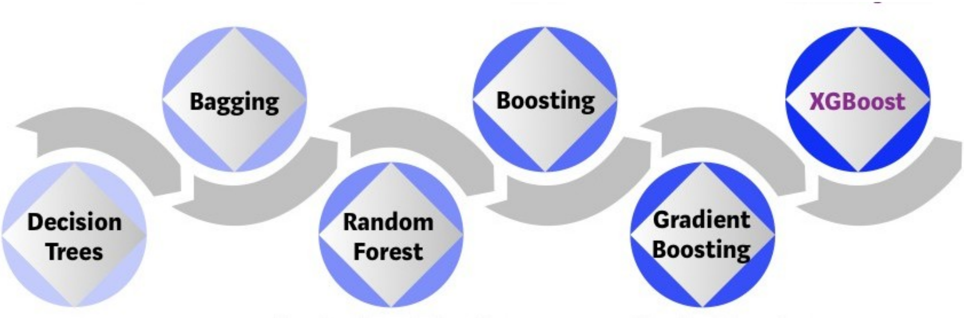
Natural Language Processing for Law and Social Science

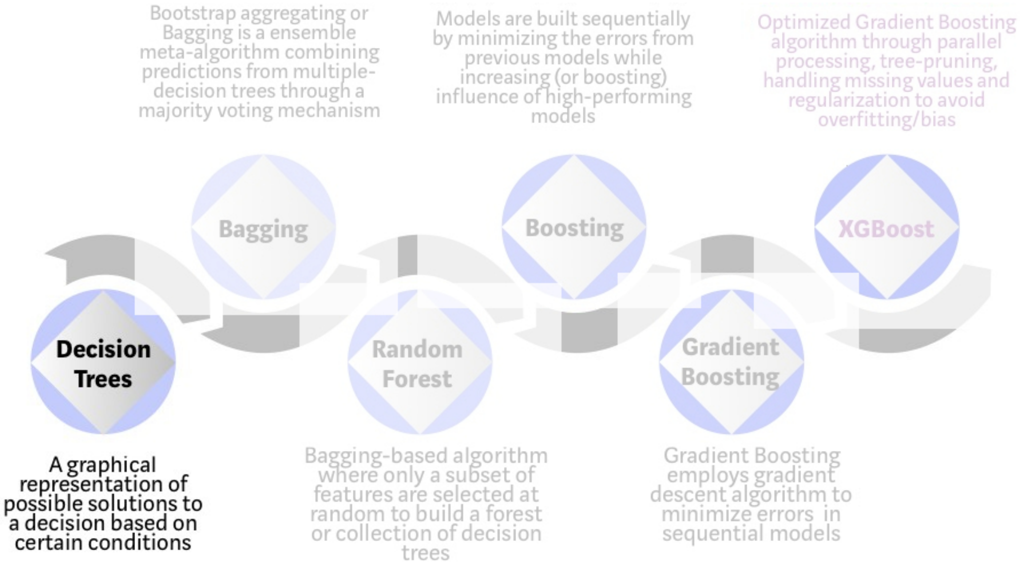
5. Ensemble Learning and Deep Learning

Outline

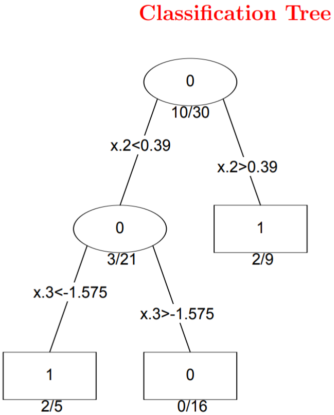
XGBoost: Overview



XGBoost Ingredients: Decision Trees

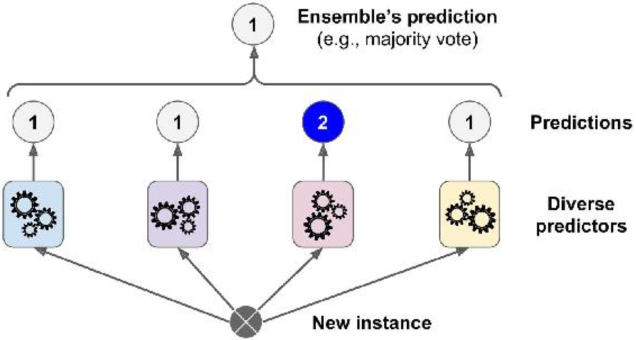


Decision Trees



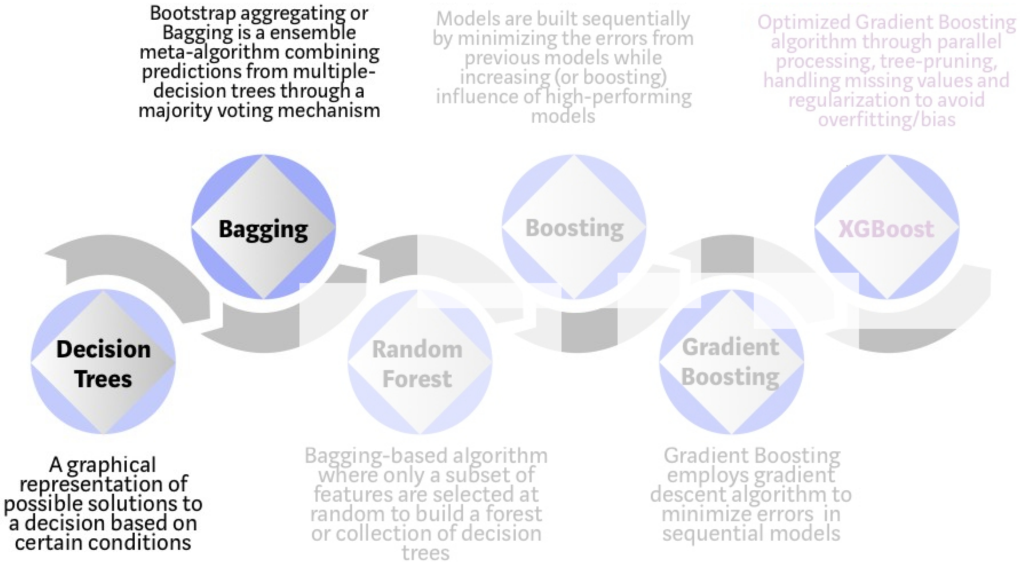
- Decision trees learn a series of binary splits in the data based on hard thresholds.
  - if yes, go right; if no, go left.
- Can have additional splits as you move through the tree.
- fast and interpretable, but performance is often poor.

Voting Classifiers



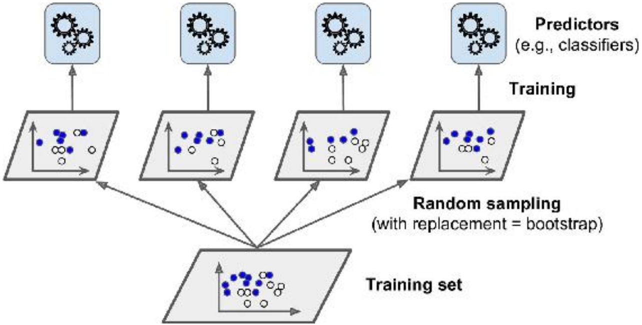
- voting classifiers (ensembles of different models that vote on the prediction) generally out-perform the best classifier in the ensemble.
  - more diverse algorithms will make different types of errors, and improve your ensemble's robustness.

XGBoost Ingredients: Bootstrapping



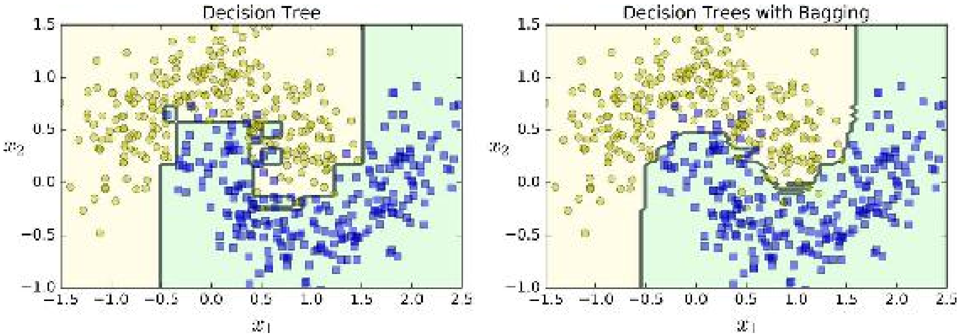
Bootstrapping

- Rather than use the same data on different classifiers, one can use different subsets of the data on the same classifier:



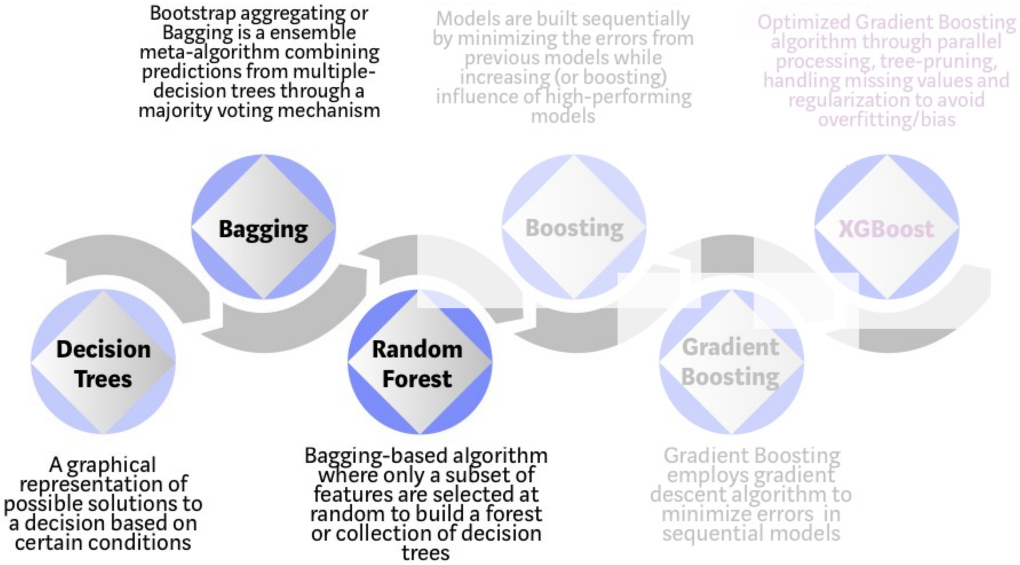
  - can also use different subsets of features across subclassifiers.

Bootstrapping Benefits



- A bootstraped ensemble generally has a similar bias but lower variance than a single predictor trained on all the data.
- Predictors can be trained in parallel using separate CPU cores.

XGBoost Ingredients: Random Forests



Random Forests

Random Forests are optimized ensembles of bootstrapped decision trees:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X,y)
```

Random Forests

Random Forests are optimized ensembles of bootstrapped decision trees:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X,y)
```

1. Each voting tree gets its own sample of data.

Random Forests

Random Forests are optimized ensembles of bootstrapped decision trees:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X,y)
```

1. Each voting tree gets its own sample of data.
2. At each tree split, a random sample of features is drawn, only those features are considered for splitting.

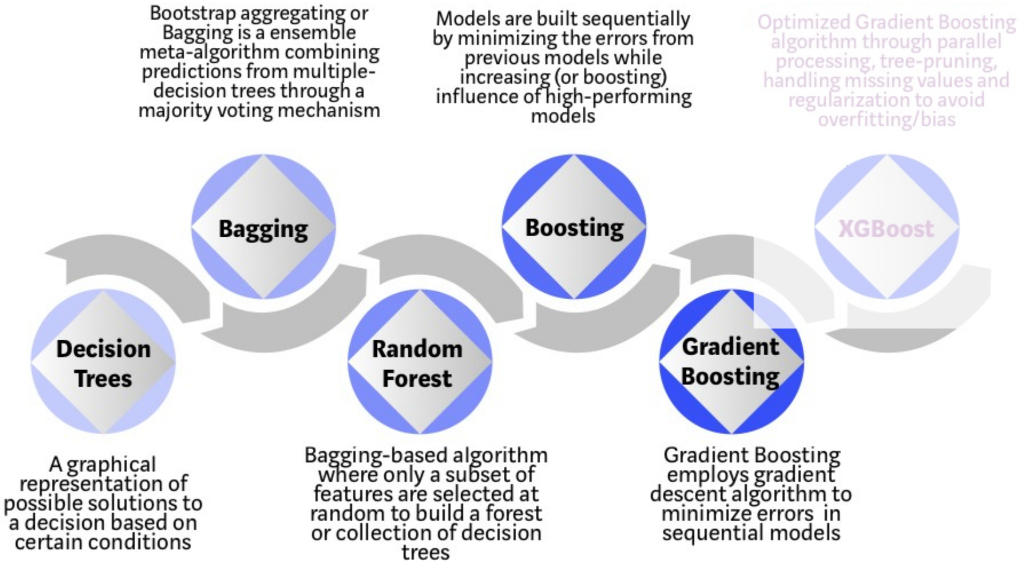
Random Forests

Random Forests are optimized ensembles of bootstrapped decision trees:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X,y)
```

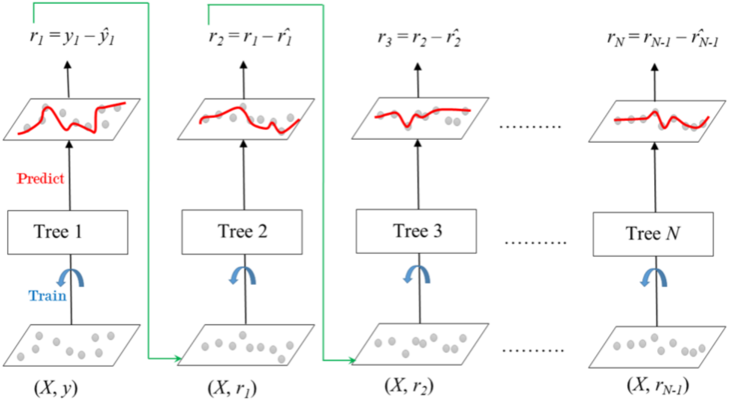
1. Each voting tree gets its own sample of data.
2. At each tree split, a random sample of features is drawn, only those features are considered for splitting.
3. For each tree, error rate is computed using data outside its bootstrap sample.

XGBoost Ingredients: Gradient Boosting



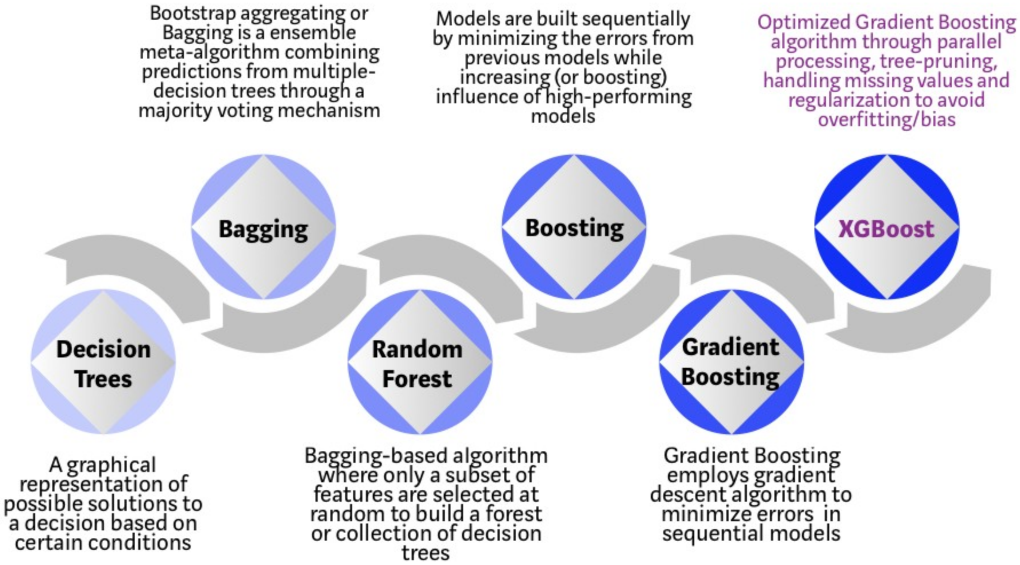
Gradient Boosting Machines

- Gradient boosting refers to an additive ensemble of trees:



- Adds additional layers of trees to fit the residuals of the first layers

XGBoost Ingredients



XGBoost

- Feurer et al (2018) find that XGBoost beats a sophisticated AutoML procedure with grid search over 15 classifiers and 18 data preprocessors.
- A good starting point for any machine learning task.

- easy to use
- actively developed
- efficient / parallelizable
- provides model explanations
- takes sparse matrices as input

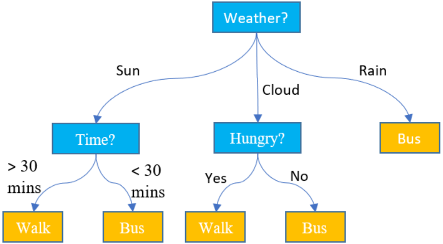
```python
from xgboost import XGBClassifier
model = XGBClassifier()

model.fit(X_train, y_train,
          early_stopping_rounds=10,
          eval_metric="logloss",
          eval_set=[(X_eval, y_eval)]
          )

y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

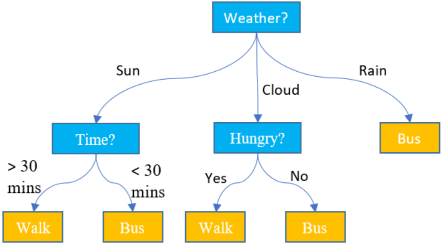
Tree Ensembles are Black Boxes

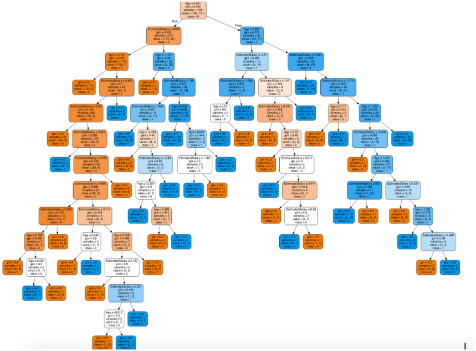
- Small decision trees have the advantage of being highly interpretable.



Tree Ensembles are Black Boxes

- Small decision trees have the advantage of being highly interpretable.





- Larger trees and ensembles (e.g. XGBoost) lose this nice feature.
- Best-performing ML models are hard to interpret because they use lots of features and exploit non-linearities and interactions.

Interpreting Tree Ensembles

XGBoost's Feature Importance Metric:

- At each decision node, compute **information gain** for feature j **(change in predicted probability**).
- Average across all nodes for each j.

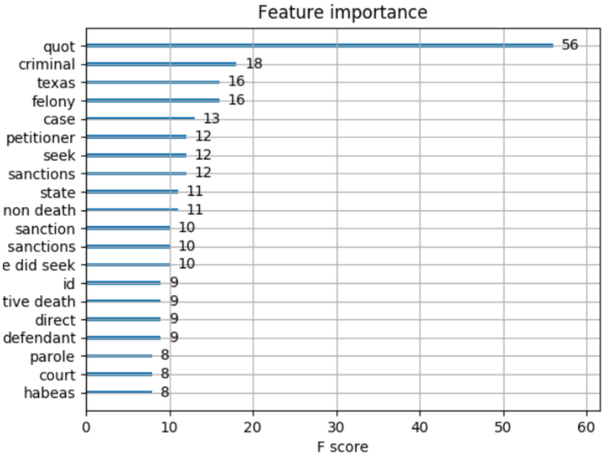
Ranks predictors by their relative contributions.

```
from xgboost import plot_importance
plot_importance(xgb_reg, max_num_features=10)
```

Feature Importance

```
from xgboost import plot_importance
plot_importance(xgb_reg, max_num_features=20)
```

```
<IPython.core.display.Javascript object>
```



- XGBoost provides a metric of feature importance that summarizes how well each feature contributes to predictive accuracy.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use cross entropy; for regression, use mean squared error.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use cross entropy; for regression, use mean squared error.
4. Evaluate model in held-out test set:
   - For classification, use balanced accuracy, confusion matrix, and calibration plot.
   - For regression, use R squared and binscatter plot.

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use cross entropy; for regression, use mean squared error.
4. Evaluate model in held-out test set:
   - For classification, use balanced accuracy, confusion matrix, and calibration plot.
   - For regression, use R squared and binscatter plot.
5. Interpret the model predictions:
   - for gradient boosting, use feature importance ranking.
   - for linear models, examine coefficients
   - look at highest and lowest ranked documents for $\hat{y}$

A baseline for machine learning using text

1. Take POS-filtered bigrams as inputs X.
2. Select a machine learning model for predicting outcome y:
   - For regression (y is one-dimensional and continuous), elastic net or gradient boosted regressor.
   - For classification (y is discrete), L2-penalized logistic regression or gradient boosted classifier.
   - *(If y is more complicated, e.g. a sequence of words, we use deep learning.)*
3. Use cross-validation grid search in training set to select model hyperparameters.
   - For classification, use cross entropy; for regression, use mean squared error.
4. Evaluate model in held-out test set:
   - For classification, use balanced accuracy, confusion matrix, and calibration plot.
   - For regression, use R squared and binscatter plot.
5. Interpret the model predictions:
   - for gradient boosting, use feature importance ranking.
   - for linear models, examine coefficients
   - look at highest and lowest ranked documents for $\hat{y}$
6. Answer the research question!

Outline

Causal inference is needed to improve the world

Consider important policy questions like:

- In light of coronavirus, should schools reopen or not for in-person teaching?

Causal inference is needed to improve the world

Consider important policy questions like:

- In light of coronavirus, should schools reopen or not for in-person teaching?
  - No matter how much we know from lab experiments about the biology/epidemiology of the virus, there will be too much uncertainty about costs/benefits to answer this.
  - We need real-world evidence, but we can't experimentally force schools to reopen or not.

Causal inference is needed to improve the world

Consider important policy questions like:

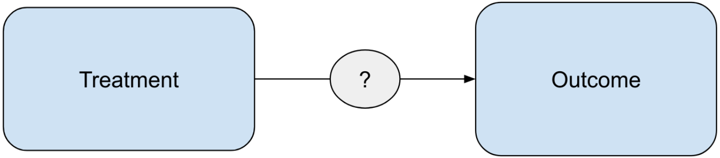
- In light of coronavirus, should schools reopen or not for in-person teaching?
  - No matter how much we know from lab experiments about the biology/epidemiology of the virus, there will be too much uncertainty about costs/benefits to answer this.
  - We need real-world evidence, but we can't experimentally force schools to reopen or not.
- Can use a **natural experiment** to produce causal estimates:
  - e.g., variation in number of coronavirus cases before/after openings, using differences in the timing of openings (differences-in-differences).

Causal inference is needed to improve the world

Consider important policy questions like:

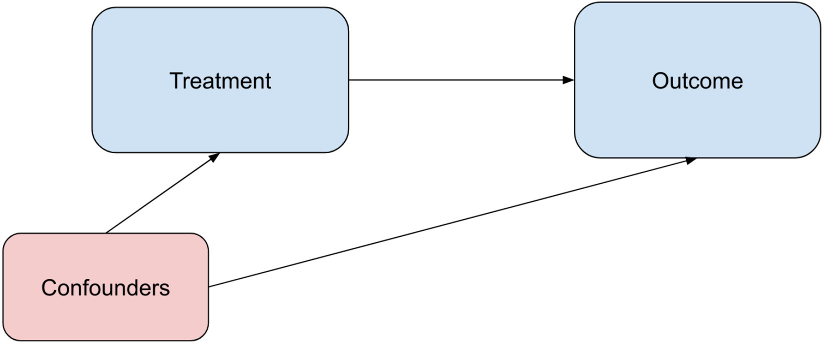
- In light of coronavirus, should schools reopen or not for in-person teaching?
  - No matter how much we know from lab experiments about the biology/epidemiology of the virus, there will be too much uncertainty about costs/benefits to answer this.
  - We need real-world evidence, but we can't experimentally force schools to reopen or not.
- Can use a **natural experiment** to produce causal estimates:
  - e.g., variation in number of coronavirus cases before/after openings, using differences in the timing of openings (differences-in-differences).
- Google/Facebook understand the importance of causal inference with A/B testing; social scientists want to use it to assist public policy.

Causal Graphs



- We are interested in estimating a causal effect (if any) of a "treatment" on an "outcome".

- ***Unobserved Confounders*** are variables that affect both the treatment and the outcome, which we don't have in our dataset:



- **Observed confounders** are not a problem, because we can adjust (control) for them in causal inference analysis (that is, including them in a regression).

- **Reverse causation**: "the outcome" affects "the "treatment".
  Joint causation: there is bidirectional causation.

- e.g., effect of tax collections on economic growth.

- **Reverse causation**: "the outcome" affects "the "treatment".
  Joint causation: there is bidirectional causation.

- e.g., effect of tax collections on economic growth.
- Resulting estimates are biased (not causal), and cannot be fixed by adjusting for observed confounders.

With joint causality, or with unobserved confounders, it is often impossible to produce statistical estimates with a causal interpretation.

With joint causality, or with unobserved confounders, it is often impossible to produce statistical estimates with a causal interpretation.

- The gold standard: randomized control trials.
  - often not available, e.g. with opening/closing schools under covid-19.

With joint causality, or with unobserved confounders, it is often impossible to produce statistical estimates with a causal interpretation.

- The gold standard: randomized control trials.
  - often not available, e.g. with opening/closing schools under covid-19.
- Second best: natural experiments.
  - differences-in-differences: use longitudinal data and look at groups or places that adopted treatment at different times.

With joint causality, or with unobserved confounders, it is often impossible to produce statistical estimates with a causal interpretation.

- The gold standard: randomized control trials.
  - often not available, e.g. with opening/closing schools under covid-19.
- Second best: natural experiments.
  - differences-in-differences: use longitudinal data and look at groups or places that adopted treatment at different times.
  - regression discontinuity: compare individuals just above or just below some discrete scoring threshold.

With joint causality, or with unobserved confounders, it is often impossible to produce statistical estimates with a causal interpretation.

- The gold standard: randomized control trials.
  - often not available, e.g. with opening/closing schools under covid-19.
- Second best: natural experiments.
  - differences-in-differences: use longitudinal data and look at groups or places that adopted treatment at different times.
  - regression discontinuity: compare individuals just above or just below some discrete scoring threshold.
  - instrumental variables: use a third variable ("instrument") that randomly shifts the probability of treatment.

Fong and Grimmer (2016): Causal effect of political messaging

- What biographical characteristics of politicians influence voter evaluations?

Fong and Grimmer (2016): Causal effect of political messaging

- What biographical characteristics of politicians influence voter evaluations?
- Could run a survey experiment:
  - `Document 1: He earned his Juris Doctor in 1997 from Yale Law School, where he operated free legal clinics for low-income residents of New Haven, Connecticut...`
  - `Document 2: He served in South Vietnam from 1970 to 1971 during the Vietnam War in the Army Rangers' 75th Ranger Regiment, attached to the 173rd Airborne Brigade. He participated in 24 helicopter assaults...`
- But hard to generalize what features drive differences.

Fong and Grimmer (2016): Approach

- Lab experiment: 1,886 participants, 5,303 responses

1. Randomly assign texts, X_i , to respondents i
   - Sees up to 3 texts from the corpus of $>$ 2200 Wikipedia biographies
2. Obtain responses Y_i for each respondent
   - Feeling thermometer rating: 0-100

Fong and Grimmer (2016): Approach

- Lab experiment: 1,886 participants, 5,303 responses

1. Randomly assign texts, X_i , to respondents i
   - Sees up to 3 texts from the corpus of $>$ 2200 Wikipedia biographies
2. Obtain responses Y_i for each respondent
   - Feeling thermometer rating: 0-100
3. Structural topic model variant ("supervised indian buffet process"):
   - Discover mapping from texts X to latent topic treatments $\vec{D}$ based on their effect on Y.

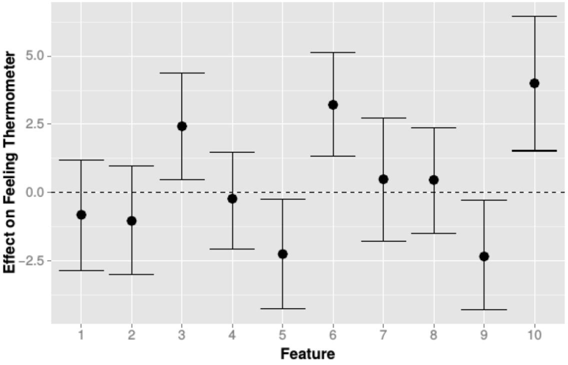
Fong and Grimmer (2016): Approach

- Lab experiment: 1,886 participants, 5,303 responses

1. Randomly assign texts, X_i , to respondents i
   - Sees up to 3 texts from the corpus of $>$ 2200 Wikipedia biographies
2. Obtain responses Y_i for each respondent
   - Feeling thermometer rating: 0-100
3. Structural topic model variant ("supervised indian buffet process"):
   - Discover mapping from texts X to latent topic treatments $\vec{D}$ based on their effect on Y.
4. Measure causal effects of these treatments on Y_i

Fong and Grimmer (2016): Results

Treatment	Keywords
3	director, university, received, president, phd, policy
5	elected, house, democratic, seat
6	united_states, military, combat, rank
9	law, school_law, law_school, juris_doctor, student
10	war, enlisted, united_states, assigned, army



Text matching for causal inference: Application to online censorship in China

Roberts, Stewart, and Nielsen (2018)

- Construct a corpus of chinese social media posts, some of which are censored.
  - 593 bloggers, 150,000 posts, 6 months

Text matching for causal inference: Application to online censorship in China

Roberts, Stewart, and Nielsen (2018)

- Construct a corpus of chinese social media posts, some of which are censored.
  - 593 bloggers, 150,000 posts, 6 months
- They use a variation of propensity score matching to identify almost identical posts, some of which were censored, and some of which were not.

Text matching for causal inference: Application to online censorship in China

Roberts, Stewart, and Nielsen (2018)

- Construct a corpus of chinese social media posts, some of which are censored.
  - 593 bloggers, 150,000 posts, 6 months
- They use a variation of propensity score matching to identify almost identical posts, some of which were censored, and some of which were not.
- Outcome:
  - Using text of subsequent posts, measure how likely they are to be censored (how censorable)

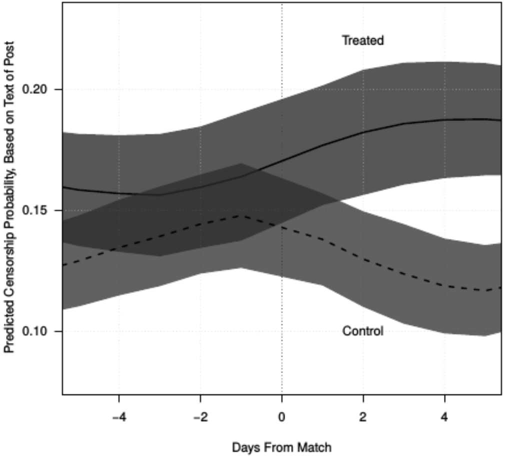
Text matching for causal inference: Application to online censorship in China

Roberts, Stewart, and Nielsen (2018)

- Construct a corpus of chinese social media posts, some of which are censored.
  - 593 bloggers, 150,000 posts, 6 months
- They use a variation of propensity score matching to identify almost identical posts, some of which were censored, and some of which were not.
- Outcome:
  - Using text of subsequent posts, measure how likely they are to be censored (how censorable)
  - Can see whether censorship has a deterrence or backlash effect.

Censorship has a backlash effect

Roberts, Stewart, and Nielsen (2018)



- Bloggers who are censored respond with more censorable content.

Outline

What have we been doing? ***Learning representations*** of the data

- Dictionary methods: document is represented as a count over the lexicon
- N-grams: document is a count over a vocabulary of phrases
- Topic models: document is a vector of shares over topics

What have we been doing? ***Learning representations*** of the data

- Dictionary methods: document is represented as a count over the lexicon
- N-grams: document is a count over a vocabulary of phrases
- Topic models: document is a vector of shares over topics
- Text classifiers: produces $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \hat{\theta})$, a vector of predicted probabilities across classes for each document i.

What have we been doing? ***Learning representations*** of the data

- Dictionary methods: document is represented as a count over the lexicon
- N-grams: document is a count over a vocabulary of phrases
- Topic models: document is a vector of shares over topics
- Text classifiers: produces $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \hat{\theta})$, a vector of predicted probabilities across classes for each document i.
  - This vector of class probabilities is a **compressed representation** of the outcome-predictive text features $\boldsymbol{x}_i$

What have we been doing? ***Learning representations*** of the data

- Dictionary methods: document is represented as a count over the lexicon
- N-grams: document is a count over a vocabulary of phrases
- Topic models: document is a vector of shares over topics
- Text classifiers: produces $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \hat{\theta})$, a vector of predicted probabilities across classes for each document i.
  - This vector of class probabilities is a **compressed representation** of the outcome-predictive text features $\boldsymbol{x}_i$
  - the vector of features, $\boldsymbol{x}_i$, is itself a compressed representation of the unprocessed document $\mathcal{D}_i$.

What have we been doing? ***Learning representations*** of the data

- Dictionary methods: document is represented as a count over the lexicon
- N-grams: document is a count over a vocabulary of phrases
- Topic models: document is a vector of shares over topics
- Text classifiers: produces $\hat{\boldsymbol{y}}_i = f(\boldsymbol{x}_i; \hat{\theta})$, a vector of predicted probabilities across classes for each document i.
  - This vector of class probabilities is a **compressed representation** of the outcome-predictive text features $\boldsymbol{x}_i$
  - the vector of features, $\boldsymbol{x}_i$, is itself a compressed representation of the unprocessed document $\mathcal{D}_i$.
- Correspondingly: the learned parameters $\hat{\theta}$ can also be understood as a **learned compressed representation of the whole dataset**:
  - it contains information about the training corpus, the text features, and the outcomes.

Information in $\hat{\theta}$

Say we train a multinomial logistic regression on a bag-of-words representation of the documents:

- Let θ be the learned matrix of parameters relating words to outcomes:
  - It contains n_y columns, which are n_x-vectors representing the outcome classes.

Information in $\hat{\theta}$

Say we train a multinomial logistic regression on a bag-of-words representation of the documents:

- Let θ be the learned matrix of parameters relating words to outcomes:
  - It contains n_y columns, which are n_x-vectors representing the outcome classes.
  - It contains n_x rows, which are n_y-vectors representing each word in the vocabulary.

Information in $\hat{\theta}$

Say we train a multinomial logistic regression on a bag-of-words representation of the documents:

- Let θ be the learned matrix of parameters relating words to outcomes:
  - It contains n_y columns, which are n_x-vectors representing the outcome classes.
  - It contains n_x rows, which are n_y-vectors representing each word in the vocabulary.
- We could already use θ! e.g.:
  - cluster the column vectors $\rightarrow$ which outcomes are similar/related.
  - cluster the row vectors $\rightarrow$ which features are similar/related.

Information in $\hat{\theta}$: Preview of Word Embeddings

θ = matrix of parameters learned from logit, relating words to outcomes.

- If $\boldsymbol{x}$ is a bag-of-words representation for a document consisting of a list of tokens $\{w_1, ..., w_t, ..., w_n\}$, we can write

$$\boldsymbol{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

  - where x_t is an n_x-dimensional one-hot vector – all entries are zero except equals one for the word at t.

Information in $\hat{\theta}$: Preview of Word Embeddings

$\theta =$ matrix of parameters learned from logit, relating words to outcomes.

- If $\boldsymbol{x}$ is a bag-of-words representation for a document consisting of a list of tokens $\{w_1, ..., w_t, ..., w_n\}$, we can write

$$\boldsymbol{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

  - where x_t is an n_x-dimensional one-hot vector – all entries are zero except equals one for the word at t.
- Let θ_t be the row of θ corresponding to the word w_t: a **word embedding** for w_t containing the outcome-relevant information for that word.

Information in $\hat{\theta}$: Preview of Word Embeddings

θ = matrix of parameters learned from logit, relating words to outcomes.

- If $\boldsymbol{x}$ is a bag-of-words representation for a document consisting of a list of tokens $\{w_1, ..., w_t, ..., w_n\}$, we can write

$$\boldsymbol{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

  - where x_t is an n_x-dimensional one-hot vector – all entries are zero except equals one for the word at t.
- Let θ_t be the row of θ corresponding to the word w_t: a **word embedding** for w_t containing the outcome-relevant information for that word.
- We can construct a **document vector**

$$\vec{\boldsymbol{d}} = \frac{1}{n}\sum_{t=1}^{n_i} \theta_t$$

  the sum of the n_y-dimensional word representations (the row vectors from above).

  - this is called the "continuous bag of words (CBOW)" representation (Goldberg 2017).
  - Note that $\vec{\boldsymbol{d}} = \theta \cdot \boldsymbol{x}$, and thus θ is a **word embedding matrix**.

Outline

Outline

- Neural networks $\leftrightarrow$ deep learning models
  - solve machine learning problems, just like logistic regression or gradient boosted machines
  - use tensorflow/keras or torch, rather than sklearn or xgboost.

- Neural networks $\leftrightarrow$ deep learning models
  - solve machine learning problems, just like logistic regression or gradient boosted machines
  - use tensorflow/keras or torch, rather than sklearn or xgboost.
- **why use neural nets?**
  - sometimes outperform standard ML techniques on standard problems
  - greatly outperform standard ML techniques on some problems, for example language modeling

- Neural networks $\leftrightarrow$ deep learning models
  - solve machine learning problems, just like logistic regression or gradient boosted machines
  - use tensorflow/keras or torch, rather than sklearn or xgboost.
- **why use neural nets?**
  - sometimes outperform standard ML techniques on standard problems
  - greatly outperform standard ML techniques on some problems, for example language modeling
- **why not use neural nets?**
  - usually worse than standard ML on standard problems, and harder to implement.
  - Computational constraints: Recent models like OpenAI's GPT-3 would take ETH Deep Learning Cluster 18 months to train.

“Neural Networks” / “Deep Learning”

- “**Neural**”:
  - NN’s do not work like the brain – such metaphors are misleading.

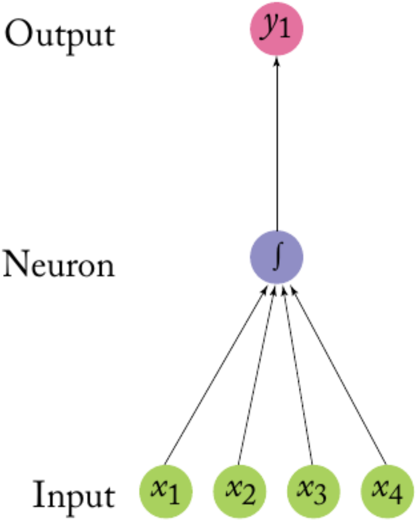
“Neural Networks” / “Deep Learning”

- “**Neural**”:
  - NN’s do not work like the brain – such metaphors are misleading.
- “**Networks**”:
  - NNs are not “networks” as that is understood in mathematical network theory or social science.

“Neural Networks” / “Deep Learning”

- “**Neural**”:
  - NN’s do not work like the brain – such metaphors are misleading.
- “**Networks**”:
  - NNs are not “networks” as that is understood in mathematical network theory or social science.
- “**Deep**” Learning:
  - does not speak to profundity or effectiveness.
  - an unfortunate source of hype.

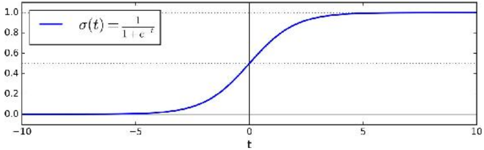
A "Neuron"



- applies dot product to vector of numerical inputs:
  - multiplies each input by a learned weight (parameter or coefficient)
  - sums these products
- applies a non-linear "activation function" to the sum
  - (e.g., the $\int$ shape indicates a sigmoid transformation)
- passes the output.

"Neuron" = Logistic Regression

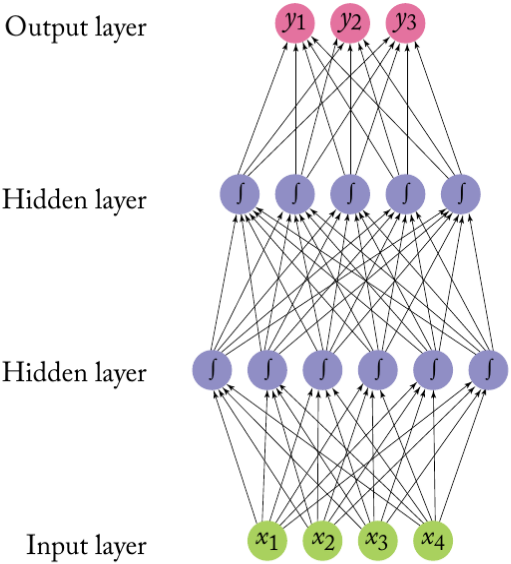
$$\hat{y} = \text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1 + \exp(-\boldsymbol{x} \cdot \theta)}$$



- applies dot product to vector of numerical inputs:
  - multiplies each input by a learned weight (parameter or coefficient)
  - sums these products
- applies a non-linear "activation function" (sigmoid) to the sum
- passes the output.

Outline

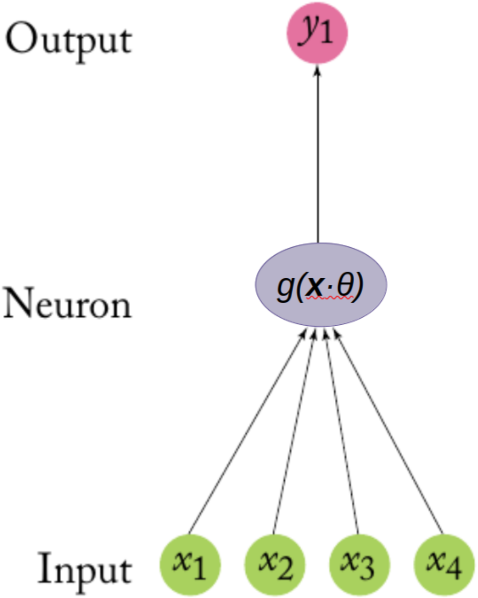
Multi-Layer Perceptron (MLP)



- A multilayer perceptron (also called a feed-forward network or sequential model) stacks neurons horizontally and vertically.
- alternatively, think of it as a stacked ensemble of logistic regression models.
- this vertical stacking is the "deep" in "deep learning"!

- MLP's are composed of "**Dense**" layers, meaning all neurons are connected.
- The tragic result in mathematics of neural nets (Hornik et al 1989, Cybenko 1989):
  - MLP with a single hidden layer, with sigmoid activation, can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$, and any mapping from one finite discrete space to another finite discrete space .
- Telgarsky (2016): NN would have to be exponentially large in many cases.

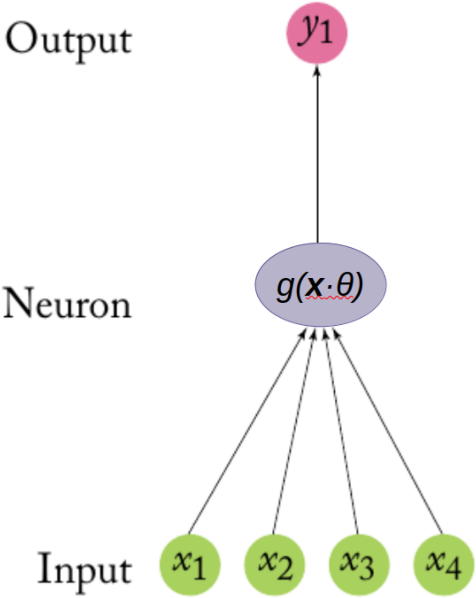
Activation functions $g(\boldsymbol{x} \cdot \theta)$



Previously we had

$$g(\boldsymbol{x} \cdot \theta) = \text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1 + \exp(-\boldsymbol{x} \cdot \theta)}$$

Activation functions $g(\boldsymbol{x} \cdot \theta)$

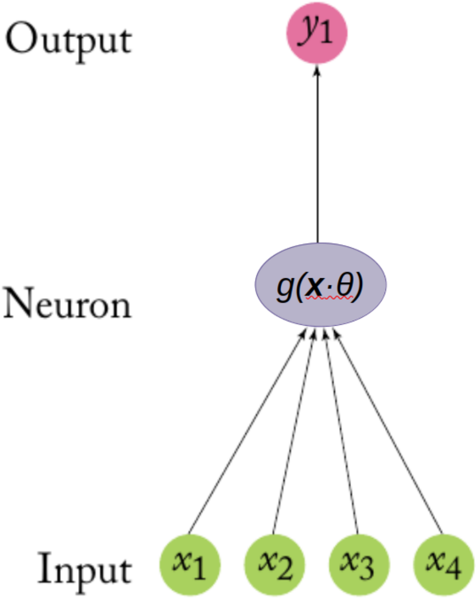


Previously we had

$$g(\boldsymbol{x} \cdot \theta)=\text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1+\exp(-\boldsymbol{x} \cdot \theta)}$$

It turns out that sigmoid does not work well in hidden layers, mainly because gradient is flat except around zero.

Activation functions $g(\boldsymbol{x} \cdot \theta)$



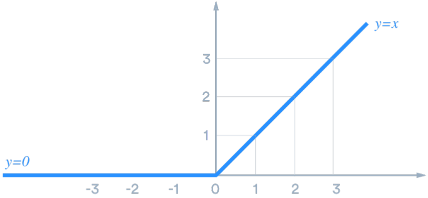
Previously we had

$$g(\boldsymbol{x} \cdot \theta) = \text{sigmoid}(\boldsymbol{x} \cdot \theta) = \frac{1}{1 + \exp(-\boldsymbol{x} \cdot \theta)}$$

It turns out that sigmoid does not work well in hidden layers, mainly because gradient is flat except around zero.

ReLU (rectified linear unit) function:

$$g(\boldsymbol{x} \cdot \theta) = \text{ReLU}(\boldsymbol{x} \cdot \theta) = \max\{0, \boldsymbol{x} \cdot \theta\}$$



Equation Notation: Multi-Layer Perceptron

- An multi-layer perceptron (MLP) with two hidden layers is

$$\boldsymbol{y} = \boldsymbol{g}_2(\boldsymbol{g}_1(\boldsymbol{x} \cdot \omega_1) \cdot \omega_2) \cdot \boldsymbol{\omega}_y$$

$$\boldsymbol{y} \in \{0,1\}^{n_y}, \boldsymbol{x} \in \mathbb{R}^{n_x}, \boldsymbol{\omega}_1 \in \mathbb{R}^{n_x \times n_1}, \boldsymbol{\omega}_2 \in \mathbb{R}^{n_1 \times n_2}, \boldsymbol{\omega}_y \in \mathbb{R}^{n_2 \times n_y}$$

  - n_1, n_2 = dimensionality in first and second hidden layer.
  - $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_y$ = set of learnable weights for the first hidden, second hidden, and output layer
  - $\boldsymbol{g}_1(\cdot), \boldsymbol{g}_2(\cdot)$ = element-wise non-linear functions for first and seond layer.

Equation Notation: Multi-Layer Perceptron

- An multi-layer perceptron (MLP) with two hidden layers is

$$\boldsymbol{y} = \boldsymbol{g}_2(\boldsymbol{g}_1(\boldsymbol{x} \cdot \omega_1) \cdot \omega_2) \cdot \boldsymbol{\omega}_y$$

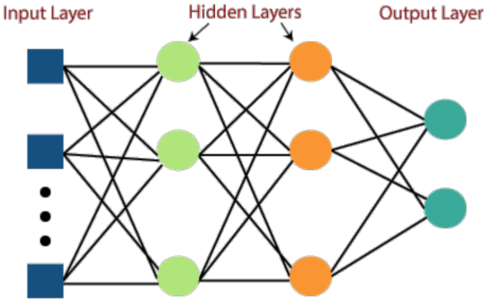
$$\boldsymbol{y} \in \{0,1\}^{n_y}, \boldsymbol{x} \in \mathbb{R}^{n_x}, \boldsymbol{\omega}_1 \in \mathbb{R}^{n_x \times n_1}, \boldsymbol{\omega}_2 \in \mathbb{R}^{n_1 \times n_2}, \boldsymbol{\omega}_y \in \mathbb{R}^{n_2 \times n_y}$$

  - n_1, n_2 = dimensionality in first and second hidden layer.
  - $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_y$ = set of learnable weights for the first hidden, second hidden, and output layer
  - $\boldsymbol{g}_1(\cdot), \boldsymbol{g}_2(\cdot)$ = element-wise non-linear functions for first and seond layer.

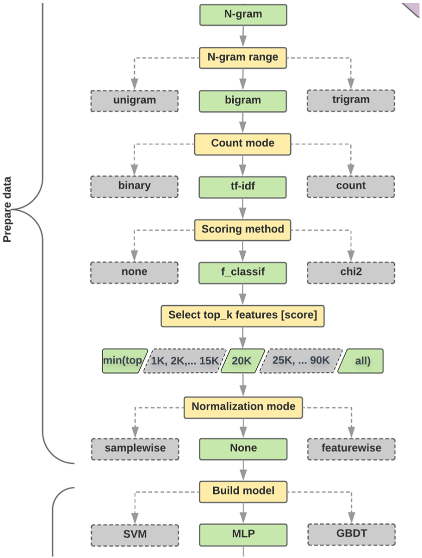
- Can also be written in decomposed notation:

$$\boldsymbol{h}_1 = \boldsymbol{g}_1(\boldsymbol{x} \cdot \omega_1)$$
$$\boldsymbol{h}_2 = \boldsymbol{g}_2(\boldsymbol{h}_1 \cdot \omega_2)$$
$$\boldsymbol{y} = \boldsymbol{h}_2 \cdot \boldsymbol{\omega}_y$$

where $\boldsymbol{h}_l$ indicate hidden layers.



- The Google Developers Guide recommends an MLP for text classification with relatively few but longer documents.:
  - $x =$ **tf-idf-weighted bigrams** as a baseline specification for text classification tasks.
    - select 20,000 features using supervised feature selection in training set.
  - $f(\cdot) =$ MLP with two hidden layers.
- A simple MLP is one of the models tried by the U.K. parliament paper (Peterson and Spirling 2018).



Outline

- See the Geron book and sample notebooks for Keras examples.
  - "Dense" layer is the DNN baseline – means that all neurons are connected.
  - performance improvements of making the model deeper than 2 layers are minimal (Reimers & Gurevych, 2017)

- See the Geron book and sample notebooks for Keras examples.
  - "Dense" layer is the DNN baseline – means that all neurons are connected.
  - performance improvements of making the model deeper than 2 layers are minimal (Reimers & Gurevych, 2017)
- Neural nets have *many* dimensions for tuning.
  - the # of layers, # of neurons, activation functions, regularization, optimizer, learning rate, normalization, etc are all tunable hyperparameters.
  - this is a major practical downside of using neural nets rather than sklearn or xgboost for most tasks.

- See the Geron book and sample notebooks for Keras examples.
  - "Dense" layer is the DNN baseline – means that all neurons are connected.
  - performance improvements of making the model deeper than 2 layers are minimal (Reimers & Gurevych, 2017)
- Neural nets have *many* dimensions for tuning.
  - the # of layers, # of neurons, activation functions, regularization, optimizer, learning rate, normalization, etc are all tunable hyperparameters.
  - this is a major practical downside of using neural nets rather than sklearn or xgboost for most tasks.
  - cross-validating these architectural chooices is usually too computationally expensive.
  - instead, make a big model (too many layers, too many neurons) and regularize with dropout and early stopping.

Dropout

An elegant regularization technique:

- at every training step, every neuron has some probability (typically $p = 0.5$) of being temporarily dropped out, so that it will be ignored at this step.
- at test time, neurons dont get dropped anymore but coefficients are down-weighted by p.

Dropout

An elegant regularization technique:

- at every training step, every neuron has some probability (typically $p = 0.5$) of being temporarily dropped out, so that it will be ignored at this step.
- at test time, neurons dont get dropped anymore but coefficients are down-weighted by p.

Why it works:

- Approximates an ensemble of N models (where N is the number of neurons).
- Neurons cannot co-adapt with neighbors; they must be independently useful.
- Layers cannot rely excessively on just a few inputs.

Early Stopping

- A second elegant regularization technique, used in both xgboost and in neural nets:
  - gradually train the model gradients while checking fit in a held-out validation set.
  - when model starts overfitting, stop training.

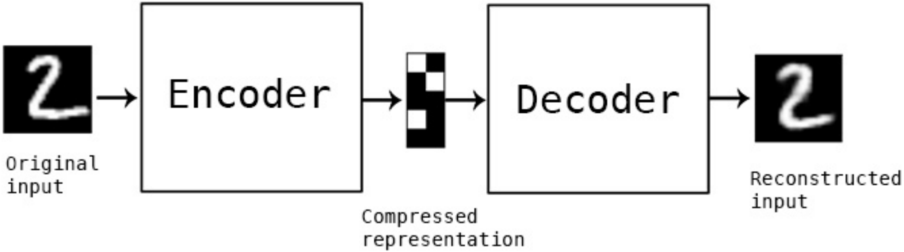
Early Stopping

- A second elegant regularization technique, used in both xgboost and in neural nets:
  - gradually train the model gradients while checking fit in a held-out validation set.
  - when model starts overfitting, stop training.
- Requires user to specify two additional parameters:
  - validation set: a third sample of the data, separate from training set and test set
  - early stopping rounds: stop training if we have done this many epochs without improving validation set performance.

Outline

Autoencoders: Optimal Compression Algorithms

- Autoencoders = neural nets that perform domain-specific lossy compression:



- Learned encodings can be decoded back to a *reconstruction* – a (minimally) lossy representation of the original data.
- AE's can memorize complex, unstructured data.

Autoencoder Architecture

- Stacked layers gradually decrease in dimensionality to create the compressed representation
- then gradually increase in dimensionality to try to reconstruct the input.

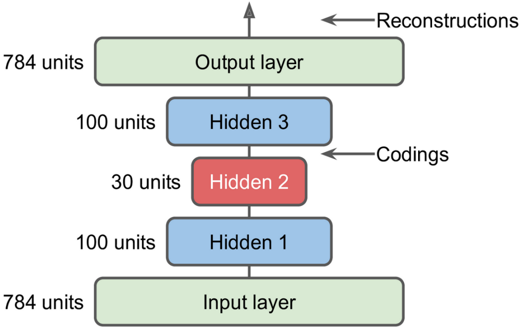


Figure 17-3. Stacked autoencoder

Reconstruction from encoded vector

Figure 17-4. Original images (top) and their reconstructions (bottom)

True/False Quiz (4 minutes)

1. Neural nets tend to out-perform xgboost on classifying long documents.
2. "Deep" neural nets have at least ten hidden layers.
3. Rectified linear unit (ReLU) should be used as the activation function in MLPs.
4. Number of hidden layers, and number of neurons per layer, are hyperparameters that can be learned by cross-validation in the training set.
5. Early stopping means splitting into three sets (train, validation, test), and training the model until performance starts decreasing in the validation set.
6. The middle layer vector of an autoencoder is a compressed embedding representing the original input.

Objectives of a (Deep) Machine Learning Project

1. **What is the question or problem?**

Objectives of a (Deep) Machine Learning Project

1. **What is the question or problem?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata

Objectives of a (Deep) Machine Learning Project

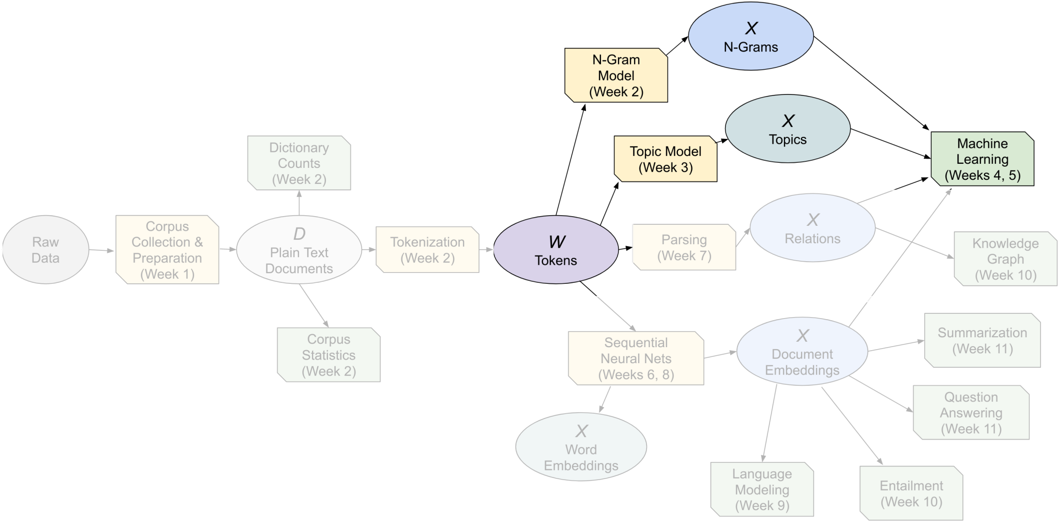
1. **What is the question or problem?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Machine learning:
   - Select a model and train it.
   - Fine-tune hyperparameters for out-of-sample fit.
   - Interpret predictions using model explanation methods.

Objectives of a (Deep) Machine Learning Project

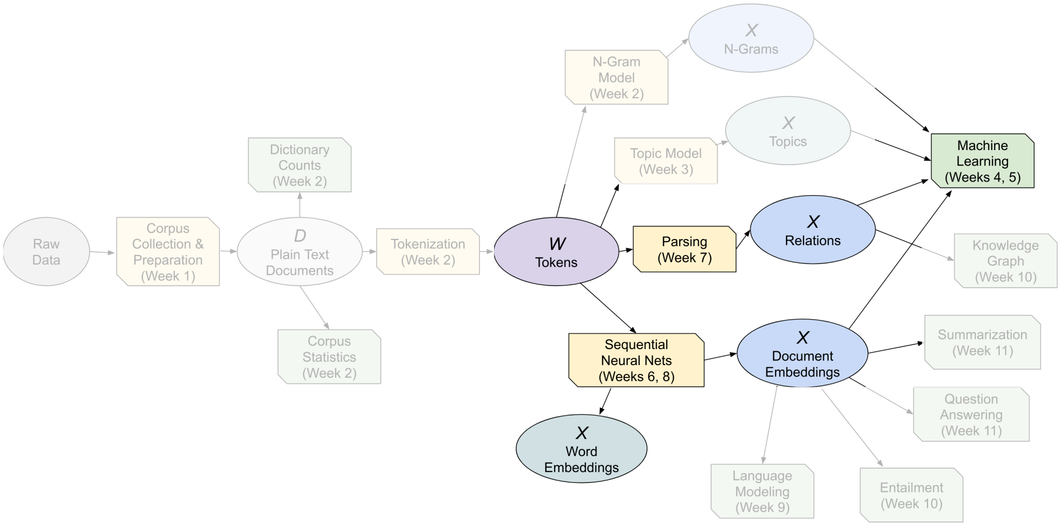
1. **What is the question or problem?**
2. Corpus and Data:
   - obtain, clean, preprocess, and link.
   - Produce descriptive visuals and statistics on the text and metadata
3. Machine learning:
   - Select a model and train it.
   - Fine-tune hyperparameters for out-of-sample fit.
   - Interpret predictions using model explanation methods.
4. Empirical analysis
   - Produce statistics or predictions with the trained model.
   - **Answer the question / solve the problem.**

Outline

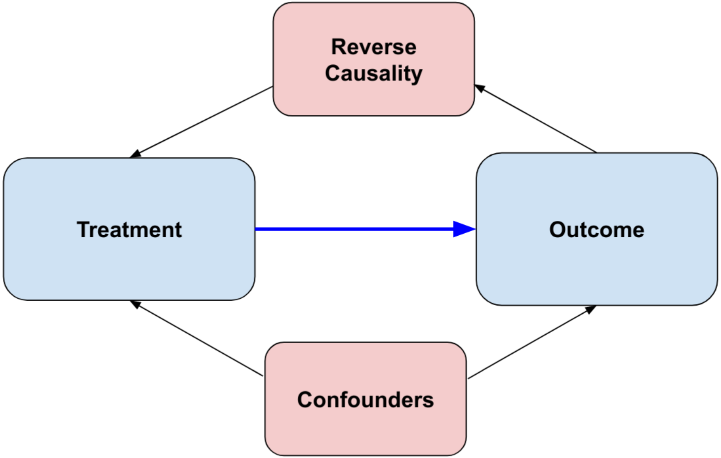
Course Progress (Weeks 2-5)



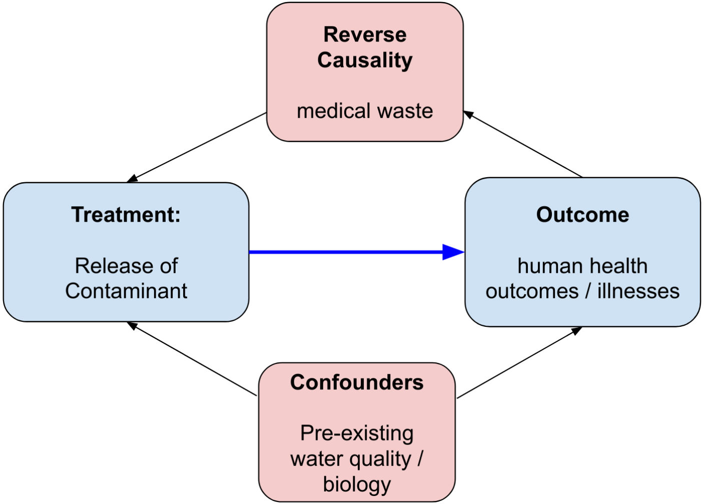
Course Progress (Weeks 5-8)



Causal Graphs



Causal Graph Example: Pollution of a River



Activity: Practice with Causal Graphs

- Think of two example causal inference questions:
  1. where you have **language as an outcome**
  2. where you have **language as a treatment**
- Try to personalize it:
  - a research question from your field
  - a policy you are interested in
  - a mystery you are fascinated by

Activity: Practice with Causal Graphs

- Think of two example causal inference questions:
  1. where you have **language as an outcome**
  2. where you have **language as a treatment**
- Try to personalize it:
  - a research question from your field
  - a policy you are interested in
  - a mystery you are fascinated by
- Link to causal graph template posted in zoom chat:
  - make a copy, fill it in
  - make your doc viewable and paste link into padlet (also in zoom chat).
  - will review these at beginning of next lecture.