Using Neural Networks to Classify Cell Characteristics

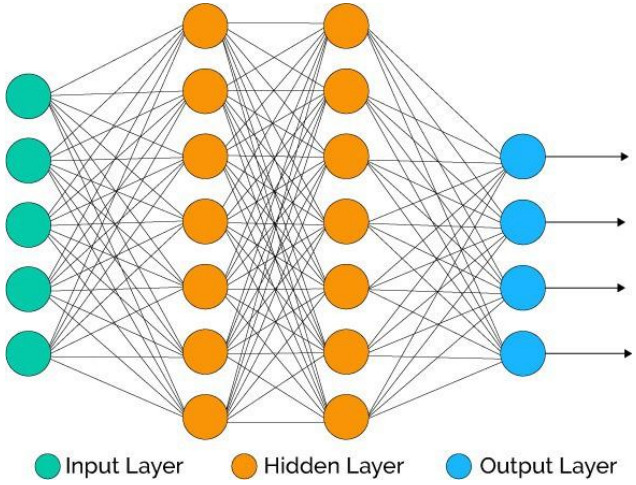
T.J. Sears
Feb 18, 2020

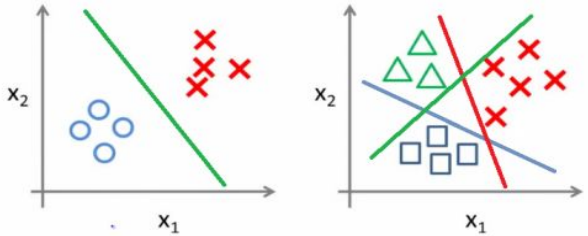
Butcher Lab

- I worked in an Immunology Lab under Dr. Butcher at Stanford
- Mostly worked on using deep learning to make educated guesses about cell characteristics
  - Guesses were based on RNA seq data
  - Main predictive tool used was Neural Networks (NN)
- I learned a lot about programming, data manipulation, and artificial intelligence
- Did some wet lab for the tired postdocs too if they asked really nicely

Machine Learning with NN's

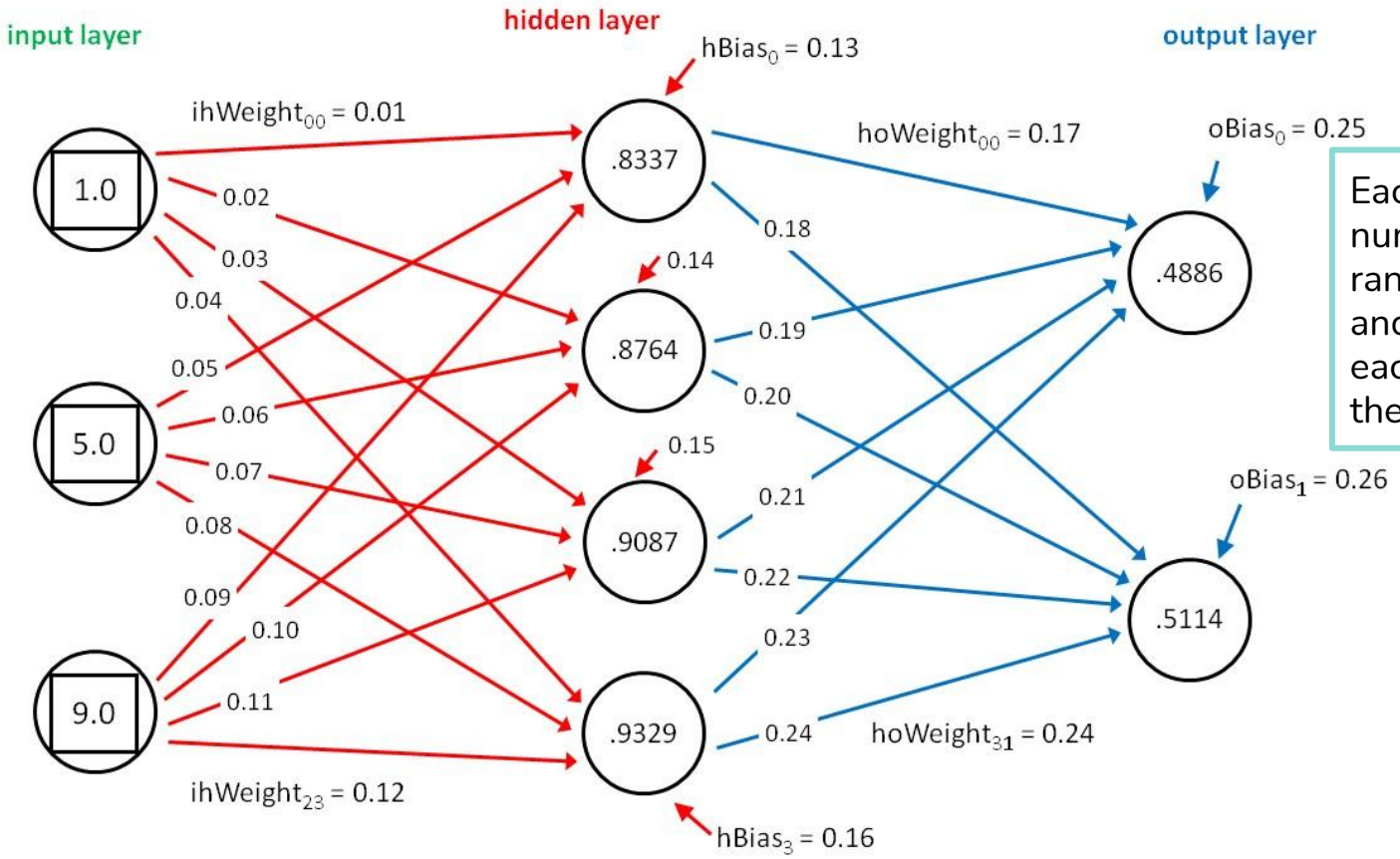
- Simplest form of a NN is a **multilayer perceptron** (MLP), which is usually applied to **supervised learning classification** tasks
- You input a **feature vector** & specify number of hidden layers and neurons per layer
- Training the MLP involves adjusting its parameters to minimize the model's error
    - Minimizing error can be thought of as using math to fit a line to data points





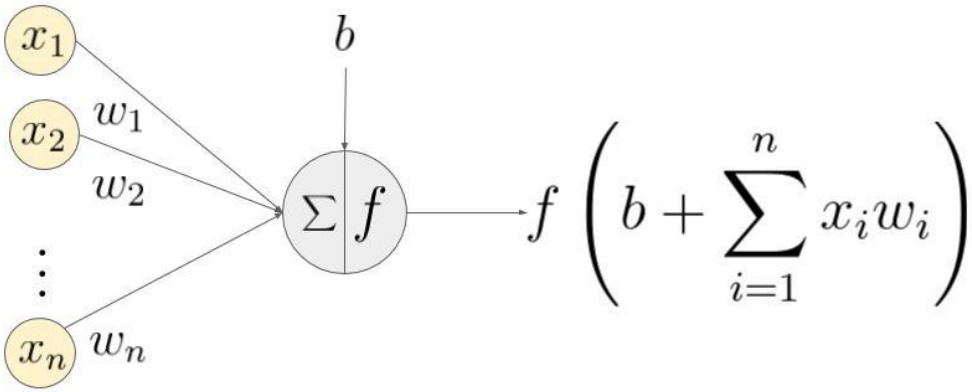
Binary vs. Multiclass Classification

The SIMPLIFIED Mathematics of NN's



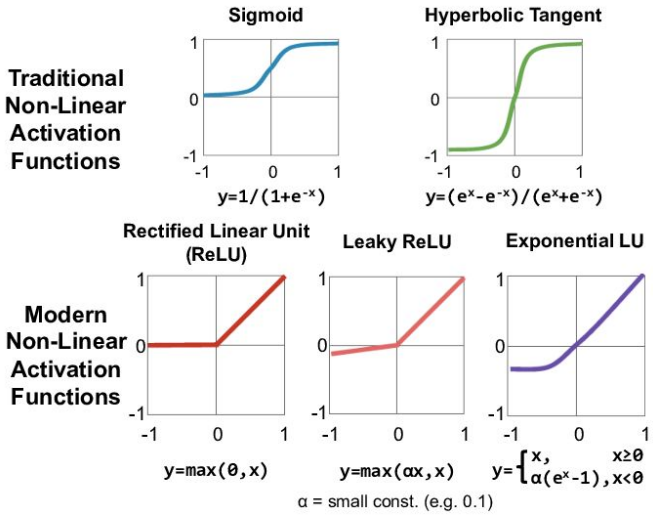
Each of these numbers are randomly initialized and altered with each Iteration of the NN

The SIMPLIFIED Mathematics of NN's



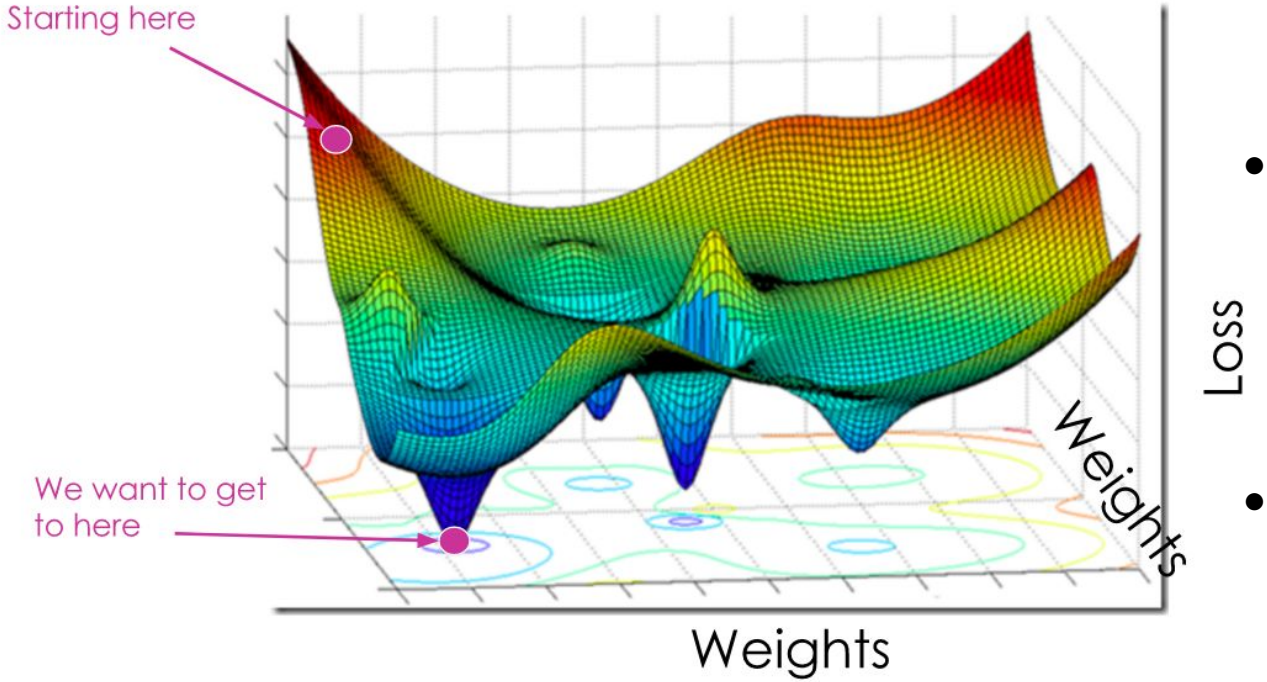
An example of a neuron showing the input (x_1 - x_n), their corresponding weights (w_1 - w_n), a bias (b) and the activation function f applied to the weighted sum of the inputs.

Each **input** from the previous page is added to one of many activation functions, then the **output** value is passed forward to the next node.



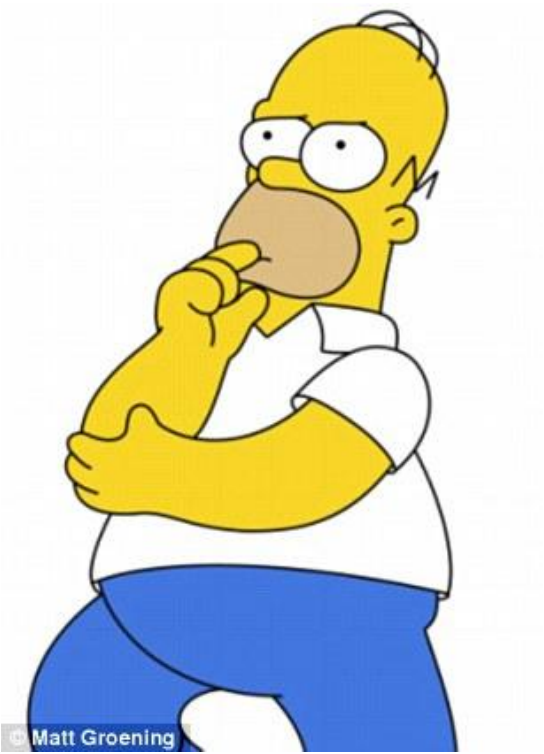
When working with RNA seq data, the n value can be up to 30,000. This creates massive vectors and matrices that only modern computers can deal with.

Loss Functions and High Dimensional Data



- This is a 3D depiction of a “feature map” with randomly initialized starting weights
- Basically, we start at a random point in the map and use **Gradient Descent:** a technique using Differential Calculus to find the closest local minimum
- Each datapoint (cell) that goes through the net should decrease the loss function towards a local minimum

Feeling Confused?



Just think of a NN as a fancy Black Box, what's important for this presentation is the results!

Classification Tasks
(classifying scRNA-seq datasets)

- Classify blood endothelial cells as dividing or resting
- Classify blood endothelial cells by their gender
- Classify blood endothelial cell type ~7ish types
- Make all of these classifiers **cross-tissue applicable***

*This is the hard part

Cell-Cycle Classification

Neural Network Architecture

- Developed in Python
- Used **scikit-learn** machine learning library
- Five layers, fifty nodes per layer
  - These structural decisions are somewhat arbitrary, I've just found that they work well

Trial 1

Train On:

70% of PLN1, PLN2, PLN3,
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

30% of PLN1, PLN2, PLN3
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

Oxa = Oxazolone, an inflammatory agent

Trial 1 Results

OVERALL ACCURACY: 98%
RESTING CELL ACCURACY: 99%
DIVIDING CELL ACCURACY: 83%

	ACTUALLY RESTING	ACTUALLY DIVIDING
PREDICTED RESTING	3932	74
PREDICTED DIVIDING	2	362

This is a special chart called a **Confusion Matrix**

Trial 2

Train On:

PLN1, PLN2, PLN3
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

Smoking and Non-Smoking Lung Dataset
(raw, log-normalized counts, no imputation)

Trial 2 Results

Predicted most lung cells as **dividing**
In reality, most lung cells were **resting**

Critical issue with cell-cycle neural network:
It was accurate when tested on lymph node datasets, but **didn't generalize accurately to new tissues** not included in the training dataset, such as this lung data

Gender Classification

Neural Network Architecture

- Developed in Python
- Used **scikit-learn** machine learning library
- Five layers, fifty nodes per layer

Trial 1

Train On:

Male and Female Cells: PLN1, PLN2, PLN3, PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

PYMT Breast Tumor Dataset
(raw counts, no imputation)

Results: Correctly predicted 13,000/13,163 PYMT cells as female

Trial 2

Train On:

Male and Female Cells: PLN1, PLN2, PLN3
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

Smoking and Non-Smoking Lung Dataset
(raw, log-normalized counts, no imputation)

Trial 2 Results

Predicted 50% of lung cells as **female**, 50% as **male**
In reality, all lung cells were **female**

Same issue with gender neural network:
It didn’t generalize accurately to new tissues not included in the training dataset, such as this lung data

How to Improve NN Architectures?

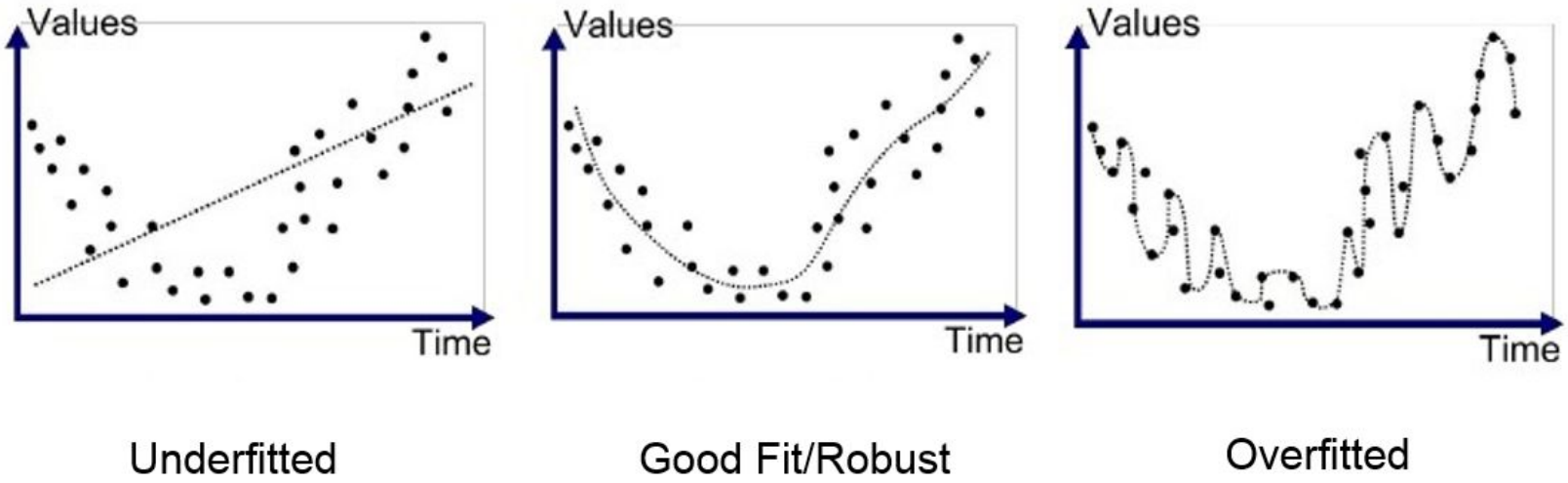
Dropout Regularization

- Solves issue of **overfitting**, when neural networks get too closely fit to their training sets, and thus do not generalize accurately (problem I was seeing)

- Dropout probabilistically “drops” a fraction of the nodes during training to avoid parameter overfitting

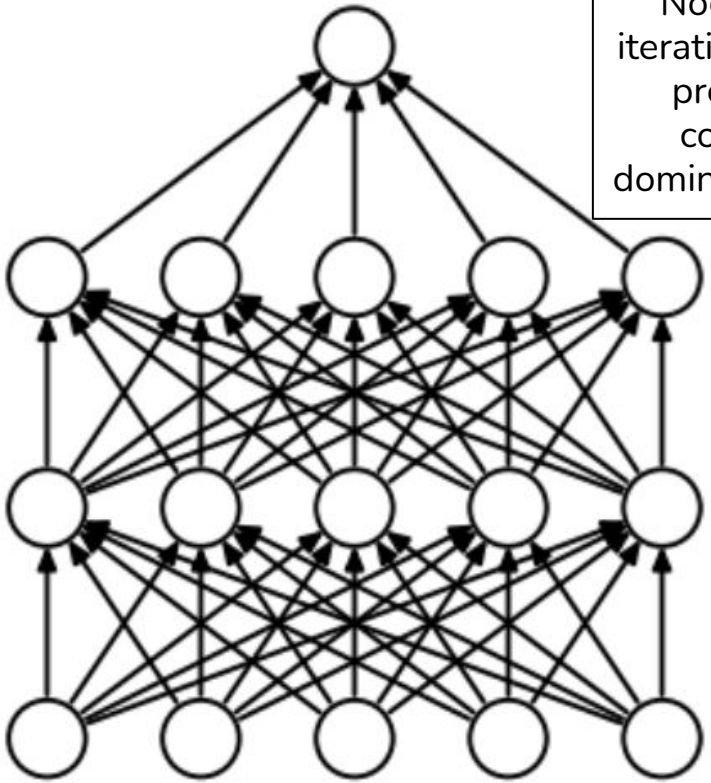
What Does Overfitting *Really* Mean?

- Overfitting = the model is trying too hard to capture the noise in the training dataset
- Noise = variation in the data due to random chance
- There is a lot of noise in RNA seq data due to the nature of the measurement—so overfitting is a real concern.

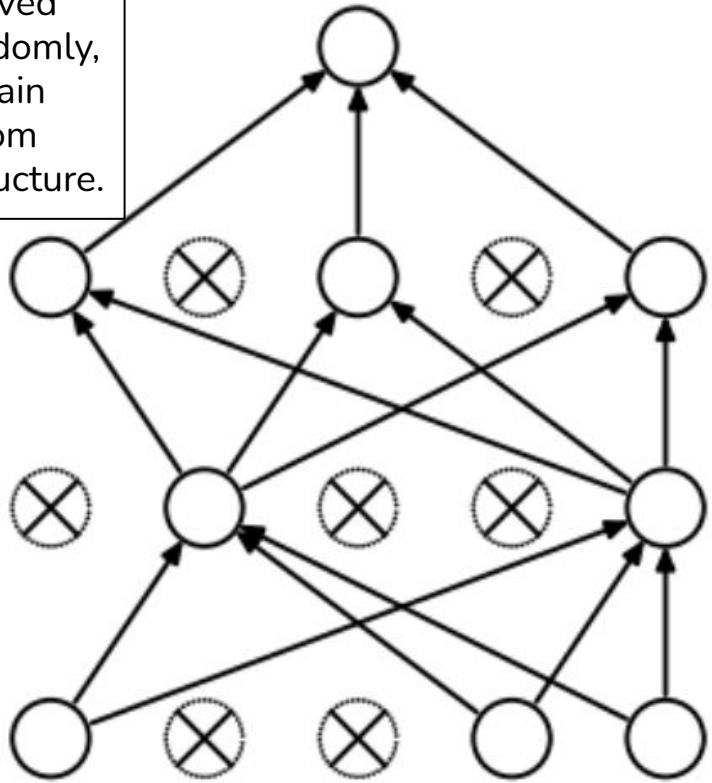


What Does Dropout *Really* Mean?

Nodes are removed iteratively and randomly, preventing certain connections from dominating the structure.



(a) Standard Neural Net



(b) After applying dropout.

New Architecture

- Developed in Python
- Used **Keras** library with **Tensorflow** backend
  - Backend = server software and hardware that the user doesn't deal with
  - Tensorflow (TF) is a powerful tool developed by google
- Four fully-connected, normal layers like before
- Add in two layers of Dropout regularization

Trial 3: Cell-Cycle

Train On:

PLN1, PLN2, PLN3
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

Smoking and Non-Smoking Lung Dataset
(raw, log-normalized counts, no imputation)

Results: Correctly predicted 385/391 lung cells as resting

Trial 3: Gender

Train On:

Male and Female Cells: PLN1, PLN2, PLN3
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

Smoking and Non-Smoking Lung Dataset
(raw, log-normalized counts, no imputation)

Results: Correctly predicted 391/391 lung cells as female

Trial 4: Cell-Cycle

Train On:

PLN1, PLN2, PLN3
PLN-Oxa12h, PLN-Oxa24h, PLN-Oxa48h
(raw, log-normalized counts, no imputation)

↓

Test On:

Heart Dataset
(raw counts, no imputation)

Results: Correctly predicted most heart cells as resting, with 10 cells (Top2a^{+} and Mki67^{+}) predicted correctly as dividing

Conclusions

- Adding Dropout to neural network architecture improved generalization capability across tissues

- When trained on lymph node dataset, neural networks for cell-cycle and gender classification could generalize accurately to lung and heart data

DeepLIFT Analysis of Neural Network Feature (Gene) Importance

What is DeepLIFT?

- DeepLIFT (Deep Learning Important FeaTures) allows us to look into the “Black Box” of machine learning
- We can take outputs and “retrace our steps” through the connections of the neural net to see which genes in the input vector had the largest contribution to the final classification decision

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017. [1]

DeepLIFT Applications

- DeepLIFT can allow us to observe connections between gene expression and classification outcomes
  - If, for example, an unexpected gene was found to have a major contribution to a cell's division classification, we could ask questions about how that gene's expression impacts cell division.
- Patterns that are not obvious to us *lowly* humans may be clear to the algorithm, and DeepLIFT allows us to observe the algorithm's "thought process"

DeepLIFT Results

Top 20 Genes for Dividing/Resting Cells

Gene Name
Stmn1
Hmgb2
Ube2c
Birc5
Pclaf
Crip1
Top2a
Cdk1
Cks2
Cdc20
Spc24
Rrm2
Hmgb1
Nusap1
Pbk

Top 20 Genes for Male/Female Cells

Gene Name
Xist
Glycam1
Ly6i
Serpina1e
Ubc
Malat1
Rnaset2b
B2m
Ms4a6b
Nxpe2
CYTB
Ftl1
Clec14a
Rps29
Rps28

The top 20 genes identified as most important to each of the neural networks made *some* biological sense.

Overall, the DeepLIFT analyses of the cell-cycle and gender neural networks were *mostly* biologically meaningful.

Quick Exploration of Top Three Genes for Each DeepLIFT Analysis

Cell Cycle Classification

1. **Stmn1**
   a. "...required for many cellular processes, such as cytoplasmic organization, cell division and cell motility" [5] "...crucial in regulating the cell cycle. [6]
2. **Hmgb2**
   a. "...suggest a role in facilitating cooperative interactions between cis-acting proteins by promoting DNA flexibility" [7]
3. **Ube2c**
   a. "...required for the destruction of mitotic cyclins and for cell cycle progression." [8]

Gender Classification

1. **Xist**
   a. "acts as a major effector of the X inactivation process" [2]
2. **Glycam1**
   a. "proteoglycan ligand expressed on cells of the high endothelial venules in lymphoid tissues" [3]
3. **Ly6i**
   a. lymphocyte antigen 6 complex [4]

Xist is obviously related to cell gender. Glycam1 and Ly6i are generally highly-expressed in lymph node tissues, but not related to gender.

TJ’s Super Awesome Classification Study (lots of graphs incoming)

Classification Architecture

```
# TrEC = 0
# HEC / HEC late = 1
# CapEC1/2 = 2
# CRP / CRP Early = 3
# HEV = 1
# Pre-Art = 5
# Vn = 4
# Art = 5
# CapIfn = 6
```

Datasets tested were PLN (Peripheral Lymph Node) and MLN (Mesenteric Lymph Node). These numbers represent some cell types in Lymph Tissues

- Developed in Python
- Used Keras libraries with TF backend
- 3 hidden layers, 7 total classes (cell types)

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 25)                776300
_________________________________________________________________
dense_1 (Dense)              (None, 10)                260
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 55
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 42
=================================================================
Total params: 776,657
Trainable params: 776,657
Non-trainable params: 0
_________________________________________________________________
```

```
Train on 6707 samples
Epoch 1/16
6707/6707 [==============================] - 27s 4ms/sample - loss: 1.1635 - accuracy: 0.5299
Epoch 2/16
6707/6707 [==============================] - 28s 4ms/sample - loss: 0.5317 - accuracy: 0.8142
Epoch 3/16
6707/6707 [==============================] - 27s 4ms/sample - loss: 0.3756 - accuracy: 0.8528
Epoch 4/16
6707/6707 [==============================] - 31s 5ms/sample - loss: 0.2712 - accuracy: 0.8742
Epoch 5/16
6707/6707 [==============================] - 36s 5ms/sample - loss: 0.2026 - accuracy: 0.8988
Epoch 6/16
6707/6707 [==============================] - 36s 5ms/sample - loss: 0.1243 - accuracy: 0.9578
Epoch 7/16
6707/6707 [==============================] - 31s 5ms/sample - loss: 0.0967 - accuracy: 0.9678
Epoch 8/16
6707/6707 [==============================] - 34s 5ms/sample - loss: 0.0817 - accuracy: 0.9708
Epoch 9/16
6707/6707 [==============================] - 32s 5ms/sample - loss: 0.0617 - accuracy: 0.9790
Epoch 10/16
6707/6707 [==============================] - 32s 5ms/sample - loss: 0.0581 - accuracy: 0.9812
Epoch 11/16
6707/6707 [==============================] - 28s 4ms/sample - loss: 0.0448 - accuracy: 0.9846
Epoch 12/16
6707/6707 [==============================] - 30s 4ms/sample - loss: 0.0381 - accuracy: 0.9861
Epoch 13/16
6707/6707 [==============================] - 36s 5ms/sample - loss: 0.0393 - accuracy: 0.9866
Epoch 14/16
6707/6707 [==============================] - 33s 5ms/sample - loss: 0.0310 - accuracy: 0.9885
Epoch 15/16
6707/6707 [==============================] - 40s 6ms/sample - loss: 0.0196 - accuracy: 0.9951
Epoch 16/16
6707/6707 [==============================] - 38s 6ms/sample - loss: 0.0286 - accuracy: 0.9914
```

Epoch = one iteration of weight adjustment for the entire dataset

Testing on Training Set

Train On: **PLN1, PLN2, PLN3**
(Imputed with MAGIC t2, log-normalized counts)

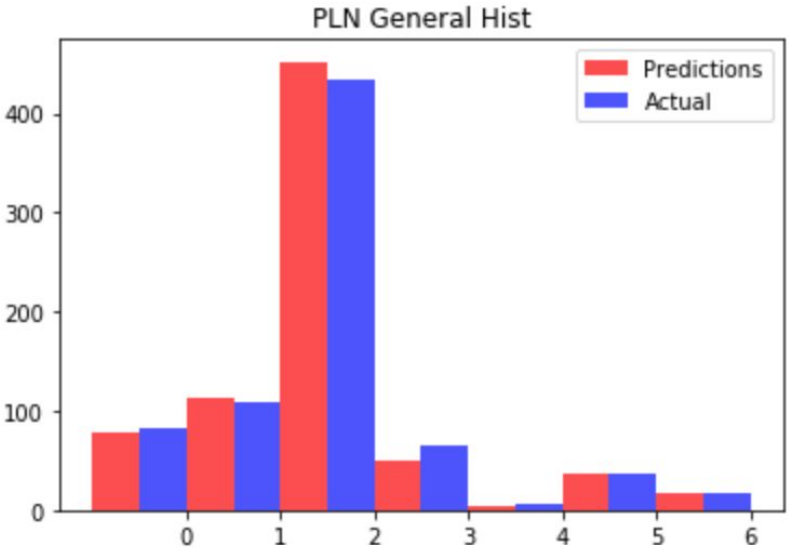
Test On: **PLN1, PLN2, PLN3**
(Imputed with MAGIC t2, log-normalized counts)

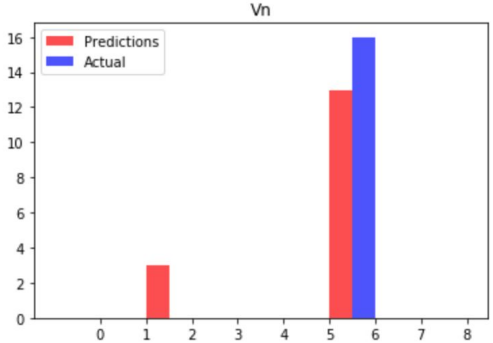
```python
# Evaluate model accuracy on test data
predictions = classifier.predict_classes(testX)
preds = predictions.tolist()

preds, testY
from sklearn.metrics import accuracy_score
print(accuracy_score(preds, testY)*100)
pd.DataFrame(preds).to_csv('/Users/tjshruti/Downloads/PLN123_predictions_sept13.csv')
pd.DataFrame(testY).to_csv('/Users/tjshruti/Downloads/PLN123_actual_sept13.csv')
```

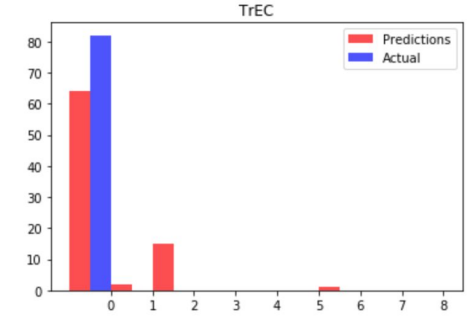
```
92.62734584450402
```

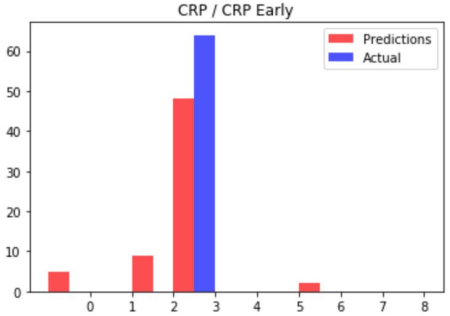
93% Accurate

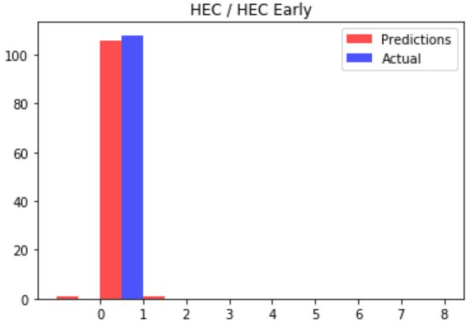
PLN Test Results

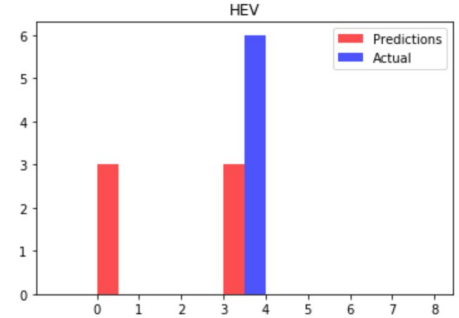


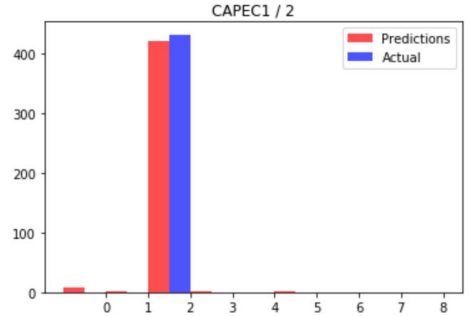


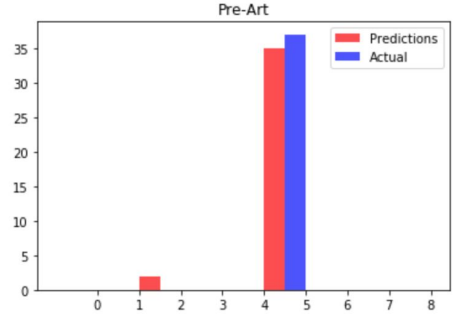












Testing on Test Set

Train On:

PLN1, PLN2, PLN3
(Imputed MAGIC t2, log-normalized counts)

Test On:

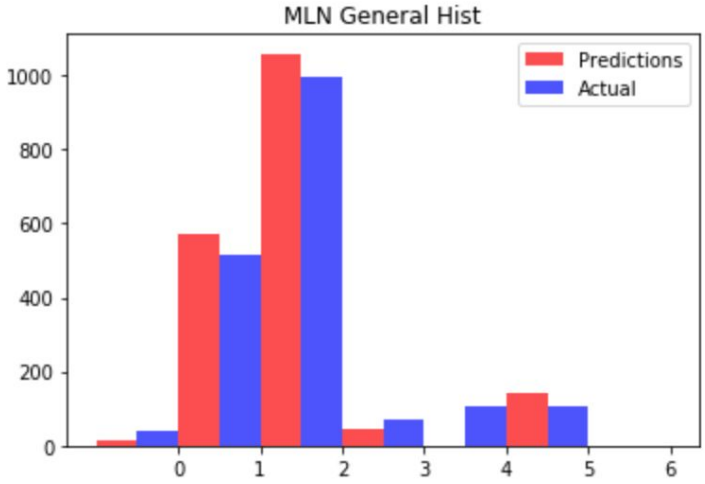
MLN1
(Imputed MAGIC t2, log-normalized counts)

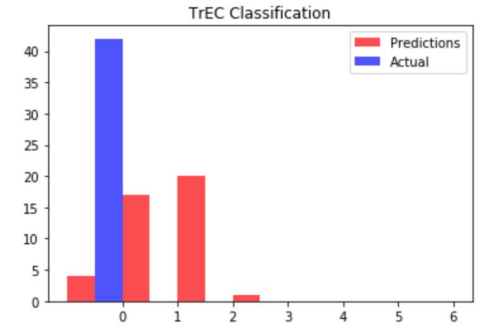
```
#Accuracy Score on MLN Tissue
from sklearn.metrics import accuracy_score
print(accuracy_score(predsMLN, Y_ArrMLN1)*100)
pd.DataFrame(predsMLN).to_csv('/Users/tjshruti/Downloads/MLN1_predictions_sept13.csv')
pd.DataFrame(Y_ArrMLN1).to_csv('/Users/tjshruti/Downloads/MLN1_actual_sept13.csv')
```

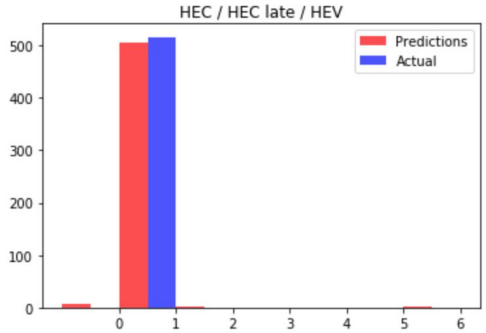
```
88.379705400982
```

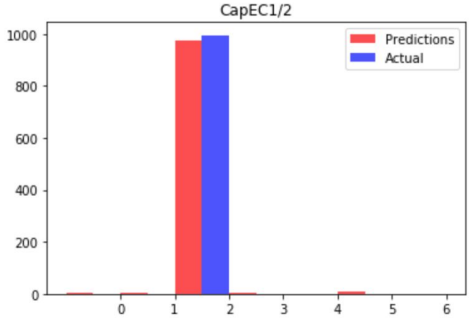
88% Accurate

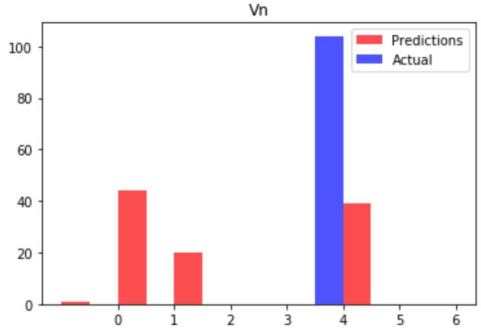
MLN Test Results

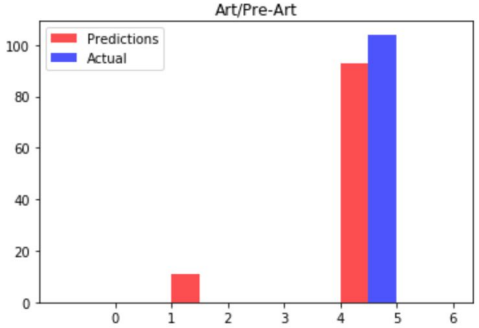






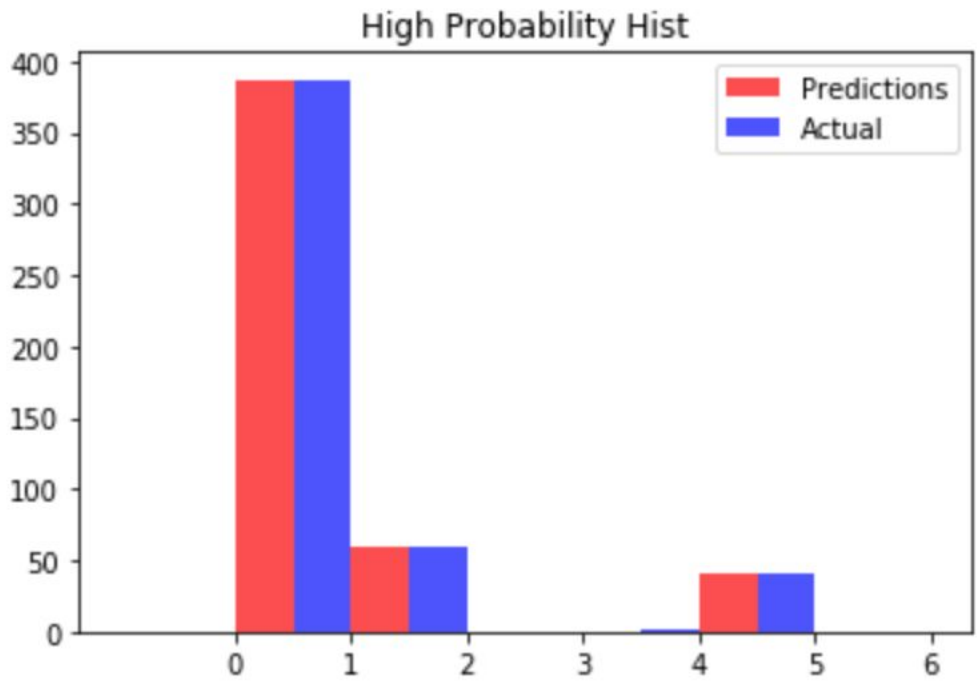






MLN Dataset had slightly different classes, so category 6 is missing.

MLN High Probability Results



- When prediction confidence had to meet a certain threshold (80%), prediction accuracy greatly increased to 98%
- This likely means that the model is tuned to the presence of specific genes, and that when they are present, prediction is confident
- Only categories 1, 2, and 5, were consistently in the high probability category--meaning that some cell types are easier to predict than others

Future Goals

- Compare different amounts of Dropout layers
- Develop model on larger datasets and test on even more tissue types
- Use DeepLIFT to explore patterns within and contributions to NN classification decisions
    - Find out which features contribute to High-Probability cells
- Explore impacts of imputation algorithms on input data on classification accuracy (MAGIC, scALIGN, etc.)

Sources

1. Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017.
2. Chow JC, Yen Z, Ziesche SM, Brown CJ (2005). "Silencing of the mammalian X chromosome". *Annual Review of Genomics and Human Genetics*.
3. Imhof, Beat A.; Dunon, Dominique (1995). *Leukocyte Migration and Adhesion*. Advances in Immunology.
4. https://www.ncbi.nlm.nih.gov/gene/?term=Ly6i
5. Kueh HY, Mitchison TJ (August 2009). "Structural plasticity in actin and tubulin polymer dynamics". *Science*. **325** (5943): 960–3.
6. Rubin CI, Atweh GF (October 2004). "The role of stathmin in the regulation of the cell cycle". *Journal of Cellular Biochemistry*. **93** (2): 242–50. doi:10.1002/jcb.20187. PMID 15368352.
7. "Entrez Gene: HMGB2 high-mobility group box 2"
8. "Entrez Gene: UBE2C ubiquitin-conjugating enzyme E2C"
9. https://scikit-learn.org/stable/

Thank you
for listening!

When your network seems to be overfitting..


Dropout
Deep Learning Engineers
Successful Company Founders


Data Scientists
Fashion Agencies
Training Models
@debo