Machine Learning pt. 2: Supervised

ACE 592 SAE

Supervised Machine Learning

The other tool kit in ML is for when you have ***labeled data***.

By far most of the applications focus on ***minimizing prediction error of labels using features***.

In this lecture, we will talk about doing both *prediction* and *feature selection* with supervised ML.

Two Algorithms We Will Go Over

1. *Least Absolute Shrinkage and Selection Operator (LASSO)*

   - For predicting a target and learning its most important predictors.

2. *Random Forest*

   - For doing out of sample prediction or classification.

Penalized Regression

In an OLS regression, we choose coefficient by solving this optimization problem:

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2\right\}$$

One problem that might arise if ***overfitting***: use of too many covariates which will fit the data too specifically.

Penalized Regression

One way to accomplish this is by “penalizing” the objective function for too many betas:

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2\right\} \text{ subject to } \sum_{j=1}^{p}|\beta_j|\leq t.$$

So now you can see that the more coefficients we can set to zero, the better we will satisfy this constraint.

Penalized Regression

Another way we can write it:

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{N} ||y - X\beta||_2^2 + \lambda ||\beta||_1 \right\}$$

Which makes it clear that lambda = 0 is just OLS. The parameter lambda is our main ***hyperparameter***.

Penalized Regression

Clearly we used the L1 norm for the penalty, but we didn't have to. We could have used L2:

$$\frac{1}{n}||Y - \mathrm{X}\,w||_2^2 + \lambda \sum_{j=1}^{d} |w_j|^2$$

This is the ***ridge regression estimator.***

How will the regression change now? ***Will it set more or less coefficients to zero compared to LASSO?***

Penalized Regression

It will set ***less coefficients to zero***. If we wanted to split the difference, we can add a hyperparameter to get an ***elastic net:***

$$\frac{1}{n}\|Y - \mathrm{X}\,w\|_2^2 + \lambda_1 \sum_{j=1}^{d} |w_j| + \lambda_2 \sum_{j=1}^{d} |w_j|^2$$

So now we have to choose both penalties, giving us ***two hyperparameters.***

Why might this be useful?

Limitations of Penalized Models

- One major limitation of LASSO is that it ***views all correlated variables as interchangeable***. This means it may arbitrarily leave a variable out, even if the variable is meaningful for the target.

- In ridge regression, it will ***hardly ever zero out coefficients***, meaning its not effective as a dimension reduction technique.

- An ***elastic net is a way to split the difference,*** so it won’t trip up on correlated variables. At the same time, it means training another parameter.

Example: LASSO for Prediction

LASSO and penalized regression has some desirable characteristics for prediction:

- Relatively simple and low cost to estimate (just OLS + a penalty).
- Prevents overfitting via choice of a good penalty term, which can be chosen using cross validation.

Example: LASSO for Variable Selection

LASSO and Elastic Nets can be a good tool to get an idea of variable selection:

- LASSO will zero out coefficients, leaving you with a reduced number of variables that actually help predict the target. These can then be included in another model as controls.

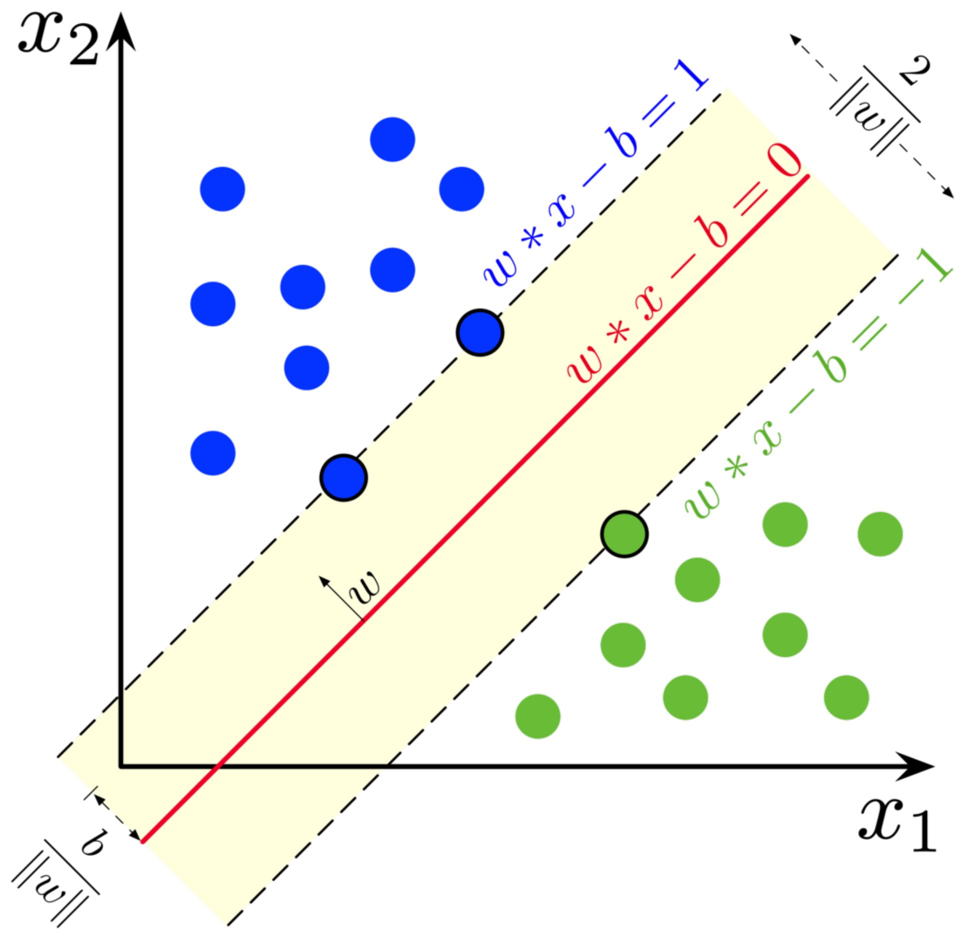
- The correlation problem can be adjusted by either 1) randomly dropping features from your list or 2) using an Elastic Net.

Fun Fact: Penalized Regression and SVM

Another type of SML model is **Support Vector Machines**, which essentially draws hyper planes to try and classify observations.

In the example on the left, blue and green dots have two difference labels.

Can be extended to non-linear classification with kernels.



Fun Fact: Penalized Regression and SVM

For classification, it tries to minimize the following objective function:

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b)\right)\right] + \lambda\|\mathbf{w}\|^2$$

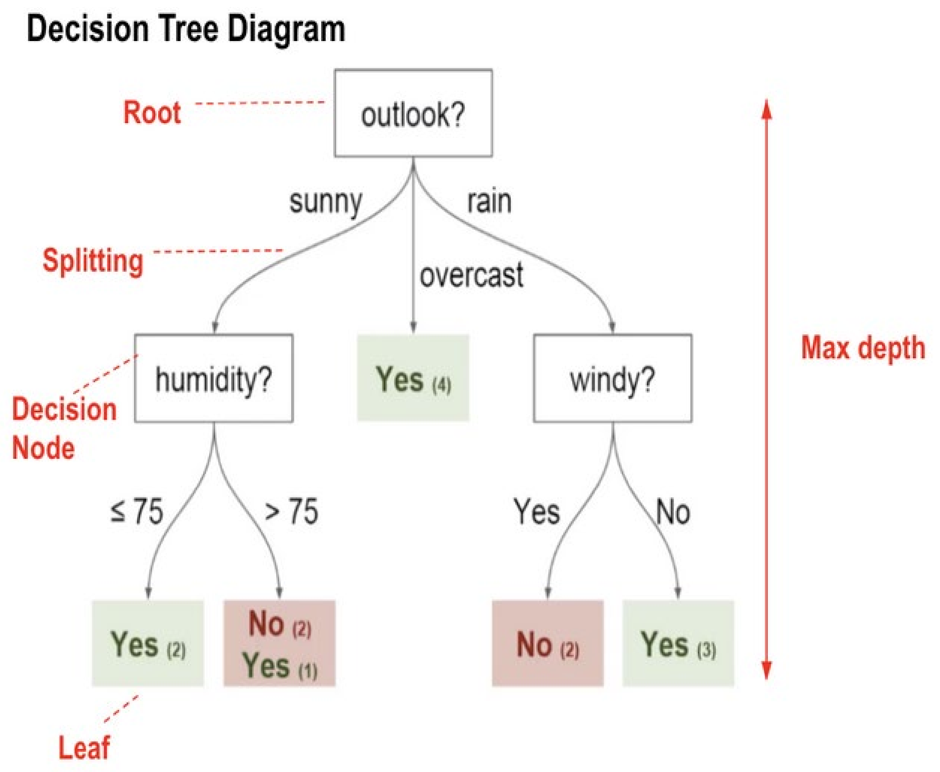
Look familiar?

In this case, the lambda hyper parameter controls the margin around the hyperplane.

LASSO Example

Random Forest Algorithm

- A popular SML algorithm that does both classification (discrete y) and regression (continuous y).
- Fast, lightweight, and effective. This makes it a popular choice for prediction.
- Deep learning tends to be best for big datasets.



A Review of the Algorithm:

1. Do a split on a certain variable in the data.
2. Examine how consistent the leaves are, usually by taking an expectation.
3. Determine whether the split gave you any ***information gain***.
4. Repeat 1-3 until some condition is met.

The Hyperparameters

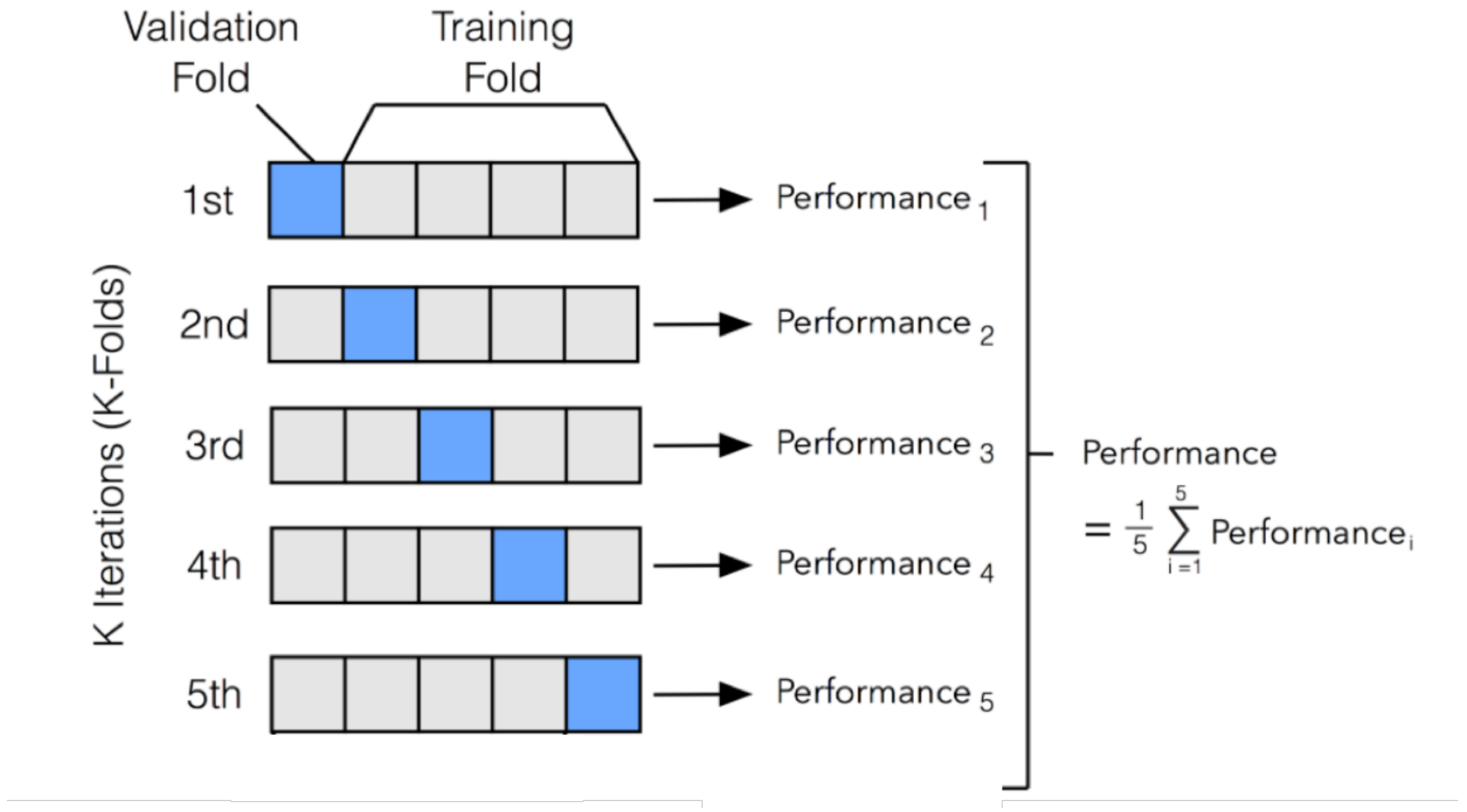
The “condition” it must meet is exactly the hyperparameter that we need to choose.

The downside of this algorithm is that there are more hyperparameters than usual:

- Number of trees.
- Max depth (how far a branch can go).
- Minimum leaf sample (how many obs must be on a leaf).

… and many others.

Reminder of How Cross Validation Works



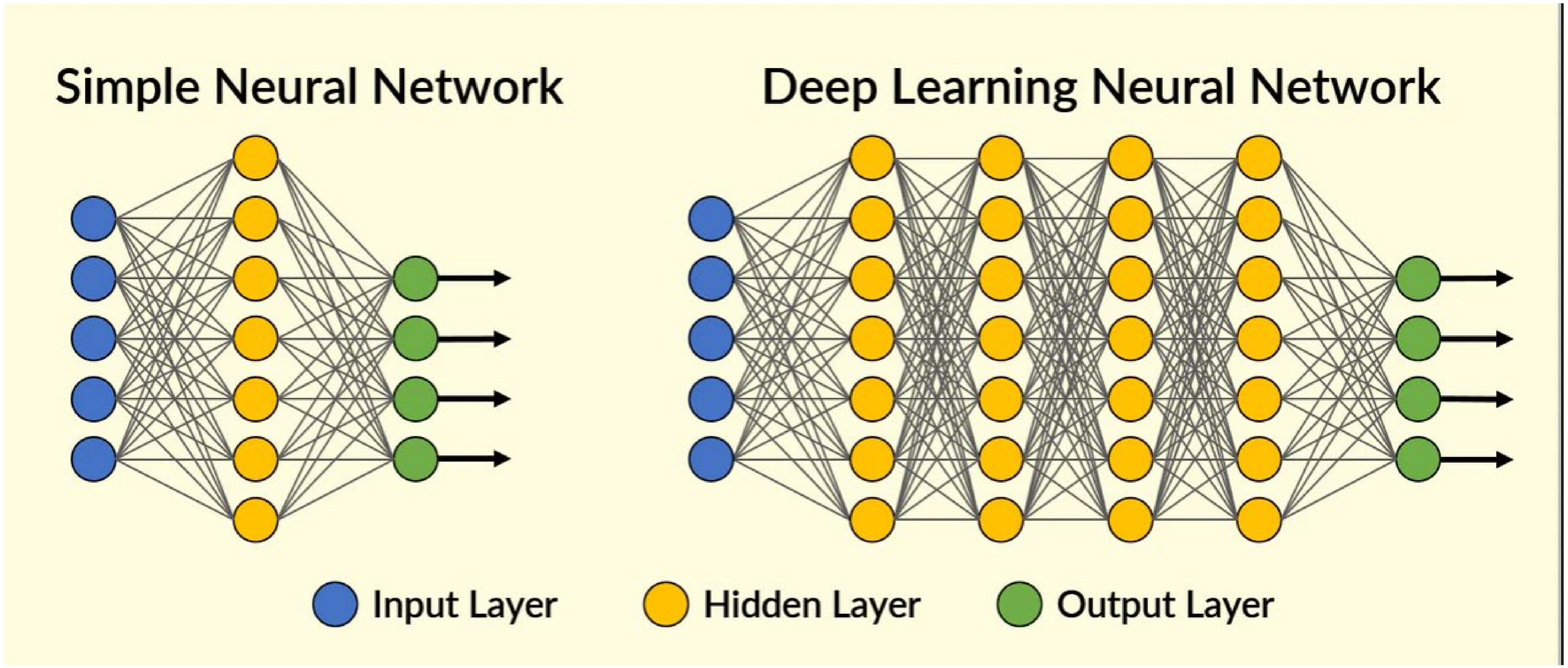
Tuning Hyperparameters

An ordinary work flow:

1. Choose a grid space over which to find parameters.
2. Do K-fold Cross Validation using your data to estimate performance of a set of parameters.
3. Choose the set that maximizes performance.

Step 2 can take a very long time, which is why we want to parallelize!

Deep Learning



Deep Learning

- Deep learning is one of the most widely used types of SML.
- Deep learning methods are constructed of ***artificial neural networks***, usually just called ***neural networks (NN).***
- Called “neural” because the algorithm is made up of connected “nodes” that are meant to model a biological brain.

How they “sorta” work:

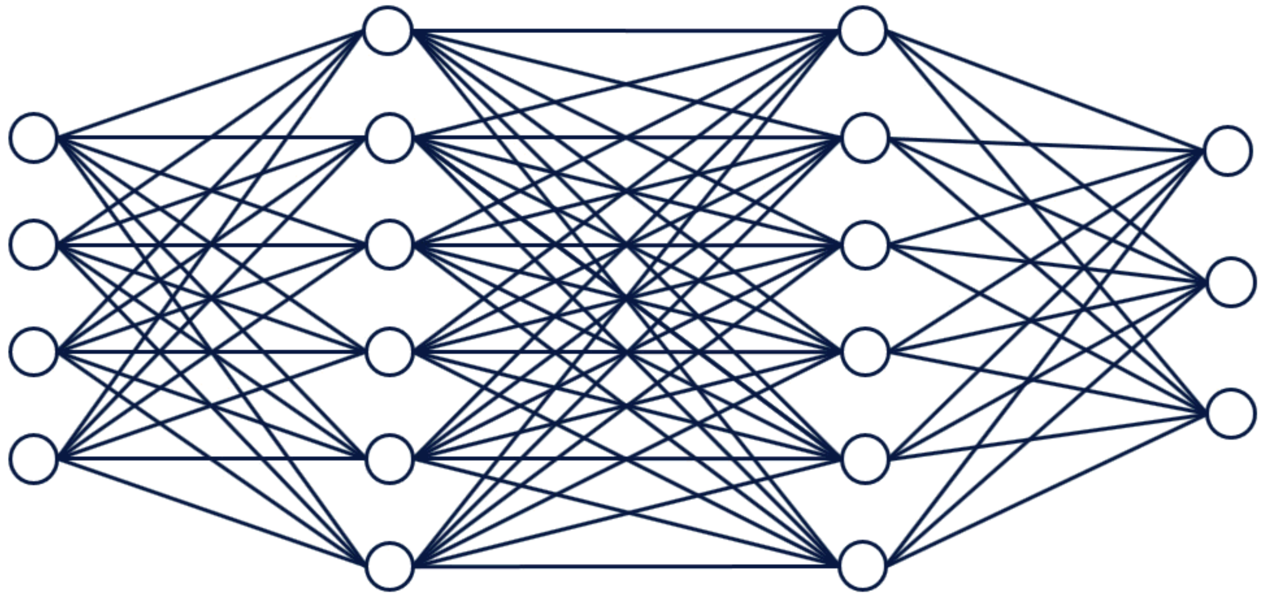
1. There are ***input layers*** that take in the features of the data.

2. The input layers aggregate the features by weighting them, and then running it through an “*activation function*” (e.g. logit function).

3. They then pass their output to ***the hidden layer if they are activated*** (meet a threshold). The process repeats.

4. Eventually, the ***hidden layer*** passes to the ***output layer***, which we compare against the true output.

An Example of Classification (2 hidden layers)



How they “sorta” work:

In essence, the training task is to figure out which “pathways” actually help you predict the target.

- If the output of a neuron helps increase accuracy, ***weight it more***.
- If the output of a neuron decreases accuracy, ***weight it less.***

If we can choose these “weights” to maximize accuracy, we can crudely do what an ***actual brain does.***

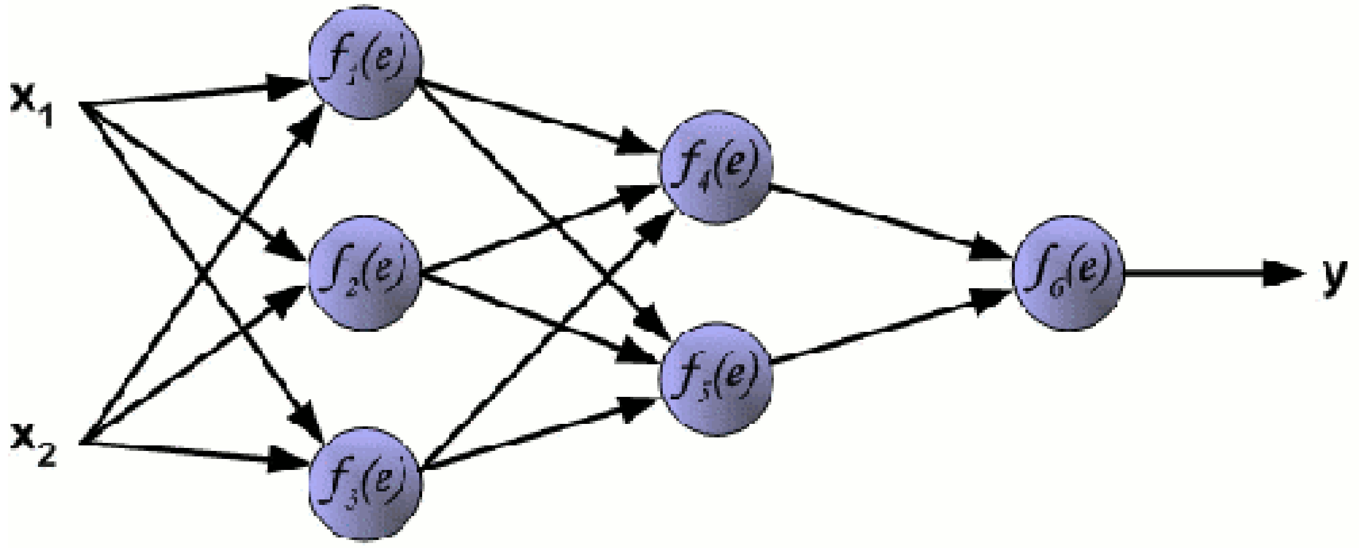
How should we pick pathways?

The most widely used algorithm is ***backpropogation***.

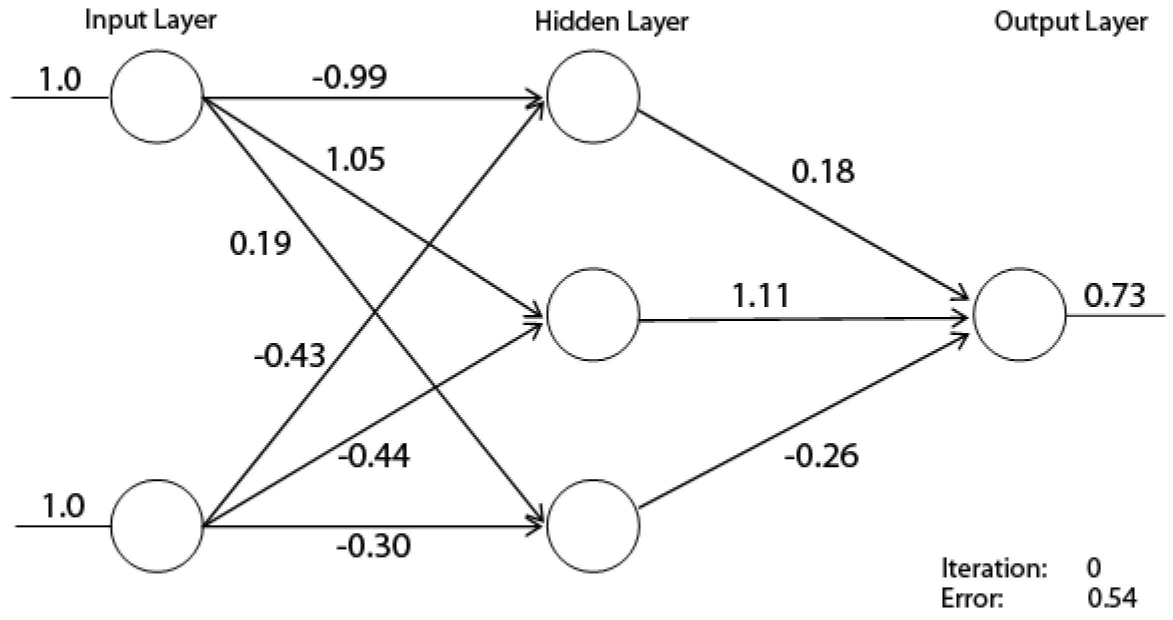
Using gradient descent, it goes ***backwards through the network*** to calculate the best weights.

It picks the ***all the weights*** that minimize the prediction error of that observation.

One Type of Weight Calculation



Another GIF


Input Layer
Hidden Layer
Output Layer
1.0
1.0
-0.99
1.05
0.19
-0.43
-0.44
-0.30
0.18
1.11
-0.26
0.73
Iteration: 0
Error: 0.54

The Power of Deep Learning

- The hyperparameters are the number of neurons, the number of hidden layers, the activation function, and many more.

- The advantage of this approach is that ***it scales incredibly well***.

- The disadvantage is that ***it is computationally costly.***

Python Packages

TensorFlow

Developed by Google

Keras

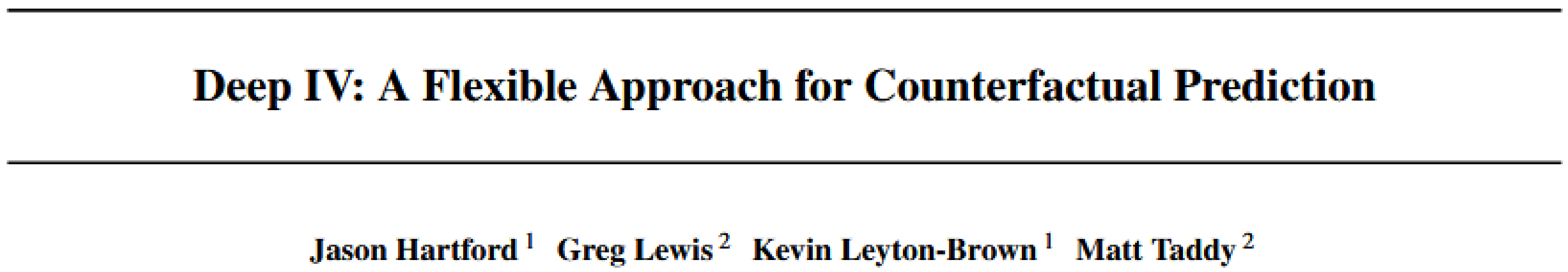
Simple. Flexible. Powerful.

Developed by Google

PyTorch

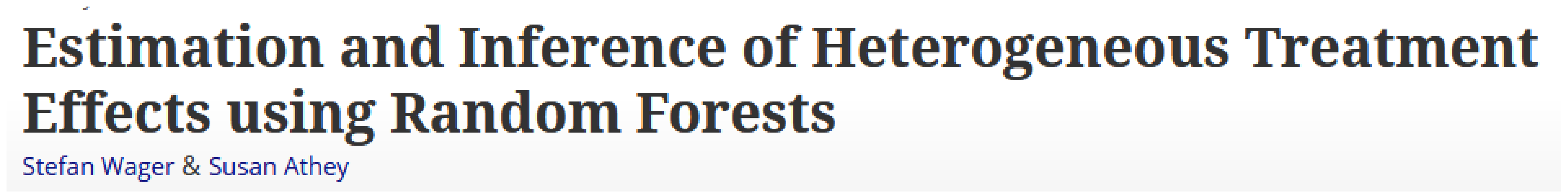
Developed by Facebook

Applications of SML in Econometrics



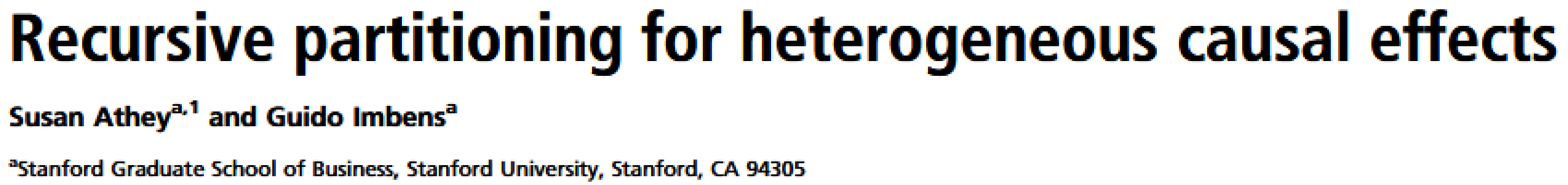
Deep IV: A Flexible Approach for Counterfactual Prediction

Jason Hartford [1] Greg Lewis [2] Kevin Leyton-Brown [1] Matt Taddy [2]



Estimation and Inference of Heterogeneous Treatment Effects using Random Forests

Stefan Wager & Susan Athey



Recursive partitioning for heterogeneous causal effects

Susan Athey[a,1] and Guido Imbens[a]

[a]Stanford Graduate School of Business, Stanford University, Stanford, CA 94305