

CS21si: AI for Social Good

Lecture 3: Training Deep Neural Networks

Deep Learning

Plan for Today

- COMPAS
- Review of neural networks
- Activation functions
- Implementing deep neural networks
- Tuning

Risk Assessment

PERSON			
Name:		Offender #:	DOR:
Gender:		Marital Status:	Agency:
Male		Single	DAJ

1. Recidivism Score

ASSESSMENT INFORMATION			
Case Identifier:	Scale Set:	Screener:	Screening Date:
	Wisconsin Core - Community Language		

2. Violence Score

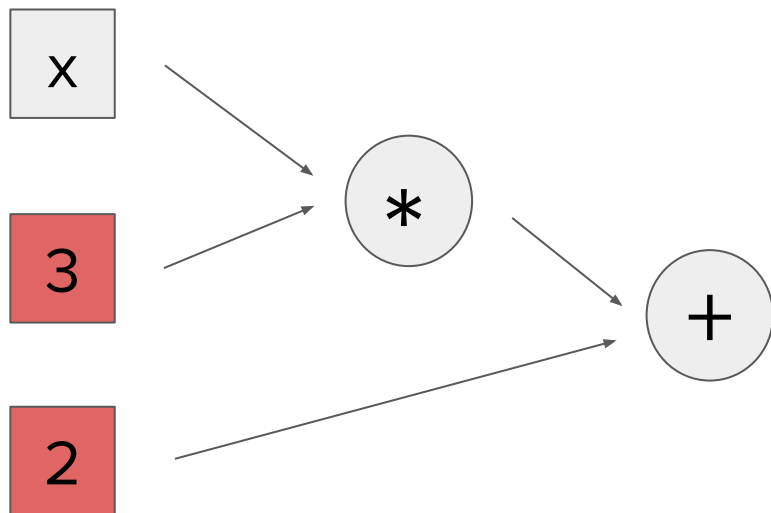
Current Charges			
<input type="checkbox"/> Homicide	<input checked="" type="checkbox"/> Weapons	<input checked="" type="checkbox"/> Assault	<input type="checkbox"/> Arson
<input type="checkbox"/> Robbery	<input type="checkbox"/> Burglary	<input type="checkbox"/> Property/Larceny	<input type="checkbox"/> Fraud
<input type="checkbox"/> Drug Trafficking/Sales	<input type="checkbox"/> Drug Possession/Use	<input type="checkbox"/> DUI/OUIL	<input checked="" type="checkbox"/> Other
<input type="checkbox"/> Sex Offense with Force	<input type="checkbox"/> Sex Offense w/o Force		
1. Do any current offenses involve family violence?			
<input checked="" type="checkbox"/> No <input type="checkbox"/> Yes			

Jupyter Exercise 1: Visualize the Data

Task: Recreate COMPAS recidivism scores from data about defendants

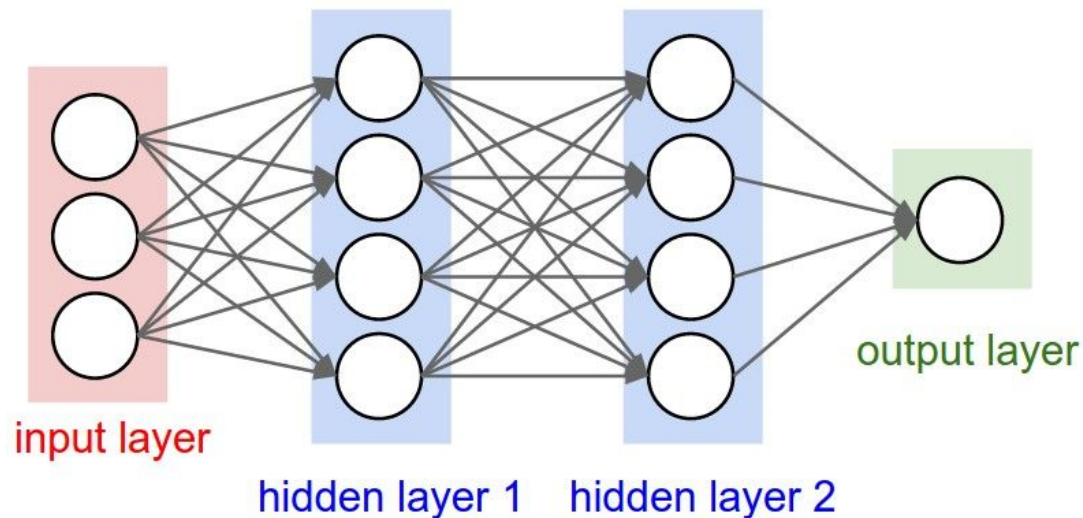
Revisiting Computational Graphs

$3x + 2$:



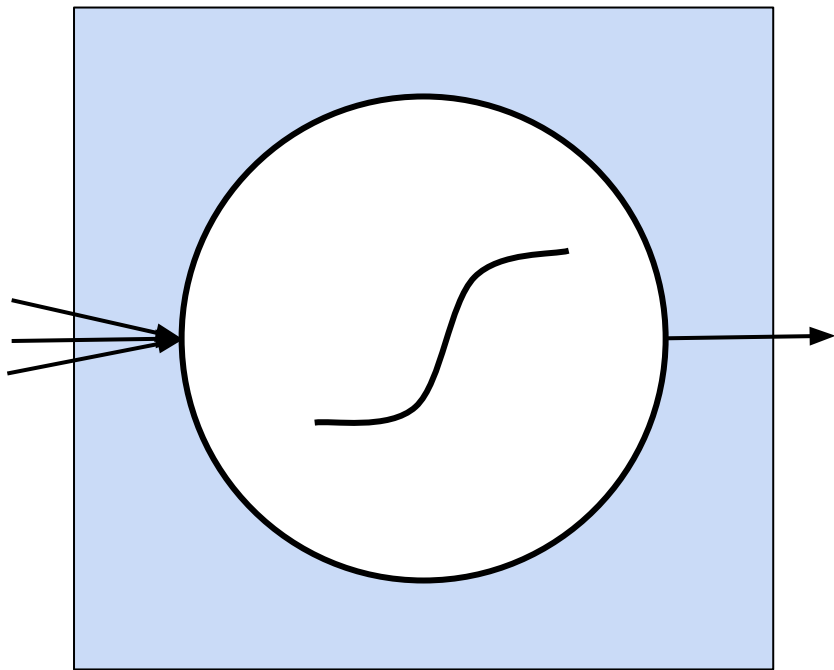
- Computational graphs can represent any function
- Forward pass to compute values/losses
- Backward pass to compute gradients

Revisiting Neural Networks



- Linear layers with non-linearities in between.
- Each arrow is a weight
- The nodes in the hidden layers represent operations...

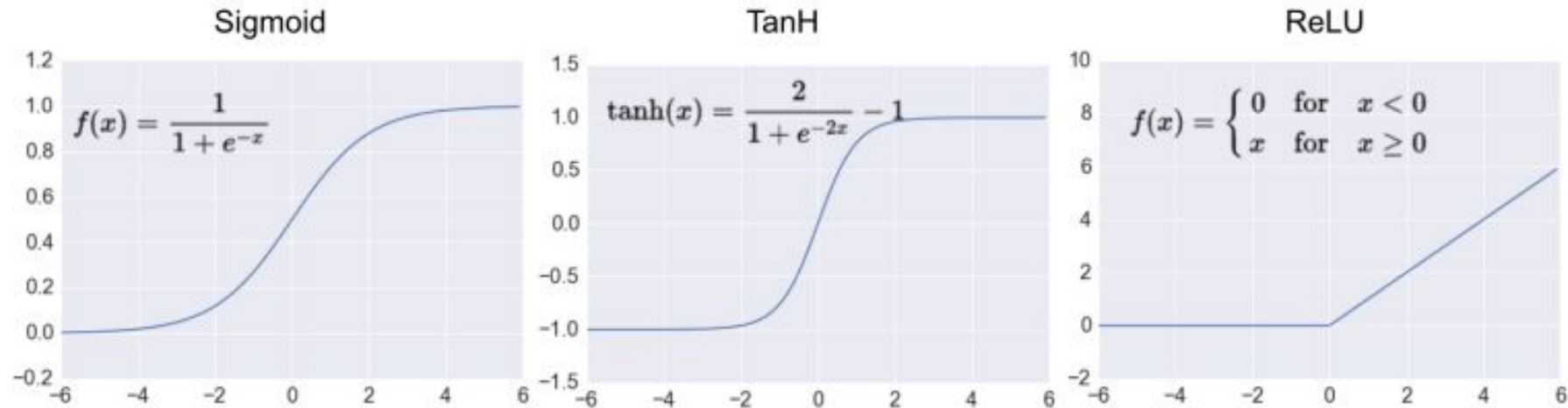
Zooming into the Hidden Layer



Each node encapsulates **a non-linearity, or an activation function**

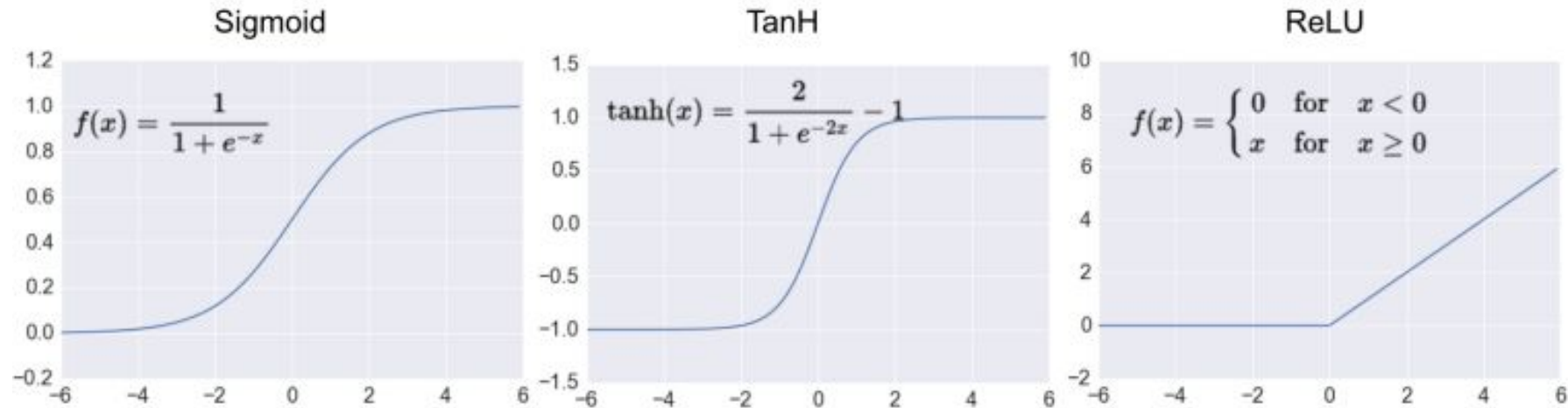
Questions?

Activation Functions



Why non-linear?

Activation Functions

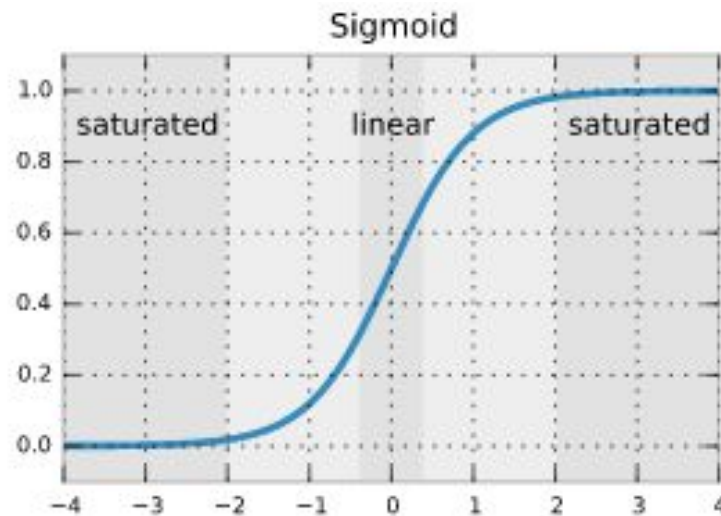


Why non-linear? Creates non-linear decision boundaries!

Activation Functions: **Sigmoid**

$$S(x) = \frac{1}{1 + e^{-x}}$$

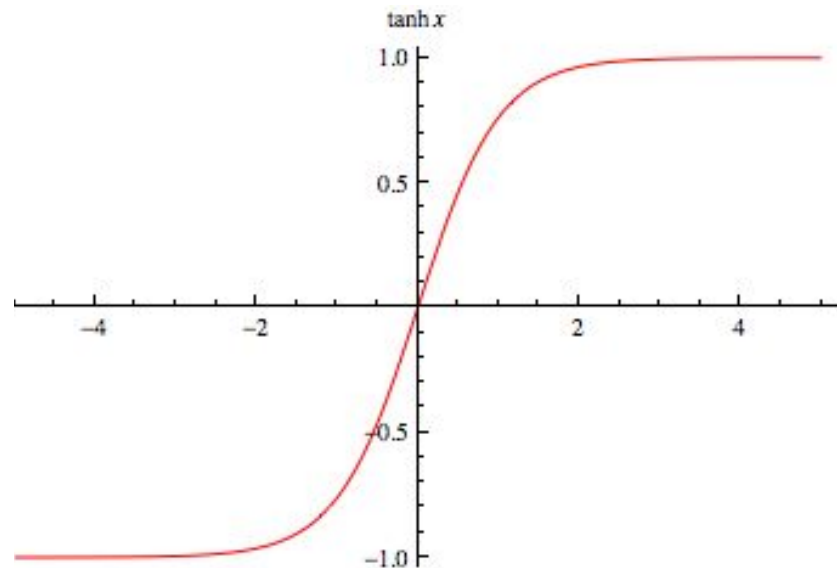
- Range: (0, 1)
- Numbers in the linear regime are good
- Saturation can kill gradients
- Not zero-centered
- $\exp()$ can be expensive



Activation Functions: **tanh**

- Range: $(-1, 1)$
- Scaled sigmoid
- Now zero-centered!
- Saturation still kills gradients...

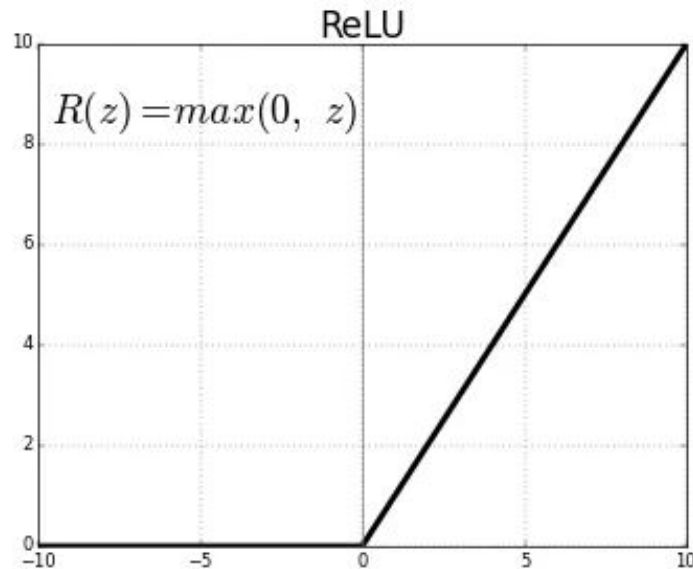
$$\begin{aligned}\tanh x &= \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \\ &= \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{1 - e^{-2x}}{1 + e^{-2x}}.\end{aligned}$$



Activation Functions: **ReLU**

$$f(x) = x^+ = \max(0, x)$$

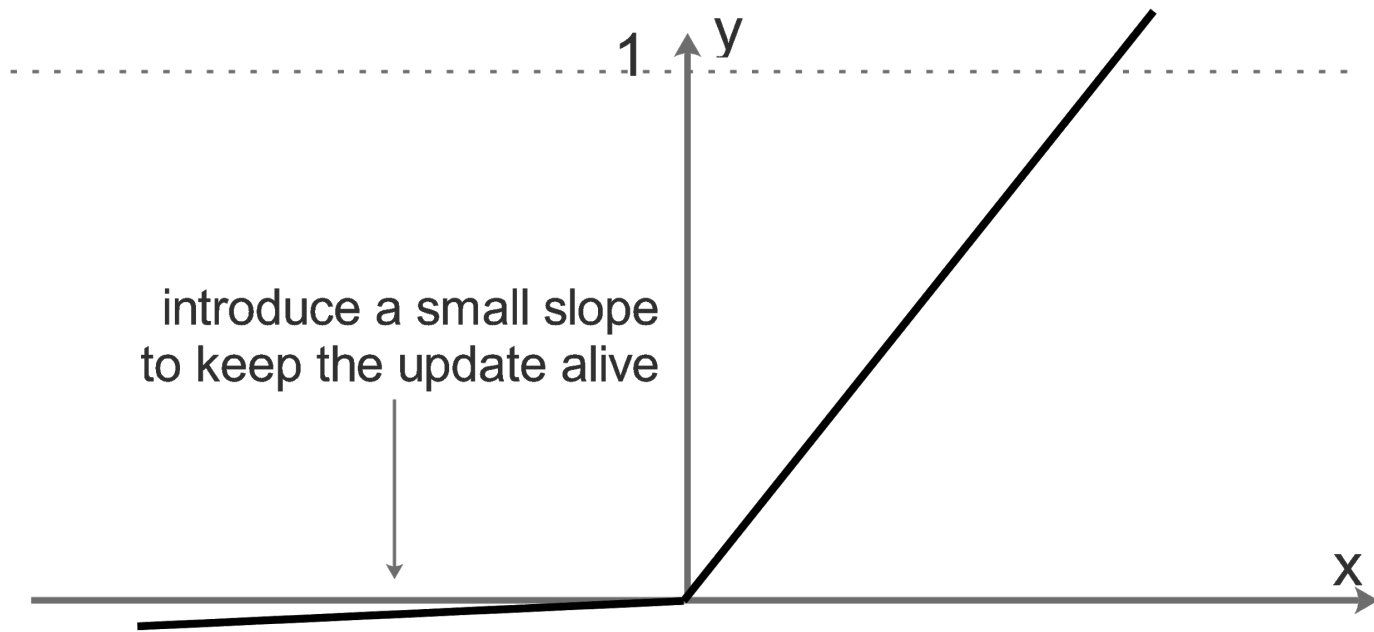
- ReLU = **R**ectified **L**inear **U**nit
- Range: $(0, +\infty)$
- Does not saturate in positive region
- Very efficient + converges fast
- Not zero-centered
- Dying ReLU



The Dying ReLU Problem

- Always outputs the same value (can't discriminate between inputs)
- Unlikely to recover once dead
- Why?
 - a. Poor initialization or high learning rate

Activation Functions: **Leaky ReLU** $f(x) = \max(0.01x, x)$



Questions?

Activation Functions: **Softmax**

- Used in the final layer of network
- Typically networks trained to optimize cross-entropy loss
- For multi-class classification
- Interpretable as probabilities

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^\top \mathbf{w}_k}}$$

Which one do I use?

Activation Functions in Practice

- Use ReLU. Be careful with your learning rate!
- Try Leaky ReLU and tanh.
- Don't use Sigmoid.*

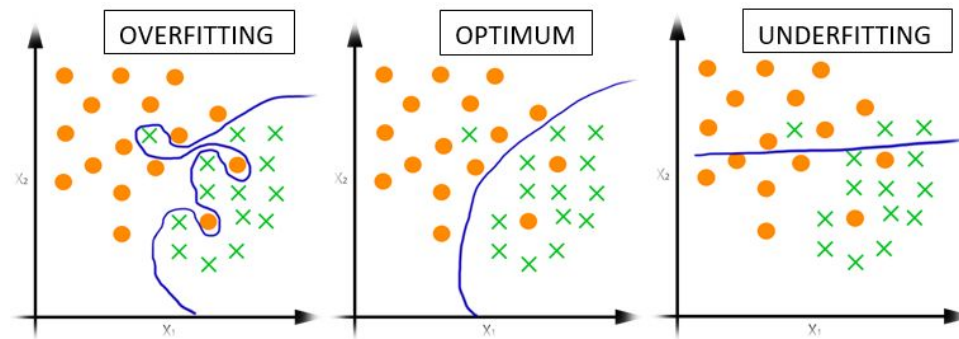
*between hidden layers; okay for output layer

Questions?

Jupyter Exercises 2-4: Baseline NN in Keras

Overfitting

- Performance gap between train and test
 - a. Difference in accuracy
 - b. Model keeps getting better on train, but plateaus or gets worse on validation set



Regularization: **L2 Weight Decay**

- Artificially discourages complex relationships in data even if they fit training set well. The idea is that such relationships are unlikely to generalize well to the future (e.g. noise).
- Penalize loss function by adding a multiple of the L2 norm of your weight vector w . You get the following equation:

$$L(X, Y) + \lambda N(w)$$

Jupyter Exercise 5: L2 Regularization

Regularization: **Dropout**

- Regularization technique where randomly selected neurons are ignored during training.
- Contributions during forward/backward pass are ignored.
- Network becomes less sensitive to the specific weights of neurons → better generalization and less overfitting.

Jupyter Exercise 6: Dropout

Preview of Homework

- You might not always see huge improvements from regularization or dropout
 - Maybe there's no overfitting...
 - Or maybe our regularization and dropout strengths are not optimal.
- You can determine the best strengths and other hyperparameters using a technique called cross-validation.

Batch Normalization

- Used to help remedy “covariate shifts”, or changes in distribution of function’s domain.
- Keep activations in the zero-mean unit-variance range.

Jupyter Exercise 7: Batch Normalization

Summary of Today

- Able to recreate COMPAS recidivism scores around 75% of the time, with a fairly simple deep neural network.
- Learned about activation functions, overfitting, regularization, dropout, and batch normalization.
- Learned how to use Keras.

Questions?