Machine Learning for Public Policy
Week 4: Prediction in a New Population

Brandon Stewart

Princeton

February 17–19, 2025

Where We've Been and Where We're Going...

Where We've Been and Where We're Going...

- Last Week
  - preview of three types of prediction
  - prediction within population
  - logistic regression, GAMs
  - tree models

Where We've Been and Where We're Going...

- Last Week
  - preview of three types of prediction
  - prediction within population
  - logistic regression, GAMs
  - tree models
- This Week
  - proposal discussion
  - prediction in a new population
  - random forests and boosting
  - neural networks

Where We've Been and Where We're Going...

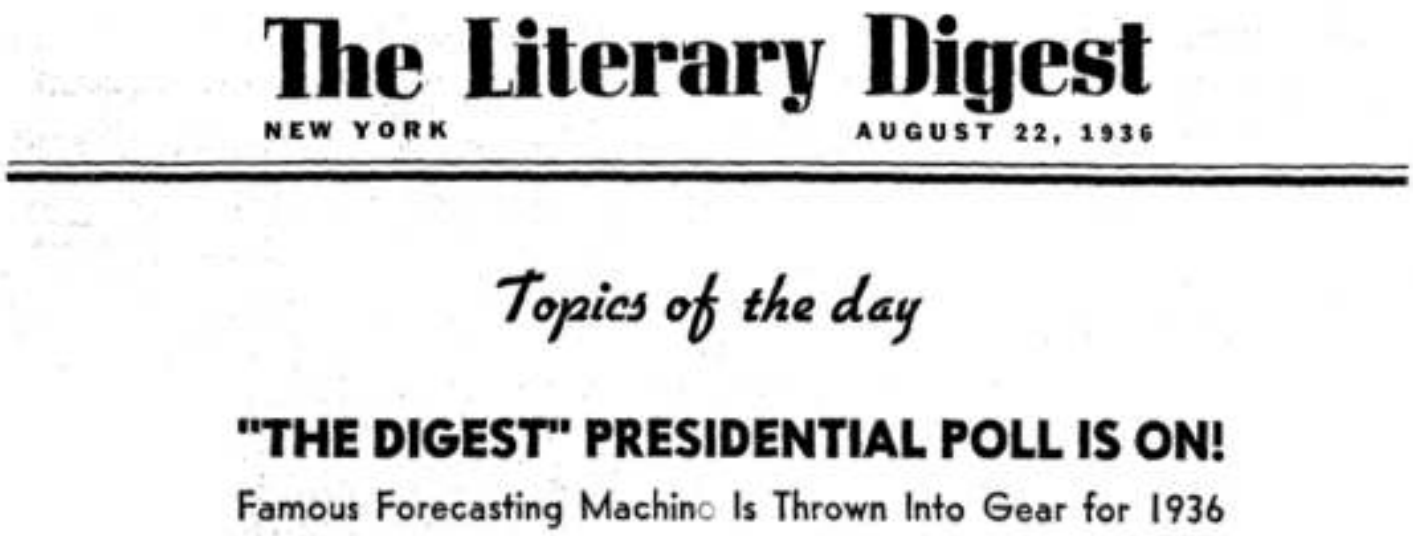
- Last Week
  - preview of three types of prediction
  - prediction within population
  - logistic regression, GAMs
  - tree models
- This Week
  - proposal discussion
  - prediction in a new population
  - random forests and boosting
  - neural networks
- Next Week
  - prediction in a counterfactual population (aka causal inference)

Where We've Been and Where We're Going...

- Last Week
  - preview of three types of prediction
  - prediction within population
  - logistic regression, GAMs
  - tree models
- This Week
  - proposal discussion
  - prediction in a new population
  - random forests and boosting
  - neural networks
- Next Week
  - prediction in a counterfactual population (aka causal inference)
- Long Run
  - target tasks $\rightarrow$ data sources $\rightarrow$ guiding values

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Literary Digest Poll



The Literary Digest

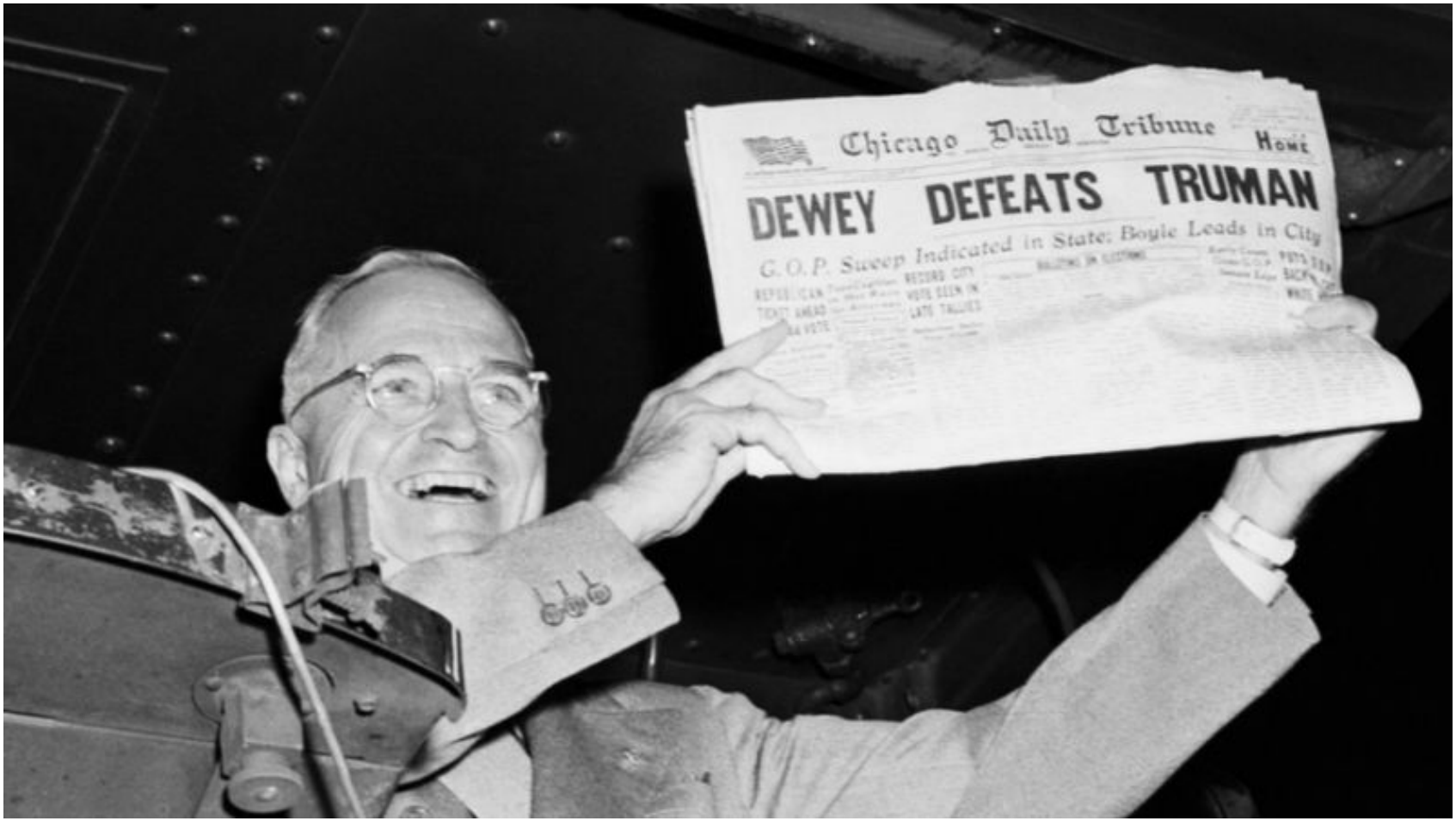
NEW YORK AUGUST 22, 1936

Topics of the day

"THE DIGEST" PRESIDENTIAL POLL IS ON!

Famous Forecasting Machine Is Thrown Into Gear for 1936

The Fable of Dewey Defeats Truman



Hubris

FiveThirtyEight

Politics Sports Science Podcasts Video

OCT. 18, 2016, AT 11:54 AM

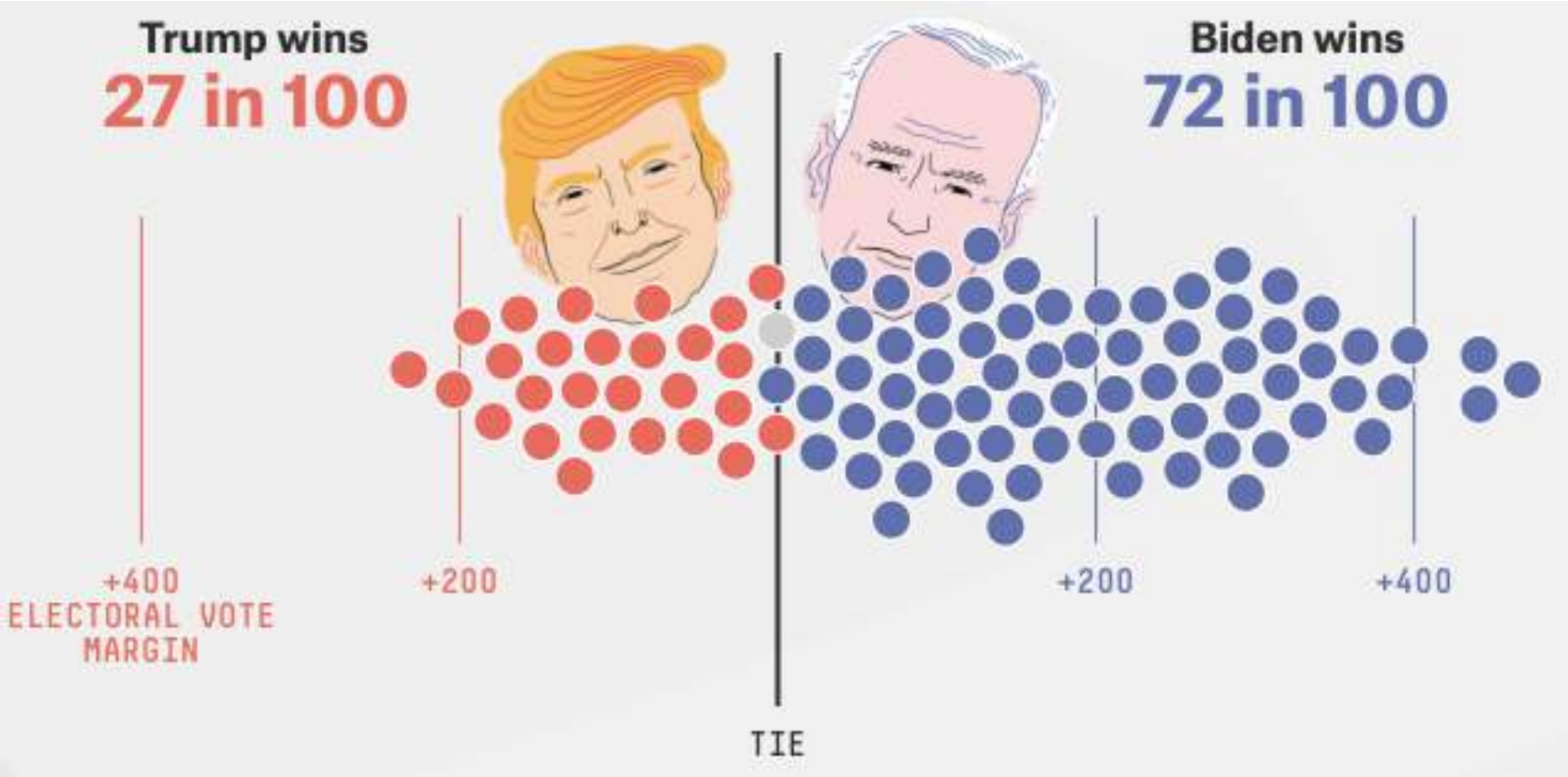
Clinton-Trump Probably Won't Be The Next 'Dewey Defeats Truman'

By Harry Enten

Filed under 2016 Election

President Harry S. Truman gleefully displays an early edition of the Chicago Daily Tribune from his train in St. Louis after his defeat of Thomas E. Dewey in the 1948 presidential election. GETTY IMAGES

Example: Modern Polling



Core Problem

Core Problem

In general, if the training data is not a random sample of the population of interest, we can be arbitrarily wrong.

Core Problem

In general, if the training data is not a random sample of the population of interest, we can be arbitrarily wrong.

Key Idea: under what assumptions can we do something reasonable?

When Is the Population Different

When Is the Population Different

- Self-selected samples/convenience samples:
  e.g. people who choose to answer a survey, images uploaded on instagram, causes of death in hospitals.

When Is the Population Different

- Self-selected samples/convenience samples:
  e.g. people who choose to answer a survey, images uploaded on instagram, causes of death in hospitals.
- Predicting the future:
  e.g. forecasting elections, forecasting sales demand etc.

When Is the Population Different

- Self-selected samples/convenience samples:
  e.g. people who choose to answer a survey, images uploaded on instagram, causes of death in hospitals.
- Predicting the future:
  e.g. forecasting elections, forecasting sales demand etc.
- Crossing domains:
  e.g. happens any time you create a product for general, arbitrary future use, using sentiment on Amazon reviews to predict movie review sentiment.

When Is the Population Different

- Self-selected samples/convenience samples:
  e.g. people who choose to answer a survey, images uploaded on instagram, causes of death in hospitals.
- Predicting the future:
  e.g. forecasting elections, forecasting sales demand etc.
- Crossing domains:
  e.g. happens any time you create a product for general, arbitrary future use, using sentiment on Amazon reviews to predict movie review sentiment.

Why is social data different?

Intuition: Why is this a Problem?

Intuition: Why is this a Problem?

Two major (related) scenarios:

Intuition: Why is this a Problem?

Two major (related) scenarios:

- The data generating process changes (drift).

Intuition: Why is this a Problem?

Two major (related) scenarios:

- The data generating process changes (drift).
- The model is an approximation which uses heuristics that no longer work (model dependence).

Implications and Simple Tools

Implications and Simple Tools

- Key tools we have discussed like train-test splits and cross-validation give us accurate estimates of generalization performance as long as we are generalize to a random draw of the same distribution.

Implications and Simple Tools

- Key tools we have discussed like train-test splits and cross-validation give us accurate estimates of generalization performance as long as we are generalize to a random draw of the same distribution.
- Big difference between settings where:

Implications and Simple Tools

- Key tools we have discussed like train-test splits and cross-validation give us accurate estimates of generalization performance as long as we are generalize to a random draw of the same distribution.
- Big difference between settings where:
  - we have a small sample of in-domain data with outcomes.

Implications and Simple Tools

- Key tools we have discussed like train-test splits and cross-validation give us accurate estimates of generalization performance as long as we are generalize to a random draw of the same distribution.
- Big difference between settings where:
  - we have a small sample of in-domain data with outcomes.
  - we have in-domain data with no outcomes.

Implications and Simple Tools

- Key tools we have discussed like train-test splits and cross-validation give us accurate estimates of generalization performance as long as we are generalize to a random draw of the same distribution.
- Big difference between settings where:
  - we have a small sample of in-domain data with outcomes.
  - we have in-domain data with no outcomes.
  - we have something that approximates the out-of-domain process (e.g. next step in forecasting).

Implications and Simple Tools

- Key tools we have discussed like train-test splits and cross-validation give us accurate estimates of generalization performance as long as we are generalize to a random draw of the same distribution.
- Big difference between settings where:
  - we have a small sample of in-domain data with outcomes.
  - we have in-domain data with no outcomes.
  - we have something that approximates the out-of-domain process (e.g. next step in forecasting).
  - we have no information beyond assumptions.

Small Sample In-Domain with Labeled Outcomes

Small Sample In-Domain with Labeled Outcomes

- This is the easiest setting of a hard problem because we can check accuracy and learn how different the new population is.

Small Sample In-Domain with Labeled Outcomes

- This is the easiest setting of a hard problem because we can check accuracy and learn how different the new population is.
- It is possible it will work if the drift is minimal.

Small Sample In-Domain with Labeled Outcomes

- This is the easiest setting of a hard problem because we can check accuracy and learn how different the new population is.
- It is possible it will work if the drift is minimal.
- With some drift, it may be we want to start with the model using the out-of-domain data and update a little using in-domain data $\rightsquigarrow$ transfer learning

Small Sample In-Domain with Labeled Outcomes

- This is the easiest setting of a hard problem because we can check accuracy and learn how different the new population is.
- It is possible it will work if the drift is minimal.
- With some drift, it may be we want to start with the model using the out-of-domain data and update a little using in-domain data $\rightsquigarrow$ transfer learning
- In other cases we can learn some kind of transformation that is easier than learning a whole new model (e.g. identifying a new optimal cutpoint)

Small Sample In-Domain with Labeled Outcomes

- This is the easiest setting of a hard problem because we can check accuracy and learn how different the new population is.
- It is possible it will work if the drift is minimal.
- With some drift, it may be we want to start with the model using the out-of-domain data and update a little using in-domain data $\rightsquigarrow$ transfer learning
- In other cases we can learn some kind of transformation that is easier than learning a whole new model (e.g. identifying a new optimal cutpoint)
- Core challenge: avoid overfitting in-domain data.

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Revisiting Verbal Autopsies

Revisiting Verbal Autopsies

- Recall the verbal autopsy settings: collect symptoms, predict cause of death

Revisiting Verbal Autopsies

- Recall the verbal autopsy settings: collect symptoms, predict cause of death
- Core challenge: differences between urban hospitals (where labeled data comes from) and rural hospitals (where you actually need the predictions)

Revisiting Verbal Autopsies

- Recall the verbal autopsy settings: collect symptoms, predict cause of death
- Core challenge: differences between urban hospitals (where labeled data comes from) and rural hospitals (where you actually need the predictions)
- This is the setting where we observe the in-domain unlabeled causes, but no labeled cases.

Potential Challenges

Potential Challenges

Potential Challenges

- different common causes of death
- different presentations of disease
- different types of data collection.

Approaching the Problem

Approaching the Problem

- Key realization in King and Liu (2008): we only need to estimate proportions of causes of death over a site, not predict individual cases.

Approaching the Problem

- Key realization in King and Liu (2008): we only need to estimate proportions of causes of death over a site, not predict individual cases.
- Without further assumptions, this isn't possible.

Approaching the Problem

- Key realization in King and Liu (2008): we only need to estimate proportions of causes of death over a site, not predict individual cases.
- Without further assumptions, this isn't possible.
- We know people have different causes of death across sites, but what if we were willing to assume that the presentation of the disease was the same across sites.

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

$$X_i = (1, 0, 0, 1, \ldots, 0)$$

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

$$X_i = (1, 0, 0, 1, \ldots, 0)$$

Define:

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

$$X_i = (1, 0, 0, 1, \ldots, 0)$$

Define:

- $P(X)$ the probability of seeing all combinations of symptoms (2^J length)

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

$$X_i = (1, 0, 0, 1, \ldots, 0)$$

Define:

- $P(X)$ the probability of seeing all combinations of symptoms (2^J length)
- $P(X|Y)$ the probability of seeing all combinations of symptoms given the cause of death (2^J for each of K causes of death)

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

$$X_i = (1, 0, 0, 1, \ldots, 0)$$

Define:

- $P(X)$ the probability of seeing all combinations of symptoms (2^J length)
- $P(X|Y)$ the probability of seeing all combinations of symptoms given the cause of death (2^J for each of K causes of death)
- $P(Y)$ the probability of each cause of death (K length vector)

VA Algorithm (King and Liu 2008)

Measure presence/absence of each symptom [J vector]

$$X_i = (1, 0, 0, 1, \ldots, 0)$$

Define:

- $P(X)$ the probability of seeing all combinations of symptoms (2^J length)
- $P(X|Y)$ the probability of seeing all combinations of symptoms given the cause of death (2^J for each of K causes of death)
- $P(Y)$ the probability of each cause of death (K length vector)

Useful probability fact: $P(X) = \sum_{k=1}^{K} P(X|Y = k)P(Y = k)$

VA Algorithm (King and Liu 2008)

VA Algorithm (King and Liu 2008)

- $P(Y)$ is what we want to know for a given site: proportion of deaths by cause.

VA Algorithm (King and Liu 2008)

- $P(Y)$ is what we want to know for a given site: proportion of deaths by cause.
- We will assume that $P(X|Y = k)$ is constant in-domain (rural) and training data (urban). Estimate in training data.

VA Algorithm (King and Liu 2008)

- $P(Y)$ is what we want to know for a given site: proportion of deaths by cause.
- We will assume that $P(X|Y = k)$ is constant in-domain (rural) and training data (urban). Estimate in training data.
- We will estimate $P(X)$ from the rural data.

VA Algorithm (King and Liu 2008)

- $P(Y)$ is what we want to know for a given site: proportion of deaths by cause.
- We will assume that $P(X|Y = k)$ is constant in-domain (rural) and training data (urban). Estimate in training data.
- We will estimate $P(X)$ from the rural data.

$$\underbrace{P(X)}_{\text{From In-Domain}} = \sum_{k=1}^{K} \underbrace{P(X|Y = k)}_{\text{From Train}} P(Y = k)$$

VA Algorithm (King and Liu 2008)

- $P(Y)$ is what we want to know for a given site: proportion of deaths by cause.
- We will assume that $P(X|Y = k)$ is constant in-domain (rural) and training data (urban). Estimate in training data.
- We will estimate $P(X)$ from the rural data.

$$\underbrace{P(X)}_{\text{From In-Domain}} = \sum_{k=1}^{K} \underbrace{P(X|Y = k)}_{\text{From Train}} P(Y = k)$$

Solve a constrained optimization problem: what distribution of diseases would give these symptoms?

VA Algorithm

- Key idea for our purposes is that we were able to use our assumptions to reason out estimate of our quantity of interest.

VA Algorithm

- Key idea for our purposes is that we were able to use our assumptions to reason out estimate of our quantity of interest.
- Could consider different strategies like $P(Y|X)$ being constant instead $\rightsquigarrow$ a different strategy (which we will talk more about later).

VA Algorithm

- Key idea for our purposes is that we were able to use our assumptions to reason out estimate of our quantity of interest.
- Could consider different strategies like $P(Y|X)$ being constant instead $\rightsquigarrow$ a different strategy (which we will talk more about later).
- In practice a key part of King and Liu (2008)'s VA algorithm is figuring out how to deal with high dimensional data (if we only had 20 symptoms $P(X)$ is over one million entries long!).

VA Algorithm

- Key idea for our purposes is that we were able to use our assumptions to reason out estimate of our quantity of interest.
- Could consider different strategies like $P(Y|X)$ being constant instead $\rightsquigarrow$ a different strategy (which we will talk more about later).
- In practice a key part of King and Liu (2008)'s VA algorithm is figuring out how to deal with high dimensional data (if we only had 20 symptoms $P(X)$ is over one million entries long!).
- They use a feature subsampling approach similar to the one in random forests and boosting.

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Google Flu

Google Flu Trends

From Wikipedia, the free encyclopedia

Google Flu Trends (**GFT**) was a web service operated by Google. It provided estimates of influenza activity for more than 25 countries. By aggregating Google Search queries, it attempted to make accurate predictions about flu activity. This project was first launched in 2008 by Google.org to help predict outbreaks of flu.[1]

Google Flu Trends stopped publishing current estimates on 9 August 2015. Historical estimates are still available for download, and current data are offered for declared research purposes.[2]

Contents [hide]

Google Flu Trends data, South Africa

Google Flu vs. Blumenstock et. al.

Google Flu vs. Blumenstock et. al.

- At first Google Flu seems similar to the Blumenstock et. al. wealth prediction, but the problem is actually much harder.

Google Flu vs. Blumenstock et. al.

- At first Google Flu seems similar to the Blumenstock et. al. wealth prediction, but the problem is actually much harder.
- Both are forms of nowcasting—trying to predict a contemporaneous measurement collected by another means.

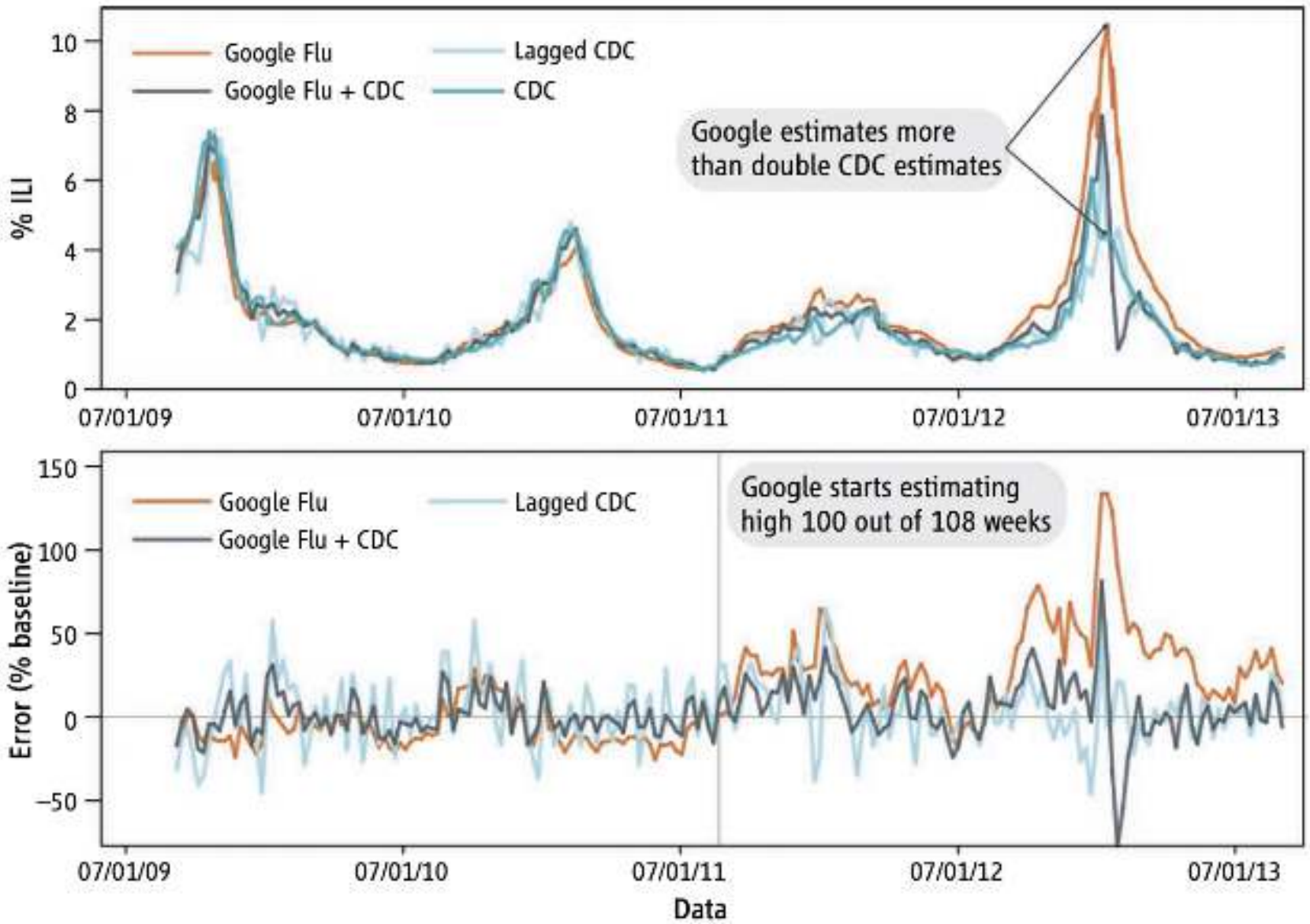
Google Flu vs. Blumenstock et. al.

- At first Google Flu seems similar to the Blumenstock et. al. wealth prediction, but the problem is actually much harder.
- Both are forms of nowcasting—trying to predict a contemporaneous measurement collected by another means.
- In Blumenstock et. al. though it is what Salganik calls amplified asking, we get a random sample of the measurement we want to nowcast.

Google Flu vs. Blumenstock et. al.

- At first Google Flu seems similar to the Blumenstock et. al. wealth prediction, but the problem is actually much harder.
- Both are forms of nowcasting—trying to predict a contemporaneous measurement collected by another means.
- In Blumenstock et. al. though it is what Salganik calls amplified asking, we get a random sample of the measurement we want to nowcast.
- Google Flu is always trying to predict data that will be released in the future.

Lazer et. al 2014



Two Mistakes Per Lazer et al.

Two Mistakes Per Lazer et al.

1) Algorithm Dynamics

the relationship between the predictors and the thing being predicted changed

Two Mistakes Per Lazer et al.

1) Algorithm Dynamics

   the relationship between the predictors and the thing being predicted changed

2) Big Data Hubris

   the often implicit assumption that big data are a substitute for, rather than a supplement to, traditional data collection and analysis

Moving Forward

Moving Forward

- The Google Flu problem has the advantage that we are always trying to learn 'just out of domain.'

Moving Forward

- The Google Flu problem has the advantage that we are always trying to learn 'just out of domain.'
- We keep getting new data that reveals to us the measurement we wanted to predict (note the distinction with Verbal Autopsy which is knowable but unknown).

Moving Forward

- The Google Flu problem has the advantage that we are always trying to learn 'just out of domain.'
- We keep getting new data that reveals to us the measurement we wanted to predict (note the distinction with Verbal Autopsy which is knowable but unknown).
- We can approximate the prediction process by testing our method by always fitting on data through some time t and then predicting at $t + 1$.

Moving Forward

- The Google Flu problem has the advantage that we are always trying to learn 'just out of domain.'
- We keep getting new data that reveals to us the measurement we wanted to predict (note the distinction with Verbal Autopsy which is knowable but unknown).
- We can approximate the prediction process by testing our method by always fitting on data through some time t and then predicting at $t + 1$.
- This doesn't guarantee that we will have a sense of the prediction error, but it is a much better indicator than cross-validation or random train-test splits.

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Using Machine Learning to Help Vulnerable Tenants in New York City

Teng Ye
tengye@umich.edu
University of Michigan, Ann Arbor

Rebecca Johnson
raj2@princeton.edu
Princeton University

Samantha Fu
samanthacfu@gmail.com
London School of Economics and Political Science

Jerica Copeny
jerica.copeny@gmail.com
DePaul University

Bridgit Donnelly
bridgit.donnelly@gmail.com
Public Engagement Unit, City of New York

Alex Freeman
freemanal@hra.nyc.gov
Public Engagement Unit, City of New York

Mirian Lima
mspencerlima@gmail.com
University of Chicago

Joe Walsh
jtwalsh@protonmail.com
University of Chicago

Rayid Ghani
rayid@uchicago.edu
University of Chicago

ABSTRACT

To keep housing affordable, the City of New York has implemented rent-stabilization policies to restrict the rate at which the rent of certain units can be increased every year. However, some landlords of these rent-stabilized units try to illegally force their tenants out in order to circumvent rent-stabilization laws and greatly increase the rent they can charge. To identify and help tenants who are vulnerable to such landlord harassment, the New York City Public Engagement Unit (NYC PEU) conducts targeted outreach to tenants to inform them of their rights and to assist them with serious housing challenges. In this paper, we[1] collaborated with NYC PEU

KEYWORDS

Machine Learning; Social Good; Public Policy; Resource Allocation; Tenant Harassment

ACM Reference Format:
Teng Ye, Rebecca Johnson, Samantha Fu, Jerica Copeny, Bridgit Donnelly, Alex Freeman, Mirian Lima, Joe Walsh, and Rayid Ghani. 2019. Using Machine Learning to Help Vulnerable Tenants in New York City. In *ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS) (COMPASS '19), July 3–5, 2019, Accra, Ghana.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3314344.3332484

Figure 1: Canvassing process with our definition of the outcome label.

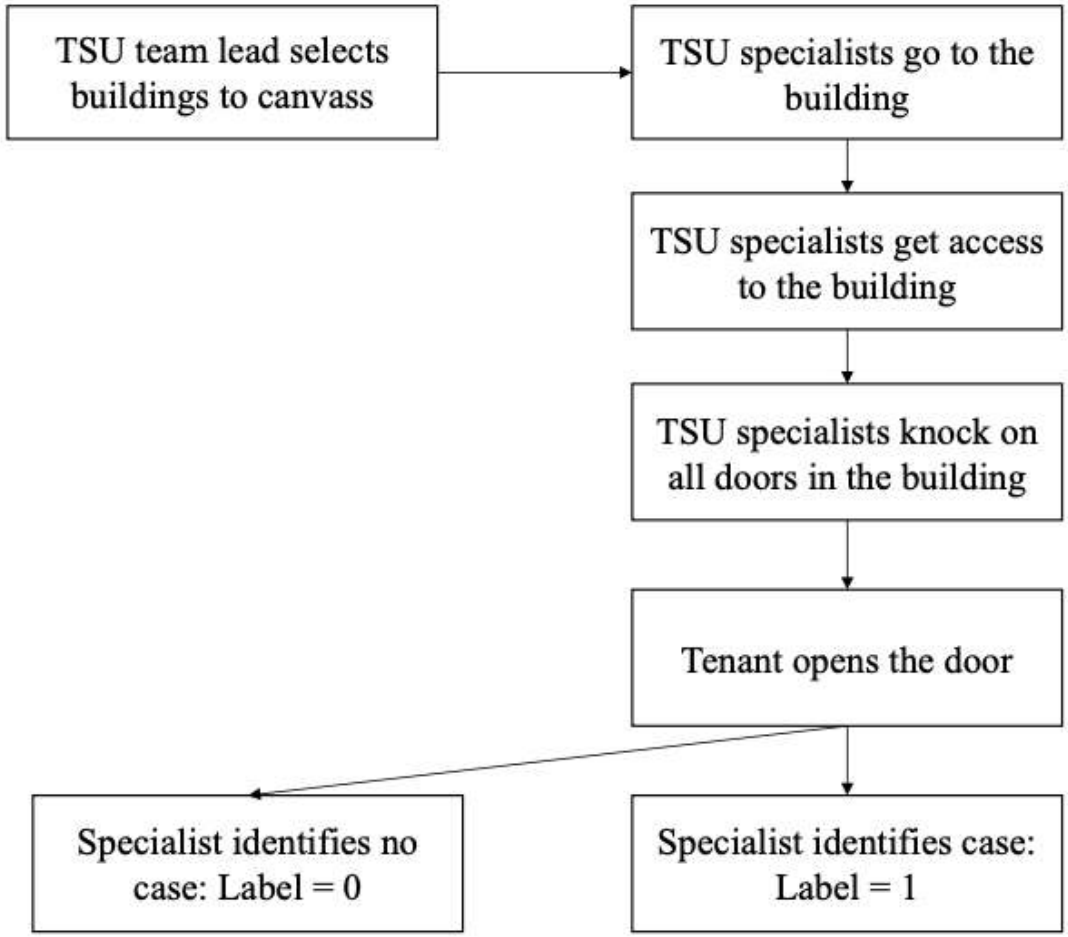


Figure 2: An example illustrating training and testing splits.

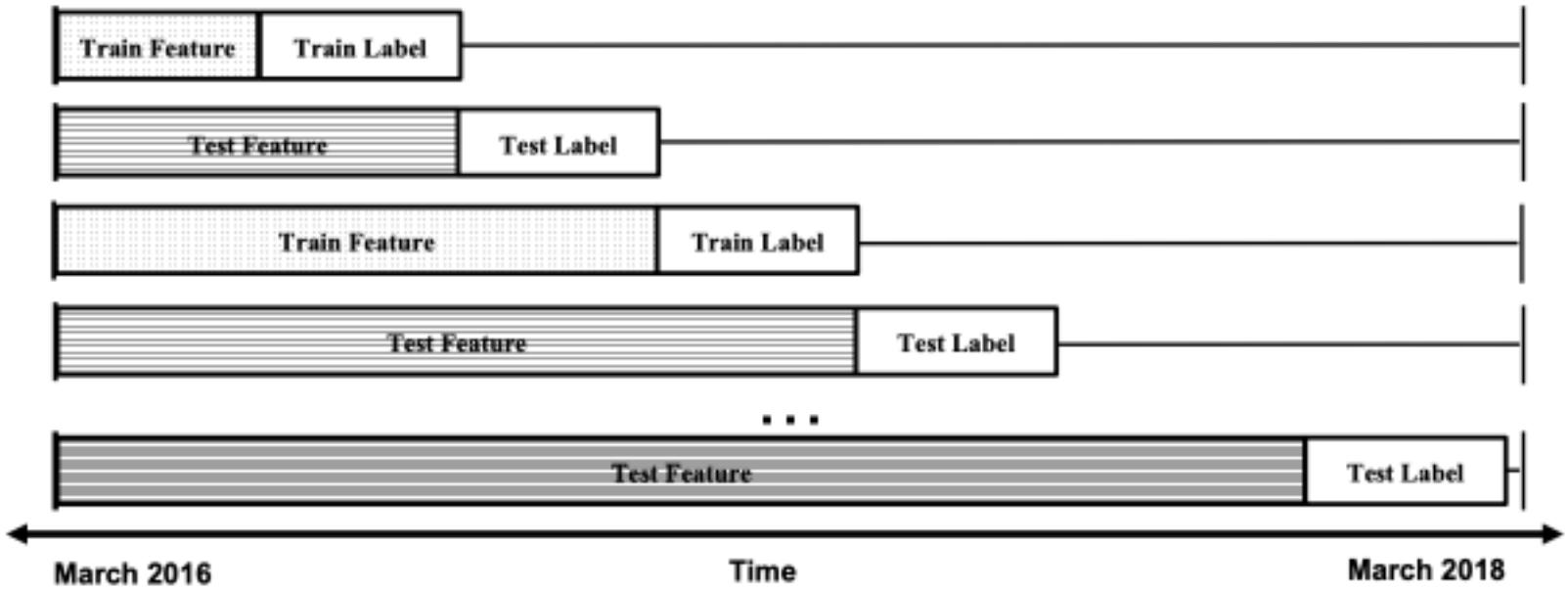


Figure 5: Precision and number of labeled data at each k proportion for the Gradient Boosting model.

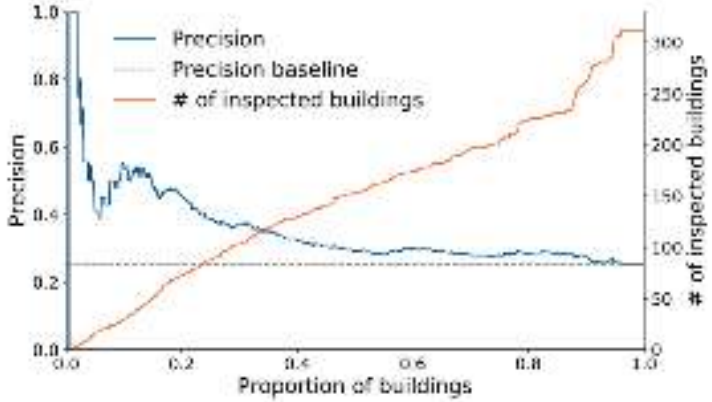
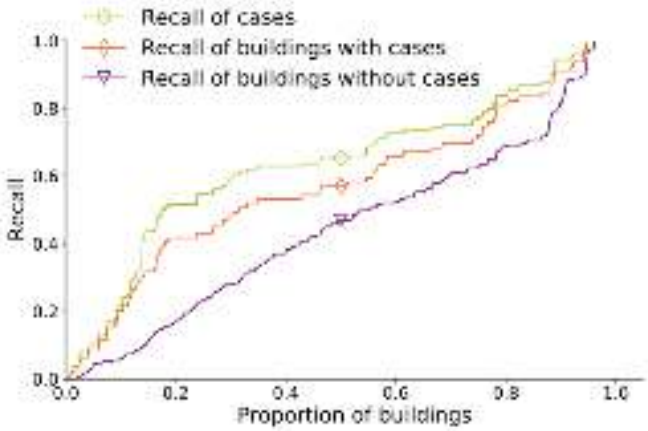


Figure 6: Recall curves at each k proportion for the Gradient Boosting model.



line). The intuition is that a good ranked list will have more (labeled) positive examples than negative examples at the top of the list and vice versa at the bottom of the list. The gap between recall on positive examples and recall on negative examples in Figure 6 allows us to see this was actually the case. The orange line goes up steeply at the beginning (more positive examples) and the purple line goes up steeply at the bottom of the list (more negative examples) giving us confidence in the ranking performance of our model.

Table 2: Confusion matrix of the best performing model.

	Actual True	Actual False
Predicted True	33	50
Predicted False	46	183

6.2 Interpreting the Models: Feature Interpretation

Figure 7 shows the top 20 features that have the highest feature importances in the best performing Gradient Boosting model.

6.2.1 Tract-level demographic features. From the figure, we see that most features in the top 20 feature list were generated from American Community Survey data, which reflects the demographic characteristics of residents in the tract in which a building is located. For example, measures of income insecurity were important in predicting harassment — these included the proportion of people receiving Supplemental Security Income (SSI) in a given tract (*Tract Percent With Supplemental Security Income SSI* in figure 7) and the percentage of households earning less than $10,000 per year (*Tract Income Percent Less Than 10000* in figure 7). These features might

This Topic in Review

This Topic in Review

- Prediction in a different population
- The difficulty of evaluating in this setting

This Topic in Review

- Prediction in a different population
- The difficulty of evaluating in this setting

Next week: Prediction in a Counterfactual Population

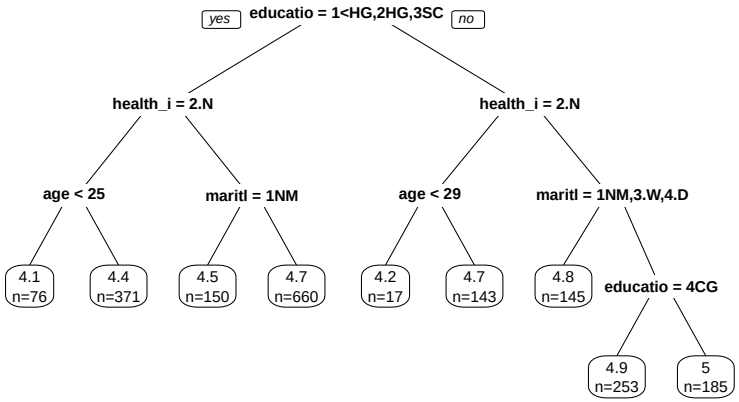
- Athey, Susan, 2017. Beyond prediction: Using big data for policy problems. *Science*, 355(6324), pp.483-485.
- Lundberg, Ian, Rebecca Johnson, and Brandon M. Stewart. 2021. "What Is Your Estimand? Defining the Target Quantity Connects Statistical Evidence to Theory." *ASR* 86, no. 3: 532–65.
- Kleinberg, J., Lakkaraju, H., Leskovec, J., Ludwig, J., and Mullainathan, S. 2018. Human decisions and machine predictions. *QJE*, 133(1), 237-293.

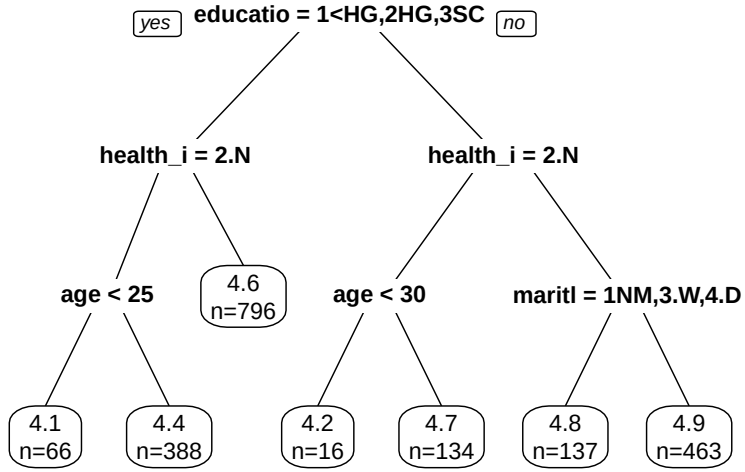
1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Random Forests

$$\text{Random Forests} \approx$$
$$\text{Trees} + \text{Bagging} + \text{Random Feature Selection}$$

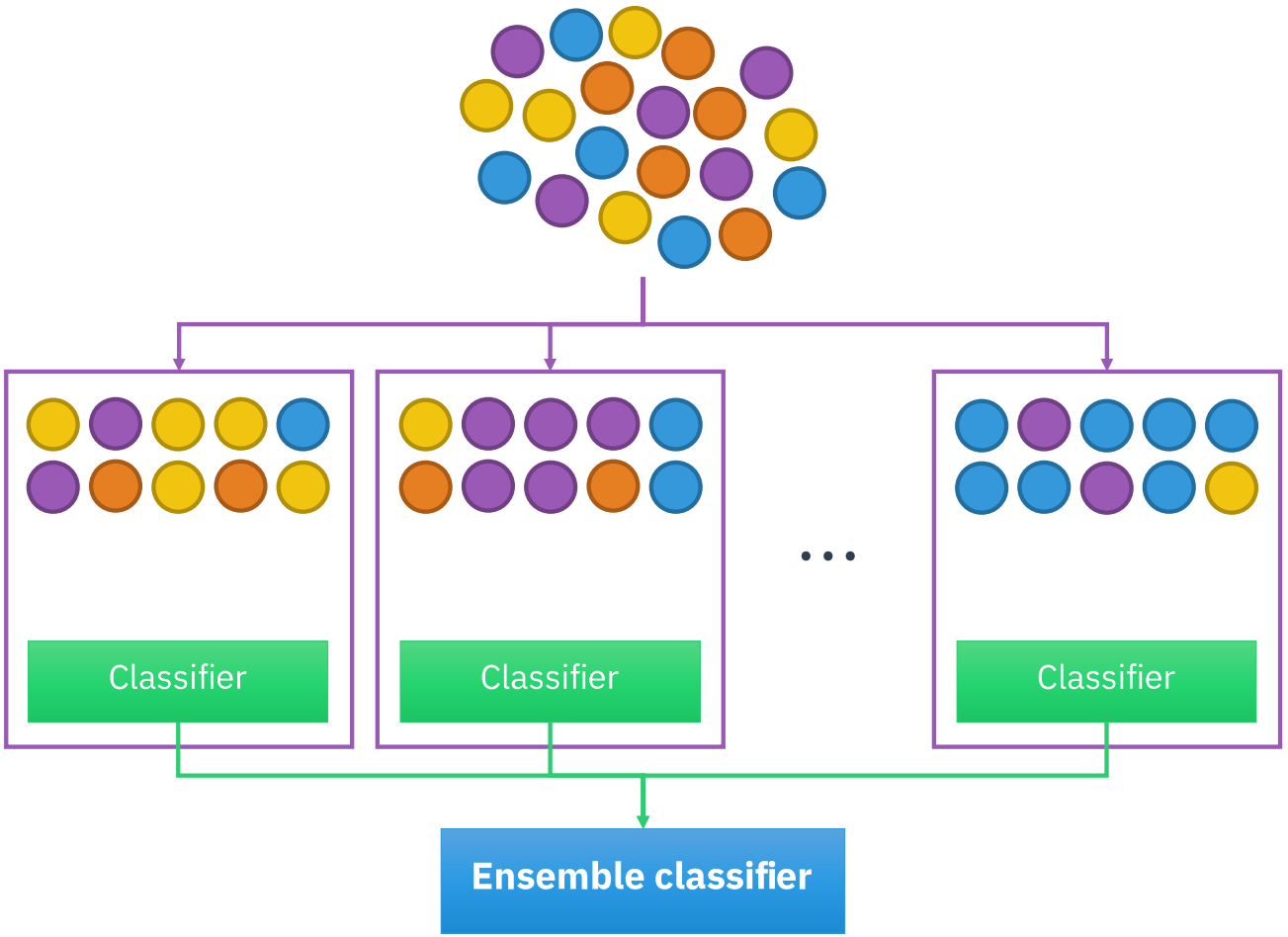
Variance in Trees





Bootstrap Averaging (Bagging)

Bootstrap Averaging (Bagging)



Original Data

Bootstrapping

Aggregating

Bagging

Image Credit: Wikipedia

Why Bagging?

Why Bagging?

- Each time, we resample with replacement n data points (the number of observations) from the data and fits a tree each time (this is the bootstrapping).

Why Bagging?

- Each time, we resample with replacement n data points (the number of observations) from the data and fits a tree each time (this is the bootstrapping).
- Bagging then averages over all the predictions.

Why Bagging?

- Each time, we resample with replacement n data points (the number of observations) from the data and fits a tree each time (this is the bootstrapping).
- Bagging then averages over all the predictions.
- Bagging stabilizes the trees by aggregating over many different variations of them. This produces a regularization type effect.

Why Bagging?

- Each time, we resample with replacement n data points (the number of observations) from the data and fits a tree each time (this is the bootstrapping).
- Bagging then averages over all the predictions.
- Bagging stabilizes the trees by aggregating over many different variations of them. This produces a regularization type effect.
- Because bootstrap samples with replacement, about $\frac{1}{3}$ of the data is unused in any given tree. This is called the out of bag (OOB) data for that tree.

Why Bagging?

- Each time, we resample with replacement n data points (the number of observations) from the data and fits a tree each time (this is the bootstrapping).
- Bagging then averages over all the predictions.
- Bagging stabilizes the trees by aggregating over many different variations of them. This produces a regularization type effect.
- Because bootstrap samples with replacement, about $\frac{1}{3}$ of the data is unused in any given tree. This is called the out of bag (OOB) data for that tree.
- The OOB data can be used to check accuracy or tune parameters because it provides 'out of sample' data for that tree (much like cross-validation).

Why Bagging?

- Each time, we resample with replacement n data points (the number of observations) from the data and fits a tree each time (this is the bootstrapping).
- Bagging then averages over all the predictions.
- Bagging stabilizes the trees by aggregating over many different variations of them. This produces a regularization type effect.
- Because bootstrap samples with replacement, about $\frac{1}{3}$ of the data is unused in any given tree. This is called the out of bag (OOB) data for that tree.
- The OOB data can be used to check accuracy or tune parameters because it provides 'out of sample' data for that tree (much like cross-validation).
- Bagging is a general strategy (not just for trees) but only matters for non-linear models.

Random Feature Selection

Random Feature Selection

- Random Feature Selection means that at every split in the tree we will consider only a random sample of the predictors (features).

Random Feature Selection

- Random Feature Selection means that at every split in the tree we will consider only a random sample of the predictors (features).
- By default we will look at $m = \sqrt{p}$ (classification) or $p/3$ (regression) possible predictors. But m is a parameter that can be set.

Random Feature Selection

- Random Feature Selection means that at every split in the tree we will consider only a random sample of the predictors (features).
- By default we will look at $m = \sqrt{p}$ (classification) or $p/3$ (regression) possible predictors. But m is a parameter that can be set.
- This has a few major effects:
  - it decorrelates the trees

Random Feature Selection

- Random Feature Selection means that at every split in the tree we will consider only a random sample of the predictors (features).
- By default we will look at $m = \sqrt{p}$ (classification) or $p/3$ (regression) possible predictors. But m is a parameter that can be set.
- This has a few major effects:
  - it decorrelates the trees
  - it forces the trees to be robust to different predictors

Random Feature Selection

- Random Feature Selection means that at every split in the tree we will consider only a random sample of the predictors (features).
- By default we will look at $m = \sqrt{p}$ (classification) or $p/3$ (regression) possible predictors. But m is a parameter that can be set.
- This has a few major effects:
  - it decorrelates the trees
  - it forces the trees to be robust to different predictors
  - it speeds things up quite a bit!

Random Feature Selection

- Random Feature Selection means that at every split in the tree we will consider only a random sample of the predictors (features).
- By default we will look at $m = \sqrt{p}$ (classification) or $p/3$ (regression) possible predictors. But m is a parameter that can be set.
- This has a few major effects:
  - it decorrelates the trees
  - it forces the trees to be robust to different predictors
  - it speeds things up quite a bit!
- In short it forces the algorithm to avoid over-reliance on any one variable.

Random Forests

Random Forests

Fix B, m, minimum node size and loss function:

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)
   b) grow a tree by recursively doing the following for each terminal node until the minimum node size is reached (no pruning!)

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)
   b) grow a tree by recursively doing the following for each terminal node until the minimum node size is reached (no pruning!)
      i) select m predictors without replacement from our p total predictors

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)
   b) grow a tree by recursively doing the following for each terminal node until the minimum node size is reached (no pruning!)
      i) select m predictors without replacement from our p total predictors
      ii) pick the best variable-split among the m according to the loss function

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)
   b) grow a tree by recursively doing the following for each terminal node until the minimum node size is reached (no pruning!)
      i) select m predictors without replacement from our p total predictors
      ii) pick the best variable-split among the m according to the loss function
      ii) split the node into two child nodes.

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)
   b) grow a tree by recursively doing the following for each terminal node until the minimum node size is reached (no pruning!)
      i) select m predictors without replacement from our p total predictors
      ii) pick the best variable-split among the m according to the loss function
      ii) split the node into two child nodes.
2) Output the set of trees. Make predictions using majority vote (classification) or average over trees (regression).

Random Forests

Fix B, m, minimum node size and loss function:

1) For each of B trees
   a) take a random sample of the data with replacement of size n (bootstrap)
   b) grow a tree by recursively doing the following for each terminal node until the minimum node size is reached (no pruning!)
      i) select m predictors without replacement from our p total predictors
      ii) pick the best variable-split among the m according to the loss function
      ii) split the node into two child nodes.
2) Output the set of trees. Make predictions using majority vote (classification) or average over trees (regression).

Note that the $\approx 1/3$ of data points chosen for each tree can be used to assess accuracy, make predictions for the training data or set hyperparameters.

Random Forests

Random Forests

- Random Forests are a pretty great off-the-shelf classifiers. In a test of 179 classifiers across 121 data sets, Fernàndez-Delgado et. al (2014) found it to perform consistently among the best.

Random Forests

- Random Forests are a pretty great off-the-shelf classifiers. In a test of 179 classifiers across 121 data sets, Fernàndez-Delgado et. al (2014) found it to perform consistently among the best.
- Even those who contest the results of the paper (Wainberg et al 2016) don't really contest that it is among the best.

Random Forests

- Random Forests are a pretty great off-the-shelf classifiers. In a test of 179 classifiers across 121 data sets, Fernàndez-Delgado et. al (2014) found it to perform consistently among the best.
- Even those who contest the results of the paper (Wainberg et al 2016) don't really contest that it is among the best.
- It is a nice off-the-shelf choice because it has very few tuning parameters and they have reasonable defaults.

Random Forests

- Random Forests are a pretty great off-the-shelf classifiers. In a test of 179 classifiers across 121 data sets, Fernàndez-Delgado et. al (2014) found it to perform consistently among the best.
- Even those who contest the results of the paper (Wainberg et al 2016) don't really contest that it is among the best.
- It is a nice off-the-shelf choice because it has very few tuning parameters and they have reasonable defaults.
- Random Forests will struggle when there are many features but almost all are noise.

Random Forests

- Random Forests are a pretty great off-the-shelf classifiers. In a test of 179 classifiers across 121 data sets, Fernàndez-Delgado et. al (2014) found it to perform consistently among the best.
- Even those who contest the results of the paper (Wainberg et al 2016) don't really contest that it is among the best.
- It is a nice off-the-shelf choice because it has very few tuning parameters and they have reasonable defaults.
- Random Forests will struggle when there are many features but almost all are noise.
- Note: Random Forests are a great example where the algorithm ran far ahead of the theory—they were introduced in 1996 and we are still figuring out the theory!

Implementation

- There are many great packages implementing random forests including `randomForest`

Implementation

- There are many great packages implementing random forests including `randomForest`
- The `ranger` package is incredibly fast and seems to be the right choice for those starting out now.

Implementation

- There are many great packages implementing random forests including `randomForest`
- The `ranger` package is incredibly fast and seems to be the right choice for those starting out now.
- The `tuneRanger` package implements parameter tuning using the out of bag cases.

Implementation

- There are many great packages implementing random forests including `randomForest`
- The `ranger` package is incredibly fast and seems to be the right choice for those starting out now.
- The `tuneRanger` package implements parameter tuning using the out of bag cases.
- `ranger` also offers a number of different extensions including quantile forests and uncertainty estimates.

Missing Data

Missing Data

- Again missing data is a real practical challenge as these algorithms largely don't know what to do with it.

Missing Data

- Again missing data is a real practical challenge as these algorithms largely don't know what to do with it.
- If the missing data is a factor variable, you can treat it as its own category.

Missing Data

- Again missing data is a real practical challenge as these algorithms largely don't know what to do with it.
- If the missing data is a factor variable, you can treat it as its own category.
- In practice we will ignore for now but this is something important to come back to later.

Summary

- Trees are great! They have okay accuracy by themselves, but really shine when included in a larger algorithm.

Summary

- Trees are great! They have okay accuracy by themselves, but really shine when included in a larger algorithm.
- Random Forests are a highly effective algorithm that averages over many trees.

Summary

- Trees are great! They have okay accuracy by themselves, but really shine when included in a larger algorithm.
- Random Forests are a highly effective algorithm that averages over many trees.

Going Deeper:

> James, Witten, Hastie and Tibshirani (2021) *An Introduction to Statistical Learning with Applications in R*. Chapters 8.0–8.2.2
> Free online at `www.statlearning.com`

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Core Idea of Boosting: Iteratively Fit the Residuals

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers
- Boosting is a similar idea except that it is

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers
- Boosting is a similar idea except that it is
  - generally not stochastic as it does not do data resampling

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers
- Boosting is a similar idea except that it is
  - generally not stochastic as it does not do data resampling
  - involves sequentially fitting classifiers

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers
- Boosting is a similar idea except that it is
  - generally not stochastic as it does not do data resampling
  - involves sequentially fitting classifiers
  - an ensemble method where we sum over the classifiers

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers
- Boosting is a similar idea except that it is
  - generally not stochastic as it does not do data resampling
  - involves sequentially fitting classifiers
  - an ensemble method where we sum over the classifiers
- Both amplify (in different ways) the effects of some simple base learner which is often a decision tree (although it need not be).

Core Idea of Boosting: Iteratively Fit the Residuals

- We just covered bagging which is:
  - stochastic due to the data resampling
  - completely parallel across classifiers
  - an ensemble method where we average over classifiers
- Boosting is a similar idea except that it is
  - generally not stochastic as it does not do data resampling
  - involves sequentially fitting classifiers
  - an ensemble method where we sum over the classifiers
- Both amplify (in different ways) the effects of some simple base learner which is often a decision tree (although it need not be).
- They are often grouped together as in practice they are interpolating classifiers.

The Many Faces of Boosting

The Many Faces of Boosting

- Boosting has many variants and goes by many names including Adaboost, gradient boosting machines, and boosted trees

The Many Faces of Boosting

- Boosting has many variants and goes by many names including Adaboost, gradient boosting machines, and boosted trees
- There are many variations on how these algorithms work and here we present the basic form of the modern system.

The Many Faces of Boosting

- Boosting has many variants and goes by many names including Adaboost, gradient boosting machines, and boosted trees
- There are many variations on how these algorithms work and here we present the basic form of the modern system.
- We then provide an introduction to `Xgboost` which adds a few tricks to create a stable general purpose classifier (think in the way that random forests added to bagging with trees)

The Many Faces of Boosting

- Boosting has many variants and goes by many names including Adaboost, gradient boosting machines, and boosted trees
- There are many variations on how these algorithms work and here we present the basic form of the modern system.
- We then provide an introduction to `Xgboost` which adds a few tricks to create a stable general purpose classifier (think in the way that random forests added to bagging with trees)
- `Xgboost` in particular has won many modern prediction competitions

The Many Faces of Boosting

- Boosting has many variants and goes by many names including Adaboost, gradient boosting machines, and boosted trees
- There are many variations on how these algorithms work and here we present the basic form of the modern system.
- We then provide an introduction to `Xgboost` which adds a few tricks to create a stable general purpose classifier (think in the way that random forests added to bagging with trees)
- `Xgboost` in particular has won many modern prediction competitions
- My general sense—boosting takes much more tuning than random forests but can often provide a bit better performance (when data is available for proper tuning)

The Boosting Procedure for Trees in Words

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)
4) For a fixed number of future trees:

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)
4) For a fixed number of future trees:
    a) Fit the residuals with a tree (i.e. rather than the outcomes)

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)
4) For a fixed number of future trees:
    a) Fit the residuals with a tree (i.e. rather than the outcomes)
    b) Shrink the tree with the regularization

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)
4) For a fixed number of future trees:
    a) Fit the residuals with a tree (i.e. rather than the outcomes)
    b) Shrink the tree with the regularization
    c) Update the residuals

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)
4) For a fixed number of future trees:
    a) Fit the residuals with a tree (i.e. rather than the outcomes)
    b) Shrink the tree with the regularization
    c) Update the residuals
    d) Repeat

The Boosting Procedure for Trees in Words

1) Fit a tree to the data of some fixed depth.
2) Regularize the tree by multiplying the prediction by some fixed value that is between 0 and 1 (usually small).
3) Calculate the residuals (i.e. subtract the tree's shrunken prediction off the truth)
4) For a fixed number of future trees:
    a) Fit the residuals with a tree (i.e. rather than the outcomes)
    b) Shrink the tree with the regularization
    c) Update the residuals
    d) Repeat
5) For future predictions run the data through all the shrunken trees and add together.

The Boosting Procedure in Math

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$
2) For $b \in 1 \ldots B$ steps

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$
2) For $b \in 1 \ldots B$ steps
    a) Fit $\tilde{g}_b$ to X, r

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$
2) For $b \in 1 \ldots B$ steps
    a) Fit $\tilde{g}_b$ to X, r
    b) Update $\hat{g}_b \leftarrow \lambda \tilde{g}_b$

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$
2) For $b \in 1 \ldots B$ steps
   a) Fit $\tilde{g}_b$ to X, r
   b) Update $\hat{g}_b \leftarrow \lambda \tilde{g}_b$
   c) Update $\hat{G}_b \leftarrow \hat{G}_{b-1} + \hat{g}_b$

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$
2) For $b \in 1 \ldots B$ steps
    a) Fit $\tilde{g}_b$ to X, r
    b) Update $\hat{g}_b \leftarrow \lambda \tilde{g}_b$
    c) Update $\hat{G}_b \leftarrow \hat{G}_{b-1} + \hat{g}_b$
    d) Update the residuals $r_i \leftarrow r_i - \hat{g}_b$

The Boosting Procedure in Math

Inputs: y, X
Settings: B, λ, classifier $\tilde{g}$

1) Set $\hat{G}_0 \leftarrow 0$. $r \leftarrow y$
2) For $b \in 1 \ldots B$ steps
   a) Fit $\tilde{g}_b$ to X, r
   b) Update $\hat{g}_b \leftarrow \lambda \tilde{g}_b$
   c) Update $\hat{G}_b \leftarrow \hat{G}_{b-1} + \hat{g}_b$
   d) Update the residuals $r_i \leftarrow r_i - \hat{g}_b$
3) Use as prediction $\hat{G}_B(X) = \sum_b \hat{g}_b(X)$

Boosting as Additive Models

Boosting as Additive Models

- Consider boosting with one split (stumps)

Boosting as Additive Models

- Consider boosting with one split (stumps)
- Each prediction only uses one variable, so this fits an additive linear model:

$$\hat{G}_B(X) = \sum_{b=1}^{B} \hat{g}_b(X) \tag{1}$$

$$= \sum_{j=1}^{p} \sum_{b \in \mathcal{B}_j} \hat{g}_b(x) \tag{2}$$

$$= \sum_{j=1}^{p} \hat{f}_j(x_j) \tag{3}$$

Boosting as Additive Models

- Consider boosting with one split (stumps)
- Each prediction only uses one variable, so this fits an additive linear model:

$$\hat{G}_B(X) = \sum_{b=1}^{B} \hat{g}_b(X) \tag{1}$$

$$= \sum_{j=1}^{p} \sum_{b \in \mathcal{B}_j} \hat{g}_b(x) \tag{2}$$

$$= \sum_{j=1}^{p} \hat{f}_j(x_j) \tag{3}$$

- It is an additive model with adaptive basis functions.

Boosting as Additive Models

- Consider boosting with one split (stumps)
- Each prediction only uses one variable, so this fits an additive linear model:

$$\hat{G}_B(X) = \sum_{b=1}^{B} \hat{g}_b(X) \tag{1}$$

$$= \sum_{j=1}^{p} \sum_{b \in \mathcal{B}_j} \hat{g}_b(x) \tag{2}$$

$$= \sum_{j=1}^{p} \hat{f}_j(x_j) \tag{3}$$

- It is an additive model with adaptive basis functions.
- In a more general sense, boosting always yields an additive model with piecewise constant basis functions which have interactions of order up to the depth of the tree minus 1.

Tuning Parameters

Tuning Parameters

- λ Shrinkage Parameter
  the degree to which the trees are shrunk, prevents overfitting (many books use η or ϵ).

Tuning Parameters

- λ Shrinkage Parameter
  the degree to which the trees are shrunk, prevents overfitting (many books use η or ϵ).
- B Number of Trees
  controls the complexity of the final model. unlike random forests where more is almost always better, makes a big difference here. Needs to be higher when λ is small.

Tuning Parameters

- λ Shrinkage Parameter
  the degree to which the trees are shrunk, prevents overfitting (many books use η or ϵ).
- B Number of Trees
  controls the complexity of the final model. unlike random forests where more is almost always better, makes a big difference here. Needs to be higher when λ is small.
- D Number of Splits in Tree
  often relatively small depth trees are used in practice.

Variations

Variations

- Many variations for different kinds of outcomes (such as binary response variables)

Variations

- Many variations for different kinds of outcomes (such as binary response variables)
- Early versions like Adaboost reweighted observations when fitting to upweight cases that had been missed.

Variations

- Many variations for different kinds of outcomes (such as binary response variables)
- Early versions like Adaboost reweighted observations when fitting to upweight cases that had been missed.
- In some settings people will take the basis functions learned by the boosting and fit a lasso model to post process them (this gives us a simpler model at the end of the day).

Implementations

`gbm` (what we've learned)

`xgboost` (gradient boosting with a twist)

Extreme Gradient Boosting (xgboost)

Many small tweaks but core ideas:

Extreme Gradient Boosting (xgboost)

Many small tweaks but core ideas:

- introduces parallelism and scalability for sparse matrices

Extreme Gradient Boosting (xgboost)

Many small tweaks but core ideas:

- introduces parallelism and scalability for sparse matrices
- column subsampling (like random forests)

Extreme Gradient Boosting (xgboost)

Many small tweaks but core ideas:

- introduces parallelism and scalability for sparse matrices
- column subsampling (like random forests)
- approximate splits for speed

Summary

Summary

- Boosting is a way of taking a weaker prediction method and allowing it to fit something more complex.

Summary

- Boosting is a way of taking a weaker prediction method and allowing it to fit something more complex.
- Bagging (by contrast) was a way of lowering the variance of very expressive prediction methods.

Summary

- Boosting is a way of taking a weaker prediction method and allowing it to fit something more complex.
- Bagging (by contrast) was a way of lowering the variance of very expressive prediction methods.
- Both ways to really improve decision tree performance at cost of interpretability.

Summary

- Boosting is a way of taking a weaker prediction method and allowing it to fit something more complex.
- Bagging (by contrast) was a way of lowering the variance of very expressive prediction methods.
- Both ways to really improve decision tree performance at cost of interpretability.

Going Deeper:

James, Witten, Hastie and Tibshirani (2013) *An Introduction to Statistical Learning with Applications in R*. Chapters 8.2.3
Free online at `www.statlearning.com`

Efron and Hastie (2016) *Computer Age Statistical Inference*. Chapters 17.2-17.5

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

The Core Idea: Linear/Logistic Regression with Adaptive Non-Linear Basis Functions

The Core Idea: Linear/Logistic Regression with Adaptive Non-Linear Basis Functions

$$\hat{y} = \hat{\beta}_0 + \sum_{m=1}^{M} \hat{\beta}_m \hat{\phi}_m(X)$$

Behind the Hype

Behind the Hype

- appear in earnest in the mid 1980's

Behind the Hype

- appear in earnest in the mid 1980's
- universal approximator

Behind the Hype

- appear in earnest in the mid 1980's
- universal approximator
- diminishing use in 1990's

Behind the Hype

- appear in earnest in the mid 1980's
- universal approximator
- diminishing use in 1990's
- reborn in $\approx$2010 as deep learning

High Level Summary

High Level Summary

- "just" a nonlinear model

High Level Summary

- "just" a nonlinear model
- tons of knobs to tune

High Level Summary

- "just" a nonlinear model
- tons of knobs to tune
- very powerful, but very finicky

High Level Summary

- "just" a nonlinear model
- tons of knobs to tune
- very powerful, but very finicky
- impressive performance on complex data that can defy intuition

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

Single Layer Feedforward Neural Network

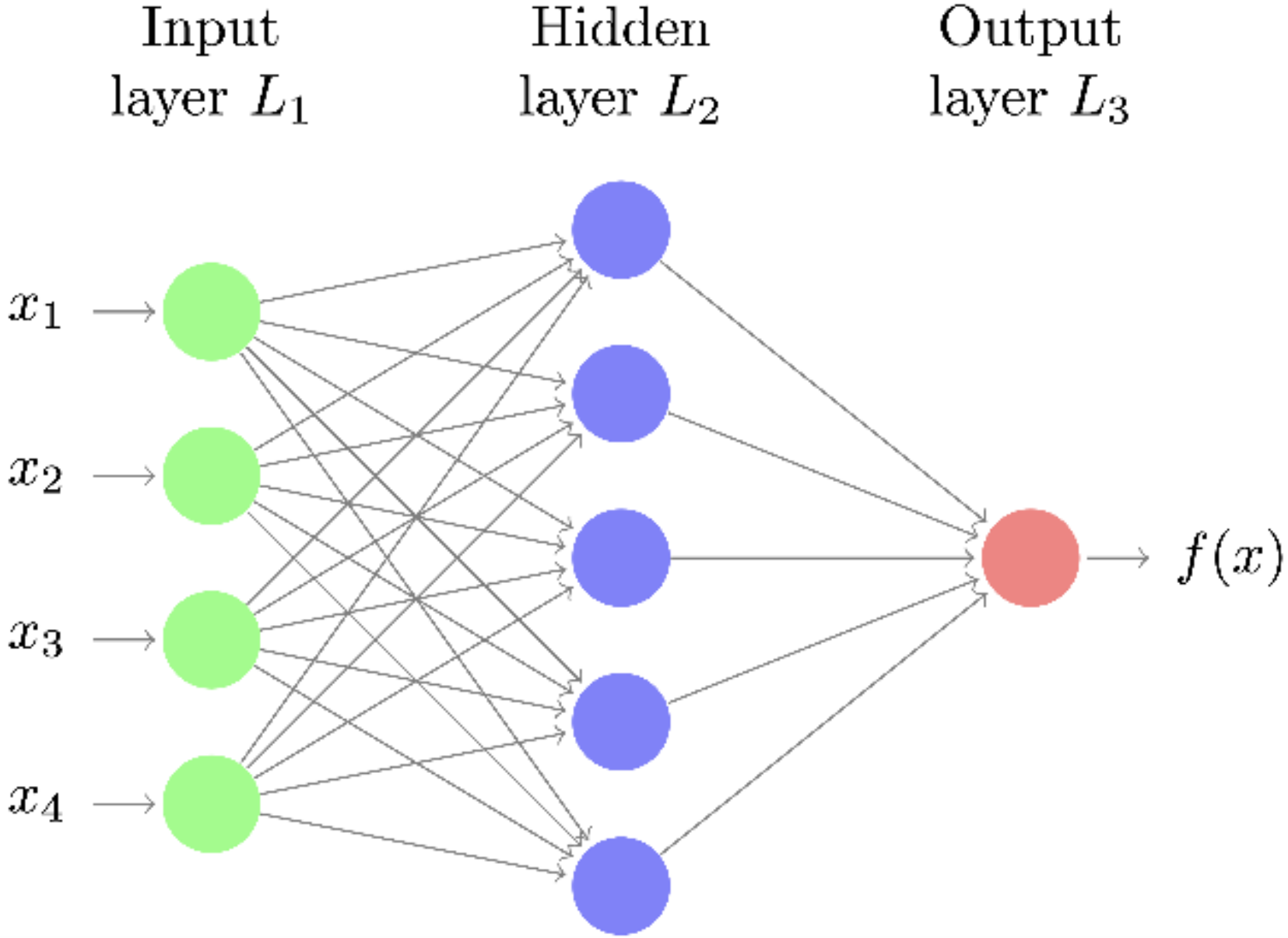


Image Credit: Efron and Hastie *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*

Single Layer Feedforward Neural Network

$$z_{i,\ell} = g\left(w_0^{(1)} + \sum_{j=1}^{4} w_{\ell,j}^{(1)} x_j\right)$$

Single Layer Feedforward Neural Network

$$
\begin{aligned}
z_{i,\ell} &= g\left(w_0^{(1)} + \sum_{j=1}^{4} w_{\ell,j}^{(1)} x_j\right) \\
\hat{Y}_i &= h\left(w_0^{(2)} + \sum_{\ell=1}^{5} w_{\ell,j}^{(2)} z_{i,\ell}\right)
\end{aligned}
$$

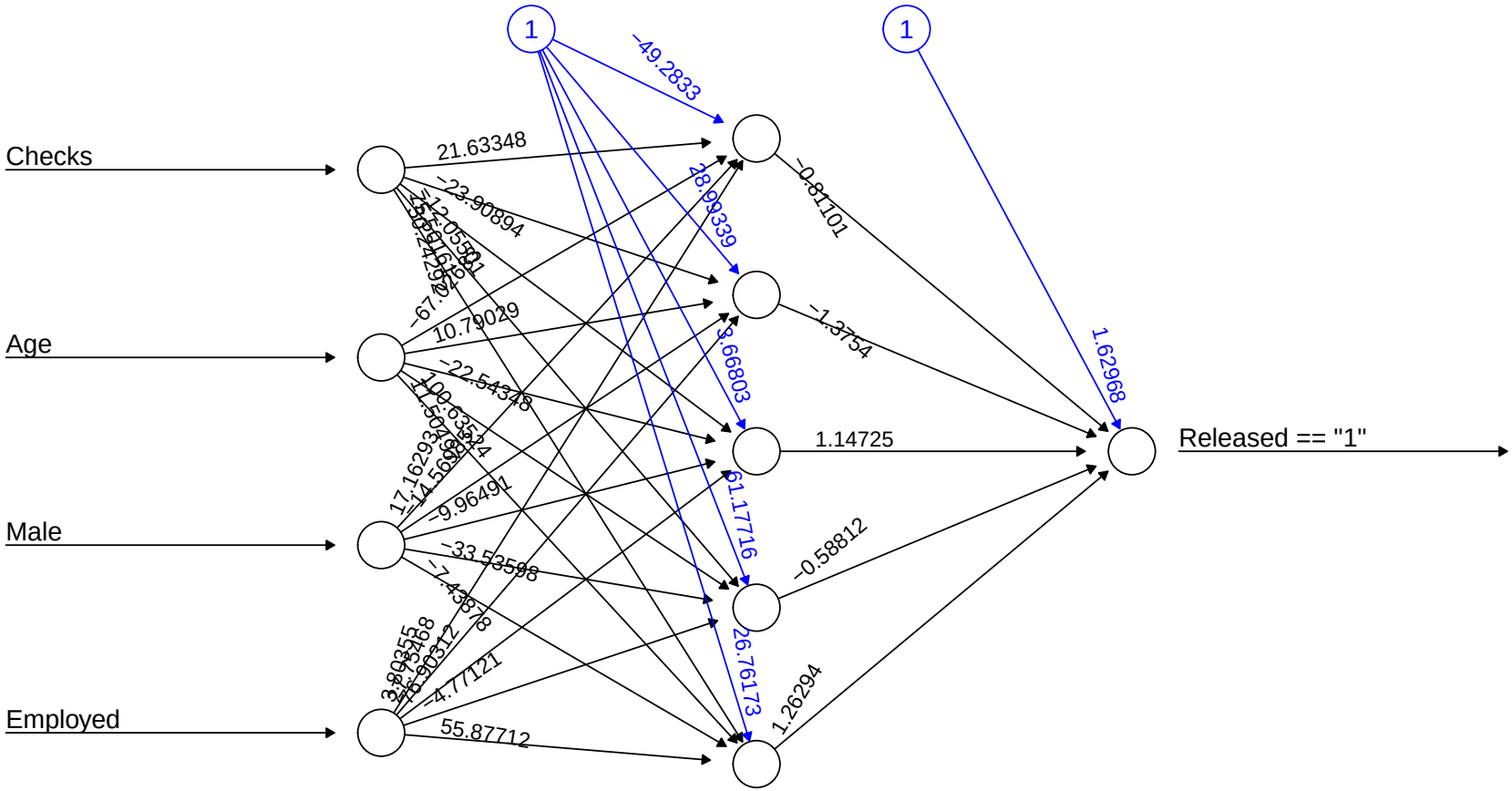
Single Layer Feedforward Neural Network

$$z_{i,\ell} = g\left(w_0^{(1)} + \sum_{j=1}^{4} w_{\ell,j}^{(1)} x_j\right)$$

$$\hat{Y}_i = h\left(w_0^{(2)} + \sum_{\ell=1}^{5} w_{\ell,j}^{(2)} z_{i,\ell}\right)$$

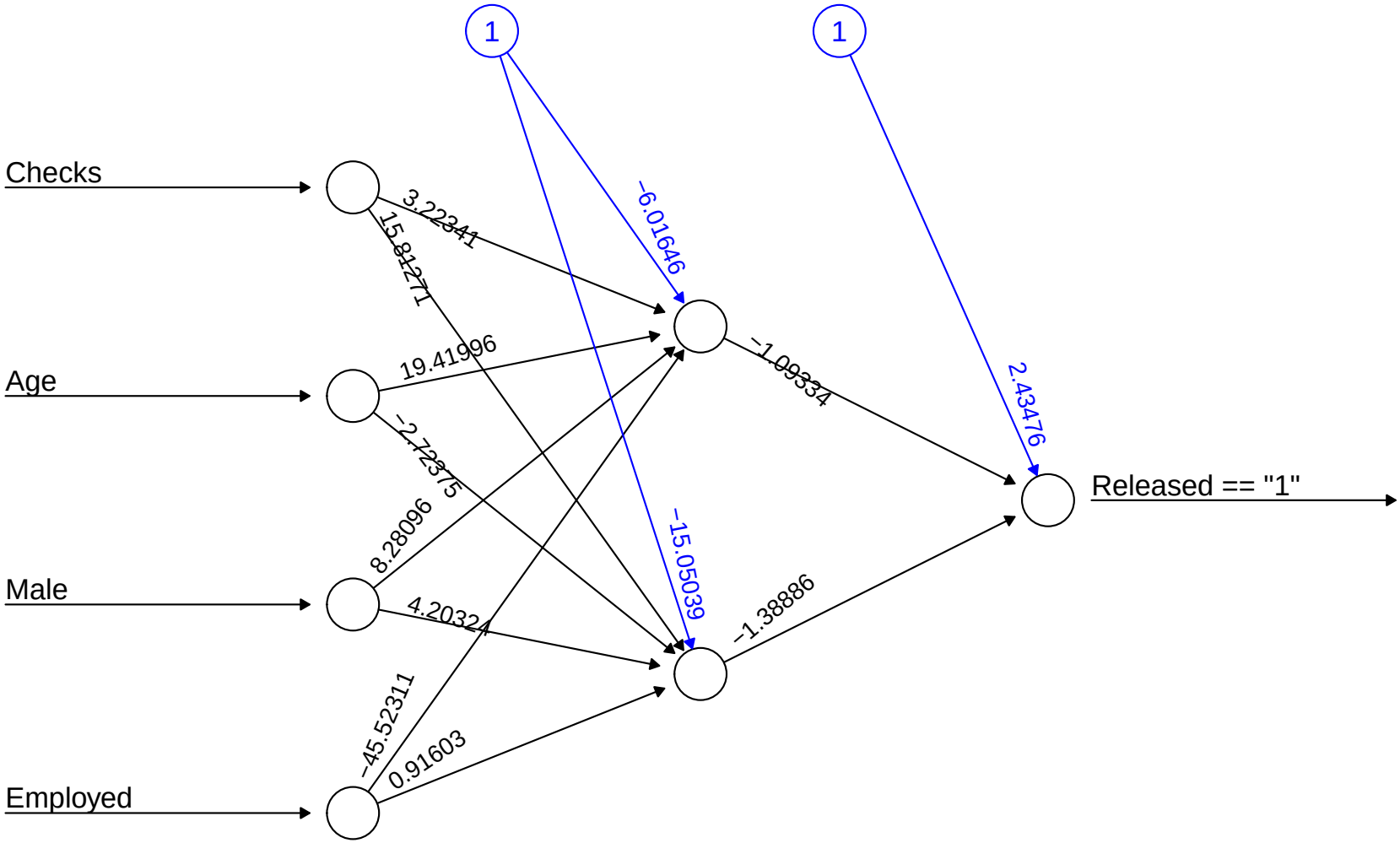
$g()$ must be non-linear (called the activation function), $h()$ will generally be the identity, logistic, or softmax.

Single Layer Feedforward Neural Network



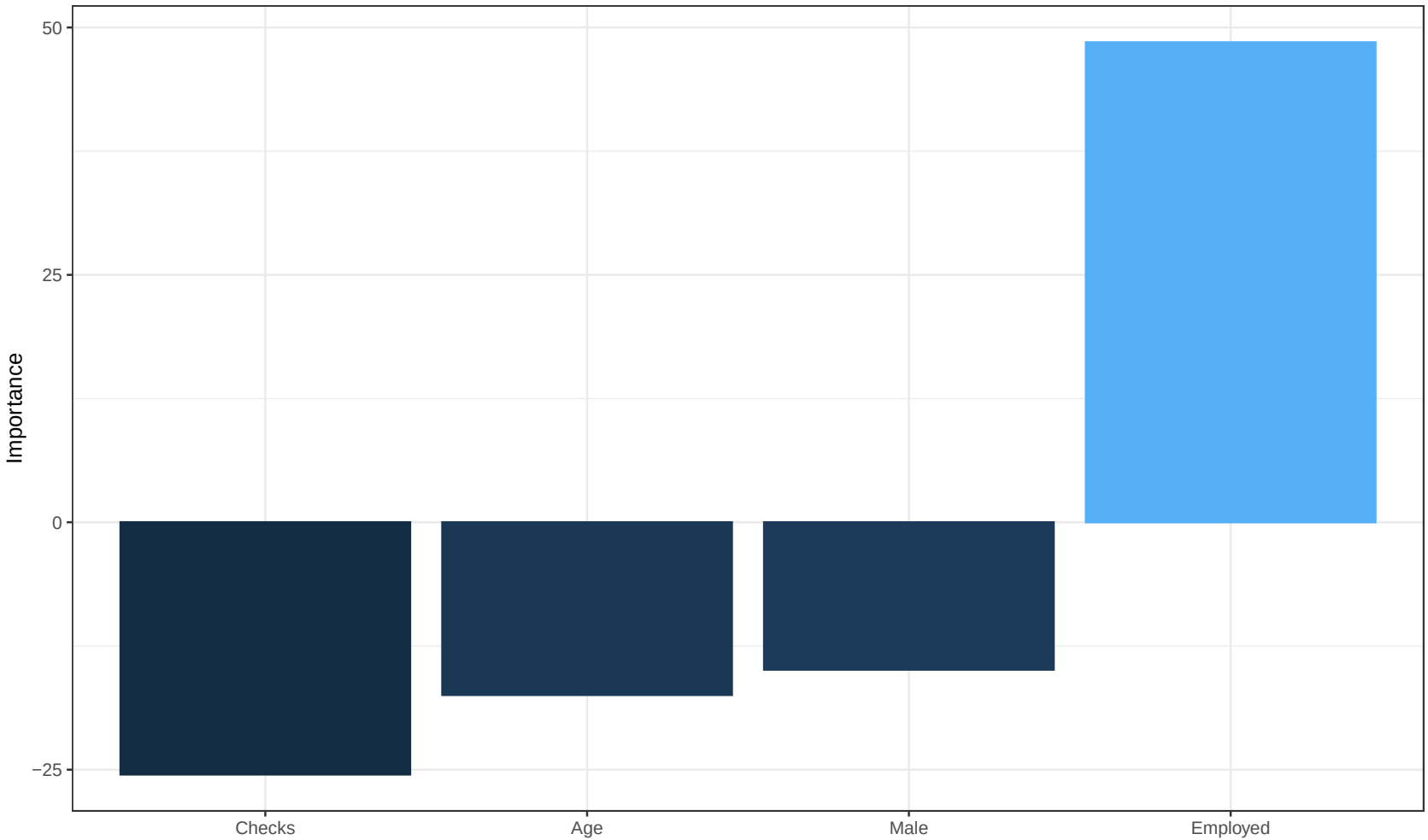
Error: 828.091808 Steps: 5058

Single Layer Feedforward Neural Network



Error: 809.07176 Steps: 1259

Single Layer Feedforward Neural Network



Optimization

- Much harder than other models!

Optimization

- Much harder than other models!
- Stochastic gradient descent! (including fancier variants)

Optimization

- Much harder than other models!
- Stochastic gradient descent! (including fancier variants)
- Careful initialization needed.

Optimization

- Much harder than other models!
- Stochastic gradient descent! (including fancier variants)
- Careful initialization needed.
- All handled by modern software packages.

Mini-batch Stochastic Gradient Descent

- Core Idea: update each step with a small piece of the data rather than the full thing.

Mini-batch Stochastic Gradient Descent

- Core Idea: update each step with a small piece of the data rather than the full thing.
- An epoch is a sequence of steps which account for one full pass through the data.

Mini-batch Stochastic Gradient Descent

- Core Idea: update each step with a small piece of the data rather than the full thing.
- An epoch is a sequence of steps which account for one full pass through the data.
- So if we have 100K data points and batch sizes of 100, there are 1000 steps per epoch.

Mini-batch Stochastic Gradient Descent

- Core Idea: update each step with a small piece of the data rather than the full thing.
- An epoch is a sequence of steps which account for one full pass through the data.
- So if we have 100K data points and batch sizes of 100, there are 1000 steps per epoch.
- Sometimes convenient to specify train time in epochs rather than steps.

Why Do We Need a Non-linearity?

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

$$\hat{Y} = w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right)$$

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

What happens if $g()$ is the identity function?

$$\hat{Y} = w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right)$$

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

What happens if $g()$ is the identity function?

$$\begin{aligned}\hat{Y} &= w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right) \\ &= \left(w_0^{(2)} + w_1^{(2)} w_0^{(1)}\right) + w_1^{(2)} w_1^{(1)} x_1 + w_1^{(2)} w_2^{(1)} x_2\end{aligned}$$

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

What happens if $g()$ is the identity function?

$$\begin{aligned}\hat{Y} &= w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right) \\ &= \left(w_0^{(2)} + w_1^{(2)} w_0^{(1)}\right) + w_1^{(2)} w_1^{(1)} x_1 + w_1^{(2)} w_2^{(1)} x_2\end{aligned}$$

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

What happens if $g()$ is the identity function?

$$\begin{aligned}
\hat{Y} &= w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right) \\
&= \left(w_0^{(2)} + w_1^{(2)} w_0^{(1)}\right) + w_1^{(2)} w_1^{(1)} x_1 + w_1^{(2)} w_2^{(1)} x_2 \\
&= \beta_0 + \beta_1 x_1 + \beta_2 x_2
\end{aligned}$$

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

What happens if $g()$ is the identity function?

$$\begin{aligned}
\hat{Y} &= w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right) \\
&= \left(w_0^{(2)} + w_1^{(2)} w_0^{(1)}\right) + w_1^{(2)} w_1^{(1)} x_1 + w_1^{(2)} w_2^{(1)} x_2 \\
&= \beta_0 + \beta_1 x_1 + \beta_2 x_2
\end{aligned}$$

Why Do We Need a Non-linearity?

Imagine a simple model with 2 covariates, 1 hidden node in 1 layer for a regression problem.

What happens if $g()$ is the identity function?

$$\begin{aligned}
\hat{Y} &= w_0^{(2)} + w_1^{(2)} g\left(w_0^{(1)} + w_1^{(1)} x_1 + w_2^{(1)} x_2\right) \\
&= \left(w_0^{(2)} + w_1^{(2)} w_0^{(1)}\right) + w_1^{(2)} w_1^{(1)} x_1 + w_1^{(2)} w_2^{(1)} x_2 \\
&= \beta_0 + \beta_1 x_1 + \beta_2 x_2
\end{aligned}$$

It reduces to a linear model!

Does It Have to Be That Non-linearity?

Does It Have to Be That Non-linearity?

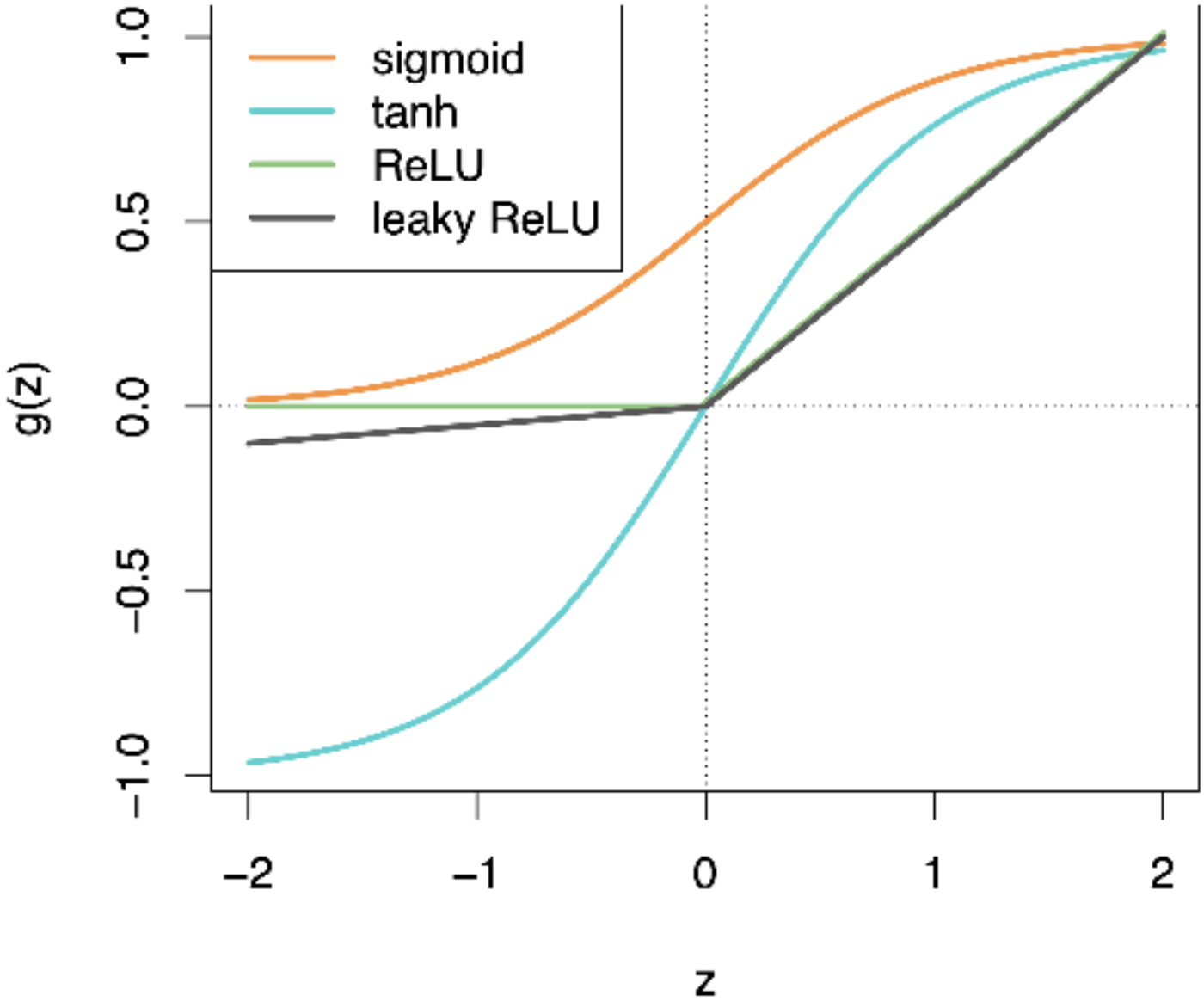


Image Credit: Efron and Hastie *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*

Regularization

Many variants of regularization

Regularization

Many variants of regularization

- "Weight Decay" (aka ridge penalty)

Regularization

Many variants of regularization

- "Weight Decay" (aka ridge penalty)
- Weight Sharing

Regularization

Many variants of regularization

- "Weight Decay" (aka ridge penalty)
- Weight Sharing
- Early Stopping

Regularization

Many variants of regularization

- "Weight Decay" (aka ridge penalty)
- Weight Sharing
- Early Stopping
- Data Augmentation

Regularization

Many variants of regularization

- "Weight Decay" (aka ridge penalty)
- Weight Sharing
- Early Stopping
- Data Augmentation
- Dropout

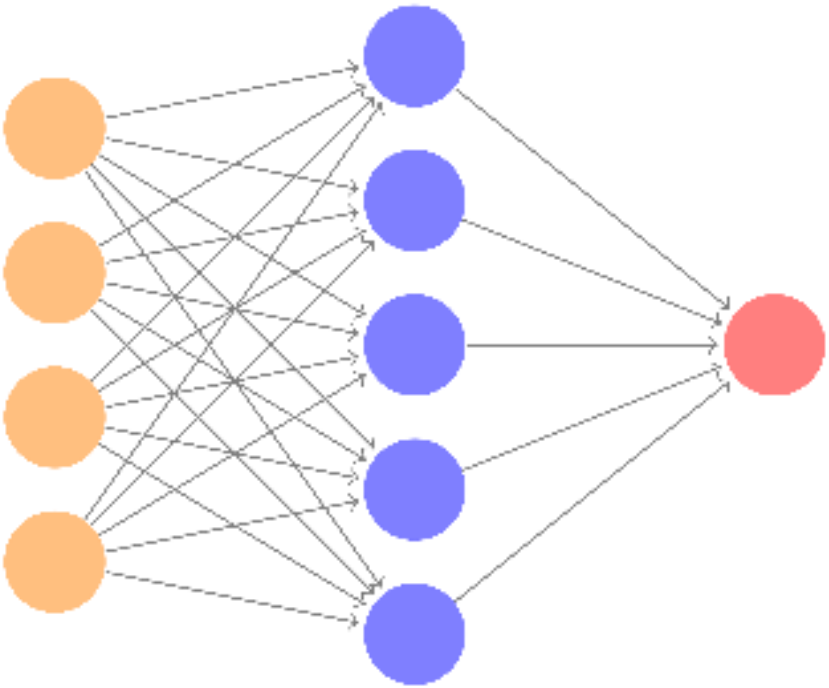
Regularization

Many variants of regularization

- "Weight Decay" (aka ridge penalty)
- Weight Sharing
- Early Stopping
- Data Augmentation
- Dropout

Some are approximately equivalent to ridge penalty.

Dropout



Dropout

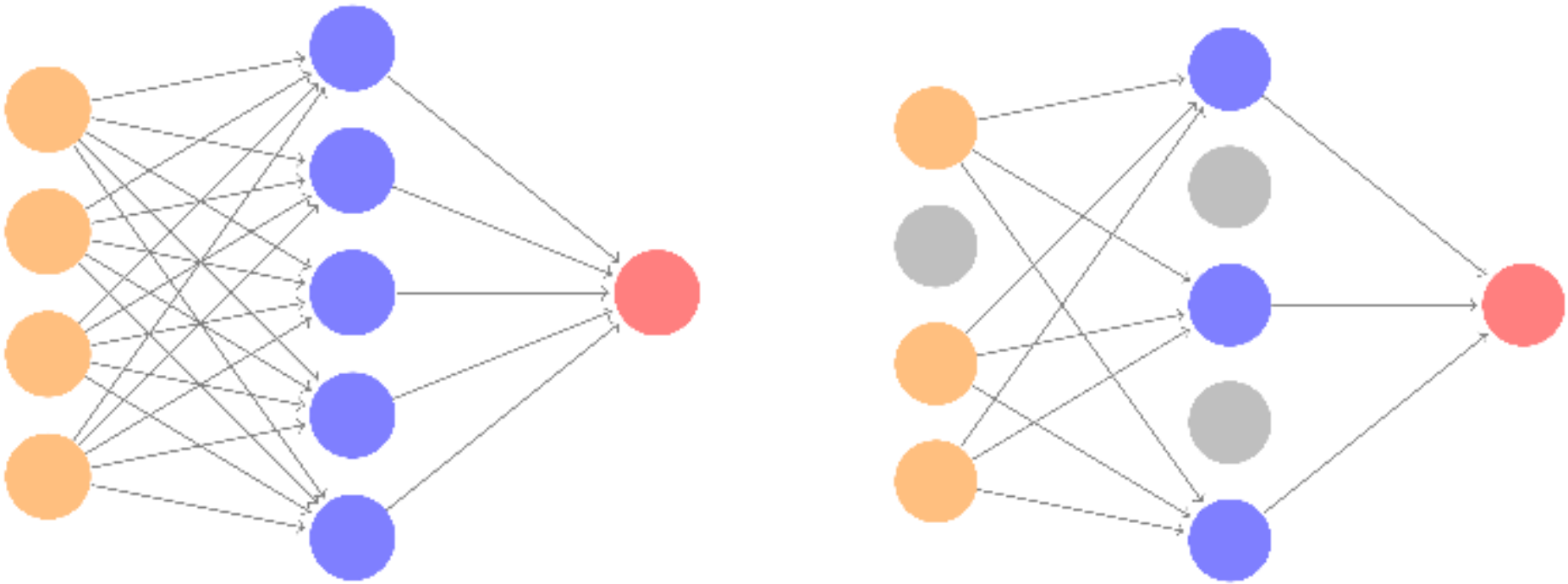


Image Credit: G. James, D. Witten, T. Hastie and R. Tibshirani. *An Introduction to Statistical Learning, with applications in R: Second Edition*

Implementation

Implementation

- `neuralnet`: simple neural networks from before deep learning

Implementation

- `neuralnet`: simple neural networks from before deep learning
- `keras`: a frontend for Tensorflow

Implementation

- `neuralnet`: simple neural networks from before deep learning
- `keras`: a frontend for Tensorflow
- `torch` and `torchvision`: an R version of PyTorch

Implementation in `torch`

Implementation in `torch`

1) Standardize the inputs.

Implementation in `torch`

1) Standardize the inputs.
2) Use `nn_module()` to define the model structure.

Implementation in `torch`

1) Standardize the inputs.
2) Use `nn_module()` to define the model structure.
3) `setup` for the loss, optimizer, and reporting.

Implementation in `torch`

1) Standardize the inputs.
2) Use `nn_module()` to define the model structure.
3) `setup` for the loss, optimizer, and reporting.
4) `set_hparams` to define inputs to initialize.

Implementation in `torch`

1) Standardize the inputs.
2) Use `nn_module()` to define the model structure.
3) `setup` for the loss, optimizer, and reporting.
4) `set_hparams` to define inputs to initialize.
5) `fit` to actually estimate the model.

Implementation in `torch`

1) Standardize the inputs.
2) Use `nn_module()` to define the model structure.
3) `setup` for the loss, optimizer, and reporting.
4) `set_hparams` to define inputs to initialize.
5) `fit` to actually estimate the model.

Two important notes:

- returns pointers vs. regular R objects
- we use fast settings to decrease computation time

1) Standardize the Inputs

```
x <- scale(model.matrix(Salary ~ . - 1, data = Gitters))
y <- Gitters$Salary
```

2) Define Model Structure

```
modnn <- nn_module(
  initialize = function(input_size) {
    self$hidden <- nn_linear(input_size, 50)
    self$activation <- nn_relu()
    self$dropout <- nn_dropout(0.4)
    self$output <- nn_linear(50, 1)
  },
  forward = function(x) {
    x %>%
      self$hidden() %>%
      self$activation() %>%
      self$dropout() %>%
      self$output()
  }
)
```

3-4) Define Other Model Components

```
modnn <- modnn %>%
  setup(
    loss = nn_mse_loss(),
    optimizer = optim_rmsprop,
    metrics = list(luz_metric_mae())
  ) %>%
  set_hparams(input_size = ncol(x))
```

5) Fit the Model

```
fitted <- modnn %>%
  fit(
    data = list(x[-testid, ],
    matrix(y[-testid], ncol = 1)),
    valid_data = list(x[testid, ],
    matrix(y[testid], ncol = 1)),
    epochs = 20
  )
```

What Can We Do With This?

- Call `predict(fitted, newdata = )` to get predictions.

What Can We Do With This?

- Call `predict(fitted, newdata = )` to get predictions.
- New! Use `as_array()` to convert the pointer to a regular R object!

What Can We Do With This?

- Call `predict(fitted, newdata = )` to get predictions.
- New! Use `as_array()` to convert the pointer to a regular R object!
- Use `plot(fitted)` to show error metrics during optimization.

1. Prediction in a New Population
2. Revisiting Verbal Autopsies
3. Google Flu
4. Landlords
5. Random Forests
6. Boosting
7. Neural Networks
   - Feedforward Neural Networks
   - Deep Learning

The Core Idea

- Including many layers in the neural network can be effective.

The Core Idea

- Including many layers in the neural network can be effective.
- Narrowing the number of hidden nodes as layers get further from data.

The Core Idea

- Including many layers in the neural network can be effective.
- Narrowing the number of hidden nodes as layers get further from data.
- Many innovations to make the above happen!

The Core Idea

- Including many layers in the neural network can be effective.
- Narrowing the number of hidden nodes as layers get further from data.
- Many innovations to make the above happen!
- When does this work?

The Core Idea

- Including many layers in the neural network can be effective.
- Narrowing the number of hidden nodes as layers get further from data.
- Many innovations to make the above happen!
- When does this work?
  complex data and high signal-to-noise ratio.

Multiple Layer Feedforward Neural Network

Multiple Layer Feedforward Neural Network

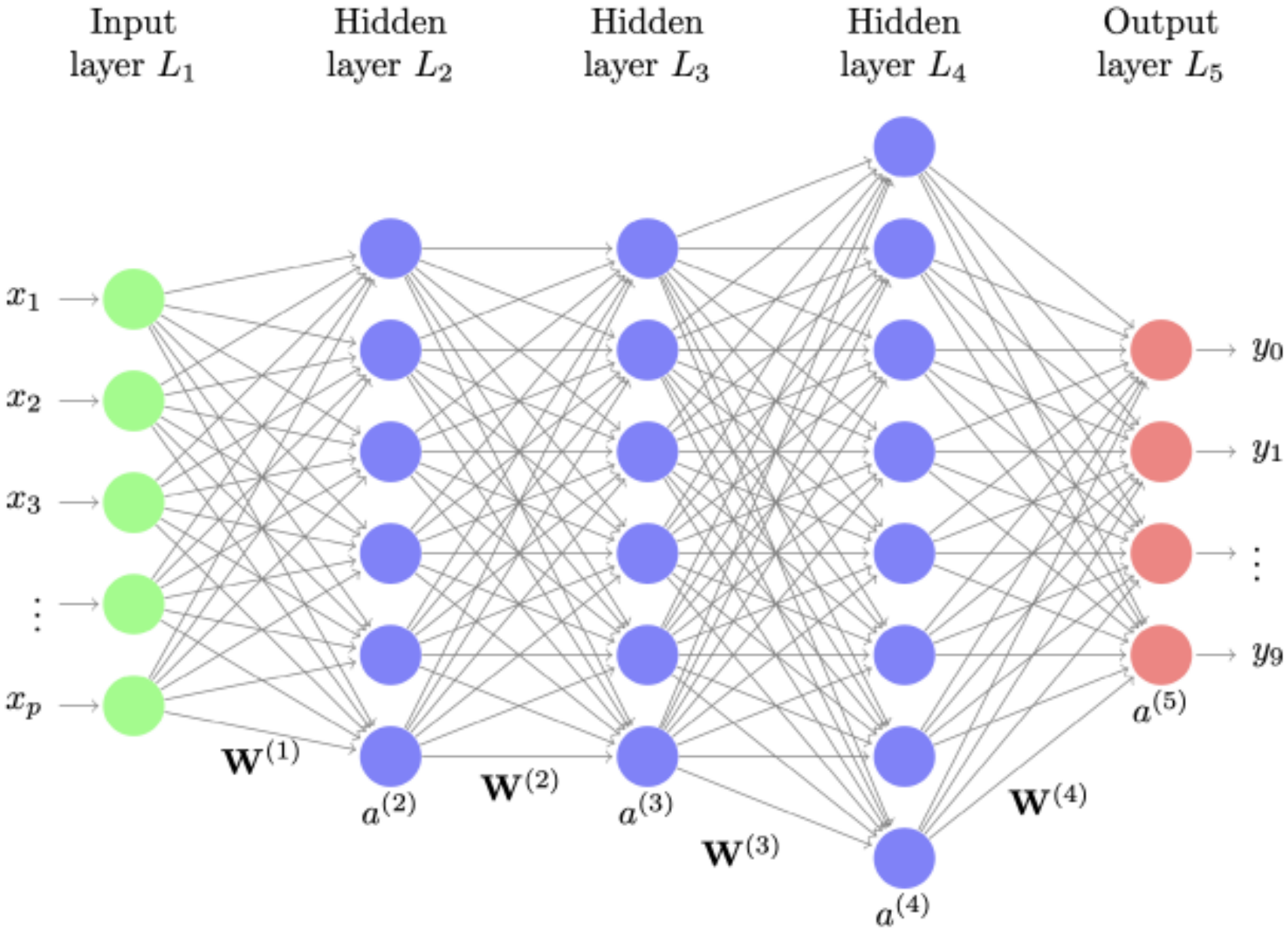
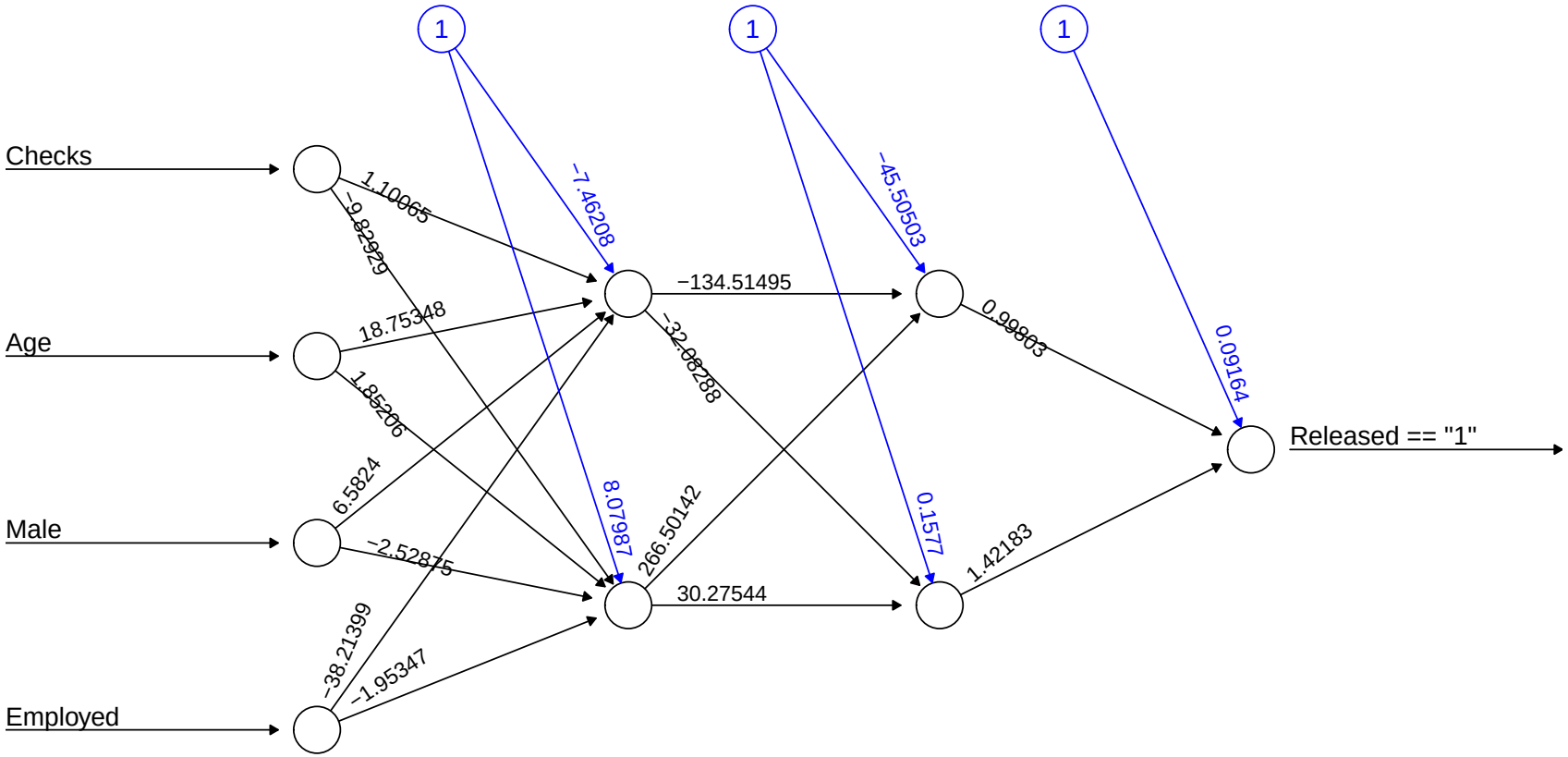


Image Credit: Efron and Hastie *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*

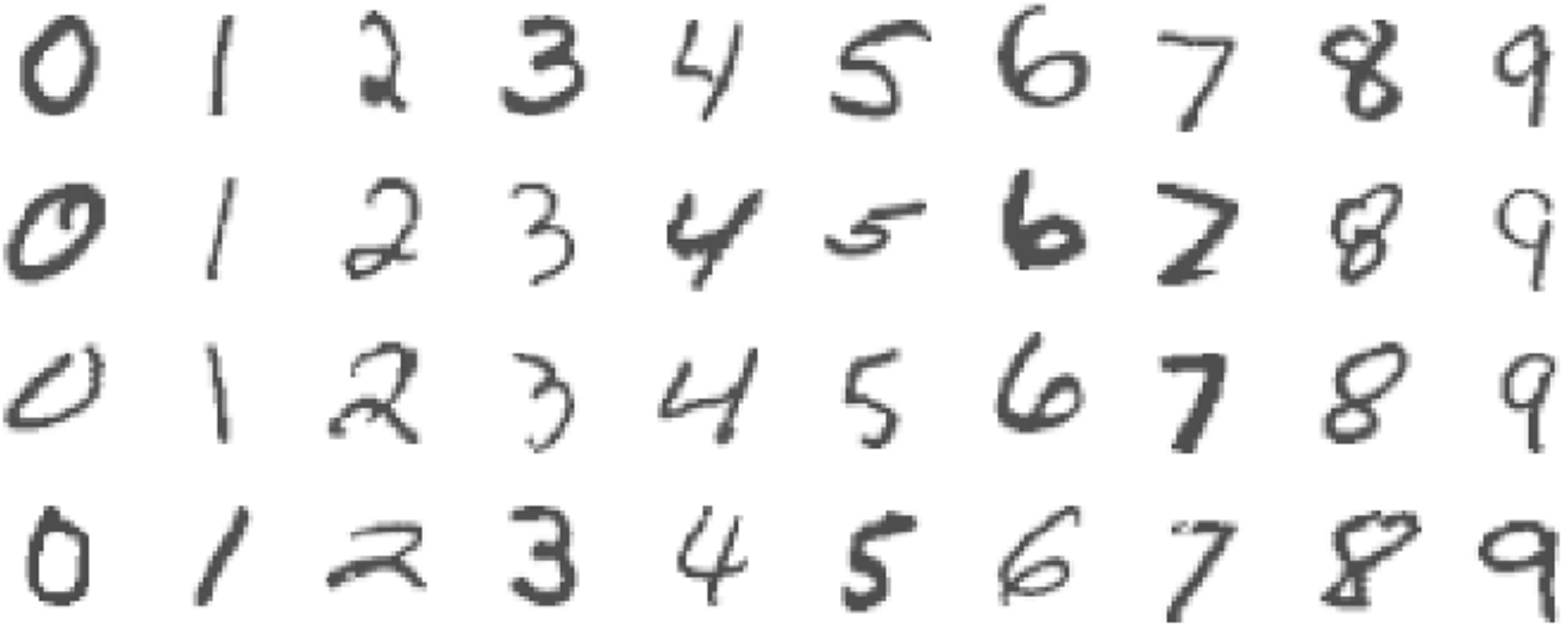
Multiple Layer Feedforward Neural Network

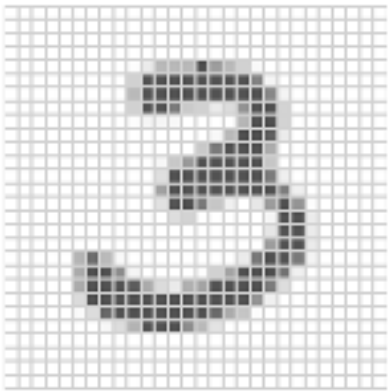
Multiple Layer Feedforward Neural Network

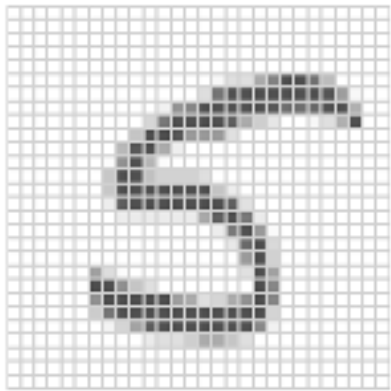


Error: 802.793986 Steps: 40904

Example: MNIST Digits







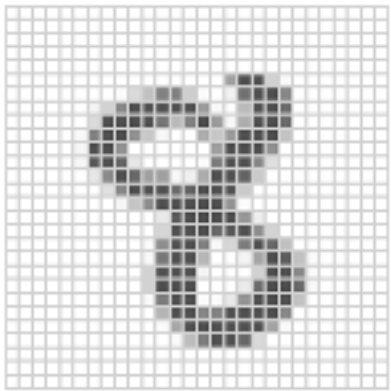


Image Credit: G. James, D. Witten, T. Hastie and R. Tibshirani. *An Introduction to Statistical Learning, with applications in R: Second Edition*

Implementation using torch

```
M_NN_struct <- nn_module(
  initialize = function() {
      self$linear1 <- nn_linear(in_features = 28*28, out_features = 256)
      self$linear2 <- nn_linear(in_features = 256, out_features = 128)
      self$linear3 <- nn_linear(in_features = 128, out_features = 10)

      self$drop1 <- nn_dropout(p = 0.4)
      self$drop2 <- nn_dropout(p = 0.3)

      self$activation <- nn_relu()
  },
  forward = function(x) {
     x %>%
     self$linear1() %>%
     self$activation() %>%
     self$drop1() %>%
     self$linear2() %>%
     self$activation() %>%
     self$drop2() %>%
     self$linear3()
}
)
```

Implementation using torch

```
modelnn <- modelnn %>%
  setup(
    loss = nn_cross_entropy_loss(),
    optimizer = optim_rmsprop,
    metrics = list(luz_metric_accuracy())
  )
```

Alternative Architectures

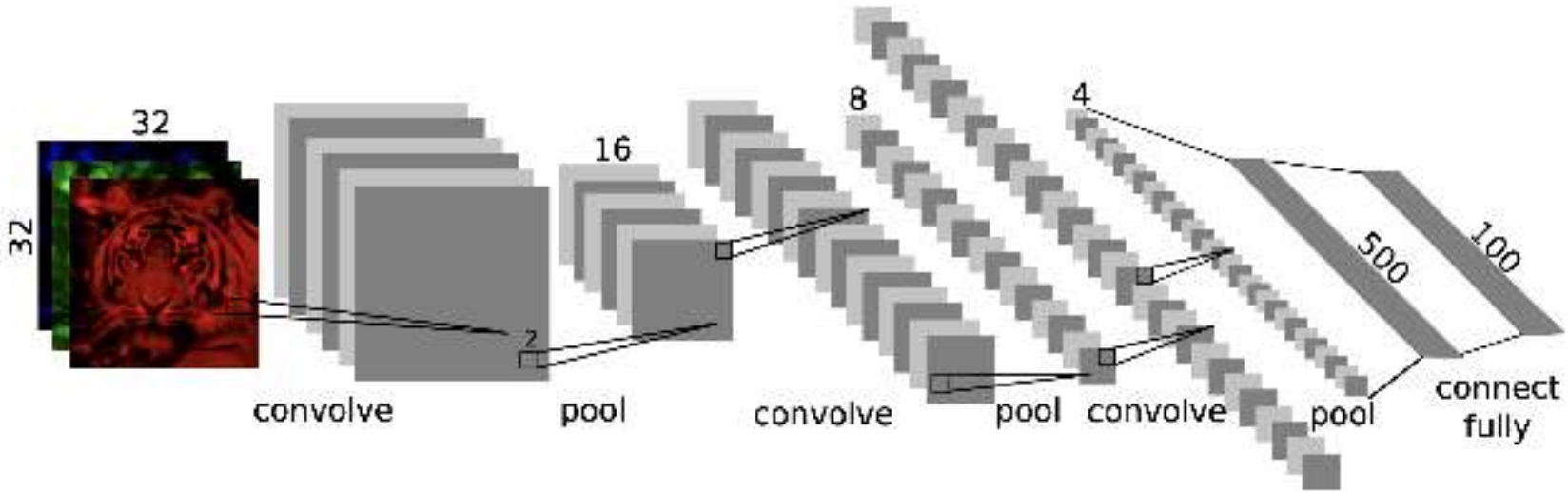


Image Credit: Efron and Hastie *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*

Convolutional Neural Networks (CNNs)

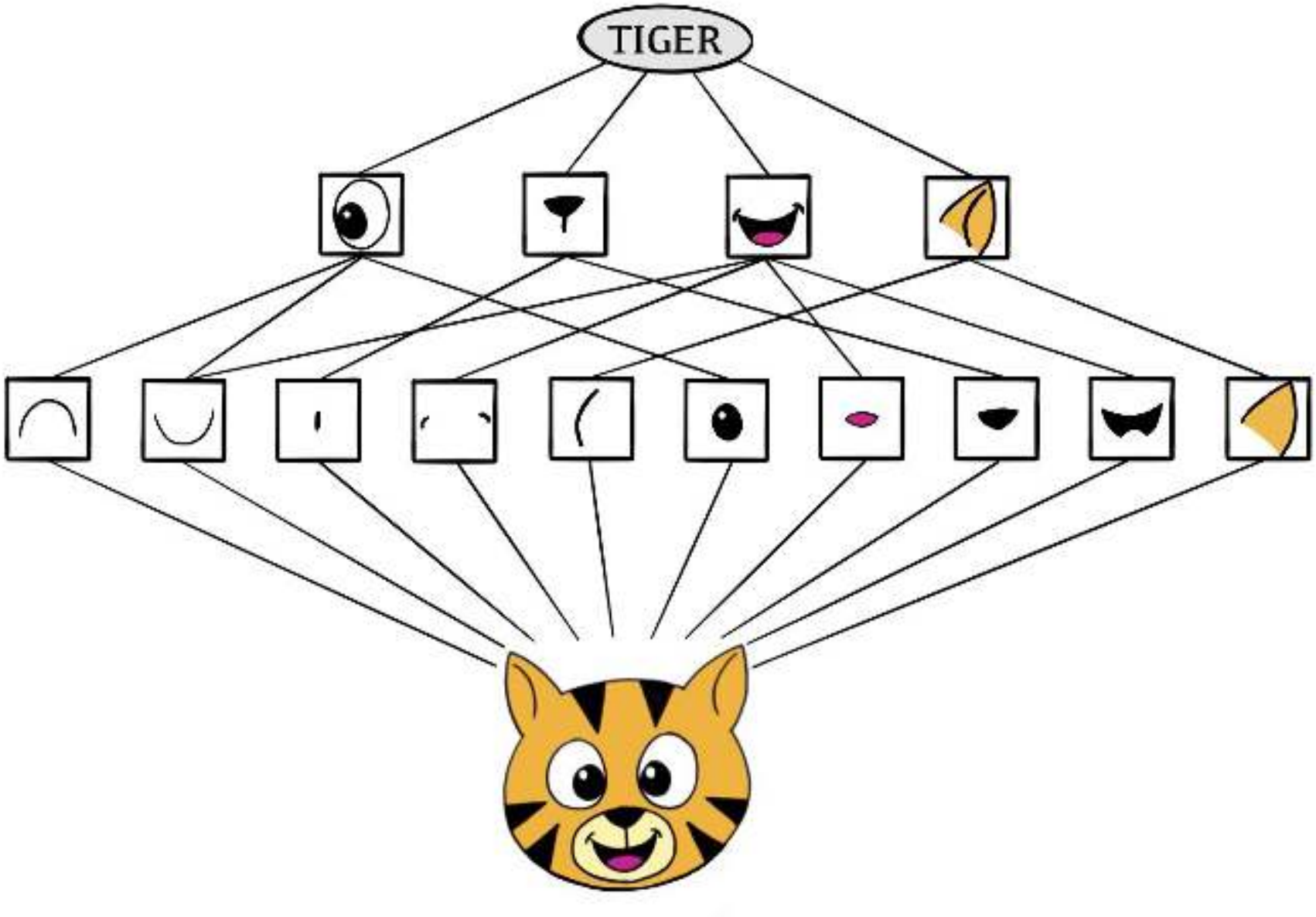
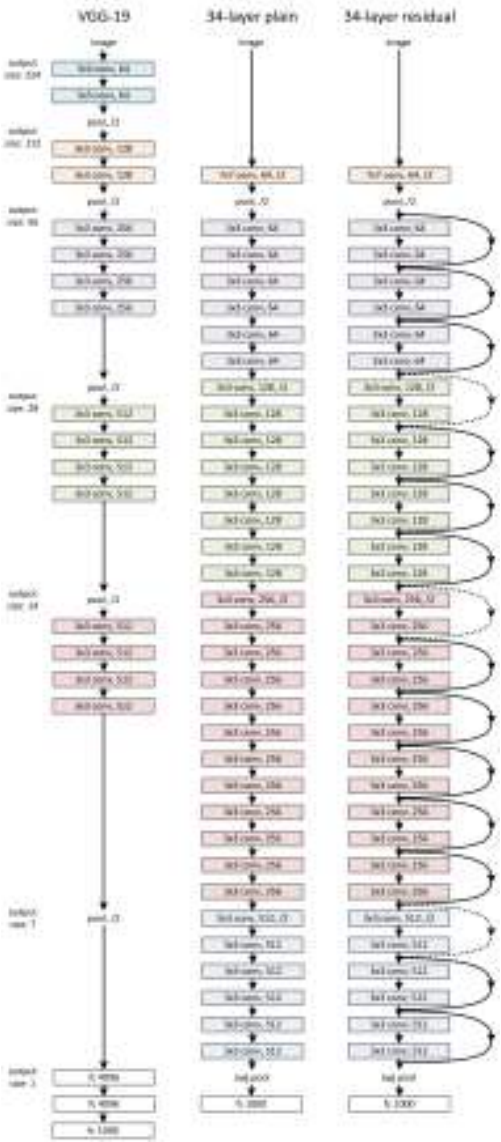


Image Credit: G. James, D. Witten, T. Hastie and R. Tibshirani. *An Introduction to Statistical Learning, with applications in R: Second Edition*

Resnet Architecture



He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

A Story About Architecture

A Story About Architecture

Turing Award Won by 3 Pioneers in Artificial Intelligence

From left, Yann LeCun, Geoffrey Hinton and Yoshua Bengio. The researchers worked on key developments for neural networks, which are reshaping how computer systems are built. From left, Facebook, via Associated Press; Aaron Vincent Elkaim for The New York Times; Chad Buchanan/Getty Images

But Also a Story About Data

But Also a Story About Data

Fei-Fei Li

From Wikipedia, the free encyclopedia

The native form of this personal name is Li Feifei. *This article uses Western name order when mentioning individuals.*

Fei-Fei Li (simplified Chinese: 李飞飞; traditional Chinese: 李飛飛; born 1976) is a Chinese-born American computer scientist, non-profit executive, and writer. She is the Sequoia Capital Professor of Computer Science at Stanford University.[2] Li is a Co-Director of the Stanford Institute for Human-Centered Artificial Intelligence, and a Co-Director of the Stanford Vision and Learning Lab.[3][4] She served as the director of the Stanford Artificial Intelligence Laboratory (SAIL)[5] from 2013 to 2018. In 2017, she co-founded AI4ALL, a nonprofit organization working to increase diversity and inclusion in the field of artificial intelligence.[6][7] Her research expertise includes artificial intelligence (AI), machine learning, deep learning, computer vision and cognitive neuroscience.[8] She was the leading scientist and principal investigator of ImageNet.[9]

Fei-Fei Li

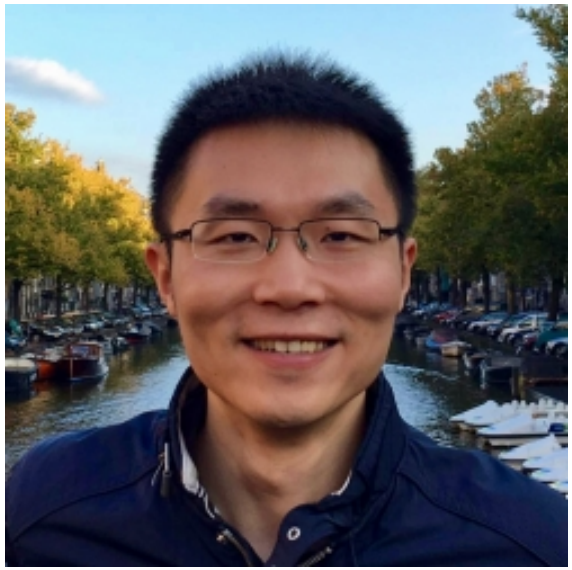
Li at AI for Good in 2017

Born	1976 (age 44–45)[1] Beijing
Nationality	American
Alma mater	Princeton University (B.A. in Physics)

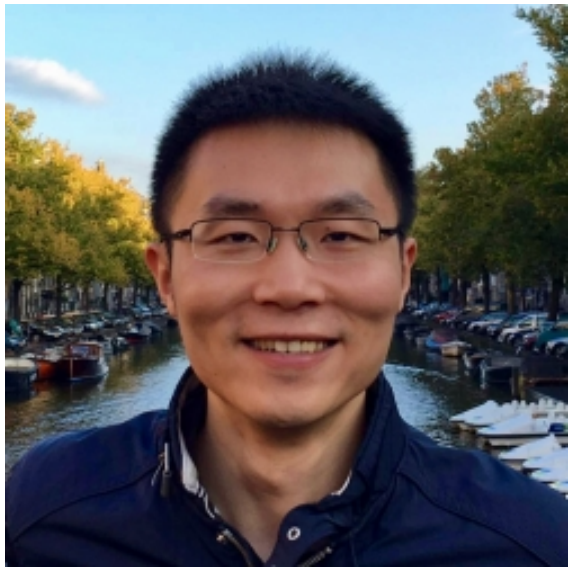
Princeton Connections!

Princeton Connections!

Princeton Connections!



Princeton Connections!



Tuning Parameters

Tuning Parameters

- number of layers

Tuning Parameters

- number of layers
- number of nodes per layer

Tuning Parameters

- number of layers
- number of nodes per layer
- regularization

Tuning Parameters

- number of layers
- number of nodes per layer
- regularization
- optimization technique

Tuning Parameters

- number of layers
- number of nodes per layer
- regularization
- optimization technique
- architecture

Powerful but difficult to use

Powerful but difficult to use

If you have ever tried to train a NN on your own, then you may know how painful and uncertain it can be. I am not talking about following the existing tutorials and demos, when all hyperparameters are already tuned for you, but about taking some data and creating something from scratch. Even with modern DL [deep learning] high-level toolkits, where all best practices such as proper weights initialization and optimizers' betas, gammas, and other options are set to sane defaults, and tons of other stuff hidden under the hood, there are still lots of decisions that you can make, hence lots of things can go wrong. As a result, your network almost never works from the first run and this is something that you should get used to.

(Lapan 2018, 66)

Summary

Summary

- neural networks

Summary

- neural networks
- deep learning as an extension of basic neural networks

Summary

- neural networks
- deep learning as an extension of basic neural networks

Going Deeper:

G. James, D. Witten, T. Hastie and R. Tibshirani. *An Introduction to Statistical Learning, with applications in R: Second Edition* Chapter 10

This Week in Review

This Week in Review

- Random Forests and Boosting
- Prediction in a different population
- The difficulty of evaluating in this setting
- Neural Networks

This Week in Review

- Random Forests and Boosting
- Prediction in a different population
- The difficulty of evaluating in this setting
- Neural Networks

Next week: Prediction in a Counterfactual Population

- Athey, Susan, 2017. Beyond prediction: Using big data for policy problems. *Science*, 355(6324), pp.483-485.
- Lundberg, Ian, Rebecca Johnson, and Brandon M. Stewart. 2021. "What Is Your Estimand? Defining the Target Quantity Connects Statistical Evidence to Theory." *ASR* 86, no. 3: 532–65.
- Kleinberg, J., Lakkaraju, H., Leskovec, J., Ludwig, J., and Mullainathan, S. 2018. Human decisions and machine predictions. *QJE*, 133(1), 237-293.