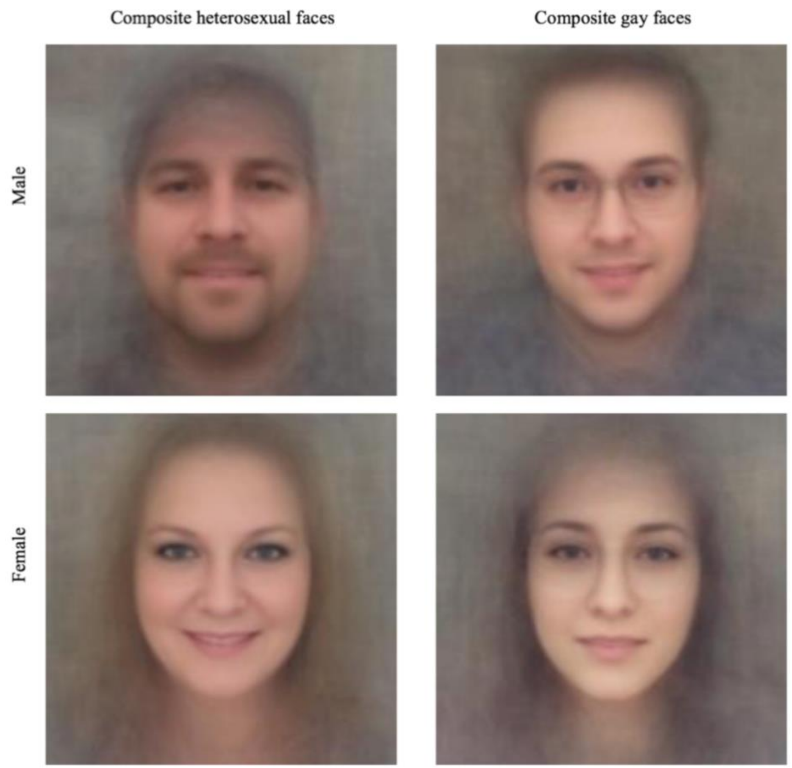
Machine translation of cortical activity to text with an encoder–decoder framework

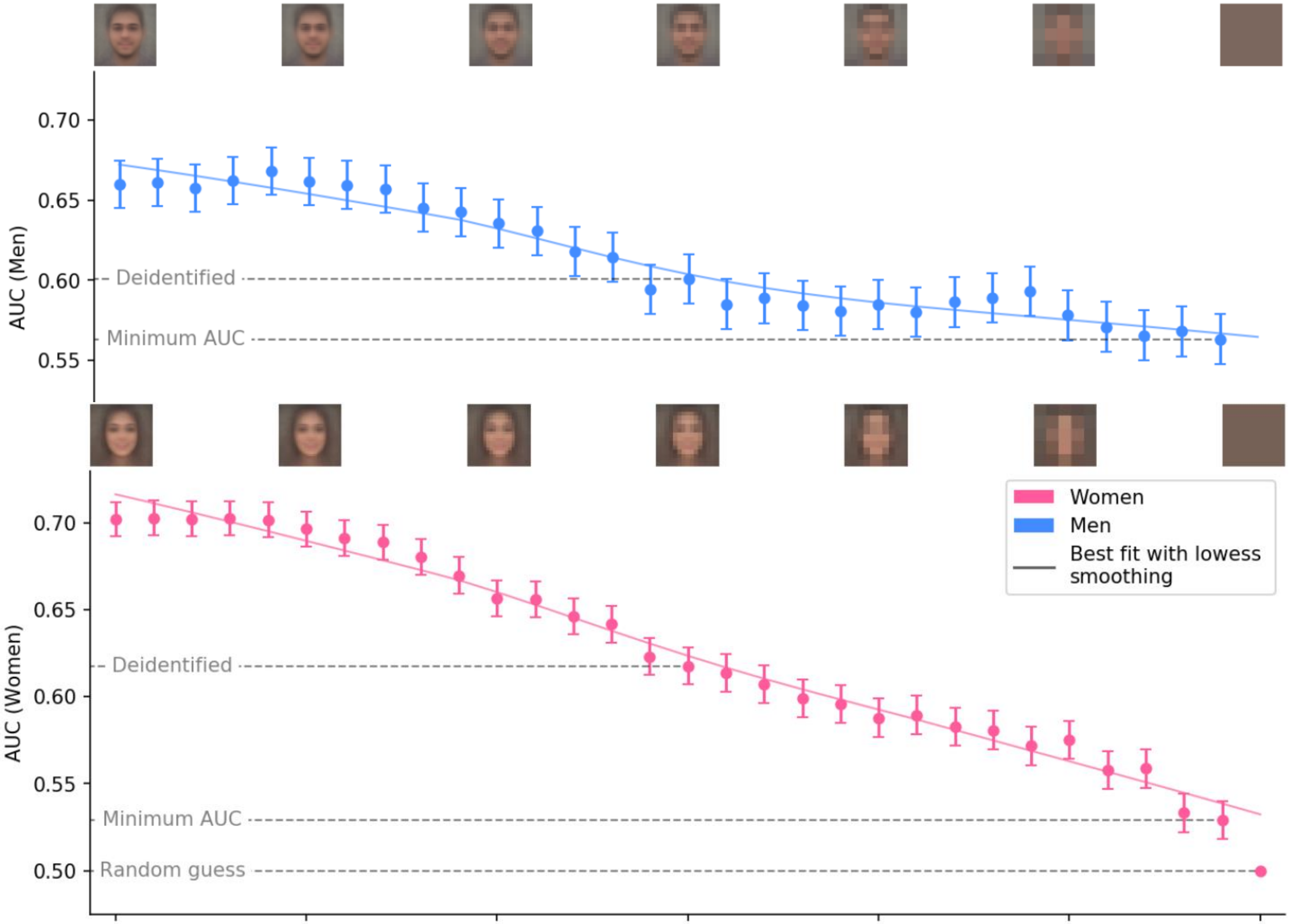
Joseph G. Makin[1,2], David A. Moses[1,2] and Edward F. Chang[1,2]



Application of Deep Neural Nets (2/2)

Predicting sexuality using convolutional neural net





Topics for today

Computations of Neural Networks

Why do neural nets work?

(Deep) Learning in Neural Networks

How do neural nets learn?

Explainable AI

What do neural nets learn?

Topics for today

Computations of Neural Networks

Why do neural nets work?

(Deep) Learning in Neural Networks

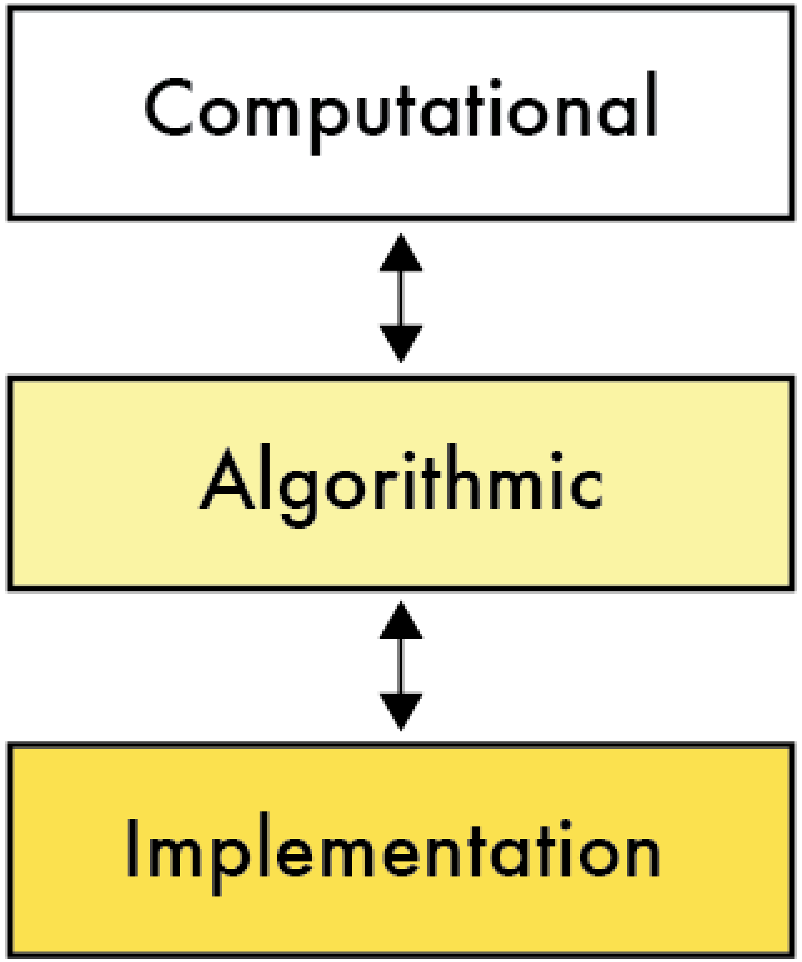
How do neural nets learn?

Explainable AI & Causal ML

What do neural nets learn?

Analysis of a Cognitive System

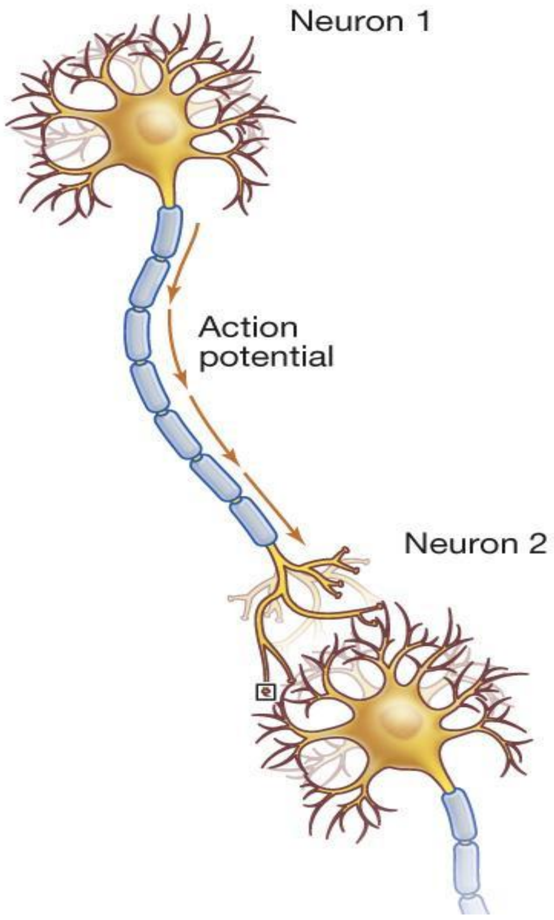
David Marr’s 3 levels of analysis:

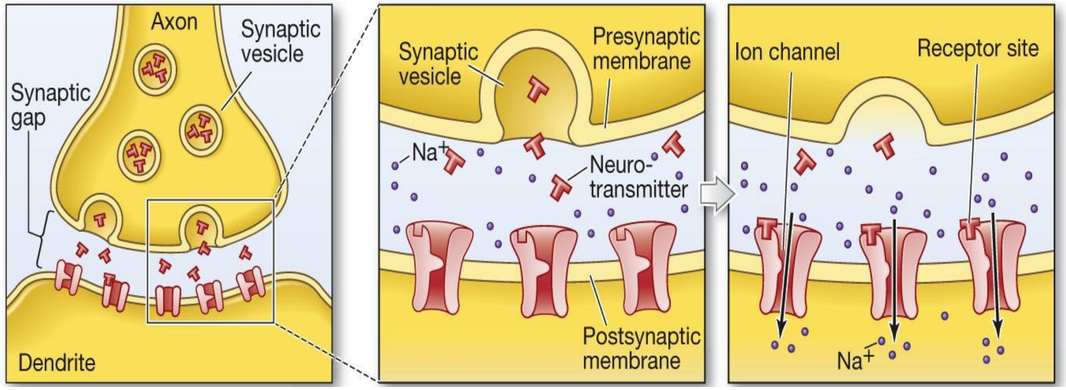


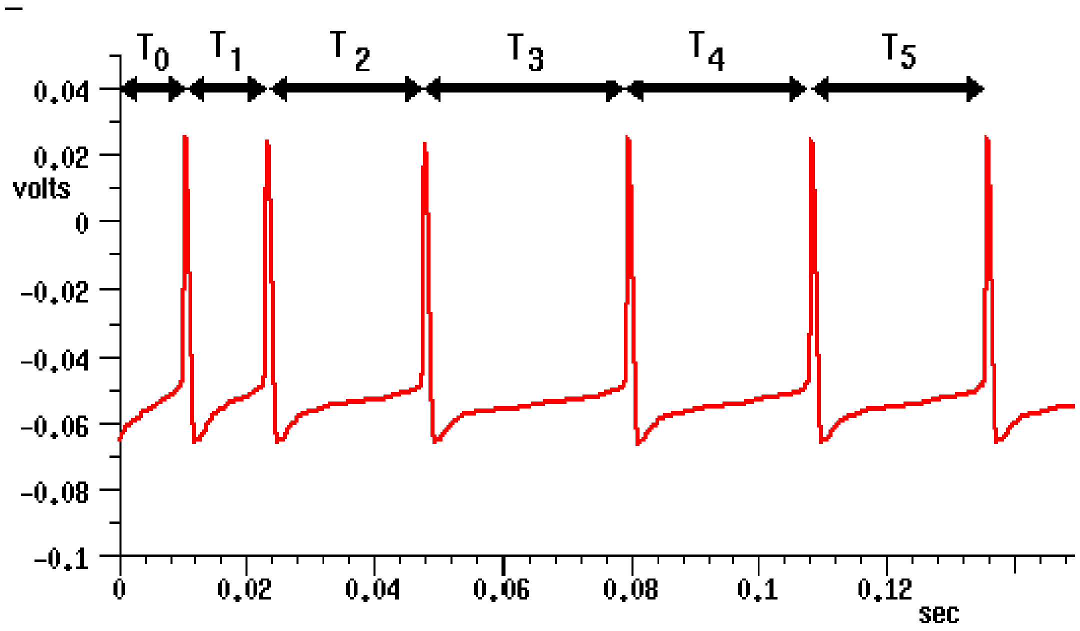
- What’s the computational problem?
- How to solve it algorithmically?
- How to implement the algorithm?

Implementation (1/5): Possible States

A neuron has two states: 0 (resting) vs. 1 (firing)

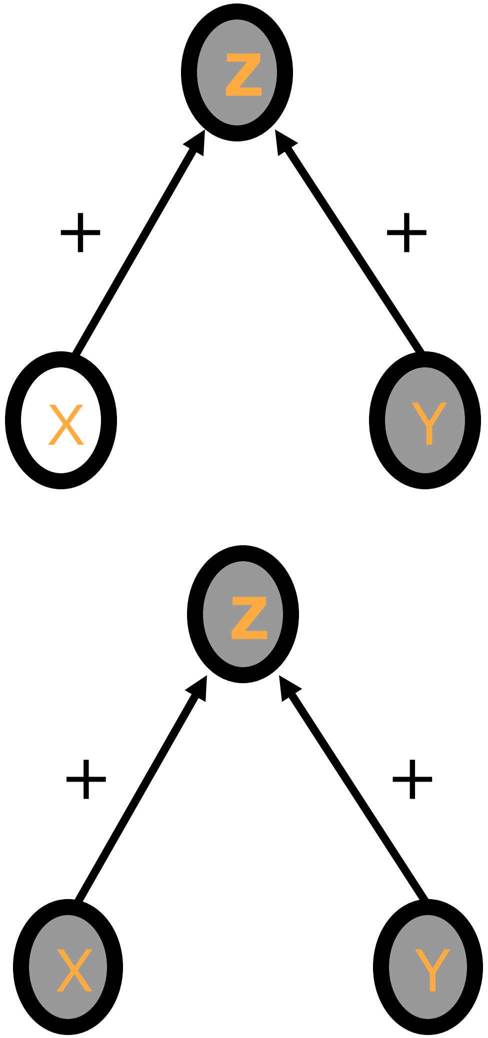






Implementation (2/5): Addition

Suppose the neuron Z has a **low** threshold

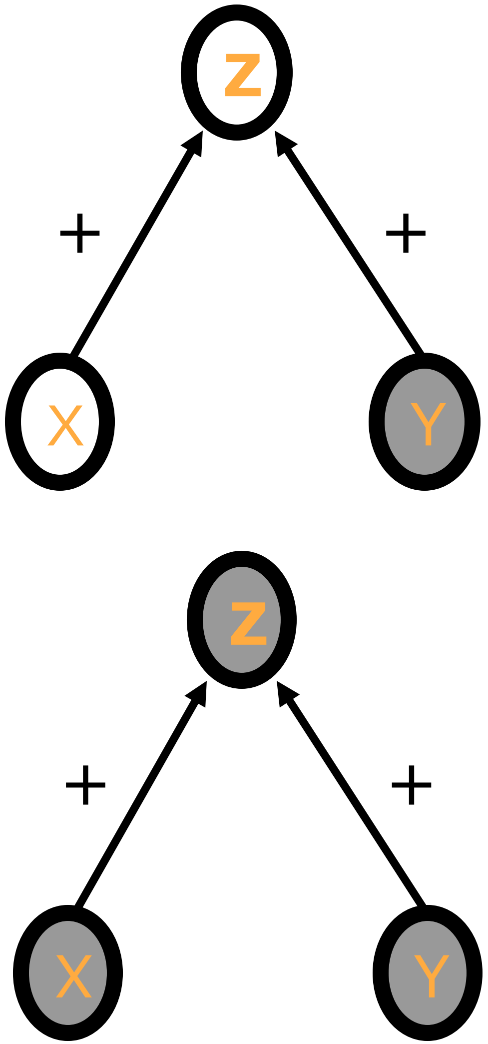


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Z=X+Y (OR)

Implementation (3/5): Multiplication

Suppose the neuron Z has a **high** threshold

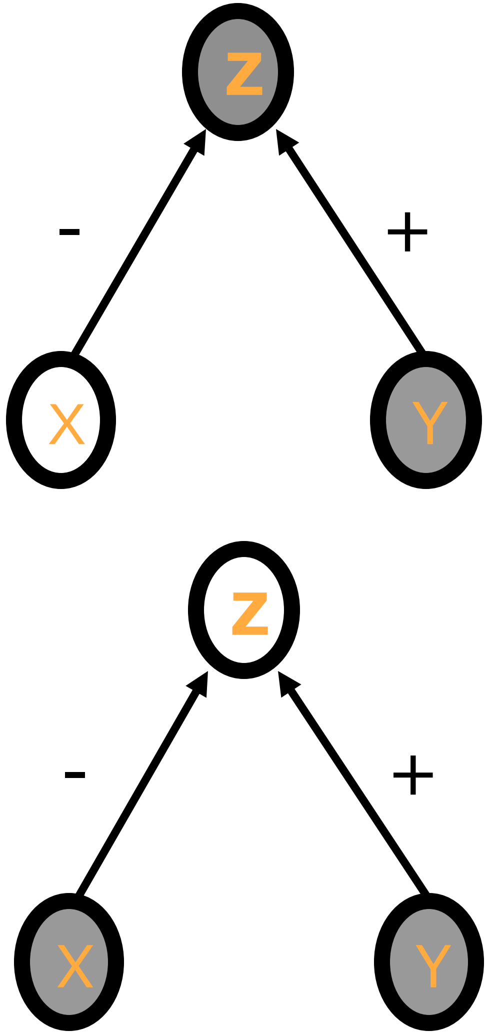


X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Z=X*Y (AND)

Implementation (4/5): Division

From excitatory to **inhibitory** connections:



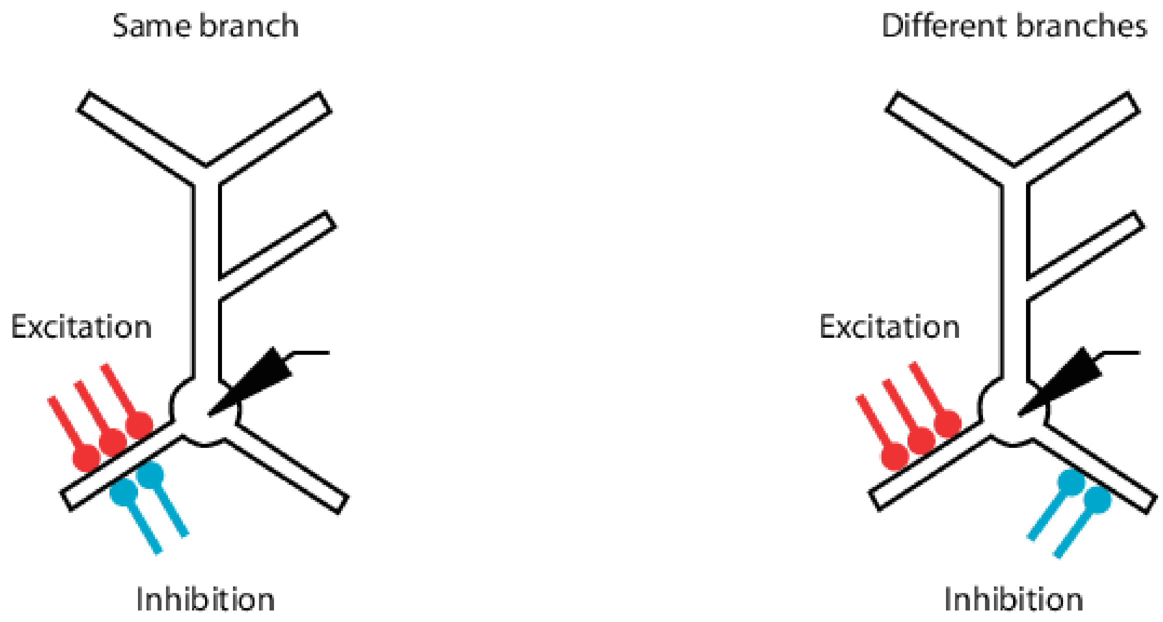
X	Z
0	1
1	0

Z=1-X (NOT)

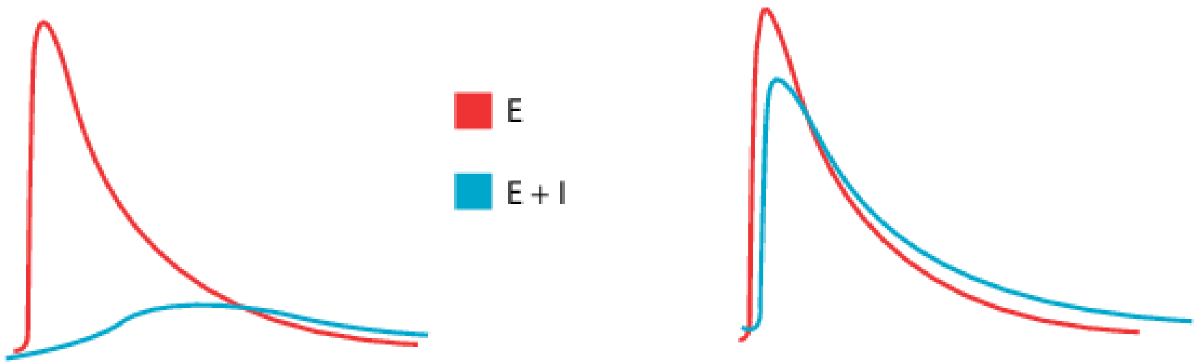
Implementation (5/5): Division

Inhibitions can lead to **subtraction** or **division**:

8→4→2→1
/2 /2 /2

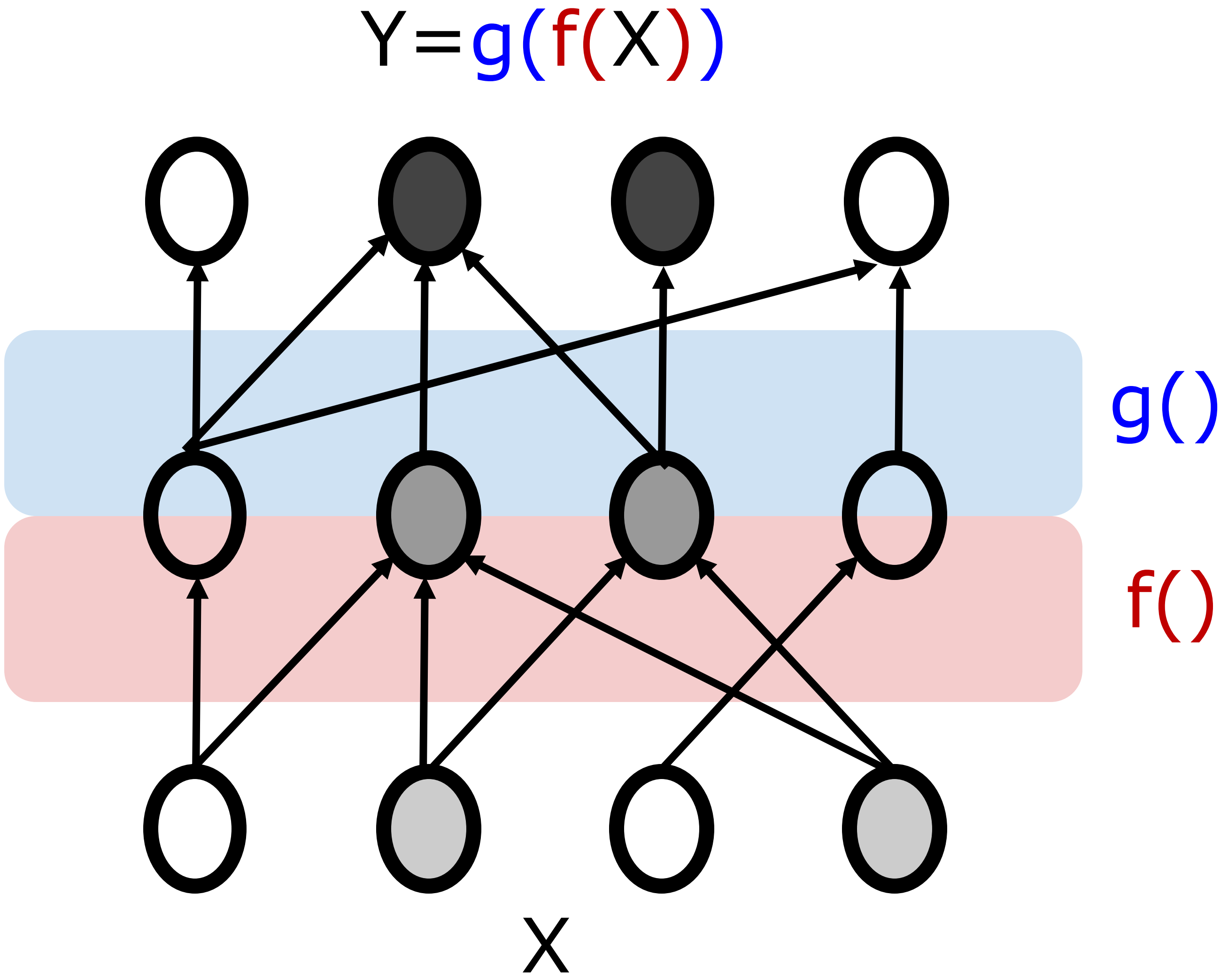


8→6→4→2
-2 -2 -2



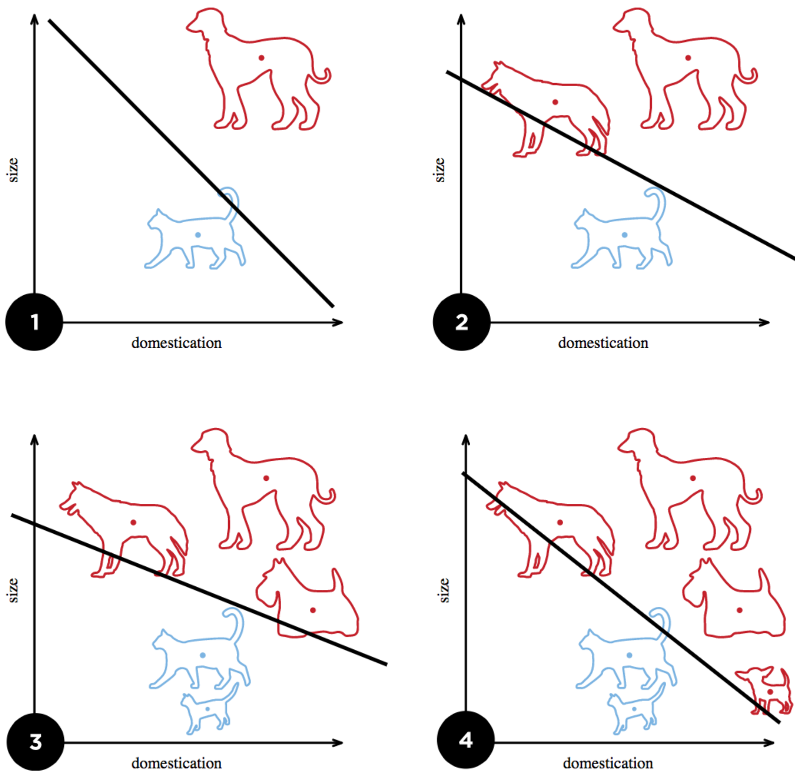
Algorithm: Arithmetic Combinations

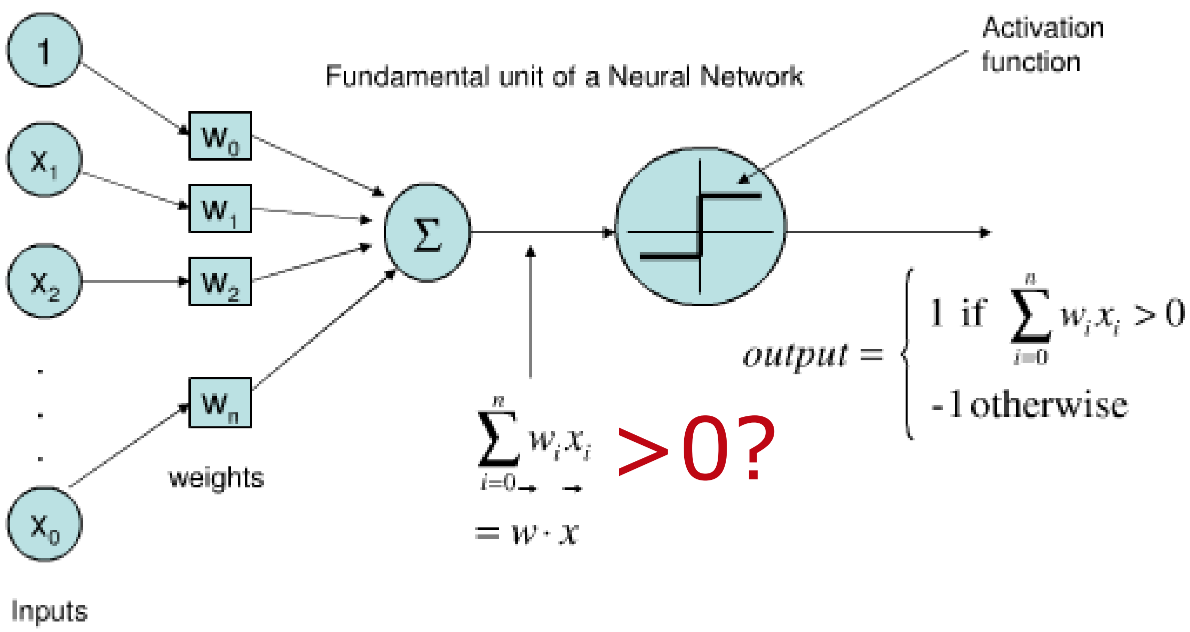
A neural network=A series of continuous transformations



Computational Problem: Recognition

Here is a dog detector where W encodes a dog template:



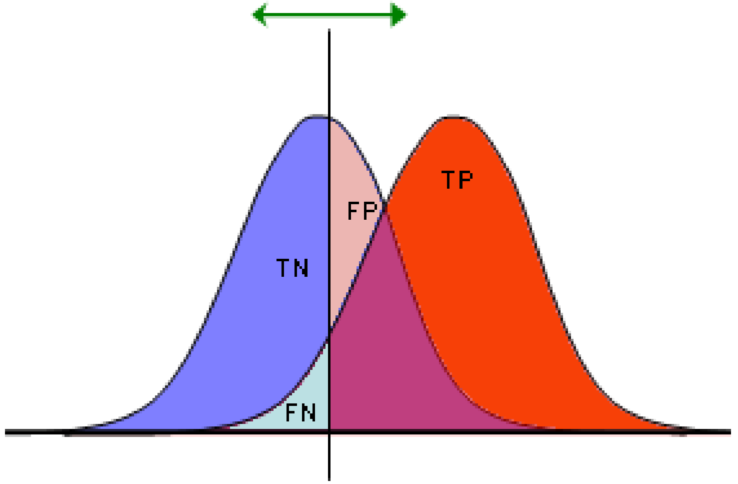


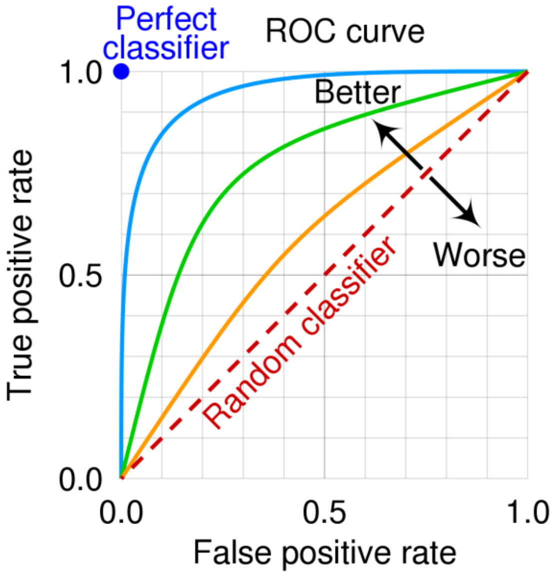
Decision Boundary: $w_2X_2=-w_1X_1-w_0 \Rightarrow w_2X_2+w_1X_1+w_01=0$

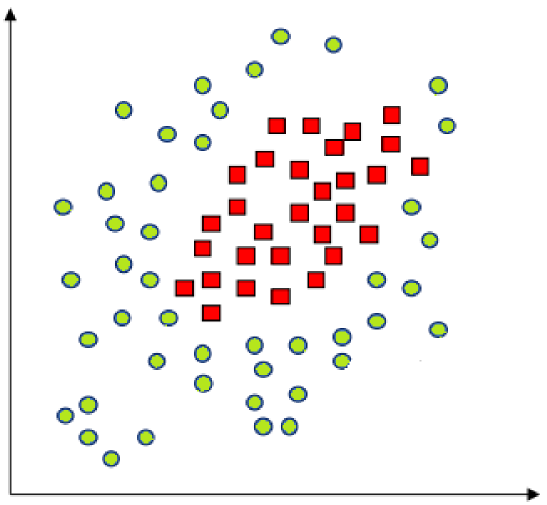
$(w_2,w_1,w_0)\cdot(X_2,X_1,1)=0$

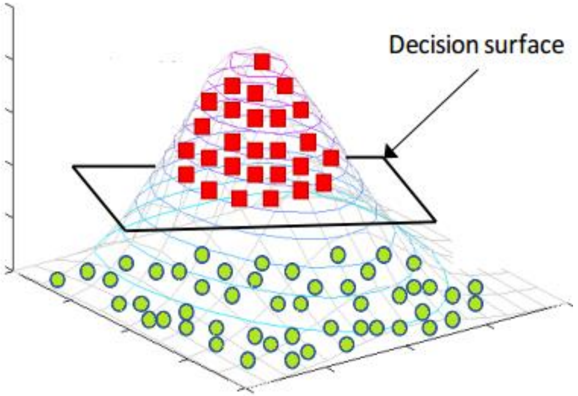
Revisiting the ROC curve

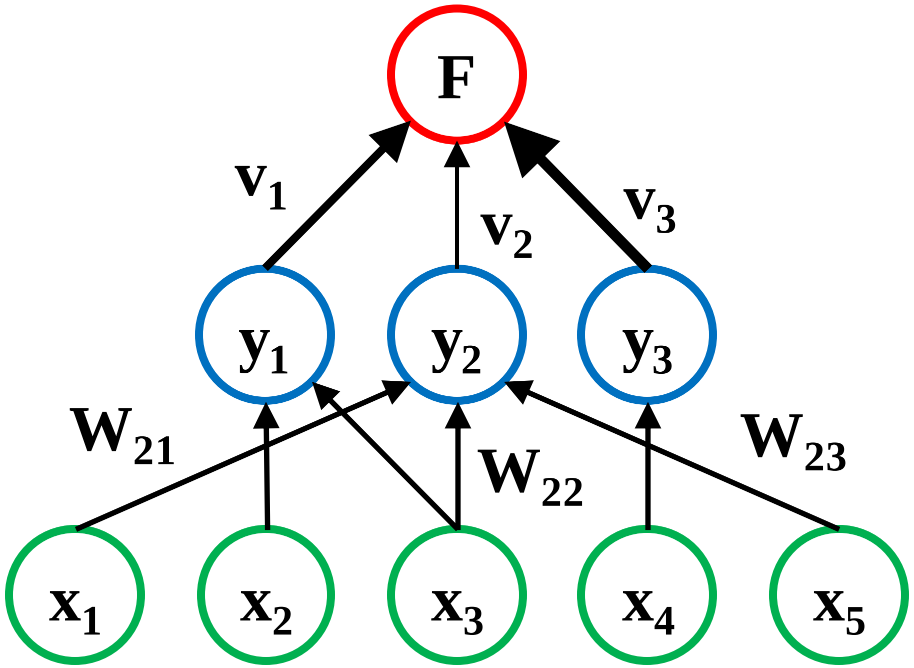
A ML can transform features to solve its problem





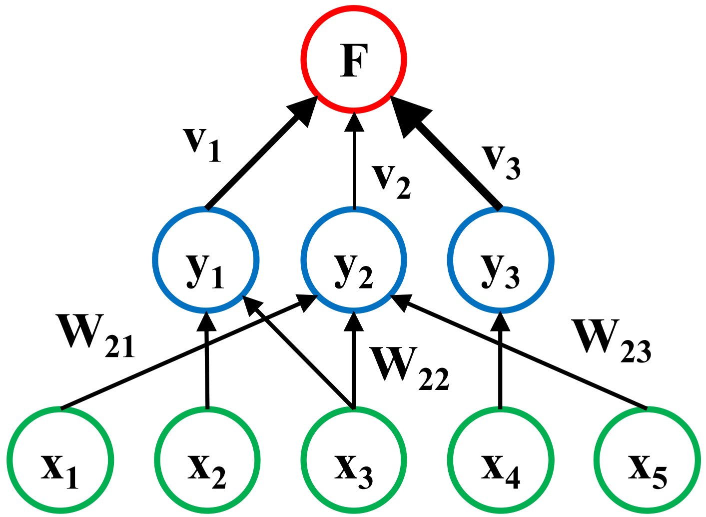


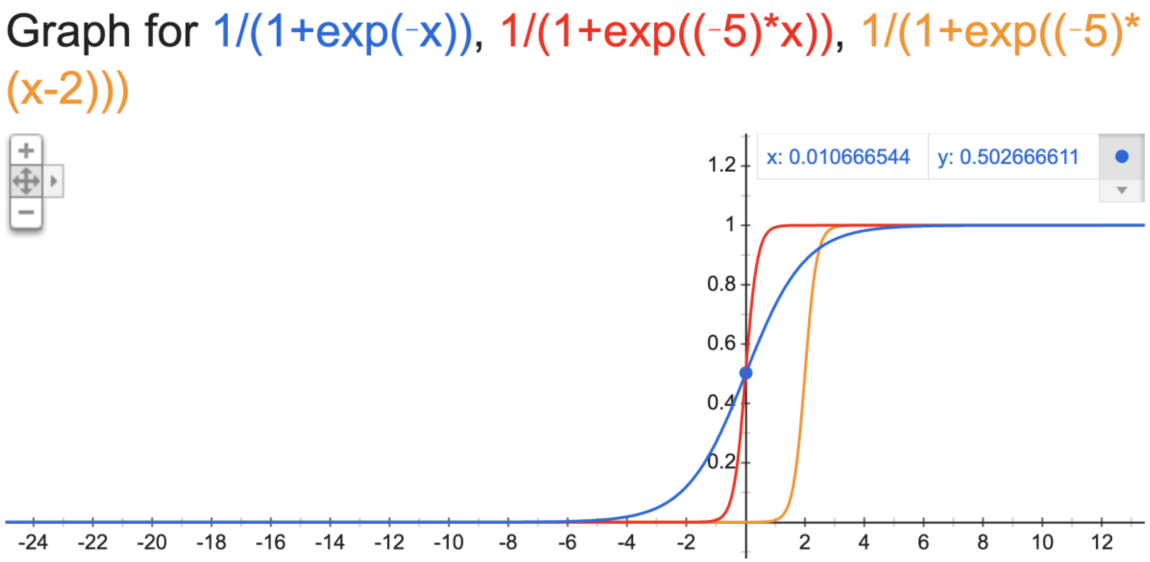




Universal Approximation Theorem

A 3-layer net can approximate any continuous function





$$\mathbf{F(x)} = \sum_{i=1}^{N} v_i \varphi \underbrace{\left(w_i^T \mathbf{x_i} - b_i\right)}_{\mathbf{y_j}}$$

as an approximate realization of the function f where f is independent of φ; that is,

$$|F(x) - f(x)| < \varepsilon$$

for all $x \in I_m$. In other words, functions of the form $F(x)$ are dense in $C(I_m)$.

A 3-layer net, Taylor Series, & Fourier Transform are special cases of **Generalized Additive Models**!

Topics for today

Computations of Neural Networks

Why do neural nets work?

(Deep) Learning in Neural Networks

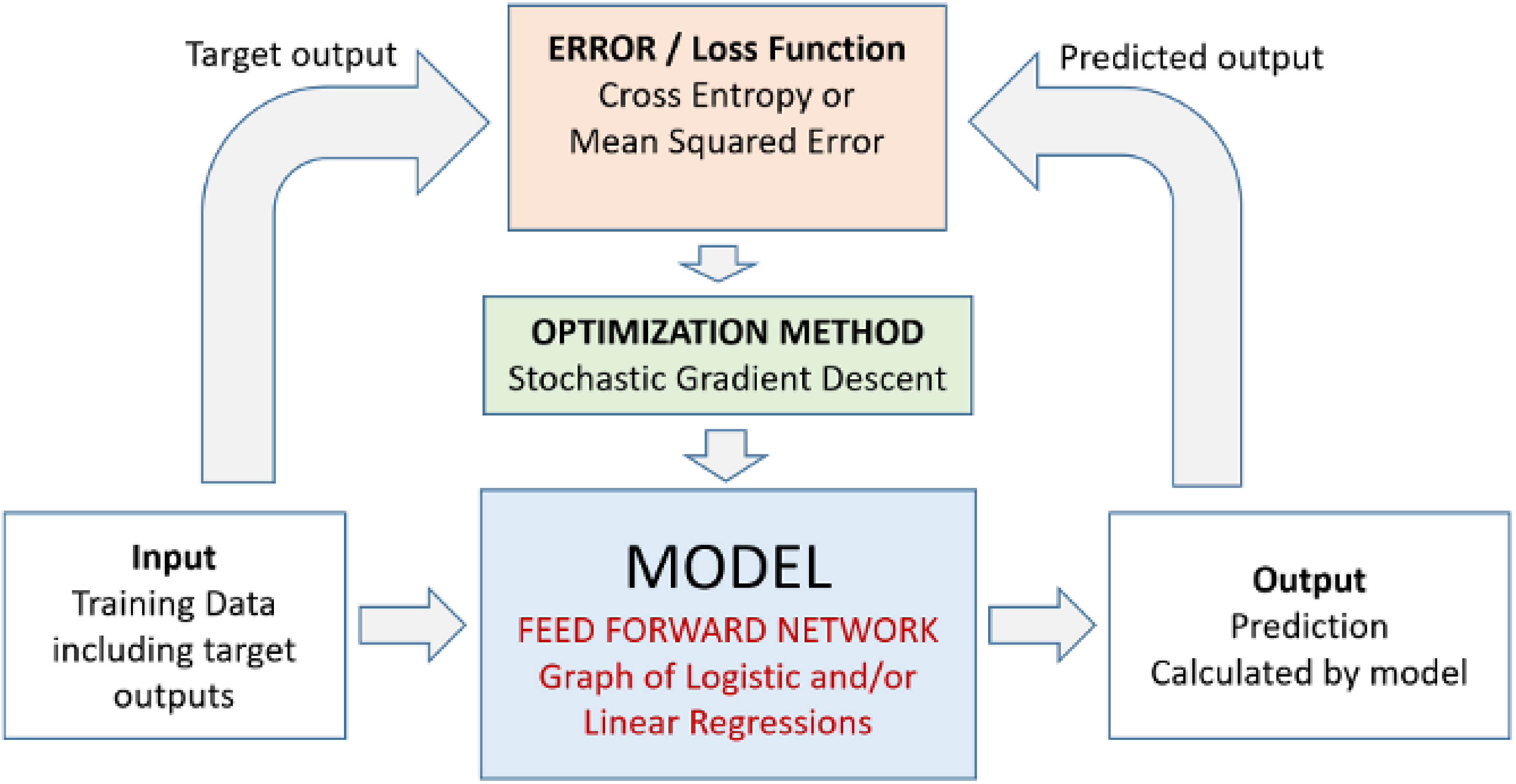
How do neural nets learn?

Explainable AI & Causal ML

What do neural nets learn?

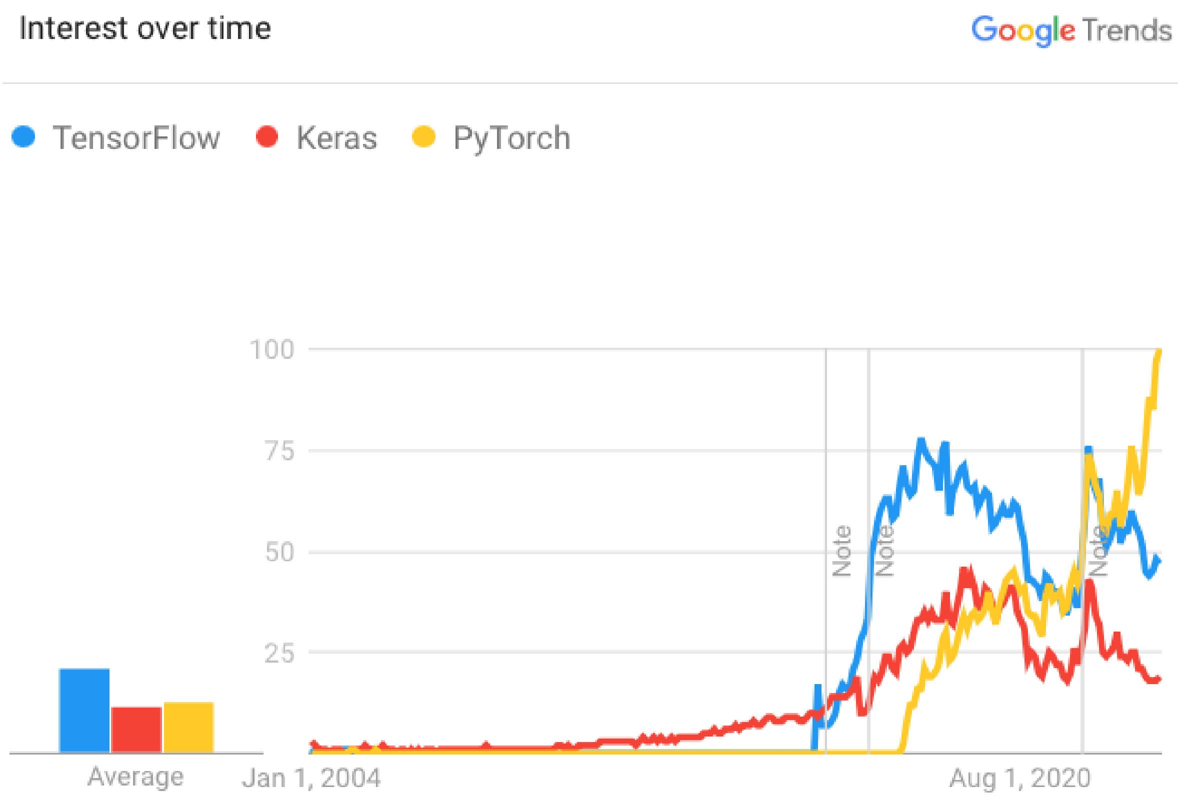
Workflow of Supervised Learning

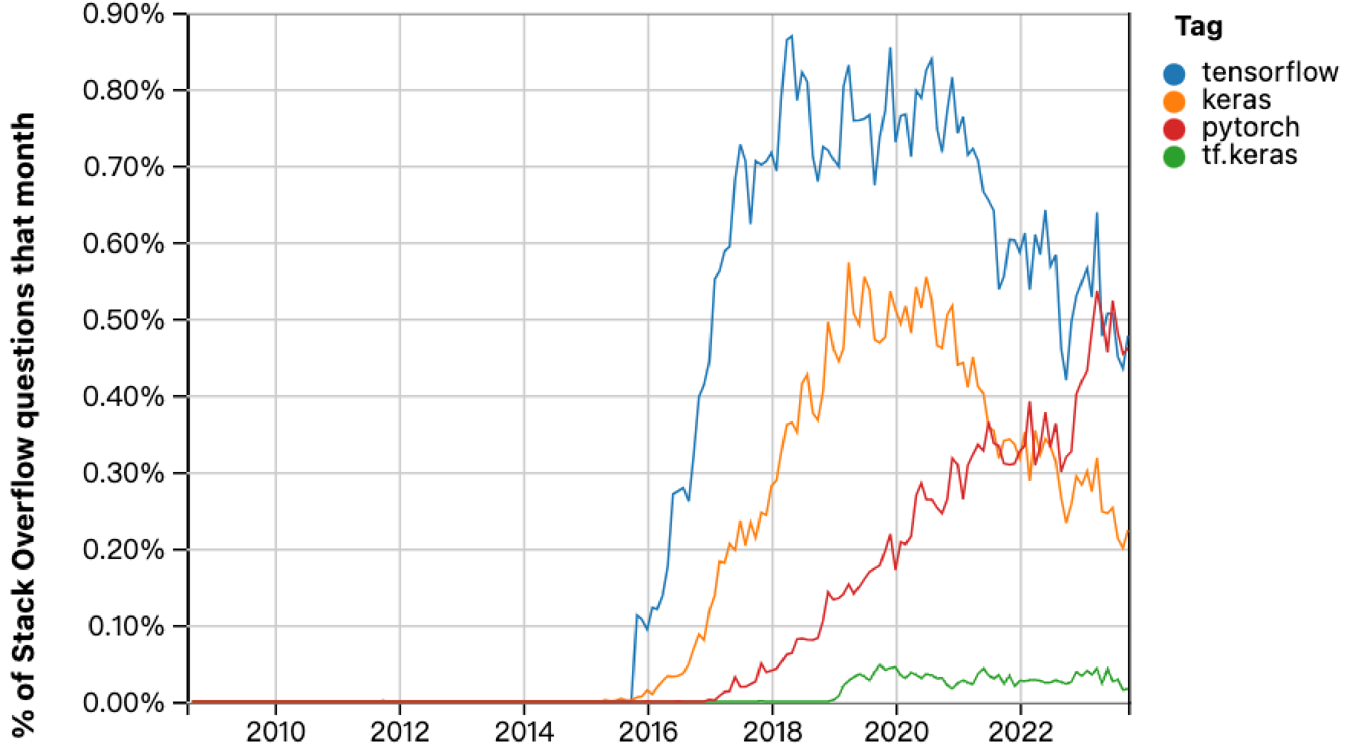
Adjusting model parameters to minimize prediction errors



Model (1/2): Frameworks

Keras is the easiest framework to build neural nets





Model (2/2): Keras Cheatsheet

ERROR / Loss Function
Cross Entropy or
Mean Squared Error

OPTIMIZATION METHOD
Stochastic Gradient Descent

MODEL
FEED FORWARD NETWORK
Graph of Logistic and/or
Linear Regressions

Python For Data Science *Cheat Sheet*

Keras

Learn Python for data science Interactively at www.DataCamp.com

Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32,
                    activation='relu',
                    input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
>>> model.fit(data,labels,epochs=10,batch_size=32)
>>> predictions = model.predict(data)
```

Data

Also see NumPy, Pandas & Scikit-Learn

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

Keras Data Sets

```
>>> from keras.datasets import boston_housing,
                               mnist,
                               cifar10,
                               imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/
ml/machine-learning-databases/pima-indians-diabetes/
pima-indians-diabetes.data"),delimiter=",")
>>> X = data[:,0:8]
>>> y = data [:,8]
```

Preprocessing

Also see NumPy & Scikit-Learn

Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

Train and Test Sets

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5,X_test5,y_train5,y_test5 = train_test_split(X,
                                                         y,
                                                         test_size=0.33,
                                                         random_state=42)
```

Standardization/Normalization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x_train2)
>>> standardized_X = scaler.transform(x_train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

Multilayer Perceptron (MLP)

Binary Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(12,
                    input_dim=8,
                    kernel_initializer='uniform',
                    activation='relu'))
>>> model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
```

Multi-Class Classification

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

Regression

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation,Conv2D,MaxPooling2D,Flatten
>>> model2.add(Conv2D(32,(3,3),padding='same',input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32,(3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))

>>> model2.add(Conv2D(64,(3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64,(3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))

>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

Inspect Model

`>>> model.output_shape`	Model output shape
`>>> model.summary()`	Model summary representation
`>>> model.get_config()`	Model configuration
`>>> model.get_weights()`	List all weight tensors in the model

Compile Model

MLP: Binary Classification

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

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='rmsprop',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
                  loss='mse',
                  metrics=['mae'])
```

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

Model Training

```
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               verbose=1,
               validation_data=(x_test4,y_test4))
```

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
                            y_test,
                            batch_size=32)
```

Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
                   optimizer=opt,
                   metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               validation_data=(x_test4,y_test4),
               callbacks=[early_stopping_monitor])
```

Error/Loss Functions

Adjusting model parameters to minimize prediction errors

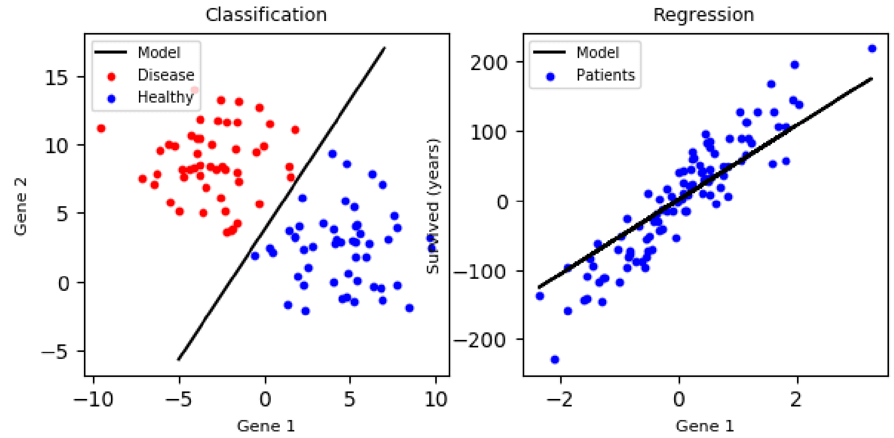
ERROR / Loss Function
Cross Entropy or
Mean Squared Error

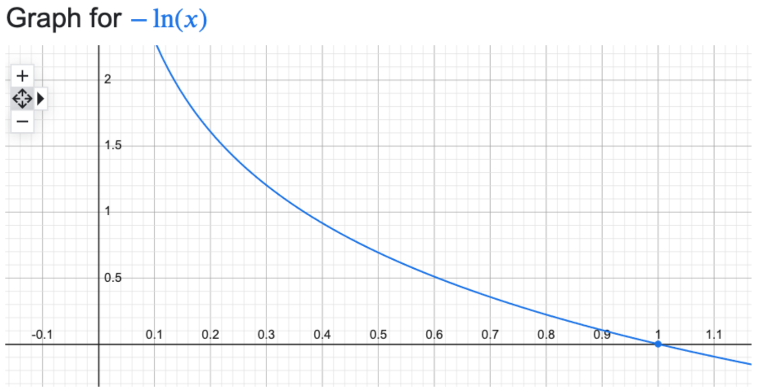
⇩

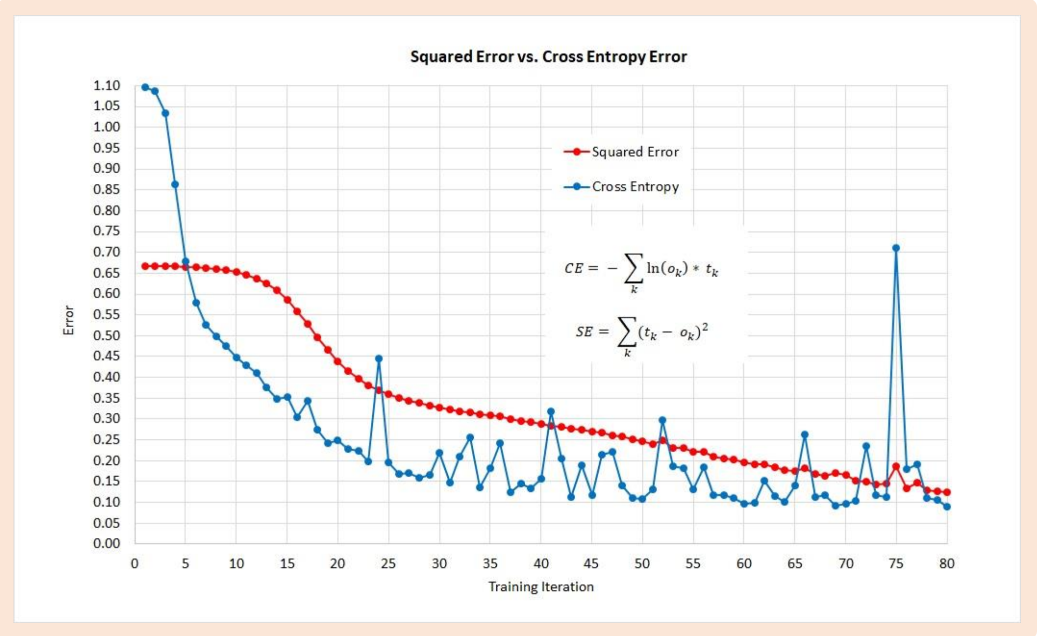
OPTIMIZATION METHOD
Stochastic Gradient Descent

⇩

MODEL
FEED FORWARD NETWORK
Graph of Logistic and/or
Linear Regressions







$$\begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix} \quad D(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_j y_j \ln \hat{y}_j \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$\hat{\mathbf{y}}$ $\mathbf{y}$

When Y_{pred}=[0.1, 1, 0.4],
CE=-ln(1)=0 but

$$MSE = \sqrt{0.1^2 + 0.4^2} \neq 0$$

Optimization (1/2): Gradient Descent

Using iterative methods to minimize error/loss functions

ERROR / Loss Function
Cross Entropy or
Mean Squared Error

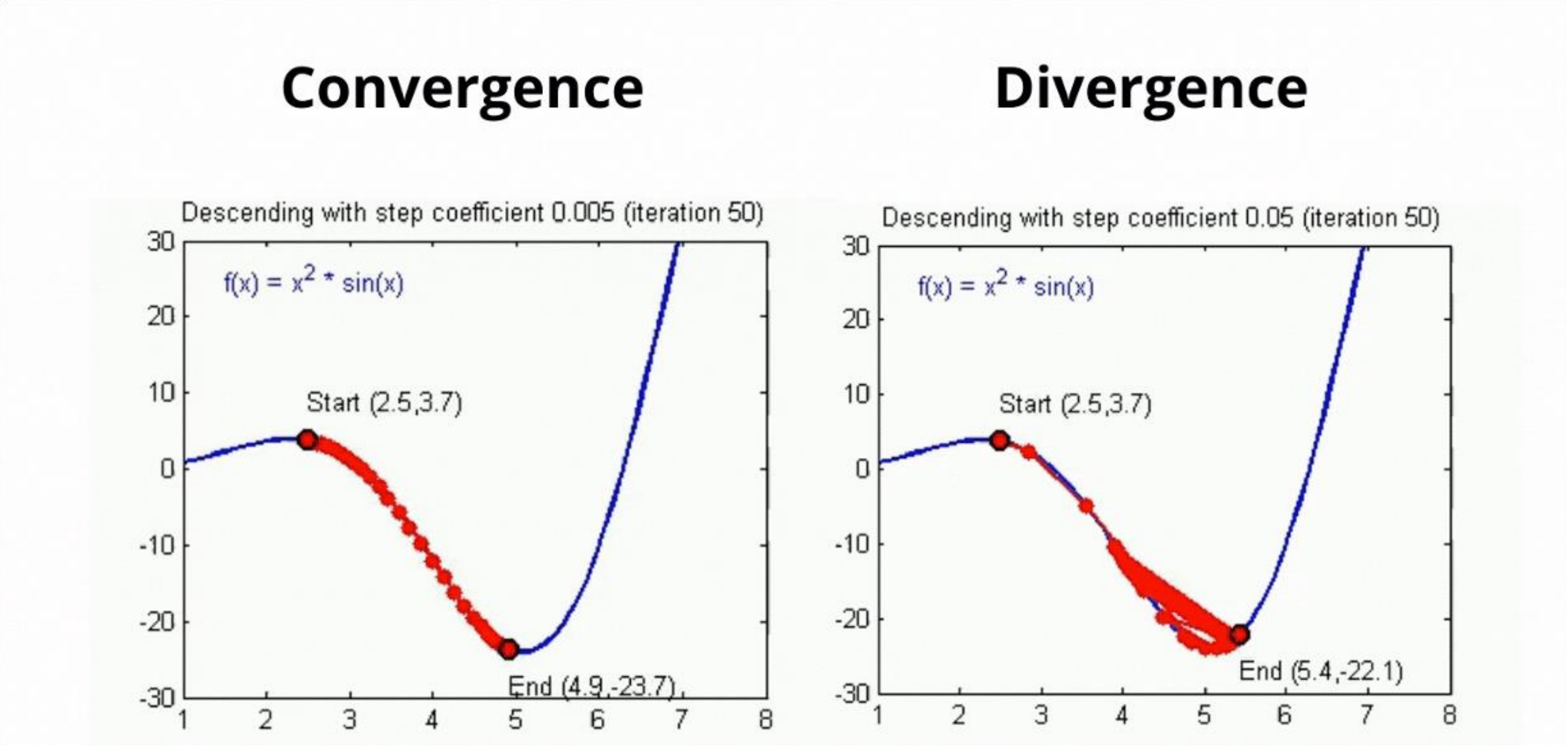
⇩

OPTIMIZATION METHOD
Stochastic Gradient Descent

⇩

MODEL
FEED FORWARD NETWORK
Graph of Logistic and/or
Linear Regressions

$$\frac{\partial L(\vec{W})}{\partial \vec{W}} = 0 \xrightarrow[\textit{over time}]{L \textit{ is changing}} \vec{W}^{new} = \vec{W}^{old} - \alpha \frac{\partial L(\vec{W})}{\partial \vec{W}}$$



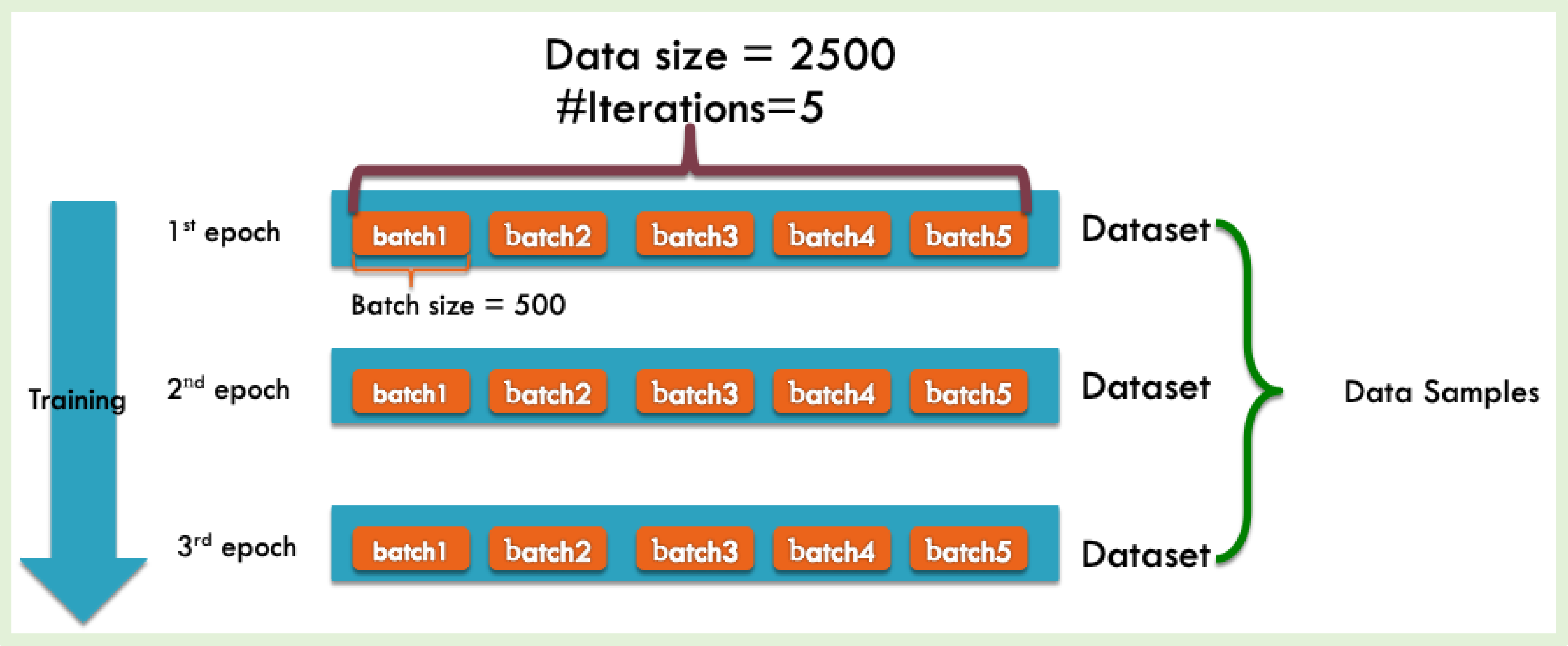
Optimization (2/2): Repetitions

1 batch/iteration = 1 adjustment of model parameters

ERROR / Loss Function
Cross Entropy or
Mean Squared Error

OPTIMIZATION METHOD
Stochastic Gradient Descent

MODEL
FEED FORWARD NETWORK
Graph of Logistic and/or
Linear Regressions

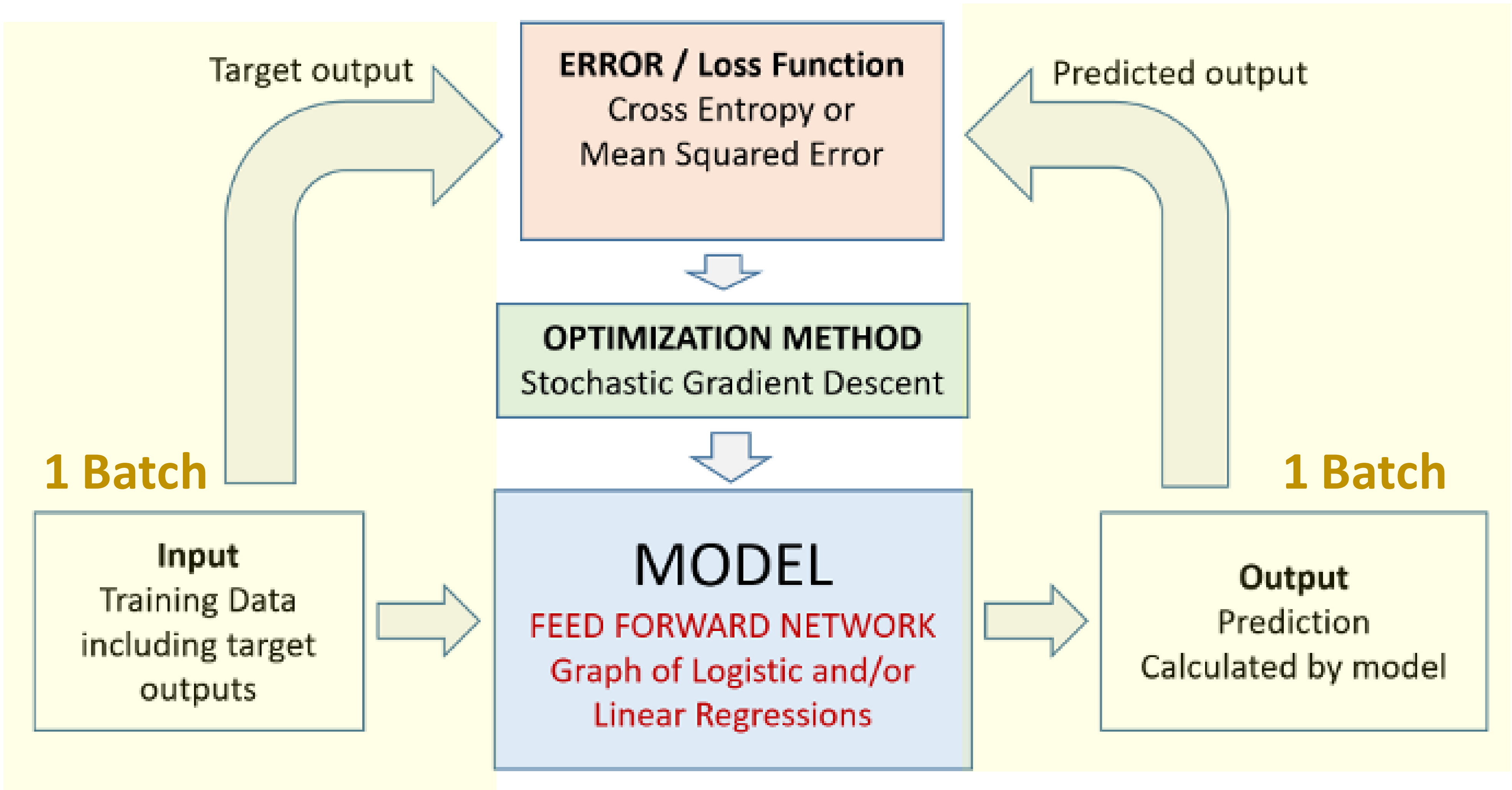


In each batch, L is different:

$$\vec{W}^{new} = \vec{W}^{old} - \alpha \frac{\partial L(\vec{W})}{\partial \vec{W}}$$

Workflow of Supervised Learning

Adjusting model parameters to minimize prediction errors



Topics for today

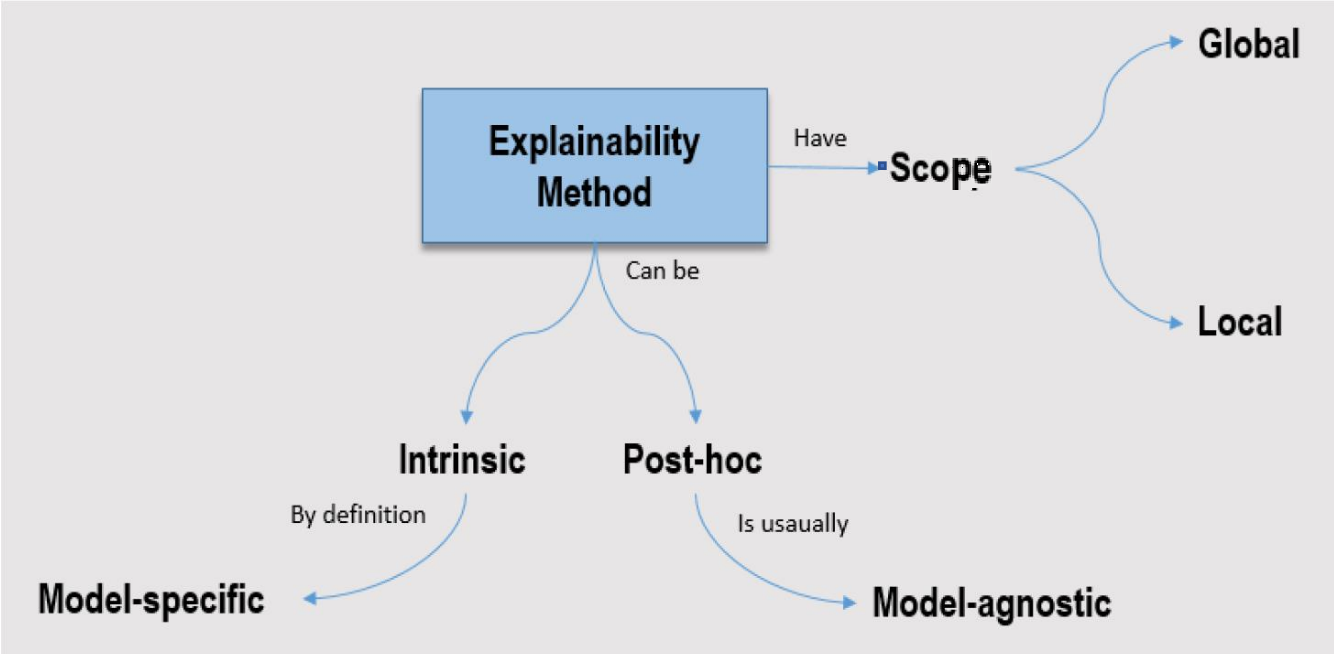
Computations of Neural Networks
Why do neural nets work?

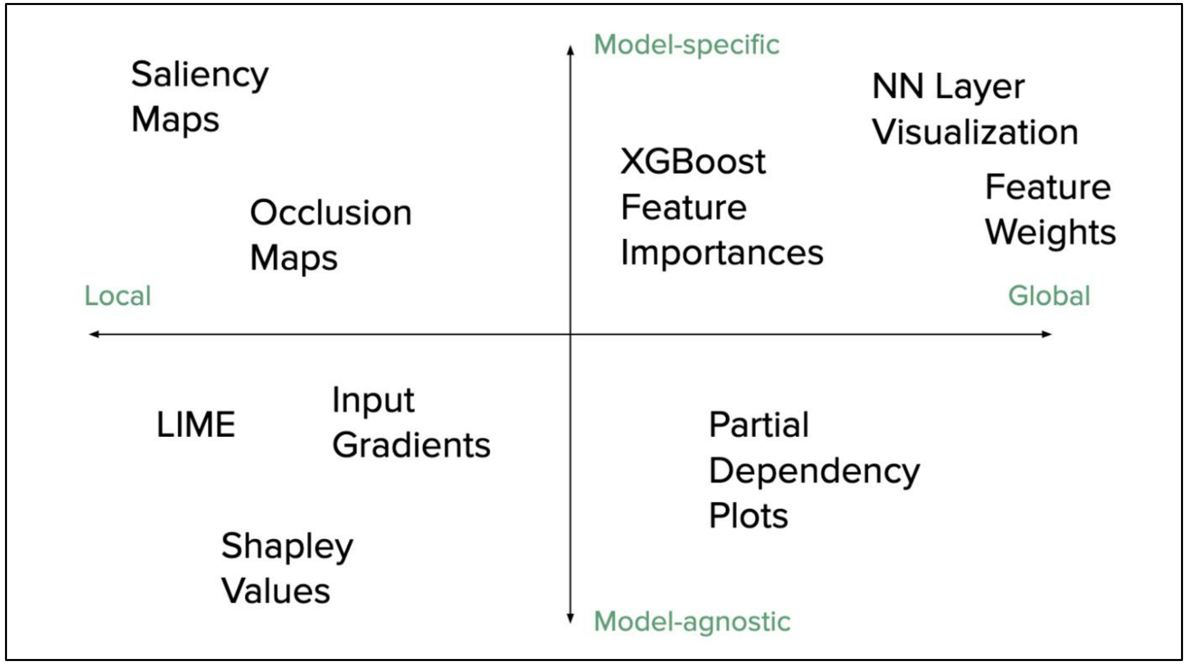
(Deep) Learning in Neural Networks
How do neural nets learn?

Explainable AI & Causal ML
What do neural nets learn?

eXplainable AI (XAI) methods

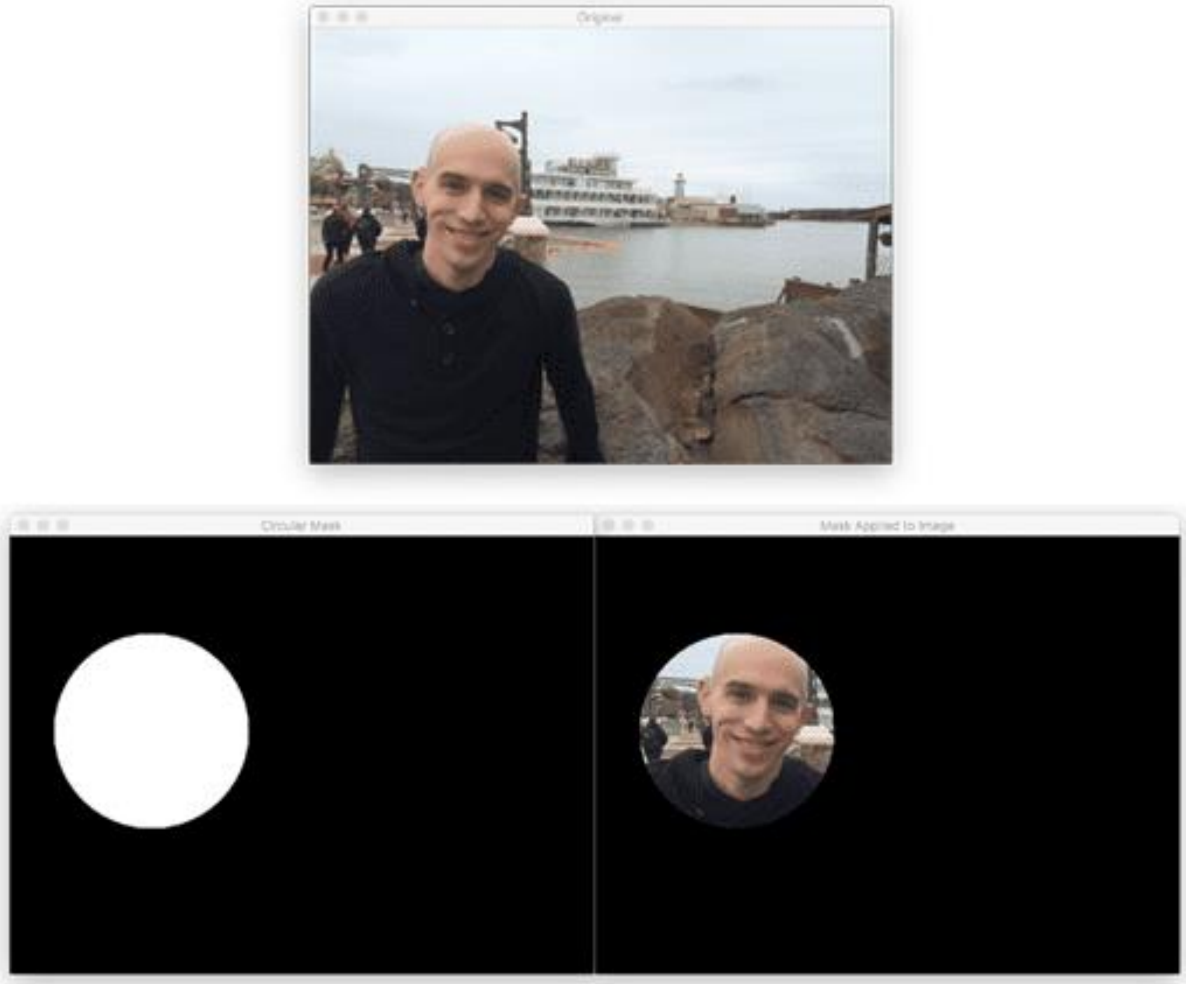
Model-agnostic: Models are accessible but not trainable

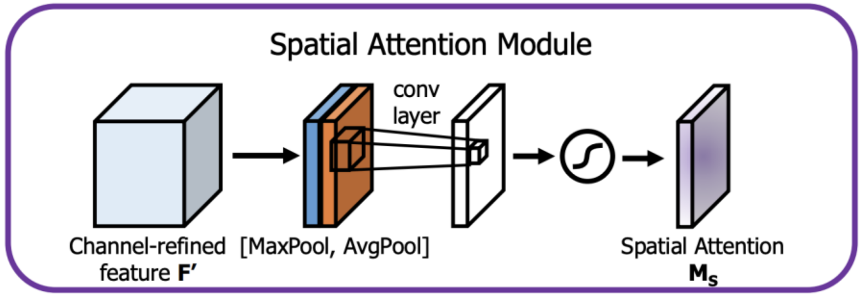


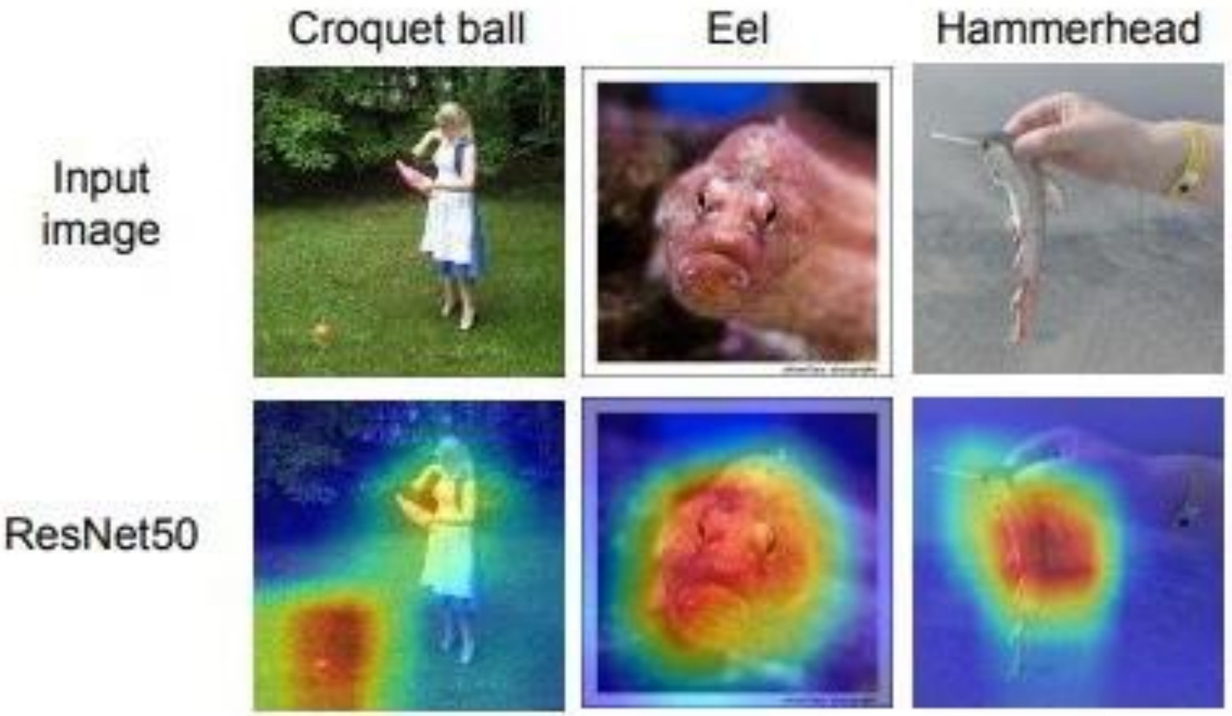


Cognitivist XAI: Model-specific

For example, simply adding a learnable attention layer

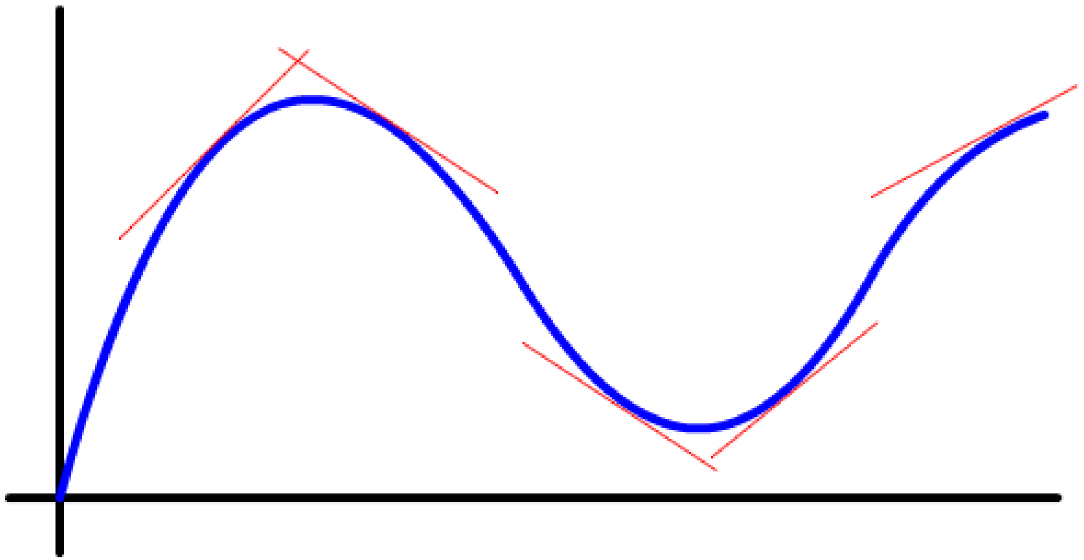


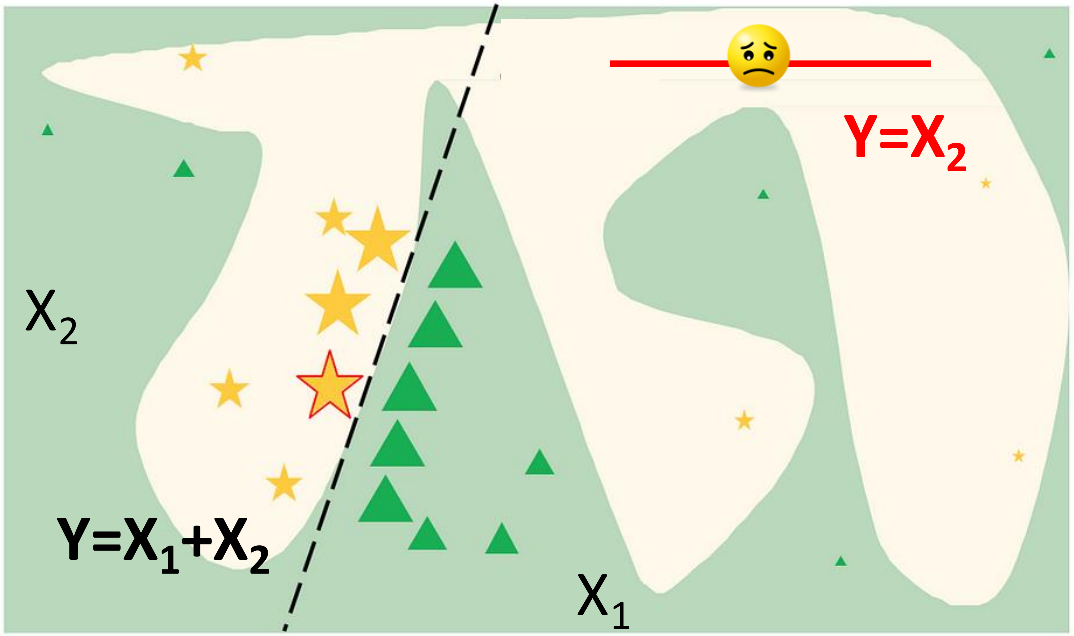




Behaviorist XAI: Model-agnostic (1/2)

LIME=Local Interpretable Model-Agnostic Explanation

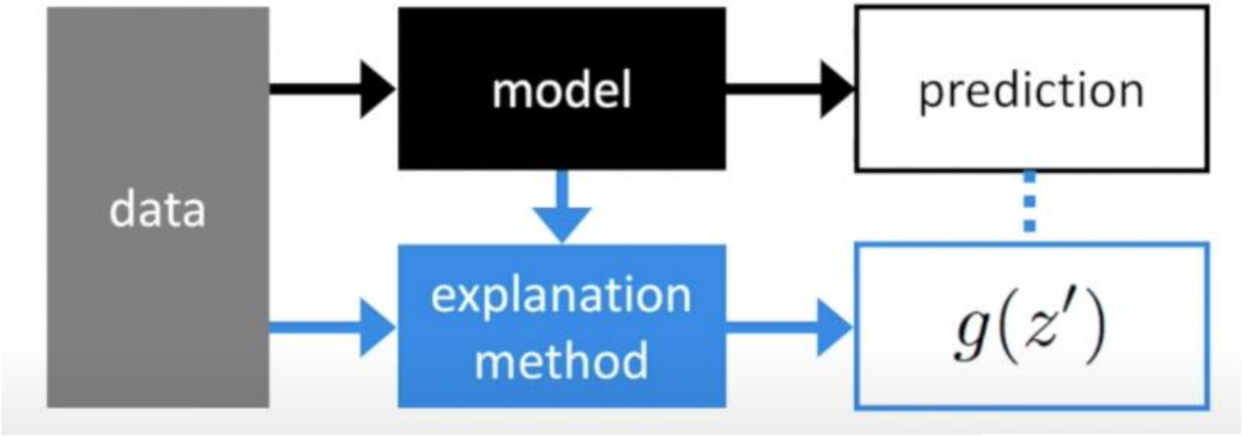




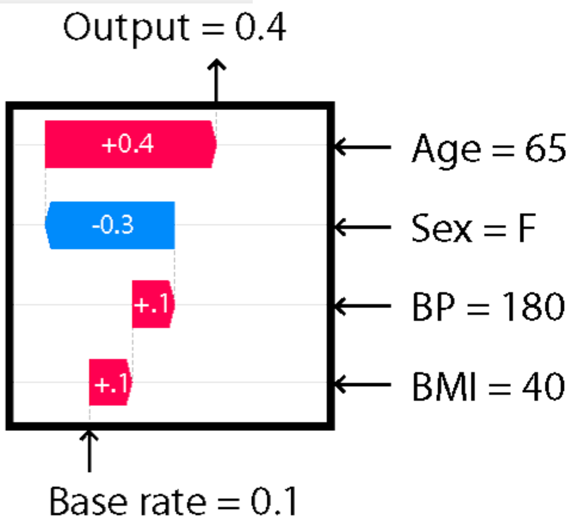
Test instance → Random data perturbation → Weight the new samples → Predict labels for new samples with original model → Train a weighted, interpretable model → Feature selection → Interpretable Representation → User

Behaviorist XAI: Model-agnostic (2/2)

LIME is a special case of SHapley Additive exPlanations (SHAP)
, which estimate feature importance **better than** feature lesion.



$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i$$



Ground Truth: $Y=X_1+X_2+X_1X_2$
Intact: $Y=\text{ModelA}(X_1,X_2)$
Lesioned: $Y=\text{ModelB}(X_2)=X_2$

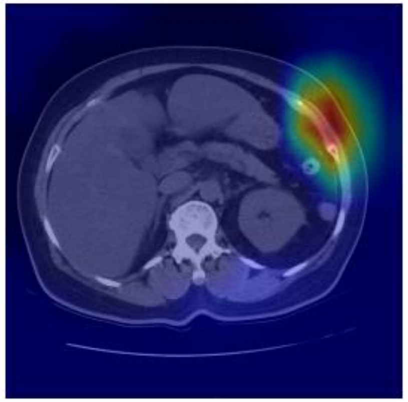
Intact-Lesioned=$X_1+X_1X_2$

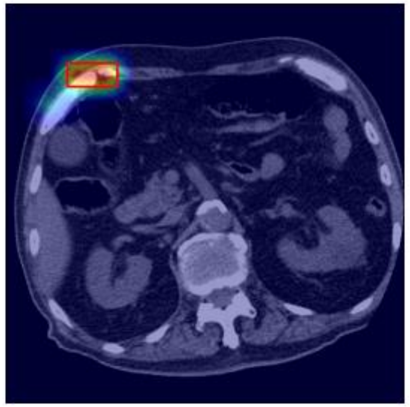
But this difference is NOT solely contributed by X_1!

Causal Machine Learning: Causality→AI

XAI may reveal that your AI is not learning real causality

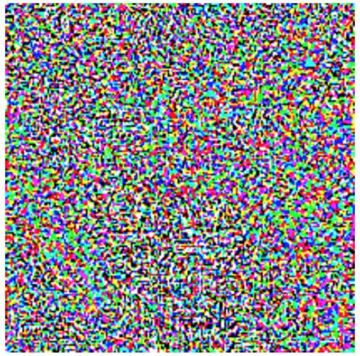
An "accurate" classifier of bile duct stone (膽管結石):





Panda (60% confidence)

+



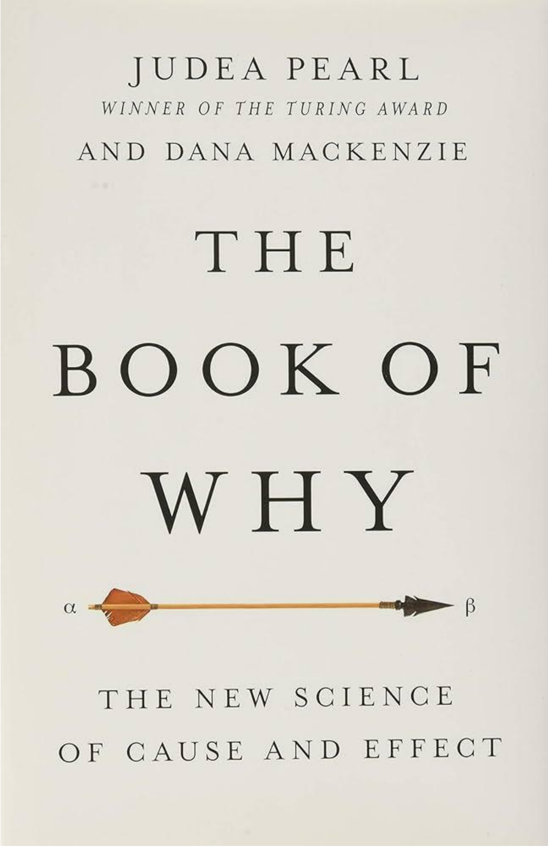
Adversarial Perturbation

=

Gibbon (99% confidence)

So what?

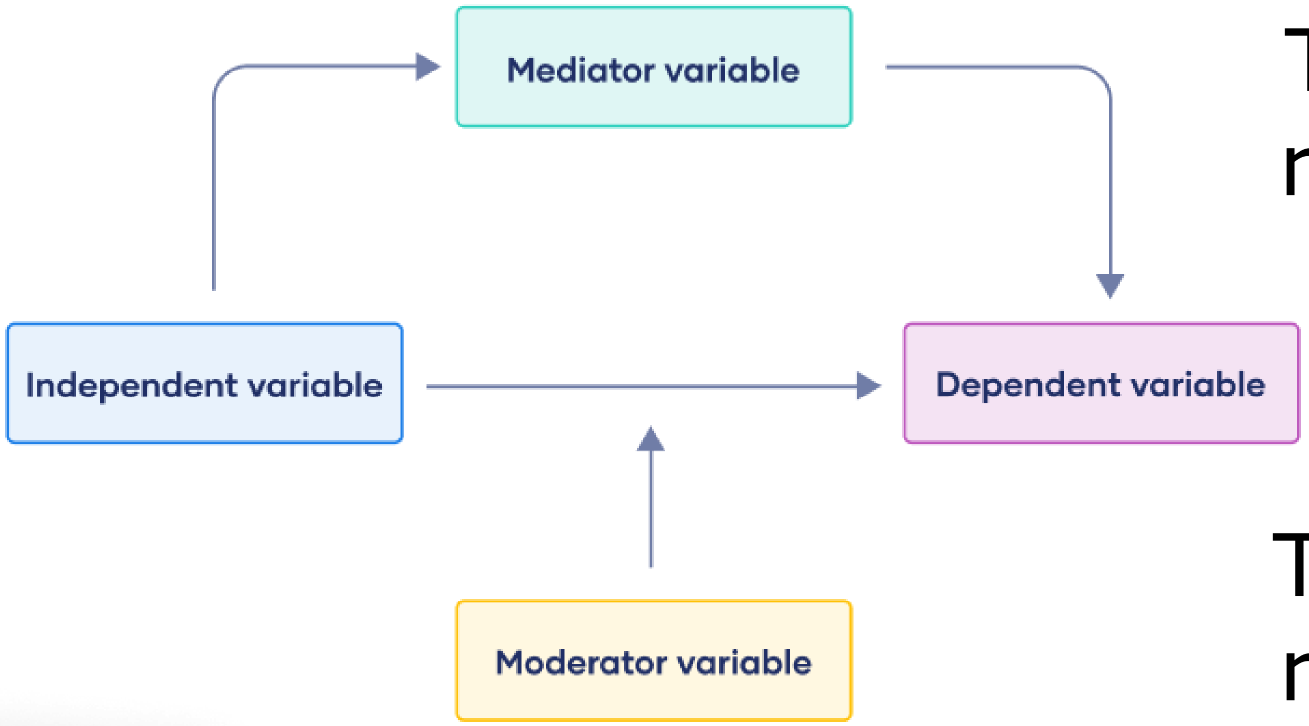
Your model can't generalize to other datasets.



Causal ML: AI→ Causality (1/2)

The feature importance of X decreases after M added:

$Y=f(X) \rightarrow Y=\mathbf{f(}X, M\mathbf{)}=\beta_X X+\beta_M M+\beta_0$



The linear function **f** can be replaced by a nonlinear ML!

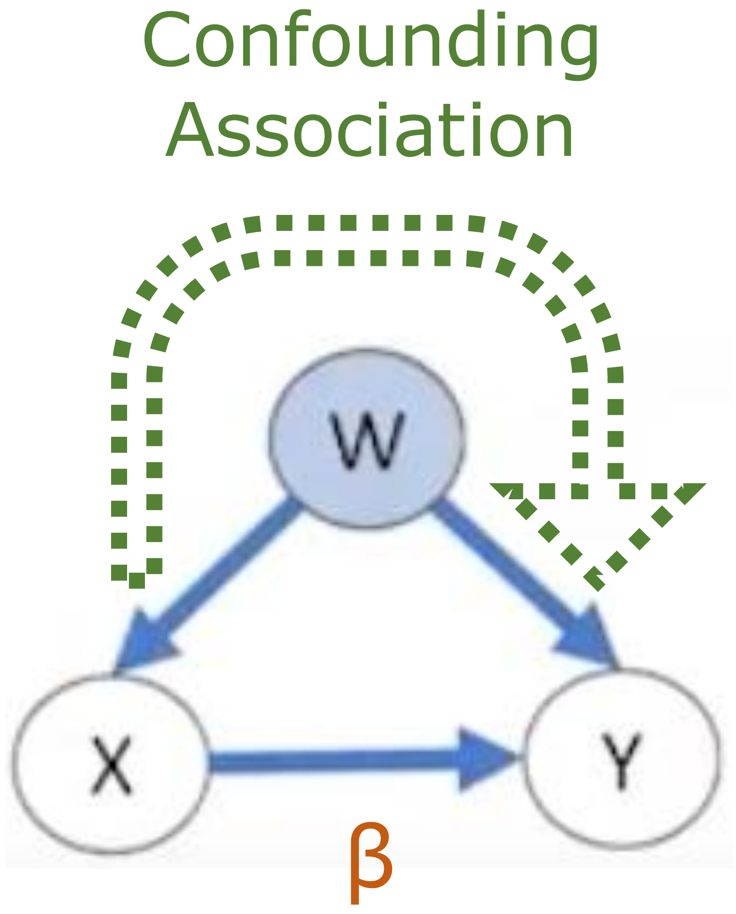
The linear function **g** can be replaced by a nonlinear ML!

More accurate predictions after M added:

$Y=g(X) \rightarrow Y=\mathbf{g(}X, M\mathbf{)}=\beta_X X+\beta_M M+\beta_{MX} MX+\beta_0$

Causal ML: AI→ Causality (2/2)

Double ML uses ML to factor out confounding associations:



$Y = f(X, W)$

Step 1:

$$Y = g(W) + Y_R$$
$$X = h(W) + X_R$$

Step 2:

$$Y_R = \beta X_R + \varepsilon$$

Topics for today

Computations of Neural Networks

Why do neural nets work?

(Deep) Learning in Neural Networks

How do neural nets learn?

Explainable AI & Causal ML

What do neural nets learn?

Game Over