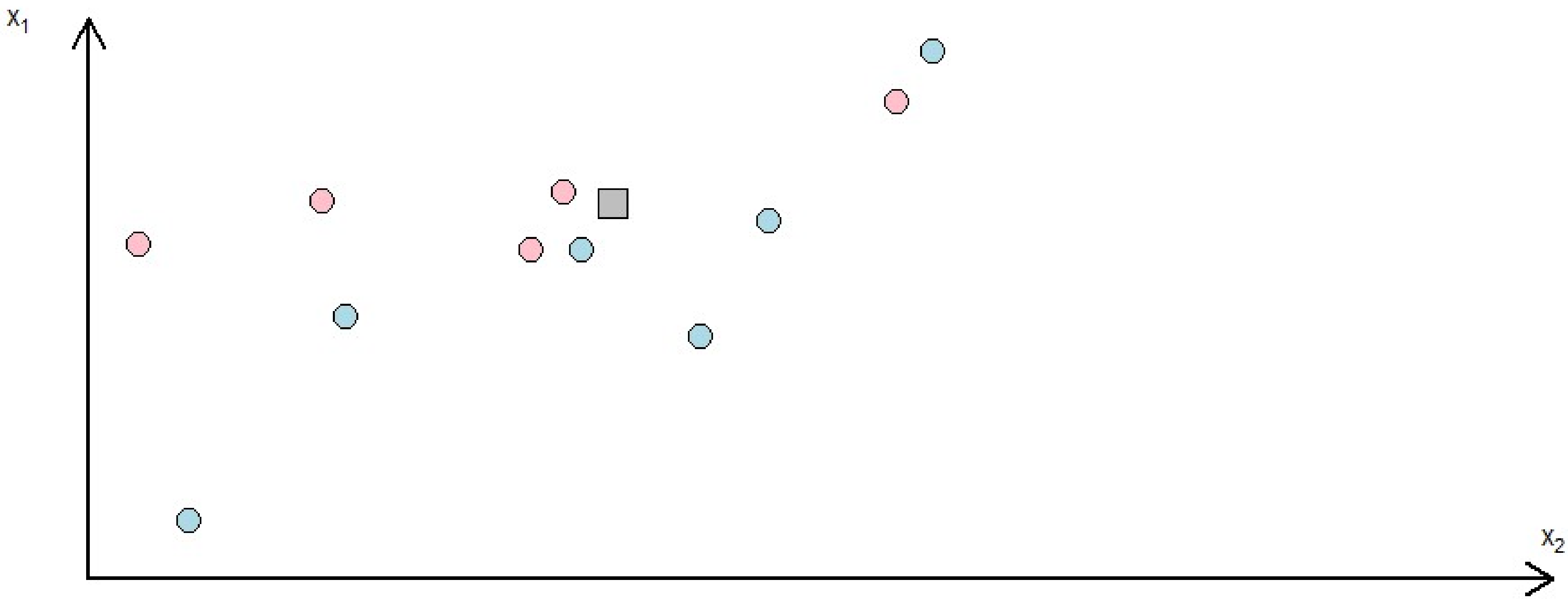

x_1
x_2

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a ‘new’ (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space,

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$:

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a ‘new’ (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

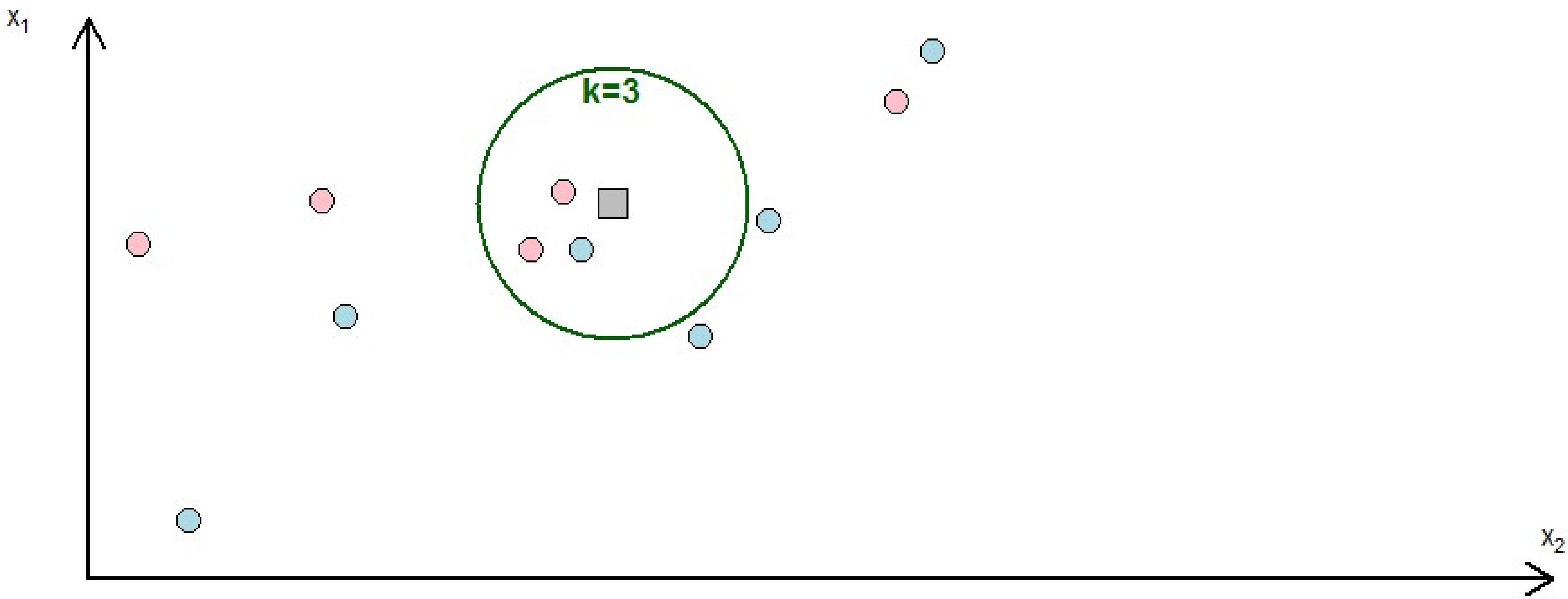
e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)


x1
k=3
x2

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a ‘new’ (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)

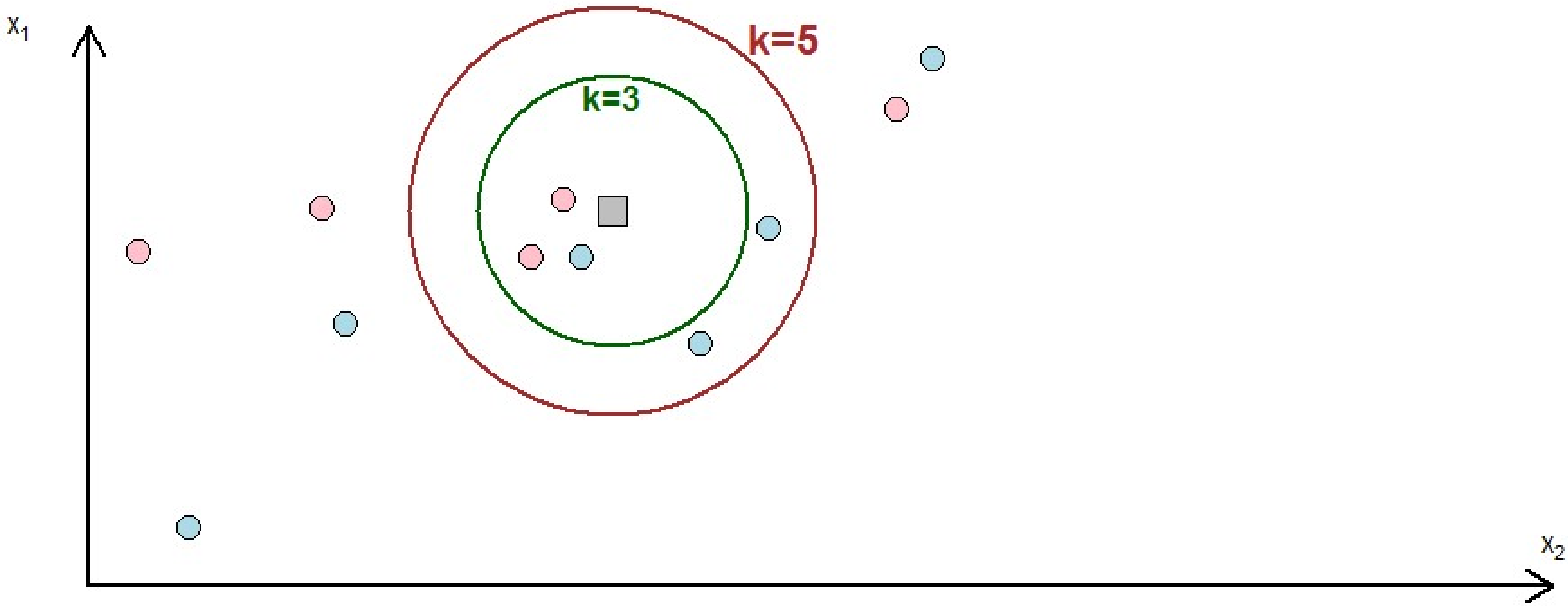
k-nearest neighbors

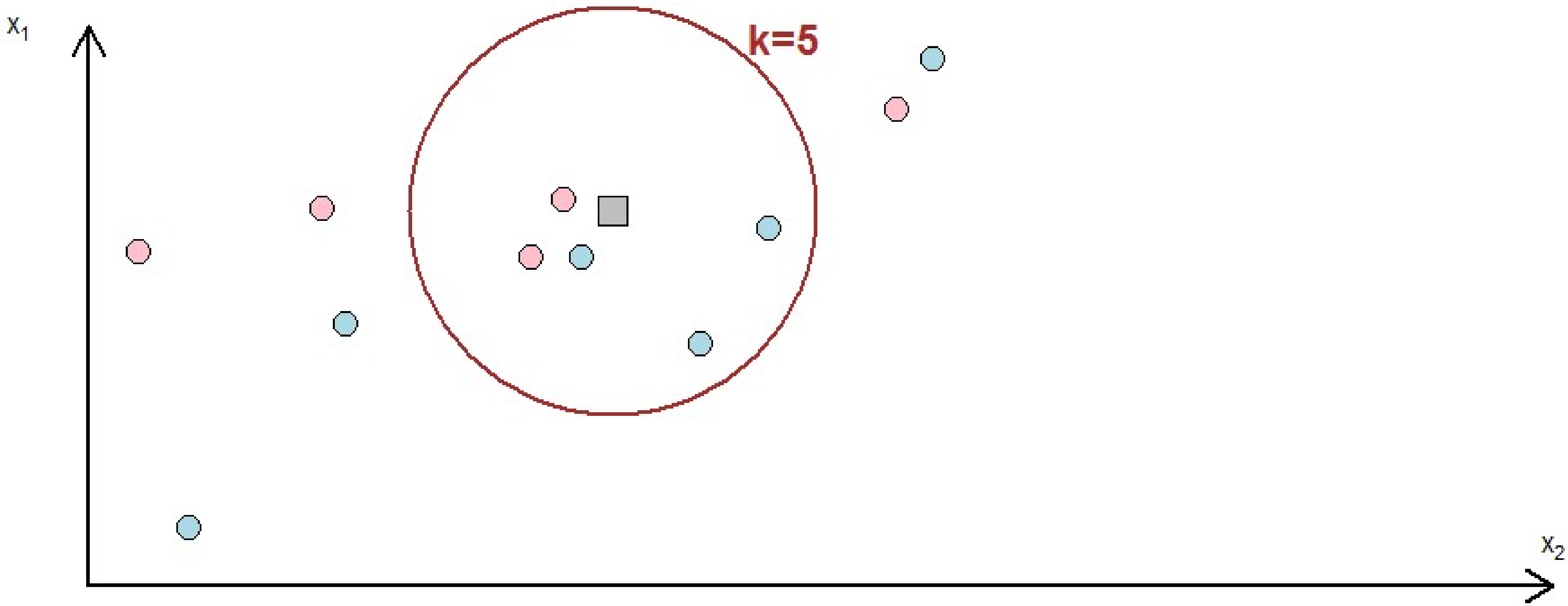
Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)

e.g. $k = 5$: she is assigned to the Democrats (3 vs 2)


x1
k=5
k=3
x2



x_1
k=5
x_2

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)

e.g. $k = 5$: she is assigned to the Democrats (3 vs 2)

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)

e.g. $k = 5$: she is assigned to the Democrats (3 vs 2)

$\rightarrow$ Can use Euclidean distance (or some other metric for the neighbors), but need to be careful about imbalance in the training set.

k-nearest neighbors

Variant of the Senate example: now suppose we interested in classifying a 'new' (test set) Senator (■) based on her feature values.

One simple idea is to look at her k nearest neighbors from the training set in the feature space, and take a majority vote in terms of what party we should assign her to.

e.g. $k = 3$: she is assigned to the Republicans (2 vs 1)

e.g. $k = 5$: she is assigned to the Democrats (3 vs 2)

→ Can use Euclidean distance (or some other metric for the neighbors), but need to be careful about imbalance in the training set.

→ Choice of k can be optimized, but generally case that noise in data causes poor classification.

Partner Exercise

Partner Exercise

1. In practice, we have to be careful about using KNN techniques when there is imbalance in terms of proportions of classes in the training set. Why?

Partner Exercise

1. In practice, we have to be careful about using KNN techniques when there is imbalance in terms of proportions of classes in the training set. Why? (hint: think about classifying a new point when 99.99% of all observations belong to one class.)

Partner Exercise

1. In practice, we have to be careful about using KNN techniques when there is imbalance in terms of proportions of classes in the training set. Why? (hint: think about classifying a new point when 99.99% of all observations belong to one class.)

2. Suppose we have picked $k = 3$. In practice, we often weight the votes of the three nearest observations by $\frac{1}{d}$ where d is the distance from the observation we wish to classify. Why?

Partner Exercise

1. In practice, we have to be careful about using KNN techniques when there is imbalance in terms of proportions of classes in the training set. Why? (hint: think about classifying a new point when 99.99% of all observations belong to one class.)

2. Suppose we have picked $k = 3$. In practice, we often weight the votes of the three nearest observations by $\frac{1}{d}$ where d is the distance from the observation we wish to classify. Why?

Advantages of kNN

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$.

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible.

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning').

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally.

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally. So, no training and no model.

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally. So, no training and no model. Can still inspect accuracy though.

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally. So, no training and no model. Can still inspect accuracy though.

Works well so long as N is large,

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally. So, no training and no model. Can still inspect accuracy though.

Works well so long as N is large, and robust to noise (random elements in the data).

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally. So, no training and no model. Can still inspect accuracy though.

Works well so long as N is large, and robust to noise (random elements in the data).

Works with any types of features,

Advantages of *kNN*

It is non-parametric in sense that don't need any assumptions about functional form of $\mathbb{E}(Y|X)$. Don't need to divide the space in big 'chunks'—very flexible. (But then, easy to overfit).

Example of instance-based training ('lazy learning'). Everything uncovered about relationship between $\mathbb{E}(Y|X)$ is done locally. So, no training and no model. Can still inspect accuracy though.

Works well so long as N is large, and robust to noise (random elements in the data).

Works with any types of features, though typically requires rescaling (normalizing) to ensure that one unit of one variable is not treated same as one unit of another (e.g. gender vs income: male is more different to female than \$10,000 is to \$10,001)

Disadvantages of kNN

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three")

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

Actually finding the nearest neighbours can be slow,

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

Actually finding the nearest neighbours can be slow, especially for long feature vectors.

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

Actually finding the nearest neighbours can be slow, especially for long feature vectors.

The curse of dimensionality.

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

Actually finding the nearest neighbours can be slow, especially for long feature vectors.

The curse of dimensionality. As number of features grows large, points look more similar than they really are (especially if features are irrelevant to actual decision).

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

Actually finding the nearest neighbours can be slow, especially for long feature vectors.

The curse of dimensionality. As number of features grows large, points look more similar than they really are (especially if features are irrelevant to actual decision). So, *kNN* often requires prior feature selection from theory,

Disadvantages of *kNN*

While interpretation of given decision is easy (“new observation is most like these other three”) the interpretation of the model is hard. Don’t typically have general lessons from *kNN*—there is no model.

Actually finding the nearest neighbours can be slow, especially for long feature vectors.

The curse of dimensionality. As number of features grows large, points look more similar than they really are (especially if features are irrelevant to actual decision). So, *kNN* often requires prior feature selection from theory, or via automated methods.

Disadvantages of *kNN*

While interpretation of given decision is easy ("new observation is most like these other three") the interpretation of the model is hard. Don't typically have general lessons from *kNN*—there is no model.

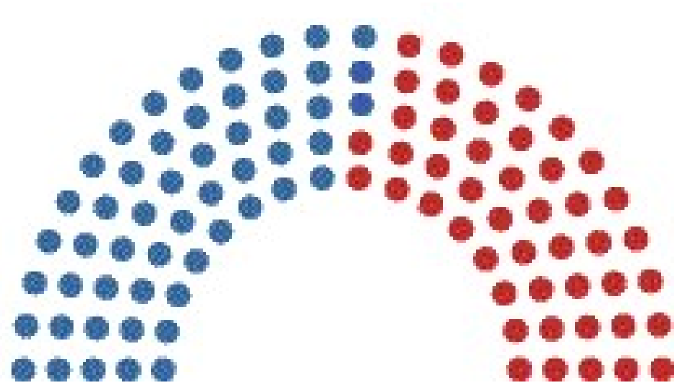
Actually finding the nearest neighbours can be slow, especially for long feature vectors.

The curse of dimensionality. As number of features grows large, points look more similar than they really are (especially if features are irrelevant to actual decision). So, *kNN* often requires prior feature selection from theory, or via automated methods. May also have to tune distance function manually (e.g weight up 'safety rating' for baby products)

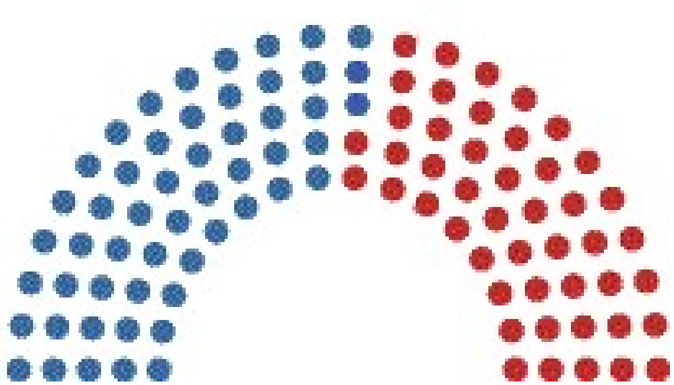
Trees and Forests

Partitioning the Senators With Trees

Partitioning the Senators With Trees

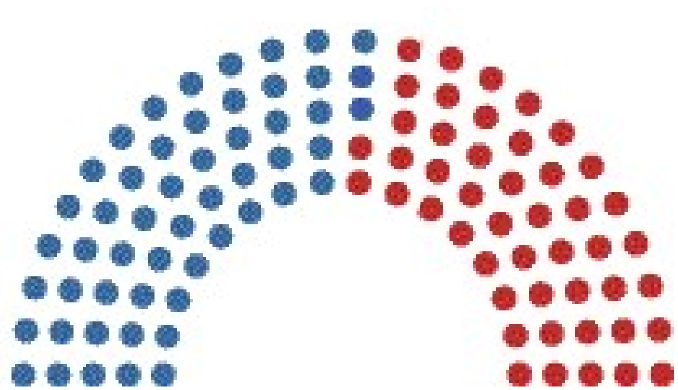


Partitioning the Senators With Trees



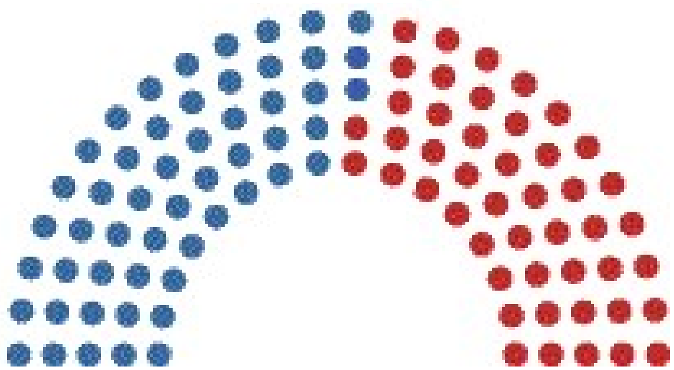
Idea our Senators are defined by their attributes.

Partitioning the Senators With Trees



Idea our Senators are defined by their attributes. Suppose we (optimally) split ('partition') the Senators with respect to x_1, such that we form two subsets of our training data.

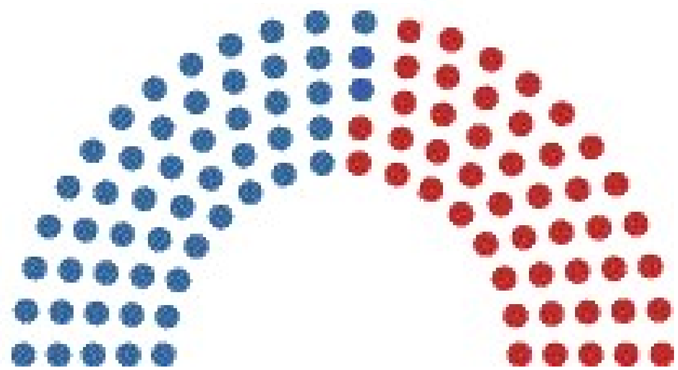
Partitioning the Senators With Trees



Idea our Senators are defined by their attributes. Suppose we (optimally) split ('partition') the Senators with respect to x_1, such that we form two subsets of our training data.

e.g suppose that Republicans generally use 'guns' more than Democrats,

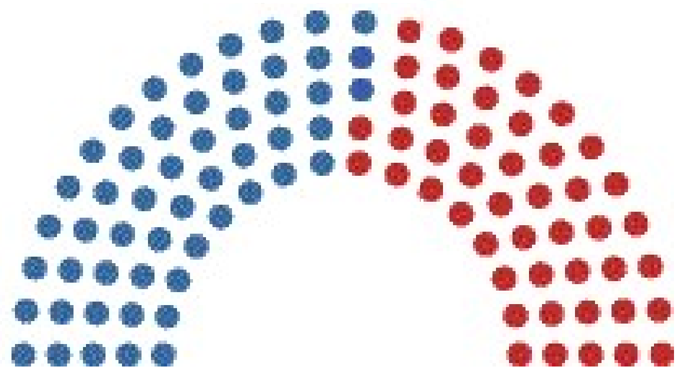
Partitioning the Senators With Trees



Idea our Senators are defined by their attributes. Suppose we (optimally) split ('partition') the Senators with respect to x_1, such that we form two subsets of our training data.

e.g suppose that Republicans generally use 'guns' more than Democrats, such that grabbing all the observations for which $x_{guns} > 0.621$ captures, say, 80% of the Republicans in our data.

Partitioning the Senators With Trees

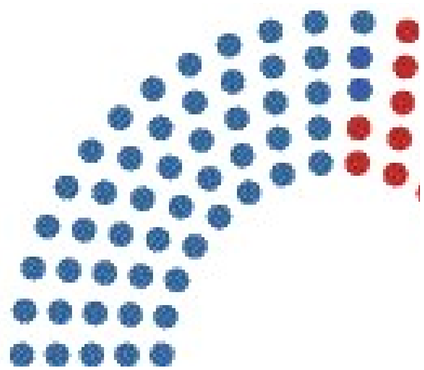


Idea our Senators are defined by their attributes. Suppose we (optimally) split ('partition') the Senators with respect to x_1, such that we form two subsets of our training data.

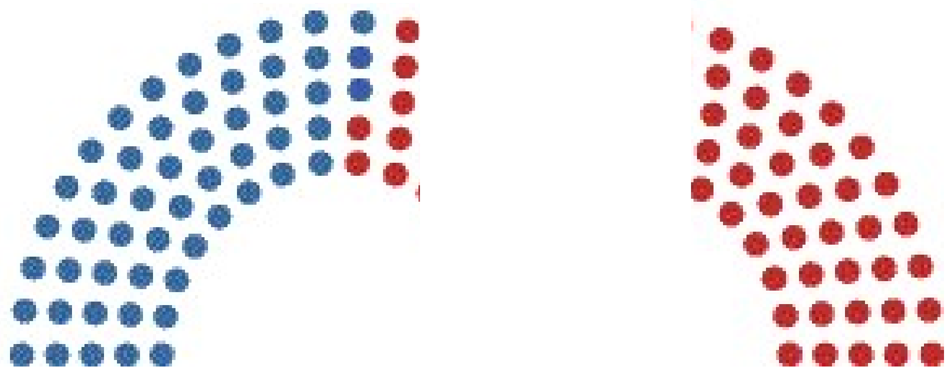
e.g suppose that Republicans generally use 'guns' more than Democrats, such that grabbing all the observations for which $x_{guns} > 0.621$ captures, say, 80% of the Republicans in our data.

Tree, stage 1

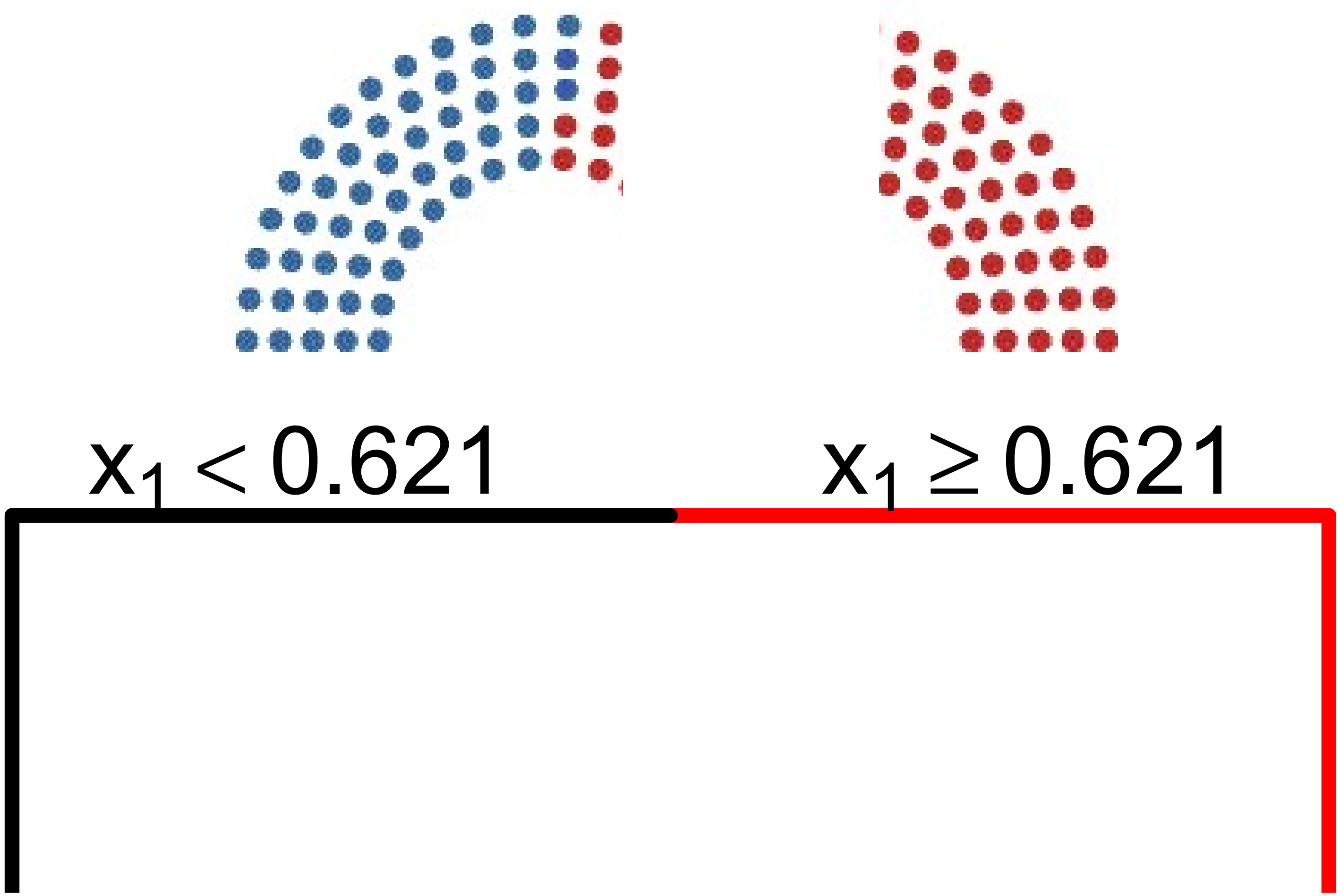
Tree, stage 1



Tree, stage 1



Tree, stage 1



Now, x_2 ...

Now, x_2 ...

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

Now, x_2 ...

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

Now, x_2 . . .

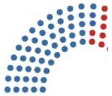
$\longrightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.

Now, x_2 . . .

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

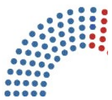
and a subset that's still a mix of Republicans and Democrats.



Now, x_2 ...

→ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.

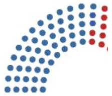


btw The set of Senators we've assigned to Republicans based on their x_1 values are called a leaf.

Now, x_2 . . .

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.



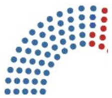
btw The set of Senators we've assigned to Republicans based on their x_1 values are called a leaf.

now suppose we take the mixed group remaining ('internal node') and split them based on x_2, which is their use of '`equality`'.

Now, x_2 ...

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.



btw The set of Senators we've assigned to Republicans based on their x_1 values are called a leaf.

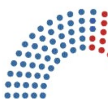
now suppose we take the mixed group remaining ('internal node') and split them based on x_2, which is their use of '`equality`'.

and it turns out that the (remaining) Republicans tend to use this less.

Now, x_2 . . .

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.



btw The set of Senators we've assigned to Republicans based on their x_1 values are called a leaf.

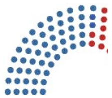
now suppose we take the mixed group remaining ('internal node') and split them based on x_2, which is their use of '`equality`'.

and it turns out that the (remaining) Republicans tend to use this less. So,

Now, x_2 . . .

$\rightarrow$ we now have **two** subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.



btw The set of Senators we've assigned to Republicans based on their x_1 values are called a **leaf**.

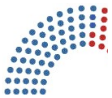
now suppose we take the mixed group remaining ('internal node') and split them based on x_2, which is their use of '`equality`'.

and it turns out that the (remaining) Republicans tend to use this less. So, when we partition according to, say, $x_2 \leq 0.56$ this enables us to perfectly divide this remaining subset into Democrats and Republicans.

Now, x_2 ...

$\rightarrow$ we now have two subsets of our training set: a bunch of Republicans (classified correctly based on x_1 alone)

and a subset that's still a mix of Republicans and Democrats.



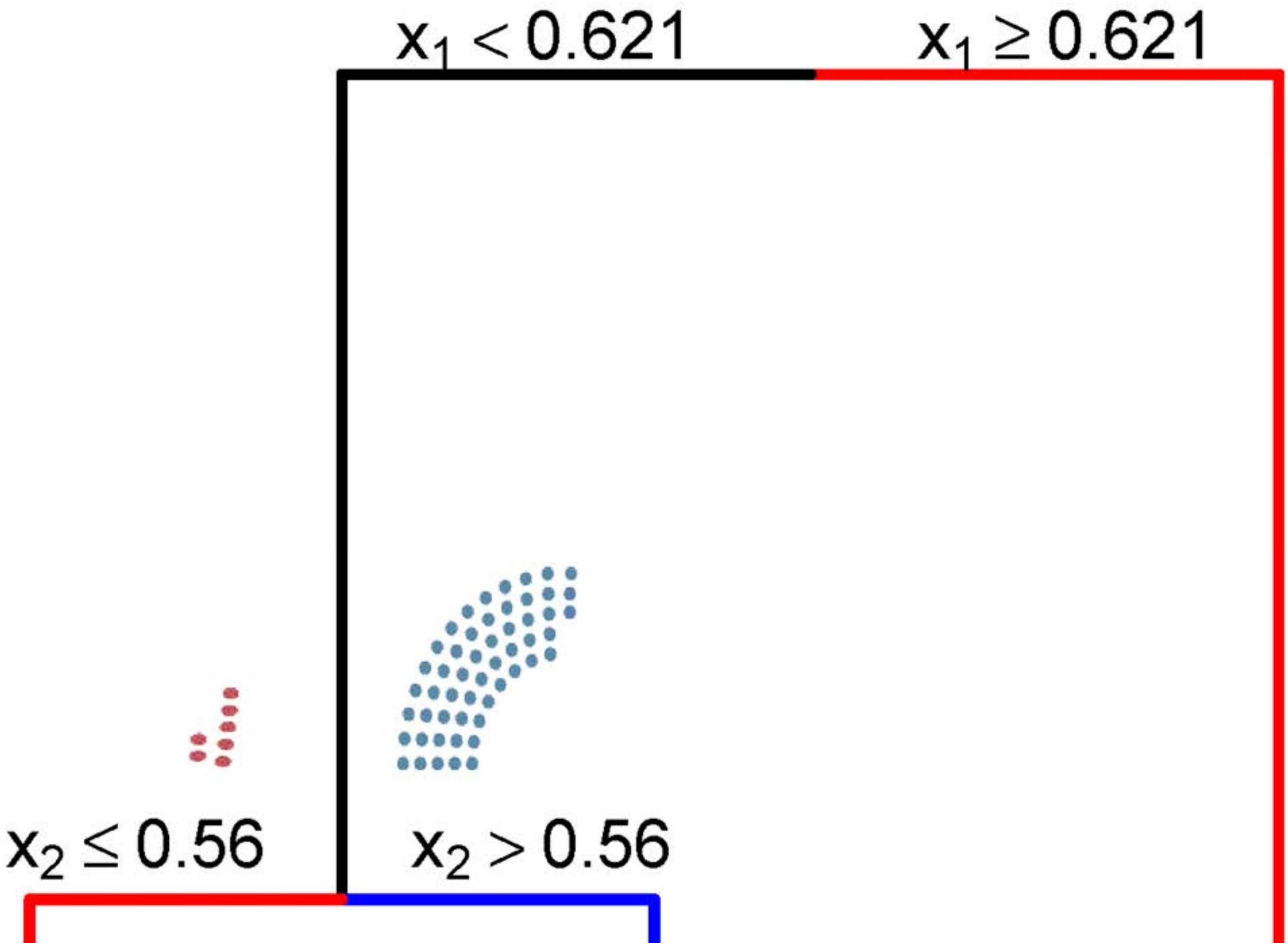
btw The set of Senators we've assigned to Republicans based on their x_1 values are called a leaf.

now suppose we take the mixed group remaining ('internal node') and split them based on x_2, which is their use of '`equality`'.

and it turns out that the (remaining) Republicans tend to use this less. So, when we partition according to, say, $x_2 \leq 0.56$ this enables us to perfectly divide this remaining subset into Democrats and Republicans.

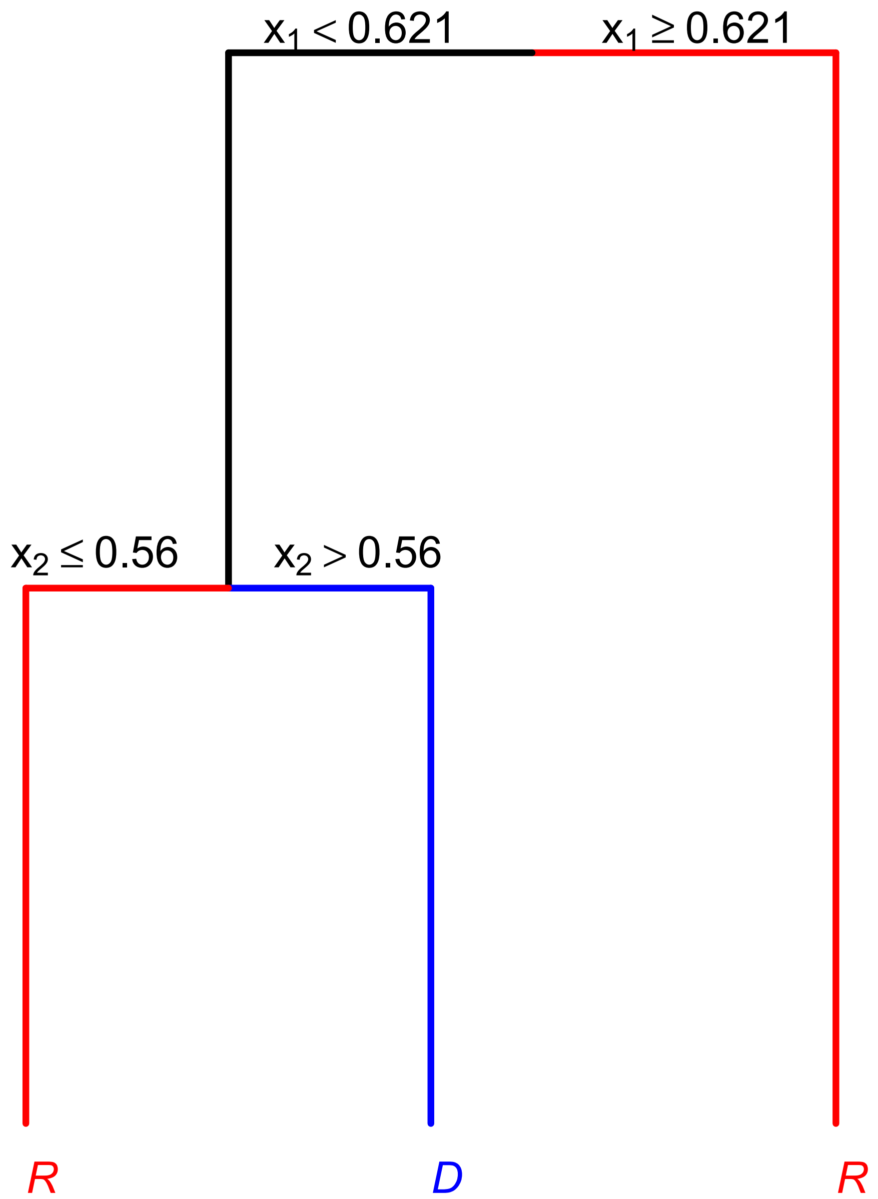
Tree, stage 2

Tree, stage 2



Complete Tree

Complete Tree



Notes

This classifier is known as a tree,

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

→ typically <u>not</u> the case that we can (or want to) classify perfectly into the leaves!

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

$\rightarrow$ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node,

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

→ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

$\rightarrow$ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x:

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

$\rightarrow$ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x: typically based on how homogenous the resulting subset of the data is

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

$\rightarrow$ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x: typically based on how homogenous the resulting subset of the data is

e.g. 'Gini impurity' and 'Variance Reduction'

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

→ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x: typically based on how homogenous the resulting subset of the data is

e.g. 'Gini impurity' and 'Variance Reduction'

+ Trees are easy to interpret,

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

$\rightarrow$ typically <u>not</u> the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x: typically based on how homogenous the resulting subset of the data is

e.g. 'Gini impurity' and 'Variance Reduction'

$+$ Trees are easy to interpret, and we can report relative variable importance statistics.

Notes

This classifier is known as a tree, and the recursive partitioning on each derived subset continues until further splits doesn't add 'much' to our classification ability.

$\rightarrow$ typically not the case that we can (or want to) classify perfectly into the leaves!

At each node, algorithmic tricks allow fast searching over all the variables to find the one that should be used.

and clearly need a metric for 'best' split in given x: typically based on how homogenous the resulting subset of the data is

e.g. 'Gini impurity' and 'Variance Reduction'

$+$ Trees are easy to interpret, and we can report relative variable importance statistics.

But. . .

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

But...

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions,

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

$\rightarrow$ related to problems of overfitting in the training data.

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

$\rightarrow$ related to problems of overfitting in the training data.

So effort is made to prune the trees back

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

→ related to problems of overfitting in the training data.

So effort is made to prune the trees back—remove less helpful branches.

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

$\rightarrow$ related to problems of overfitting in the training data.

So effort is made to prune the trees back—remove less helpful branches.

Or can construct many trees (from slightly different samples of the data) and average over them:

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

$\rightarrow$ related to problems of overfitting in the training data.

So effort is made to prune the trees back—remove less helpful branches.

Or can construct many trees (from slightly different samples of the data) and average over them: known as bagging ('bootstrap aggregating').

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

$\rightarrow$ related to problems of overfitting in the training data.

So effort is made to prune the trees back—remove less helpful branches.

Or can construct many trees (from slightly different samples of the data) and average over them: known as bagging ('bootstrap aggregating').

$\rightarrow$ random forests combines *many* trees (forest) and

But. . .

These basic CART (Classification and Regression Trees) approaches show instability in practice:

i.e. minor changes to training data can have large consequences for classification decisions, because any error is propagated down the tree.

$\rightarrow$ related to problems of overfitting in the training data.

So effort is made to prune the trees back—remove less helpful branches.

Or can construct many trees (from slightly different samples of the data) and average over them: known as bagging ('bootstrap aggregating').

$\rightarrow$ random forests combines *many* trees (forest) and at each split a *random sample* of features is considered (rather than all features).

Using Random Forests (in regression context). . .

Using Random Forests (in regression context). . .

what stems are most important predictors of treaty harshness? (note: $p >> n$)

Using Random Forests (in regression context)...

what stems are most important predictors of treaty harshness? (note: $p >> n$)

use reduction in mean square error when that term is used in a tree

Using Random Forests (in regression context)...

what stems are most important predictors of treaty harshness? (note: $p >> n$)

use reduction in mean square error when that term is used in a tree (increase in MSE when it is not used)

Using Random Forests (in regression context)...

what stems are most important predictors of treaty harshness? (note: $p >> n$)

use reduction in mean square error when that term is used in a tree (increase in MSE when it is not used)

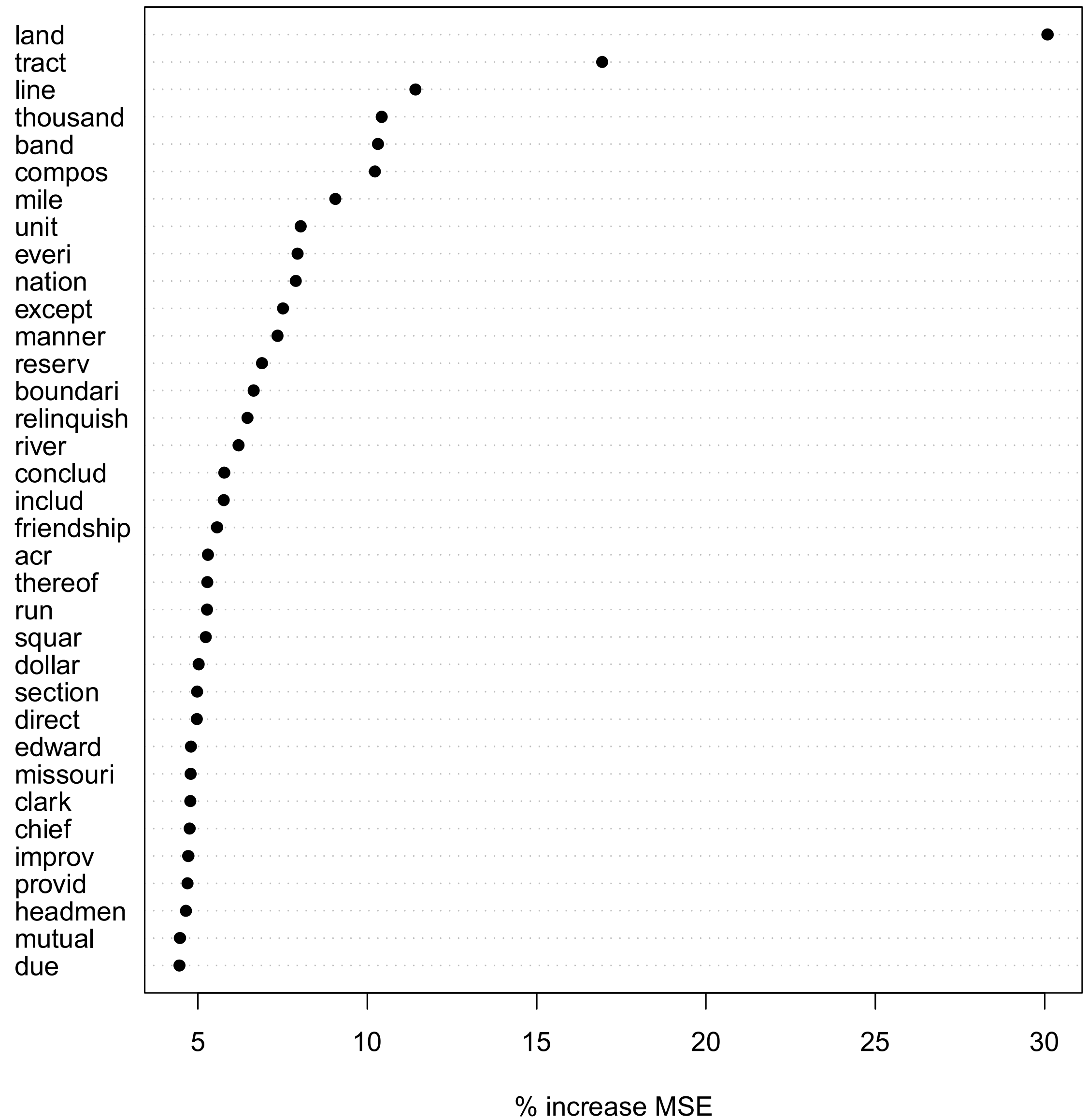
fairly robust to TDM construction choices

Using Random Forests (in regression context)...

what **stems** are most important predictors of treaty harshness? (note: $p >> n$)

use **reduction in mean square error** when that term is used in a tree (increase in MSE when it is not used)

fairly **robust** to TDM construction choices



Associations

Associations

stem	appearance	common phrasing (frequency)	ρ
`friendship`	friendship	“A treaty of peace and friendship” (15)	0.504
`mutual`	mutually	“shall be mutually forgiven and forgot” (19)	0.255
`peac`	peace	“A treaty of peace and friendship” (15)	0.179

Associations

stem	appearance	common phrasing (frequency)	ρ
friendship	friendship	"A treaty of peace and friendship" (15)	0.504
mutual	mutually	"shall be mutually forgiven and forgot" (19)	0.255
peac	peace	"A treaty of peace and friendship" (15)	0.179
cession	cession	"In consideration of the foregoing cession" (15)	-0.205
relinquish	relinquish	"cede and relinquish to the United States" (4)	-0.208
boundari	boundary	"land included within the following boundaries" (4)	-0.214
tract	tract	"One tract," (14)	-0.442
dollar	dollars	"forty dollars" (11)	-0.457
land	lands	"one section of land" (29)	-0.567
reserv	reservation	"one other reservation" (5)	-0.622

Better yet. . . Ensembles

Better yet. . . Ensembles

Plus can up-weight the most problematic observations

Better yet. . . Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers

Better yet. . . Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees:

Better yet. . . Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Better yet... Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually,

Better yet. . . Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general:

Better yet. . . Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general: ensemble learning involves bringing together different algorithms to produce better results (better prediction) than any one technique.

Better yet... Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general: ensemble learning involves bringing together different algorithms to produce better results (better prediction) than any one technique.

e.g. Hillard, Purpura and Wilkerson are interested in predicting topic (20) and sub-topic (226) of $379,000$ Congressional bills.

Better yet... Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general: ensemble learning involves bringing together different algorithms to produce better results (better prediction) than any one technique.

e.g. Hillard, Purpura and Wilkerson are interested in predicting topic (20) and sub-topic (226) of $379,000$ Congressional bills.

Better yet. . . Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general: ensemble learning involves bringing together different algorithms to produce better results (better prediction) than any one technique.

e.g. Hillard, Purpura and Wilkerson are interested in predicting topic (20) and sub-topic (226) of $379,000$ Congressional bills.

Why? State of the art requires *months* to train human coders,

Better yet... Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general: ensemble learning involves bringing together different algorithms to produce better results (better prediction) than any one technique.

e.g. Hillard, Purpura and Wilkerson are interested in predicting topic (20) and sub-topic (226) of $379,000$ Congressional bills.

Why? State of the art requires *months* to train human coders, although intercoder agreement is high.

Better yet... Ensembles

Plus can up-weight the most problematic observations and upweight the most accurate classifiers when combining lots of trees: *boosting*.

Actually, these ideas—bagging, forests, boosting—can be made more general: ensemble learning involves bringing together different algorithms to produce better results (better prediction) than any one technique.

e.g. Hillard, Purpura and Wilkerson are interested in predicting topic (20) and sub-topic (226) of $379,000$ Congressional bills.

Why? State of the art requires *months* to train human coders, although intercoder agreement is high.

Q Can they use supervised learning to do better? (better in terms of time: assume humans are accurate)

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

“Computer Assisted Topic Classification for Mixed-Methods Social Science Research”

Split human coded data in two:

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing,

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

“Computer Assisted Topic Classification for Mixed-Methods Social Science Research”

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set
4. Boostexter: boosting algorithm that combines weak(er) classifiers

“Computer Assisted Topic Classification for Mixed-Methods Social Science Research”

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set
4. Boostexter: boosting algorithm that combines weak(er) classifiers

In topic stage,

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set
4. Boostexter: boosting algorithm that combines weak(er) classifiers

In topic stage, compare predictions $\hat{y}_i \in \{1, \ldots, 20\}$ from methods:

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set
4. Boostexter: boosting algorithm that combines weak(er) classifiers

In topic stage, compare predictions $\hat{y}_i \in \{1, \ldots, 20\}$ from methods: allocate document to winner via some 'vote' aggregation system.

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set
4. Boostexter: boosting algorithm that combines weak(er) classifiers

In topic stage, compare predictions $\hat{y}_i \in \{1, \ldots, 20\}$ from methods: allocate document to winner via some 'vote' aggregation system.

In subtopic stage,

"Computer Assisted Topic Classification for Mixed-Methods Social Science Research"

Split human coded data in two: $187,000$ in training set, $187,000$ in test set.

Some preprocessing, and then use four methods:

1. Naive Bayes
2. SVM
3. MaxEnt: like NB but features are weighted in way that maximizes likelihood of training set
4. Boostexter: boosting algorithm that combines weak(er) classifiers

In topic stage, compare predictions $\hat{y}_i \in \{1, \ldots, 20\}$ from methods: allocate document to winner via some 'vote' aggregation system.

In subtopic stage, use SVM (as best performer).

Results

Results

TABLE 3. Bill Title Interannotator Agreement for Five Model Types

	SVM	MaxEnt	Boostexter	Naïve Bayes	Ensemble
Major topic $N = 20$	88.7% (.881)	86.5% (.859)	85.6% (.849)	81.4% (.805)	89.0% (.884)
Subtopic $N = 226$	81.0% (.800)	78.3% (.771)	73.6% (.722)	71.9% (.705)	81.0% (.800)

Note. Results are based on using approximately 187,000 human-labeled cases to train the classifier to predict approximately 187,000 other cases (that were also labeled by humans but not used for training). Agreement is computed by comparing the machine's prediction to the human assigned labels. (AC1 measure presented in parentheses).

Results

TABLE 3. Bill Title Interannotator Agreement for Five Model Types

	SVM	MaxEnt	Boostexter	Naïve Bayes	Ensemble
Major topic $N = 20$	88.7% (.881)	86.5% (.859)	85.6% (.849)	81.4% (.805)	89.0% (.884)
Subtopic $N = 226$	81.0% (.800)	78.3% (.771)	73.6% (.722)	71.9% (.705)	81.0% (.800)

Note. Results are based on using approximately 187,000 human-labeled cases to train the classifier to predict approximately 187,000 other cases (that were also labeled by humans but not used for training). Agreement is computed by comparing the machine's prediction to the human assigned labels. (AC1 measure presented in parentheses).

AC1 is intercoder reliability corrected for chance agreement.

Results

TABLE 3. Bill Title Interannotator Agreement for Five Model Types

	SVM	MaxEnt	Boostexter	Naïve Bayes	Ensemble
Major topic *N* = 20	88.7% (.881)	86.5% (.859)	85.6% (.849)	81.4% (.805)	89.0% (.884)
Subtopic *N* = 226	81.0% (.800)	78.3% (.771)	73.6% (.722)	71.9% (.705)	81.0% (.800)

Note. Results are based on using approximately 187,000 human-labeled cases to train the classifier to predict approximately 187,000 other cases (that were also labeled by humans but not used for training). Agreement is computed by comparing the machine's prediction to the human assigned labels. (AC1 measure presented in parentheses).

AC1 is intercoder reliability corrected for chance agreement.

Improvement over SVM alone is real,

Results

TABLE 3. Bill Title Interannotator Agreement for Five Model Types

	SVM	MaxEnt	Boostexter	Naïve Bayes	Ensemble
Major topic $N = 20$	88.7% (.881)	86.5% (.859)	85.6% (.849)	81.4% (.805)	89.0% (.884)
Subtopic $N = 226$	81.0% (.800)	78.3% (.771)	73.6% (.722)	71.9% (.705)	81.0% (.800)

Note. Results are based on using approximately 187,000 human-labeled cases to train the classifier to predict approximately 187,000 other cases (that were also labeled by humans but not used for training). Agreement is computed by comparing the machine's prediction to the human assigned labels. (AC1 measure presented in parentheses).

AC1 is intercoder reliability corrected for chance agreement.

Improvement over SVM alone is real, though small.

Results

TABLE 3. Bill Title Interannotator Agreement for Five Model Types

	SVM	MaxEnt	Boostexter	Naïve Bayes	Ensemble
Major topic N = 20	88.7% (.881)	86.5% (.859)	85.6% (.849)	81.4% (.805)	89.0% (.884)
Subtopic N = 226	81.0% (.800)	78.3% (.771)	73.6% (.722)	71.9% (.705)	81.0% (.800)

Note. Results are based on using approximately 187,000 human-labeled cases to train the classifier to predict approximately 187,000 other cases (that were also labeled by humans but not used for training). Agreement is computed by comparing the machine's prediction to the human assigned labels. (AC1 measure presented in parentheses).

AC1 is intercoder reliability corrected for chance agreement.

Improvement over SVM alone is real, though small.

Classification especially good in cases where methods agree on topic.

Results

TABLE 3. Bill Title Interannotator Agreement for Five Model Types

	SVM	MaxEnt	Boostexter	Naïve Bayes	Ensemble
Major topic $N = 20$	88.7% (.881)	86.5% (.859)	85.6% (.849)	81.4% (.805)	89.0% (.884)
Subtopic $N = 226$	81.0% (.800)	78.3% (.771)	73.6% (.722)	71.9% (.705)	81.0% (.800)

Note. Results are based on using approximately 187,000 human-labeled cases to train the classifier to predict approximately 187,000 other cases (that were also labeled by humans but not used for training). Agreement is computed by comparing the machine's prediction to the human assigned labels. (AC1 measure presented in parentheses).

AC1 is intercoder reliability corrected for chance agreement.

Improvement over SVM alone is real, though small.

Classification especially good in cases where methods agree on topic.

Partner Exercise

Suppose you have trained an SVM or a classification tree on the yelp restaurant data (and cross validated appropriately!).

Partner Exercise

Suppose you have trained an SVM or a classification tree on the yelp restaurant data (and cross validated appropriately!). Now, new data comes in on a daily basis,

Partner Exercise

Suppose you have trained an SVM or a classification tree on the yelp restaurant data (and cross validated appropriately!). Now, new data comes in on a daily basis, and you want to 'update' your model according to the new training and test sets every day.

Partner Exercise

Suppose you have trained an SVM or a classification tree on the yelp restaurant data (and cross validated appropriately!). Now, new data comes in on a daily basis, and you want to 'update' your model according to the new training and test sets every day.

Such 'incremental' learning tasks are difficult/costly with the techniques presented above. Why?

Partner Exercise

Suppose you have trained an SVM or a classification tree on the yelp restaurant data (and cross validated appropriately!). Now, new data comes in on a daily basis, and you want to 'update' your model according to the new training and test sets every day.

Such 'incremental' learning tasks are difficult/costly with the techniques presented above. Why?

Imagine you apply the model from the 2015 data to 2016, and then every year thereafter: 2017, 2018...2024, 2025. Would you expect it to get more or less accurate over time? Why?