University of Salford MANCHESTER

Artificial Intelligent – Week 7

Dr Salem Ameen

S.A.AMEEN1@SALFORD.AC.UK

Week 7

- Today

1. Ensemble Methods
2. SVM
3. Logistic Regression
4. Naïve Bayes

Week 7

- DT (Last Week)

1. interpretable by humans
2. have sufficiently complex decision boundaries
3. the decision boundaries are locally linear, each component of the decision boundary is simple to describe mathematically.

Week 7
Ensemble Methods / Majority voting

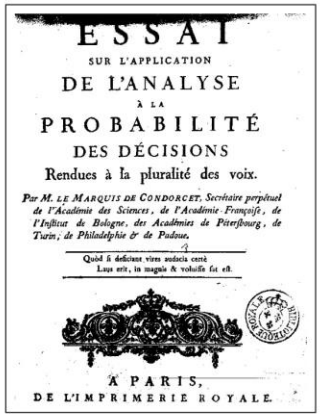
University of Salford MANCHESTER

- A randomly chosen hyperplane has an expected error of 0.5.
- Many random hyperplanes combined by majority vote will still be random.
- Suppose we have m classifiers, performing slightly better than random, that is error = 0.5-ε.
- Combine: make a decision based on majority vote?
- What if we combined these m slightly-better-than-random classifiers? Would majority vote be a good choice?

Week 7
Ensemble Methods / Majority voting

- Assumptions:
  - 1. Each individual makes the right choice with a probability p.
  - 2. The votes are independent.
- If p > 0.5, then adding more voters increases the probability that the majority decision is correct. if p < 0.5, then adding more voters makes things worse.

Marquis de Condorcet Application of Analysis to the Probability of Majority Decisions. 1785.


ESSAI
SUR L'APPLICATION
DE L'ANALYSE
À LA
PROBABILITÉ
DES DÉCISIONS
Rendues à la pluralité des voix.
Par M. LE MARQUIS DE CONDORCET, Secrétaire perpétuel de l'Académie des Sciences, de l'Académie Françoise, de l'Institut de Bologne, des Académies de Pétersbourg, de Turin, de Philadelphie & de Padoue,
Quid fi deficiens vires audacia certæ
Laus erit, in magnis & voluiffe fat eft.
A PARIS,
DE L'IMPRIMERIE ROYALE.

Week 7
Ensemble Methods / Majority voting

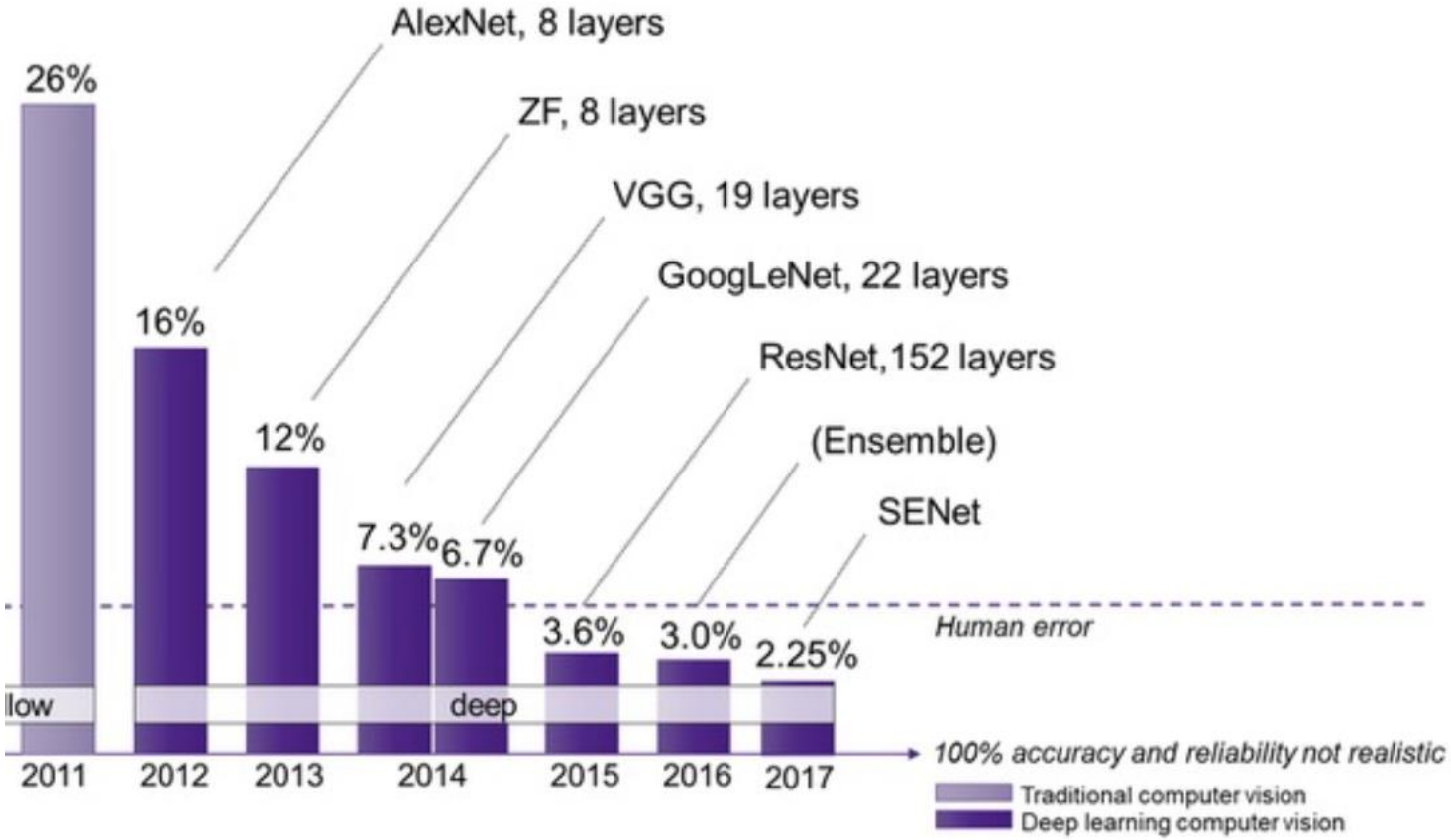
University of Salford MANCHESTER

- Classification (Voting)

- Regression (Average)

Week 7
Ensemble Methods / Majority voting

- Netflix Prize
- ImageNet



Week 7
Ensemble Methods / Majority voting

University of Salford MANCHESTER

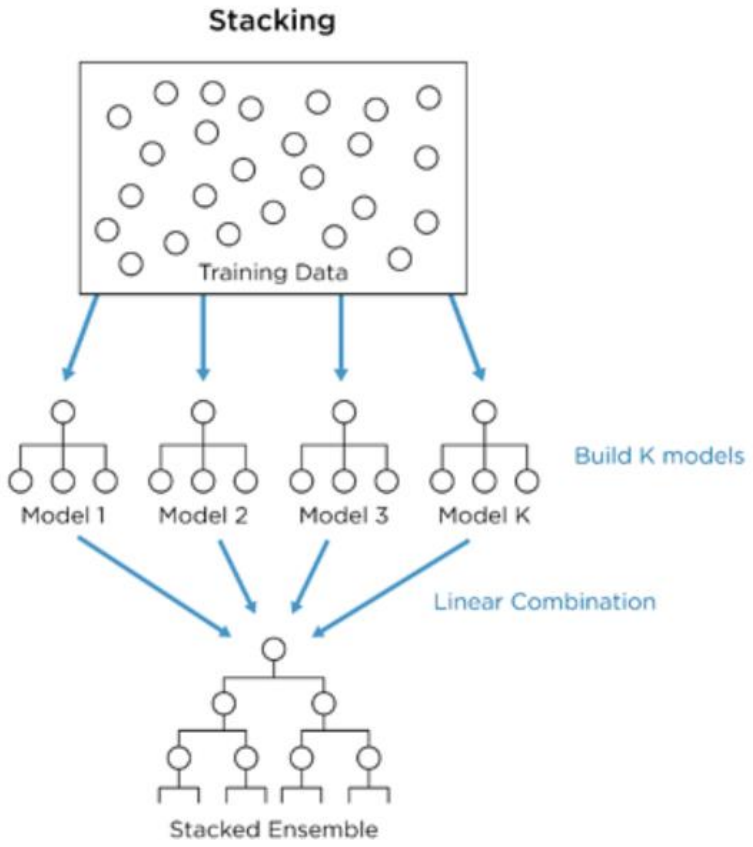
```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier

clf1 = LogisticRegression(multi_class='multinomial', random_state=1)
clf2 = GaussianNB()

eclf1 = VotingClassifier(estimators=[ ('lr', clf1), ('gnb', clf2)])

eclf1 = eclf1.fit(X_train, y_train)
eclf1.predict(X_test)
```

Week 7
Ensemble Methods / Majority voting

University of Salford MANCHESTER

```
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import VotingRegressor

r1 = LinearRegression()
r2 = KNeighborsRegressor()

Er = VotingRegressor([('lr', r1), ('rf', r2)])

Er.fit(X_train,y_train)

Er.predict(X_test)
```

Week 7
Ensemble Methods / Majority voting / Stacking



Week 7
Ensemble Methods / Majority voting / Stacking

University of Salford MANCHESTER

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import StackingClassifier

estimators = [  ('rf', RandomForestClassifier(n_estimators=10)),
('svr', make_pipeline(StandardScaler(), LinearSVC(random_state=42))) ]

 clf = StackingClassifier( estimators=estimators, final_estimator=LogisticRegression()
```

Week 7
Ensemble Methods / Majority voting

University of Salford MANCHESTER

- An Ensemble Method combines the predictions of many individual classifiers by majority voting.
- Such individual classifiers, called weak learners, are required to perform slightly better than random.
- How do we produce independent weak learners using the same training data?
- Use a strategy to obtain relatively independent weak learners!
- Different methods:
  1. **Bagging**
  2. **Random Forests**
  3. **Boosting**

Week 7
Ensemble Methods / Majority voting

University of Salford MANCHESTER

- **Bagging (bootstrap aggregation)**
  - Each weak learner is trained on a random subset of the data.
- **Random forests**
  - Bagging with tree classifiers as weak learners.
  - Uses an additional step to remove dimensions that carry little information.
- **Boosting**
  - After training each weak learner, data is modified using weights.
  - Deterministic algorithm.

Week 7
Bagging

University of Salford MANCHESTER

- One way to adjust for the **high variance** of the output of an experiment is to perform the experiment multiple times and then average the results.
- The same idea can be applied to high variance models:
    1. **(Bootstrap)**we generate multiple samples of training data, via bootstrapping. We train a full decision tree on each sample of data.
    2. **(Aggregate)**for a given input, we output the averaged outputs of all the models for that input.
- For classification, we return the class that is outputted by the plurality of the models. For regression we return the average of the outputs for each tree.
- This method is called ***Bagging*** (Breiman, 1996), short for, of course, Bootstrap Aggregating.

Week 7
Bagging

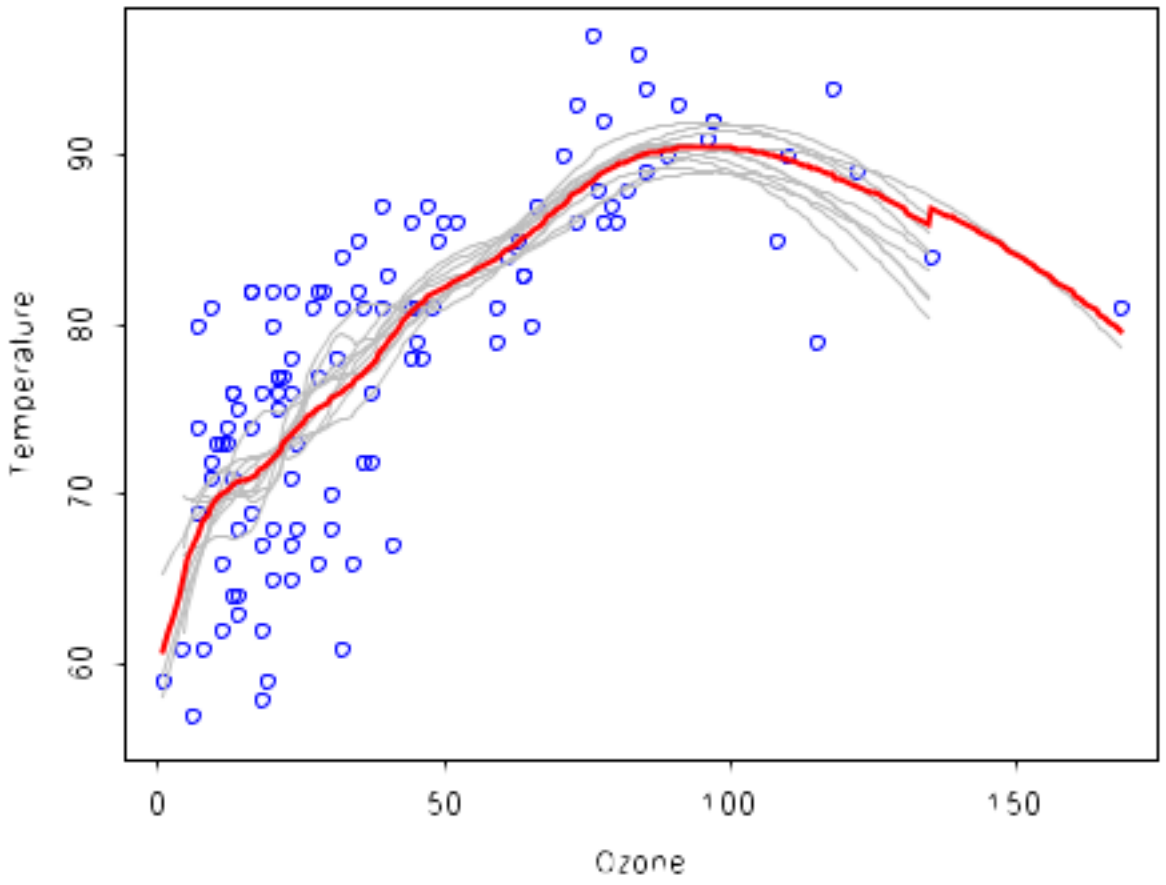
University of Salford MANCHESTER

- Note that bagging enjoys the benefits of:

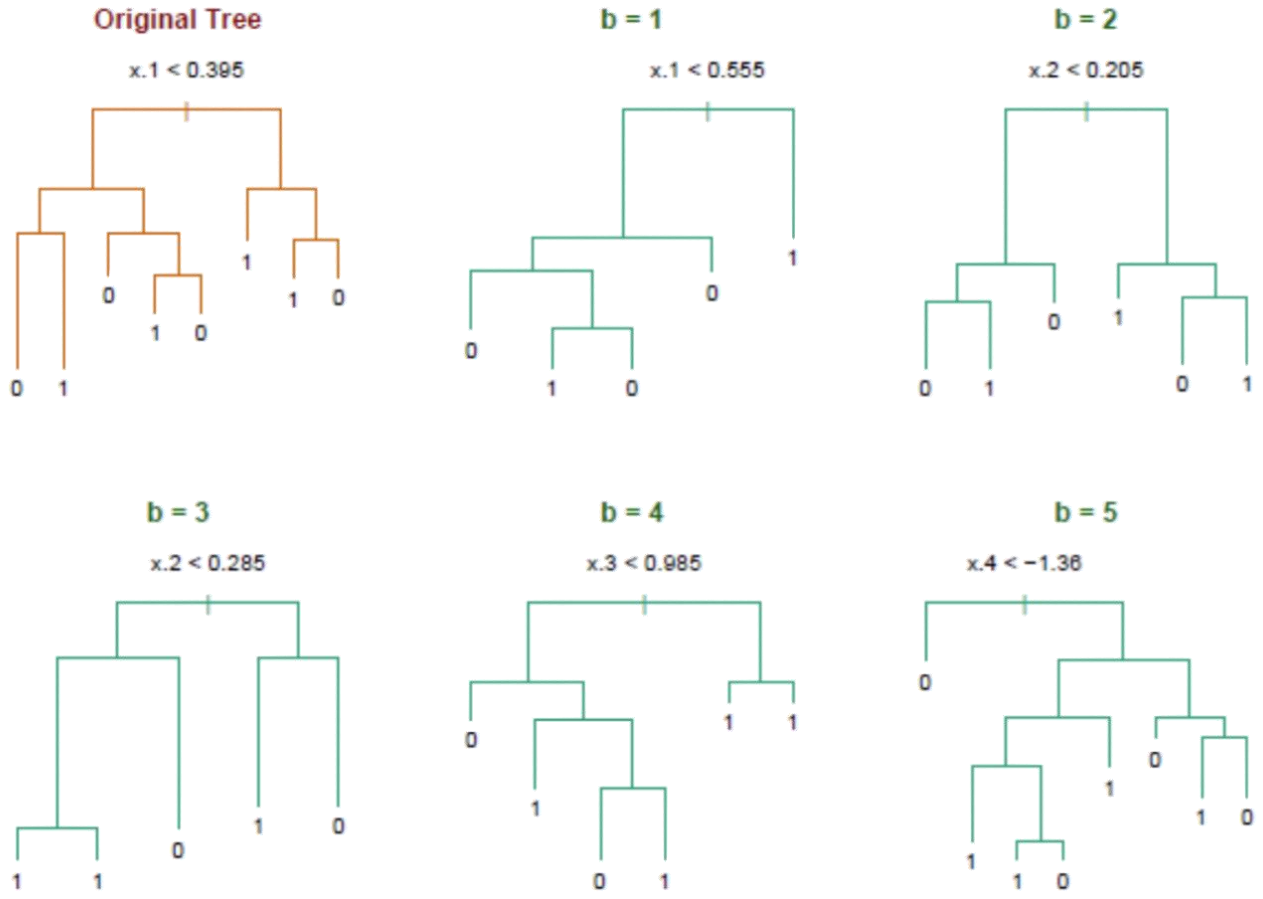
1. High expressiveness - by using full trees each model is able to approximate complex functions and decision boundaries.

2. Low variance - averaging the prediction of all the models reduces the variance in the final prediction, assuming that we choose a sufficiently large number of trees.

Week 7
Bagging (regression)



Week 7
Bagging (classification)

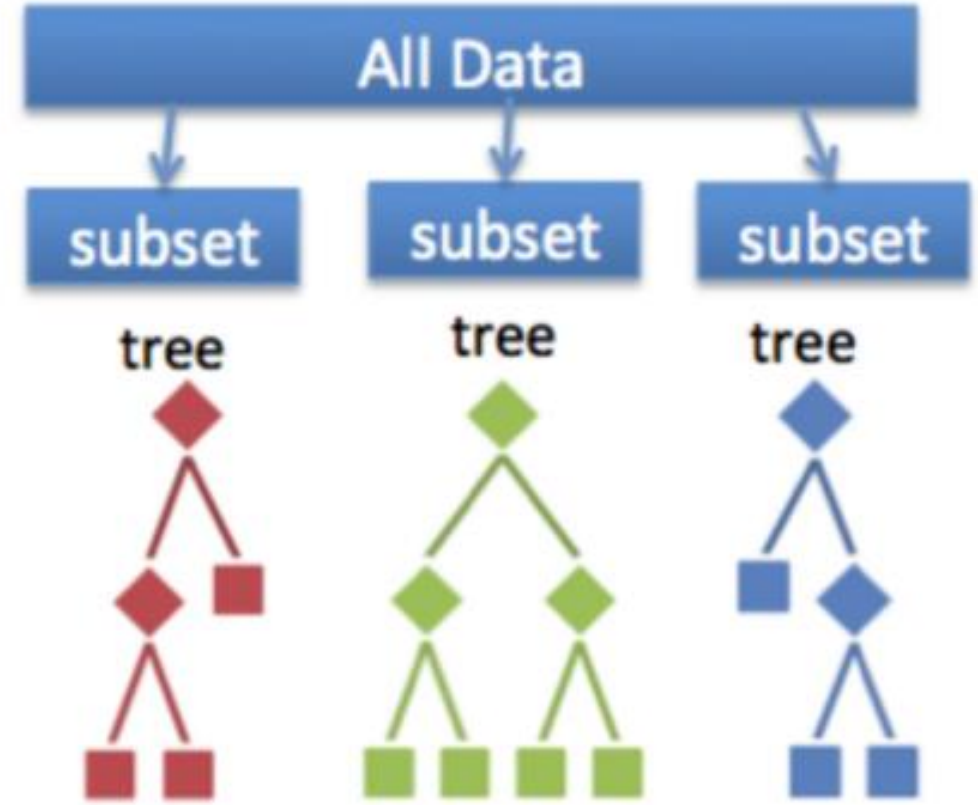


Week 7
Bagging

- **Question:** Do you see any problems?

- Still some overfitting if the trees are too large.
- If trees are too shallow it can still underfits.
- Interpretability: The **major drawback** of bagging (and other ***ensemble methods*** that we will study) is that the averaged model is no longer easily interpretable - i.e. one can no longer trace the 'logic' of an output through a series of decisions based on predictor values!

Week 7
Bagging



Week 7
Bagging

University of Salford MANCHESTER

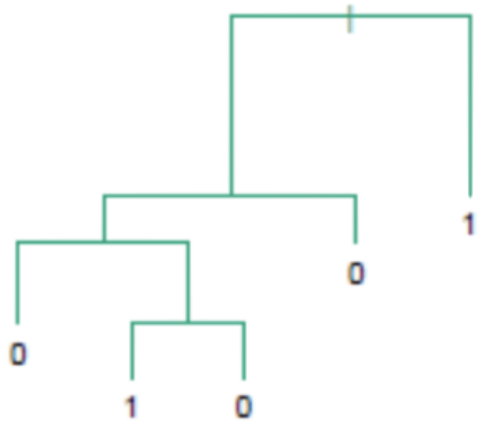
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bootstrap Sample 1

X	Y
X_4	y_4
X_{14}	y_{14}
X_{11}	y_{11}
X_2	y_2
X_{35}	y_{35}
⋮	⋮
X_k	y_k

Decision Tree 1



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Week 7
Bagging

University of Salford MANCHESTER

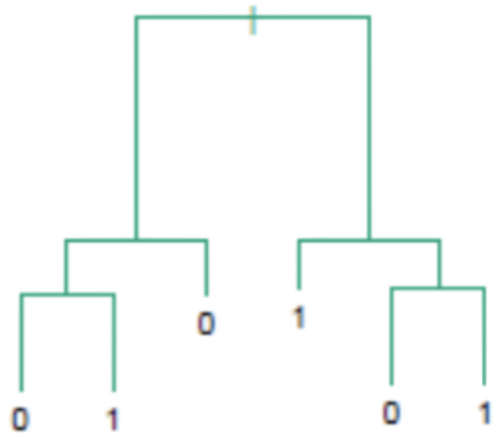
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bootstrap Sample 2

X	Y
X_5	y_5
X_3	y_3
X_{12}	y_{12}
X_{43}	y_{43}
X_1	y_1
⋮	⋮
X_k	y_k

Decision Tree 2



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Week 7
Bagging

University of Salford MANCHESTER

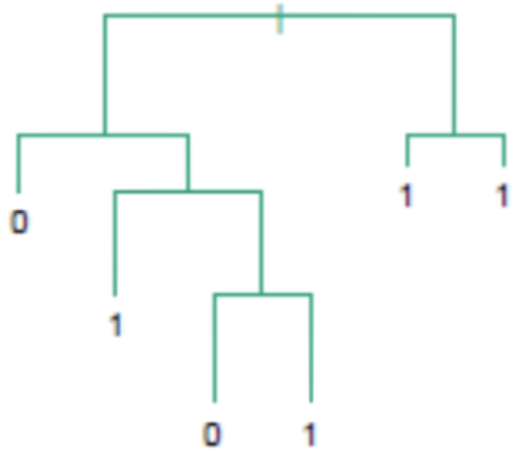
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

Bootstrap Sample 3

X	Y
X_9	y_9
X_4	y_4
X_1	y_1
X_1	y_1
X_{65}	y_{65}
⋮	⋮
X_k	y_k

Decision Tree 3



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
⋮	⋮
X_n	y_n

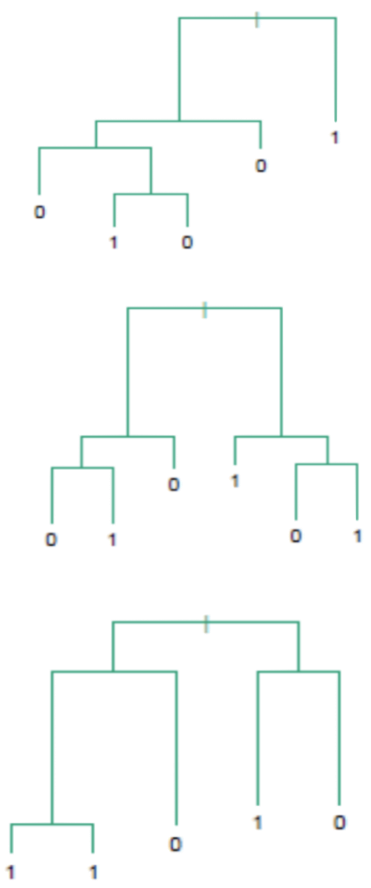
Week 7
Point-wise out-of-bag error

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
⋮	⋮
X_i	y_i
⋮	⋮
X_n	y_n

Week 7
Point-wise out-of-bag error

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
⋮	⋮
X_i	y_i
⋮	⋮
X_n	y_n

B Trees that did not see $\{X_i, y_i\}$



Classification

$$\hat{y}_{i,pw} = \text{majority}(\hat{y}_i)$$

$$e_i = \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

Regression

$$\hat{y}_{i,pw} = \sum_{j \in B} \hat{y}_{i,j}$$

$$e_i = \left(y_i - \hat{y}_{i.pw}\right)^2$$

Week 7
OOB Error

University of Salford MANCHESTER

- We average the point-wise out-of-bag error over the full training set.

Classification

$$Error_{OOB} = \sum_{i}^{n} e_i = \sum_{i}^{n} \mathbb{I}(\hat{y}_{i,pw} \neq \mathrm{y_i})$$

Regression

$$Error_{OOB} = \sum_{i}^{n} e_i = \sum_{i}^{n} \left(y_i - \hat{y}_{i,pw}\right)^2$$

Week 7
Out-of-Bag Error

University of Salford MANCHESTER

- Bagging is an example of an ***ensemble method***, a method of building a single model by training and aggregating multiple models.
- With ensemble methods, we get a new metric for assessing the predictive performance of the model, the ***out-of-bag error***.
- Given a training set and an ensemble of models, each trained on a bootstrap sample, we compute the ***out-of-bag error*** of the averaged model by

    1. For each point in the training set, we average the predicted output for this point over the models whose bootstrap training set excludes this point. We compute the error or squared error of this averaged prediction. Call this the point-wise out-of-bag error.
    2. We average the point-wise out-of-bag error over the full training set.

Week 7
Improving on Bagging

University of Salford MANCHESTER

- In practice, the ensembles of trees in Bagging tend to be highly correlated.

- Suppose we have an extremely strong predictor, x_j , in the training set amongst moderate predictors. Then the greedy learning algorithm ensures that most of the models in the ensemble will choose to split on x_j in early iterations.

- That is, each tree in the ensemble is identically distributed, with the expected output of the averaged model the same as the expected output of any one of the trees.

Week 7
Bagging – Sklearn - Classification

University of Salford MANCHESTER

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

clf = BaggingClassifier(base_estimator= DecisionTreeClassifier(),  n_estimators=10, random_state=0)
clf.fit(X_train, y_train)

 clf.predict(X_test)
```

Week 7
Bagging – Sklearn - Classification

University of Salford MANCHESTER

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

clf = BaggingClassifier(base_estimator= DecisionTreeClassifier(),  n_estimators=10, random_state=0)
clf.fit(X_train, y_train)

 clf.predict(X_test)
```

```
from sklearn.svm import SVM
from sklearn.ensemble import BaggingClassifier

clf = BaggingClassifier(base_estimator= SVM(),  n_estimators=10, random_state=0)
clf.fit(X_train, y_train)

 clf.predict(X_test)
```

Week 7
Bagging – Sklearn - Regression

University of Salford MANCHESTER

```
from sklearn.svm import SVR
from sklearn.ensemble import BaggingRegressor

regr = BaggingRegressor(base_estimator=SVR(), n_estimators=10, random_state=0).fit(X_train, y_train)

regr.predict(X_test)
```

Week 7
Random Forests

University of Salford MANCHESTER

1. Random forests: modifies bagging with trees to reduce correlation between trees.
2. Tree training optimizes each split over all dimensions.
3. But for Random forests, **choose a different subset of dimensions at each split**. Number of dimensions chosen m.
4. Optimal split is chosen within the subset.
5. The subset is chosen at random out of all dimensions $1, \ldots, d$.
6. Recommended $m = \sqrt{d}$ or smaller. d/3 for regression.

Week 7
Random Forests

University of Salford MANCHESTER

```
from sklearn.ensemble import RandomForestClassifier


clf = RandomForestClassifier(max_depth=2, random_state=0)

clf.fit(X_train, y_train)

clf.predict(X_test)
```

Week 7
Random Forests

University of Salford MANCHESTER

```
from sklearn.ensemble import RandomForestRegressor


clf = RandomForestRegressor(max_depth=2, random_state=0)

clf.fit(X_train, y_train)

clf.predict(X_test)
```

Week 7
Boosting

University of Salford MANCHESTER

- The predictions from all of the $G_m,\ m \in \{1, \cdots, M\}$ are combined with a weighted majority voting.
- α_m is the contribution of each weak learner G_m.
- Computed by the boosting algorithm to give a weighted importance to the classifiers in the sequence.
- The decision of a highly-performing classifier in the sequence should weight more than less important classifiers in the sequence.
- This is captured in:

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$$

Week 7
Boosting / AdaBoost

University of Salford MANCHESTER

1. Initialize the example weights $w_i = \frac{1}{n},\ i = 1, \cdots, n.$
2. For $m = 1$ to M (number of weak learners)
   (a) Fit a classifier $G_m(x)$ to training data using the weights w_i.
   (b) Compute

$$err_m := \frac{\sum_{i=1}^{n} \quad w_i \quad 1\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^{n} w_i}$$

   (c) Compute

$$\alpha_m = log(\frac{1 - err_m}{err_m})$$

   (d) $w_i \leftarrow w_i.exp[\alpha_m.1(y_i \neq G_m(x_i))] \qquad for\ i = 1, \cdots, n.$
3. Output

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)]$$

Week 7
Boosting and Bagging

University of Salford MANCHESTER

Bagging	vs.	Boosting
Resamples data points		Reweights data points (modifies their distribution)
Weight of each classifier is the same		Weight is dependent on classifier's accuracy
Only variance reduction		Both bias and variance reduced – learning rule becomes more complex with each iteration

Week 7
AdaBoost - Sklearn

University of Salford MANCHESTER

```
from sklearn.ensemble import AdaBoostClassifier
```

```
clf = AdaBoostClassifier(n_estimators=100, random_state=0)
clf.fit(X, y)
```

AdaBoostRegressor

Week 7
Gradient Boosting

University of Salford MANCHESTER

- The key intuition behind boosting is that one can take an ensemble of simple models $\{T_h\}_{h \in H}$ and additively combine them into a single, more complex model.

- Each model T_h might be a poor fit for the data, but a linear combination of the ensemble

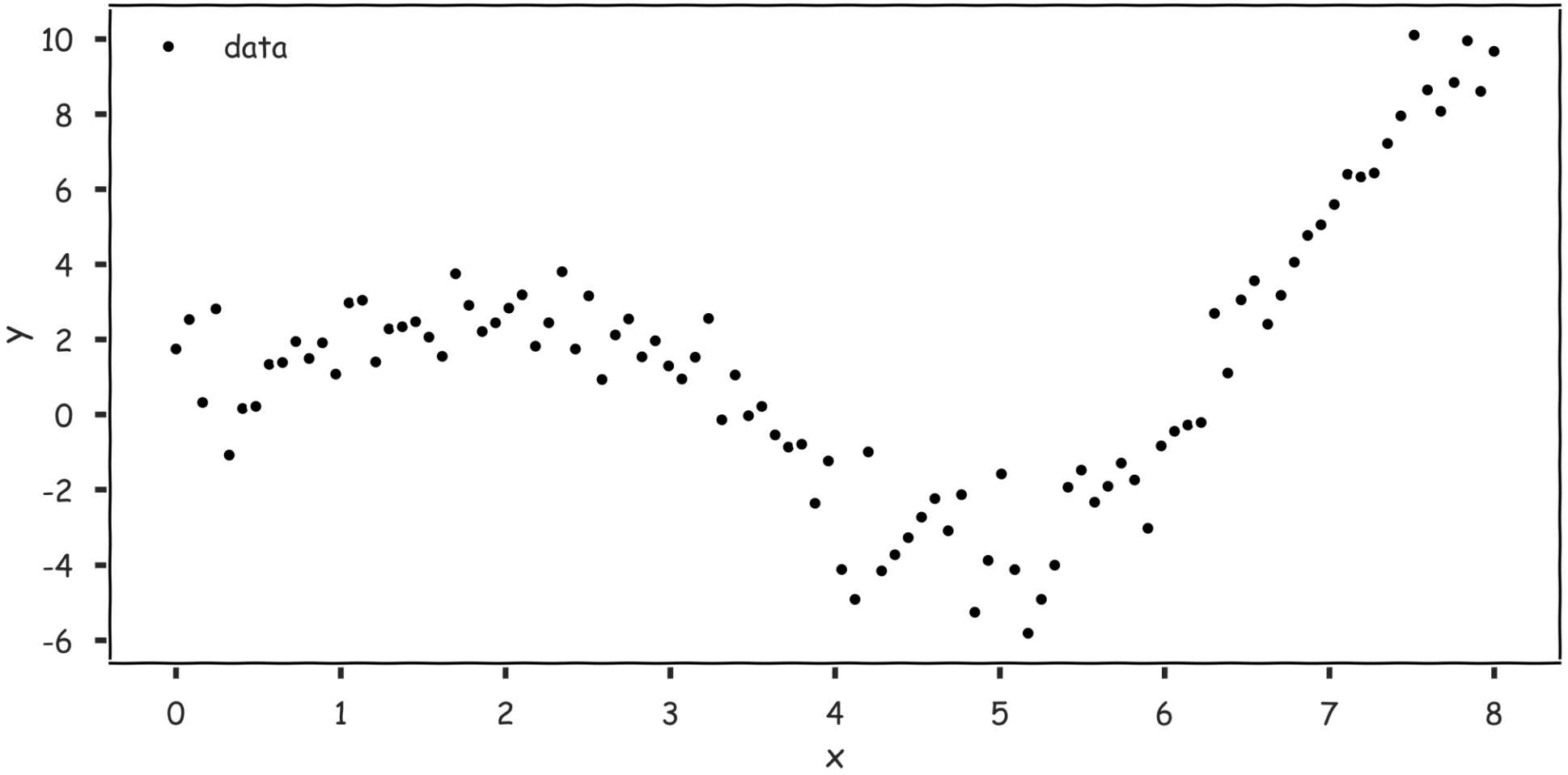
- can be expressive/flexible. $$T = \sum_h \lambda_h T_H$$

- **Question:** But which models should we include in our ensemble? What should the coefficients or weights in the linear combination be?

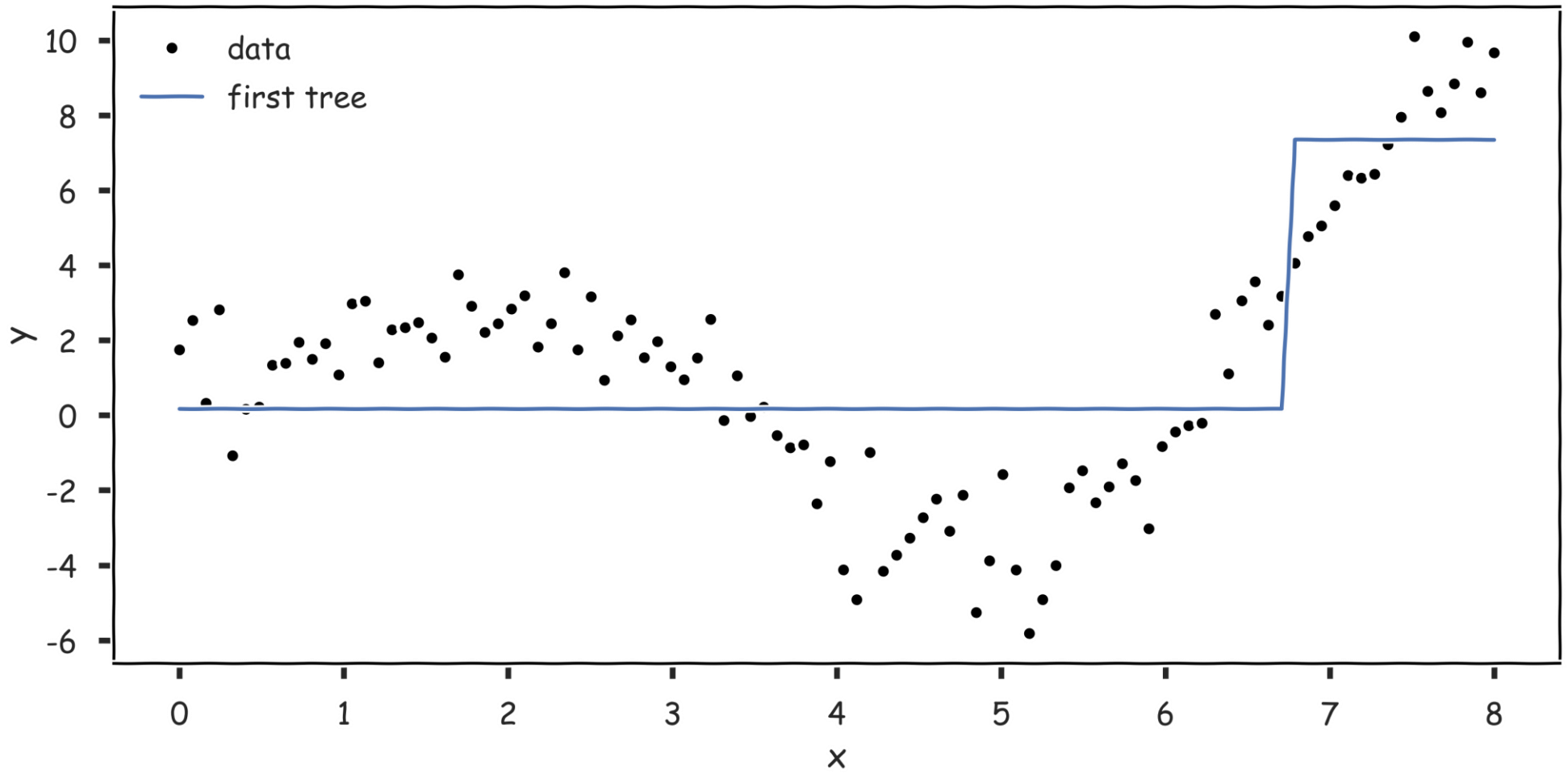
Week 7
Gradient Boosting

- ***Gradient boosting*** is a method for iteratively building a complex regression model T by adding simple models. Each new simple model added to the ensemble compensates for the weaknesses of the current ensemble.
- 1. Fit a simple model $T^{(0)}$ on the training data
  - $\{(x_1, y_1), \ldots, (x_N, y_N)\}$
- Set $T \leftarrow T^{(0)}$. Compute the residuals $\{r_1, \ldots, r_N\}$ for T.
- 2. Fit a simple model, $T^{(1)}$, to the current ***residuals***, i.e. train using
  - $\{(x_1, r_1), \ldots, (x_N, r_N)\}$
- 3. Set $T \leftarrow T + \lambda T^{(1)}$
- 4. Compute residuals, set $r_n \leftarrow r_n - \lambda T^i(x_n), \ n = 1, \ldots, N$
- 5. Repeat steps 2-4 until ***stopping*** condition met.
- where λ is a constant called the ***learning rate***.

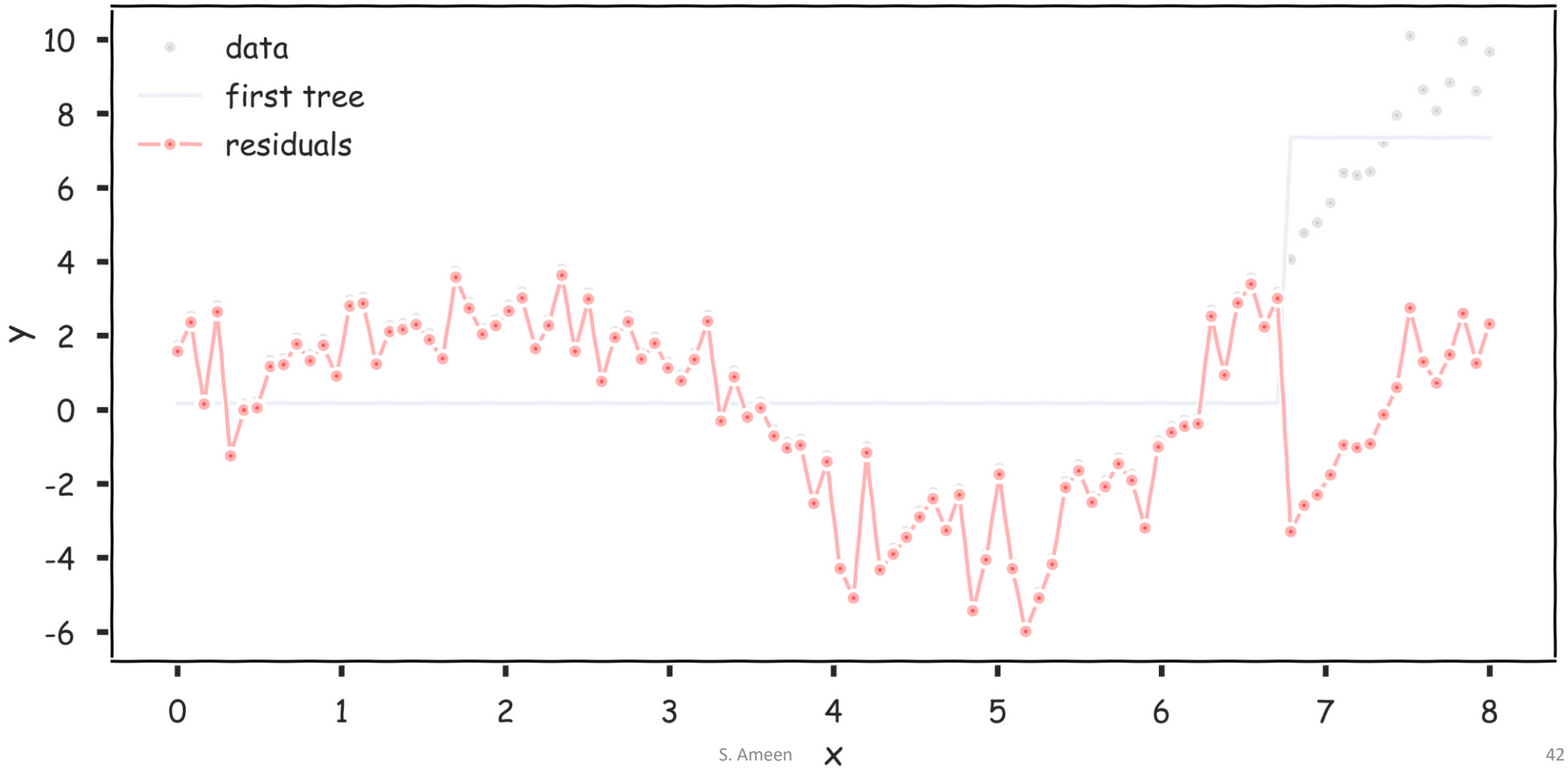
Week 7
Gradient Boosting: illustration



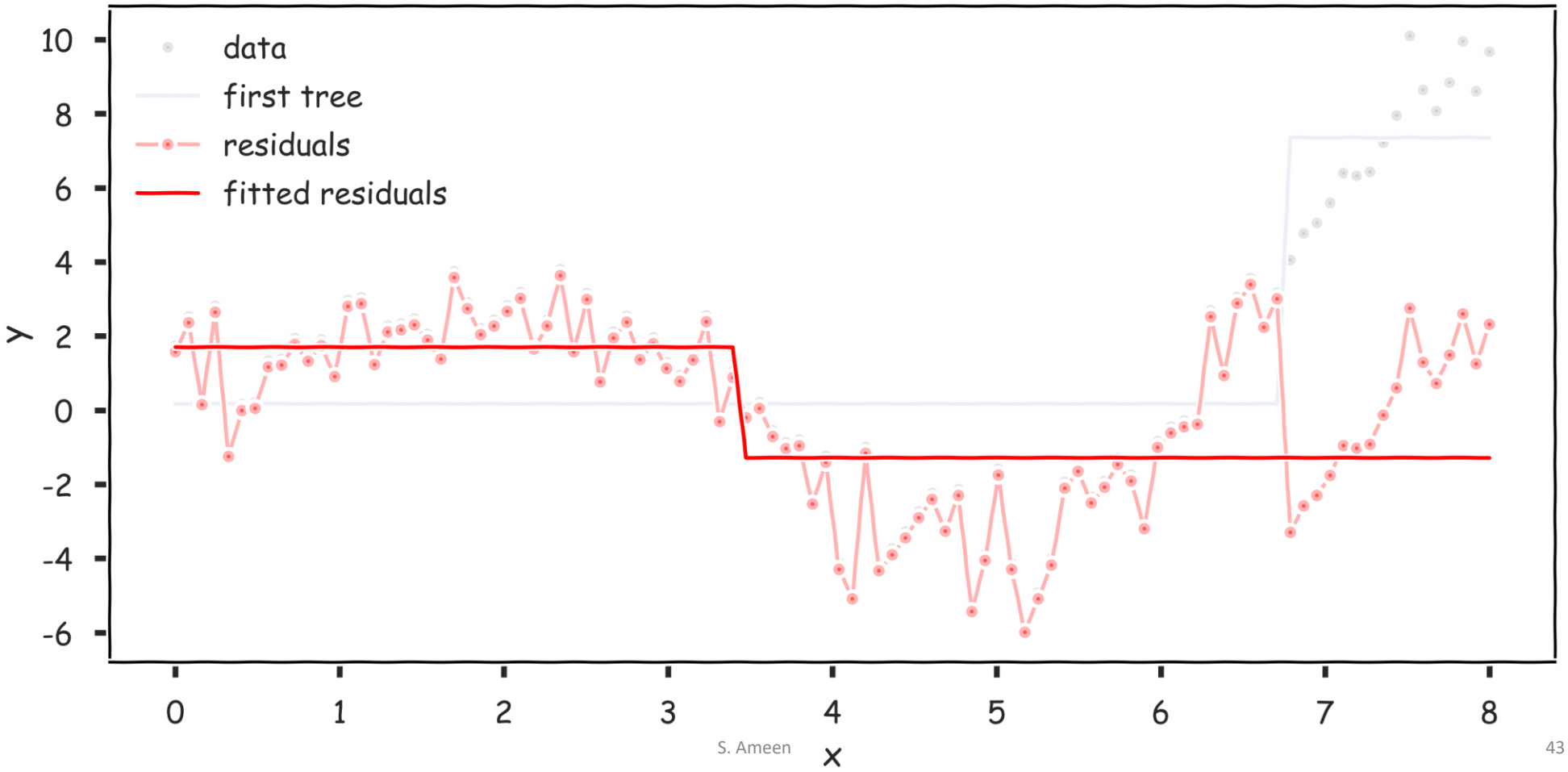
Week 7
Gradient Boosting: illustration



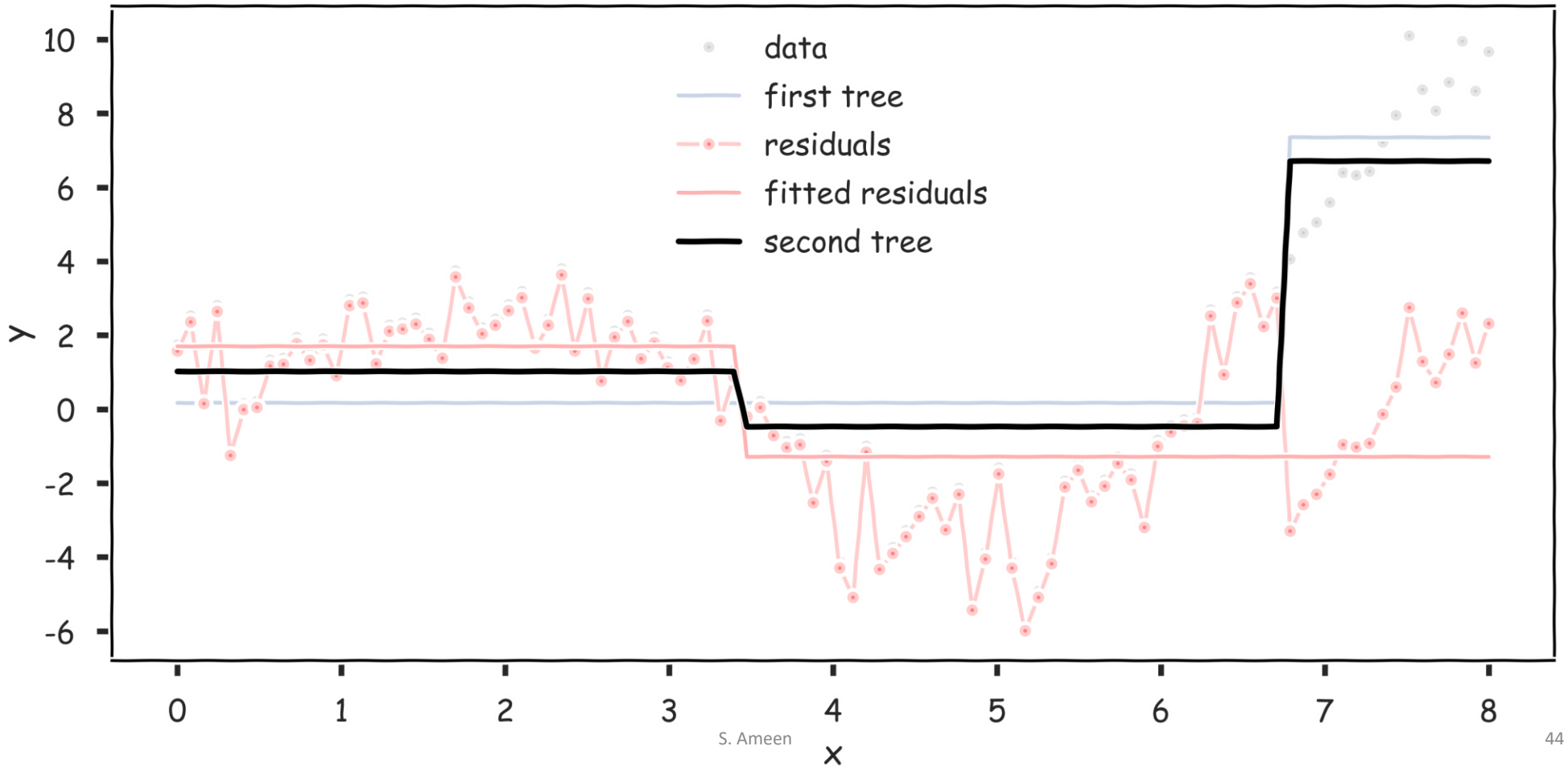
Week 7
Gradient Boosting: illustration



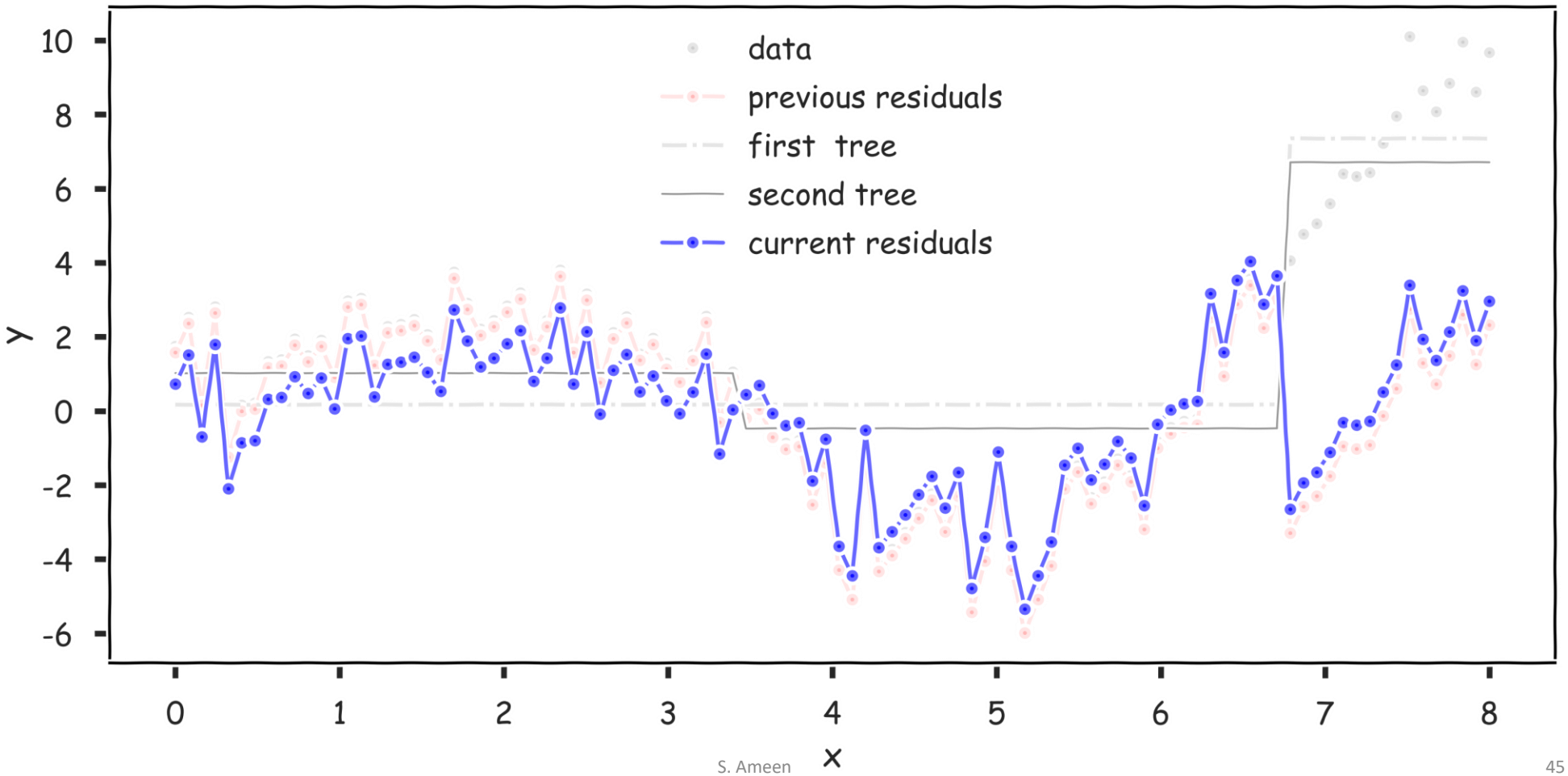
Week 7
Gradient Boosting: illustration



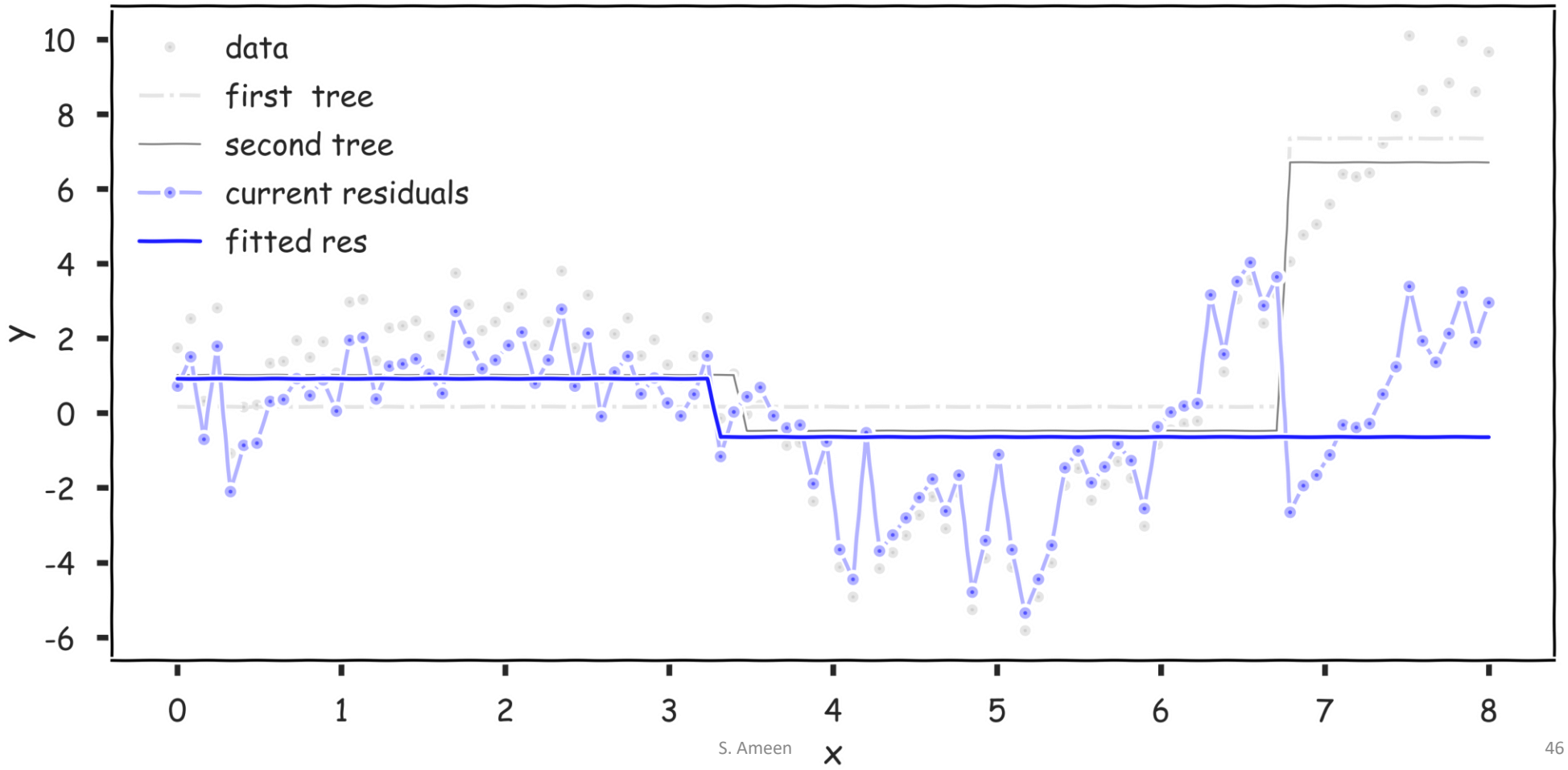
Week 7
Gradient Boosting: illustration



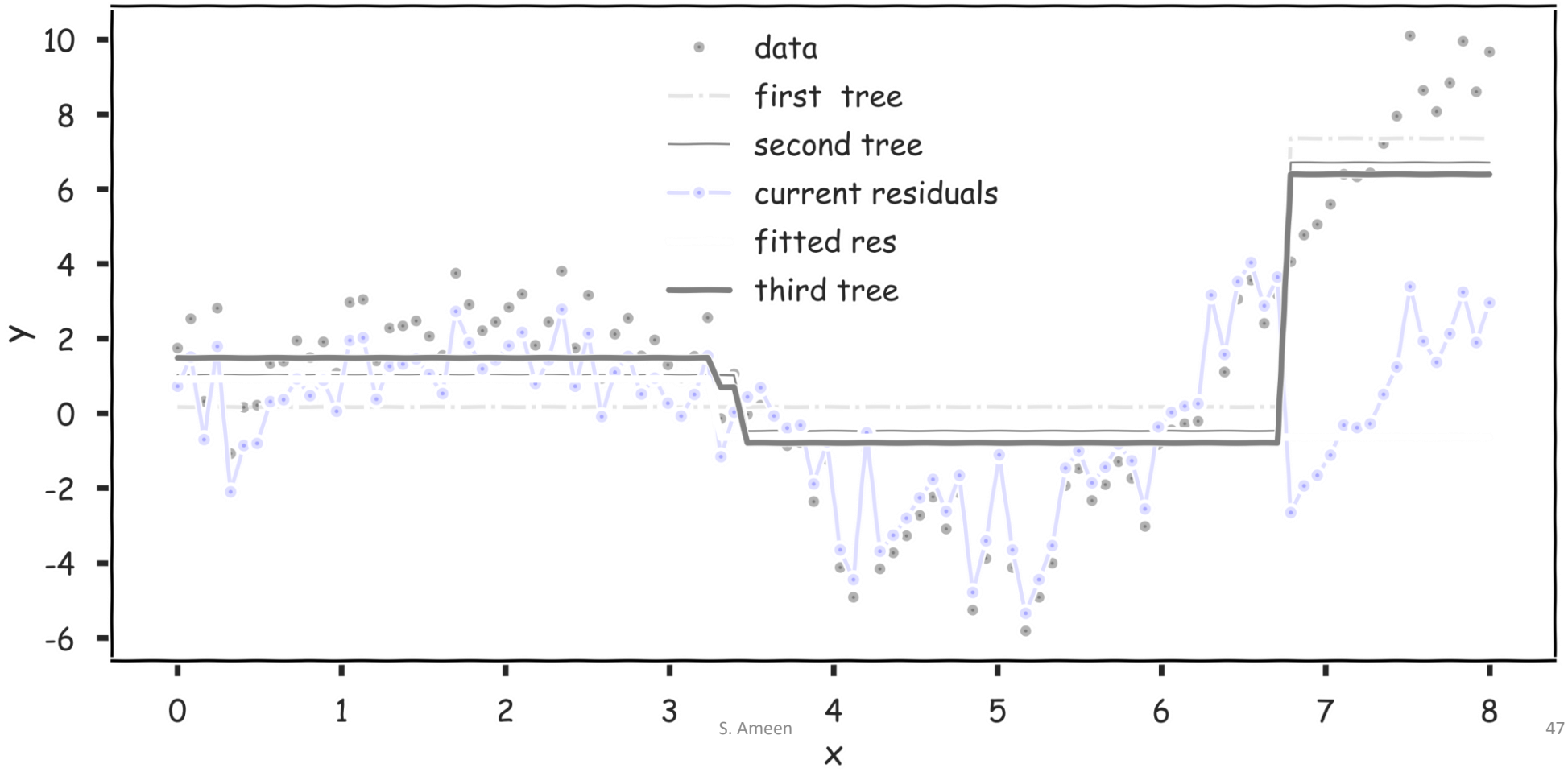
Week 7
Gradient Boosting: illustration



Week 7
Gradient Boosting: illustration



Week 7
Gradient Boosting: illustration



Week 7
Gradient Boosting - Sklearn

University of Salford MANCHESTER

```
from sklearn.ensemble import GradientBoostingClassifier


clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=2,
random_state=0)

clf.fit(X_train, y_train)
```

GradientBoostingRegressor

Week 7
Other Gradient Boosting

University of Salford MANCHESTER

```
import xgboost as xgb
import lightgbm as lgb
```

Week 7
Other Gradient Boosting

University of Salford MANCHESTER

```
num_folds = 2
from operator import itemgetter
def report(grid_scores, n_top=5):
    params = None
    top_scores = sorted(grid_scores, key=itemgetter(1), reverse=True)[:n_top]
    for i, score in enumerate(top_scores):
        print("Parameters with rank: {0}".format(i + 1))
        print("Mean validation score: {0:.4f} (std: {1:.4f})".format(
              score.mean_validation_score, np.std(score.cv_validation_scores)))
        print("Parameters: {0}".format(score.parameters))
        print("")
```

```
        if params == None:
            params = score.parameters

    return params

# The most common value for the max number of features to look at in each split␣
 ↪is sqrt(# of features)
sqrtfeat = np.sqrt(X.shape[1])
grid_test1 = { "max_depth"      : [1,2,3,4,5]}

# Large randomized test using max_depth to control tree size (5000 possible␣
 ↪combinations)
random_test1 = { "max_depth"      : np.arange(1,6)}


forest = xgboost.XGBClassifier()

#print ("Hyperparameter optimization using GridSearchCV...")
#grid_search = GridSearchCV(forest, grid_test1, n_jobs=-1, cv=num_folds)
#grid_search.fit(X, Y)
#print(grid_search.best_estimator_)
#print ('++++++++++++++++++++++++++++++++++++++++++++++')
print ("Hyperparameter optimization using RandomizedSearchCV")
Random_search = RandomizedSearchCV(forest, random_test1, n_jobs=-1,␣
 ↪cv=num_folds, n_iter=5)
Random_search.fit(X, Y)
print(Random_search.best_estimator_)
```

Week 7
Numerical vs Categorical Attributes

University of Salford MANCHESTER

- Note that the ‘compare and branch’ method by which we defined classification tree works well for numerical features.

- However, if a feature is categorical (with more than two possible values), comparisons like feature < threshold does not make sense.

- How can we handle this?

- A simple solution is to encode the values of a categorical feature using numbers and treat this feature like a numerical variable. This is indeed what some computational libraries (e.g. sklearn) do, however, this method has drawbacks.

Week 7
Numerical vs Categorical Attributes

University of Salford MANCHESTER

	Gender	City	Temperature	Rating
0	female	New York	low	4
1	female	London	medium	3
2	male	New Delhi	high	2

Week 7
Numerical vs Categorical Attributes

University of Salford MANCHESTER

```
from sklearn.preprocessing import LabelEncoder
data['City_encoded'] = LabelEncoder().fit_transform(data['City'])
data[['City', 'City_encoded']] # special syntax to get just these two columns
```

	City	City_encoded
0	New York	2
1	London	0
2	New Delhi	1

	Gender	City	Temperature	Rating
0	female	New York	low	4
1	female	London	medium	3
2	male	New Delhi	high	2

Week 7
Numerical vs Categorical Attributes
Beyond DT

University of Salford MANCHESTER

	Gender	City	Temperature	Rating
0	female	New York	low	4
1	female	London	medium	3
2	male	New Delhi	high	2

	City_London	City_New Delhi	City_New York
0	0.0	0.0	1.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0

	City	City_encoded
0	New York	2
1	London	0
2	New Delhi	1

One-hot encoding

Week 7
Numerical vs Categorical Attributes
Beyond DT

University of Salford MANCHESTER

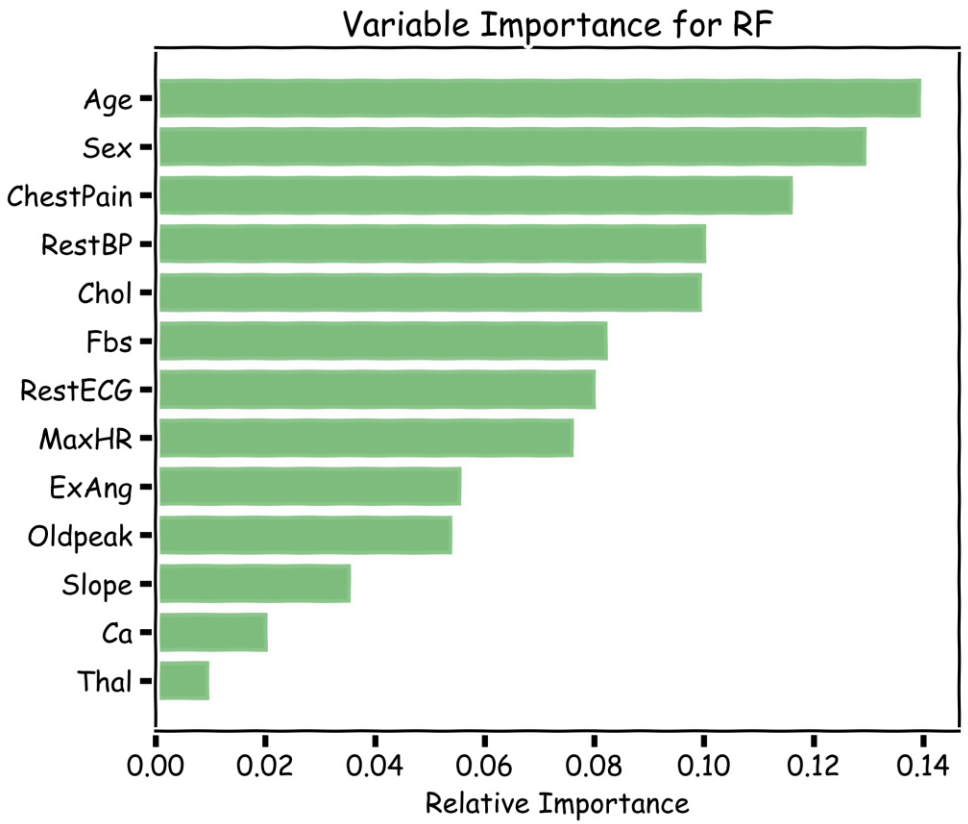
from sklearn.preprocessing import **OneHotEncoder**

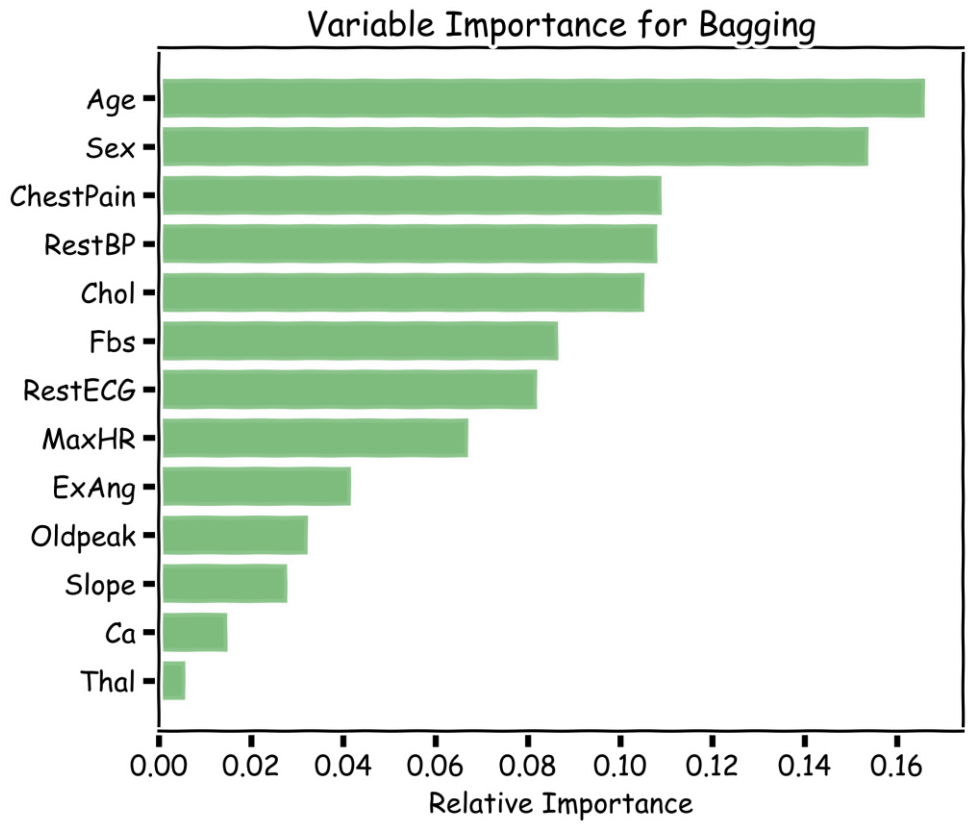
data['City_encoded'] = OneHotEncoder.fit_transfor(data['city'])
data[['city_encoded']]

	City_London	City_New Delhi	City_New York
0	0.0	0.0	1.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0

	Gender	City	Temperature	Rating
0	female	New York	low	4
1	female	London	medium	3
2	male	New Delhi	high	2

Week 7
Variable Importance for Bagging



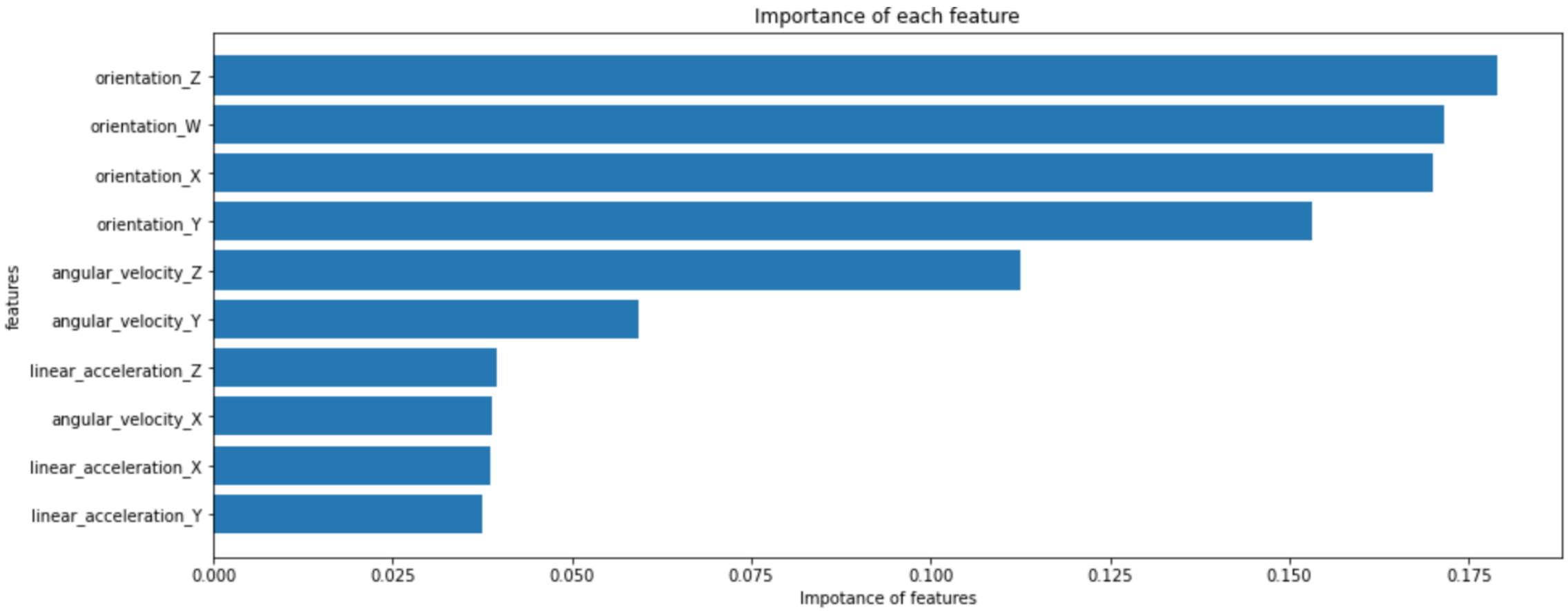


Week 7
Variable Importance for Bagging

University of Salford MANCHESTER

```
model = ExtraTreesClassifier()
#model = RandomForestClassifier(n_estimators = 15)
model.fit(X_train, y_train)
# display the relative importance of each attribute
print(model.feature_importances_)
imp = model.feature_importances_
imp, names = zip(*sorted(zip(imp,names)))
plt.barh(range(len(names)), imp, align = 'center')
plt.yticks(range(len(names)), names)
plt.rcParams['figure.figsize'] = (20, 30)
plt.xlabel('Impotance of features')
plt.ylabel('features')
plt.title('Importance of each feature')
plt.show()
```

Week 7
Variable Importance for Bagging



Week 7
Support Vector Machine (SVM)

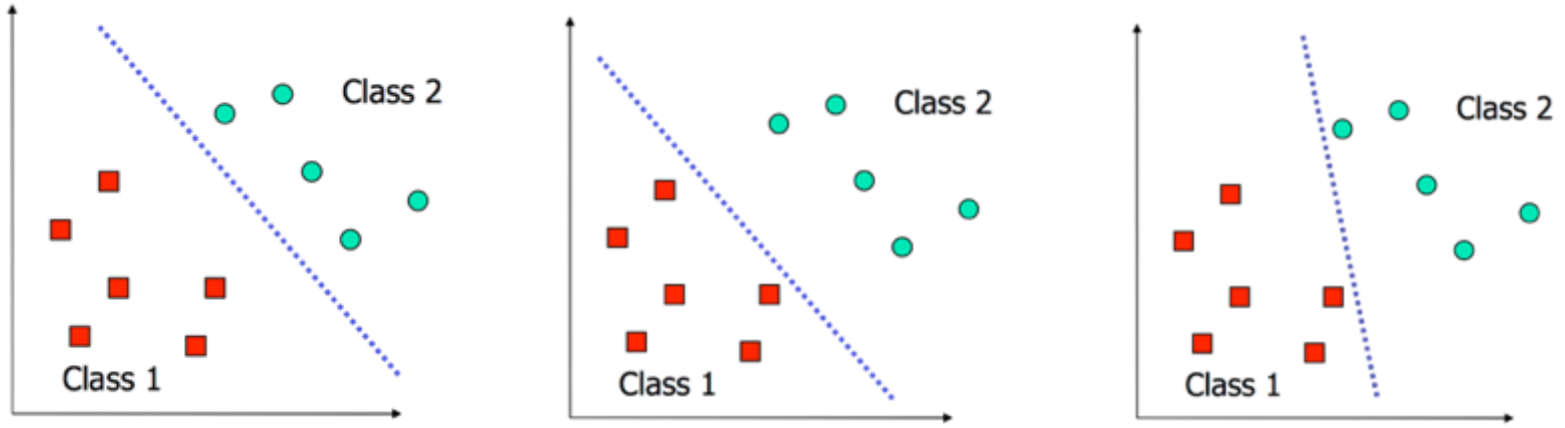
University of Salford MANCHESTER

- Classifying Linear Separable Data
- Classifying Linear Non-Separable Data
  - Kernel Trick

Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

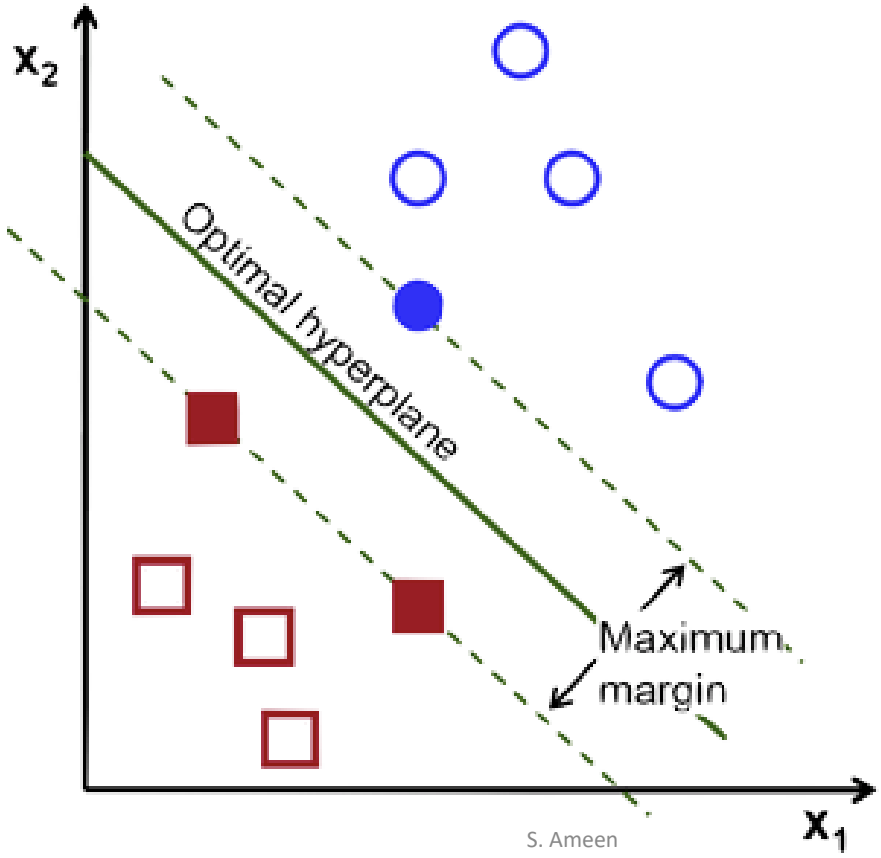
- Classifying Linear Separable Data



Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

- Classifying Linear Separable Data

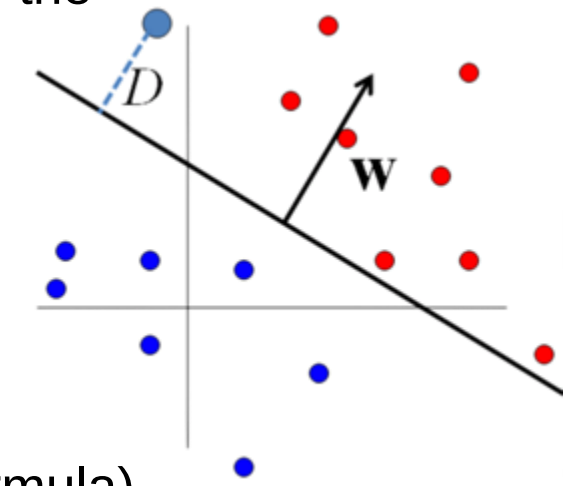


Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

- Classifying Linear Separable Data

Now, using some geometry, we can compute the distance between any point to the decision boundary using *w* and *b*.



The signed distance from a point $x \in \mathbb{R}^n$ to the decision boundary is

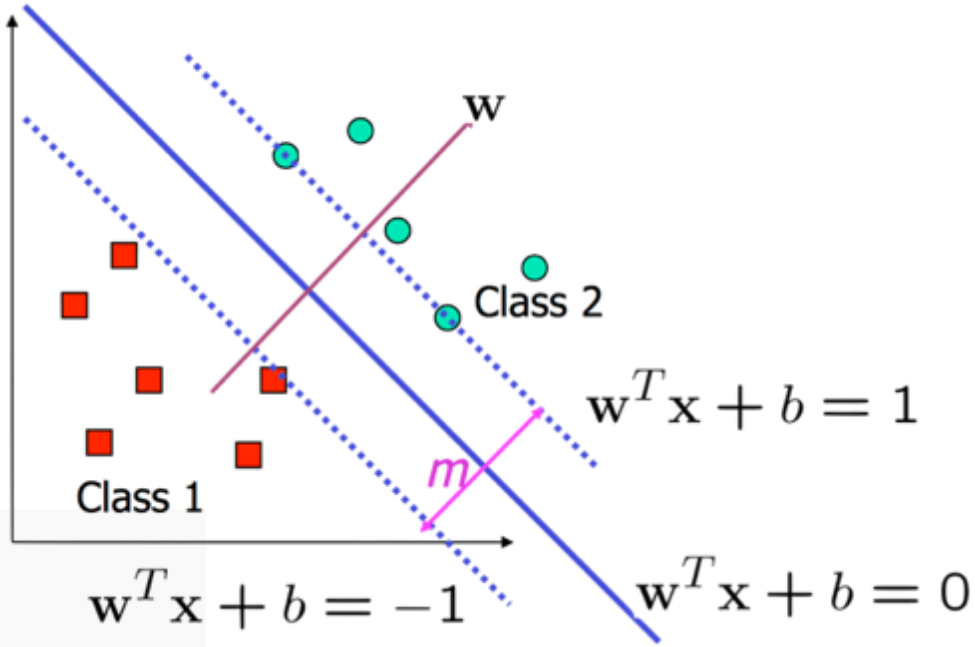
$$D(x) = \frac{w^\top x + b}{\|w\|}$$

(Euclidean Distance Formula)

Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

- Classifying Linear Separable Data



$$\begin{cases} \max\limits_{w,b} M \\ \text{such that } |D(x_n)| = \dfrac{y_i(w^\top x_n + b)}{\|w\|} \geq M,\ n = 1, \ldots, N \end{cases}$$

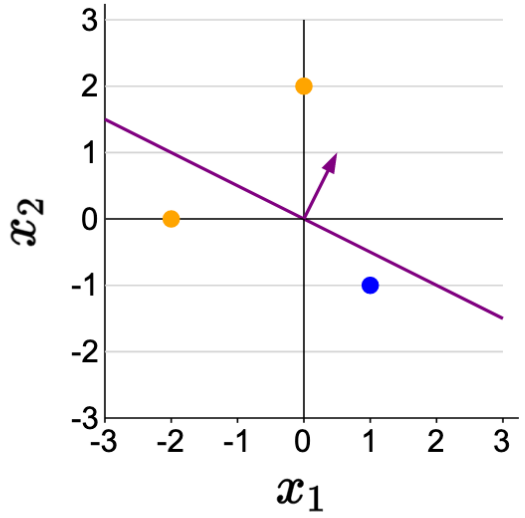
Week 7 (From week 3 Linear Model) Support Vector Machine (SVM)

University of Salford MANCHESTER

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$
$$\mathbf{w} = [0.5, 1]$$
$$\phi(x) = [x_1, x_2]$$

training data $\mathcal{D}_{\text{train}}$

x_1	x_2	y
0	2	1
-2	0	1
1	-1	-1



$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{1 - (\mathbf{w} \cdot \phi(x))y, 0\}$$

$$\text{Loss}([0,2], 1, [0.5,1]) = \max\{1 - [0.5,1] \cdot [0,2](1), 0\} = 0$$
$$\text{Loss}([-2,0], 1, [0.5,1]) = \max\{1 - [0.5,1] \cdot [-2,0](1), 0\} = 2$$
$$\text{Loss}([1,-1], -1, [0.5,1]) = \max\{1 - [0.5,1] \cdot [1,-1](-1), 0\} = 0.5$$
$$\text{TrainLoss}([0.5,1]) = 0.83$$

$$\nabla\text{Loss}([0,2], 1, [0.5,1]) = [0,0]$$
$$\nabla\text{Loss}([-2,0], 1, [0.5,1]) = [2,0]$$
$$\nabla\text{Loss}([1,-1], -1, [0.5,1]) = [1,-1]$$
$$\nabla\text{TrainLoss}([0.5,1]) = [1,-0.33]$$

Week 7 (From week 3 Linear Model) Support Vector Machine (SVM)

University of Salford MANCHESTER

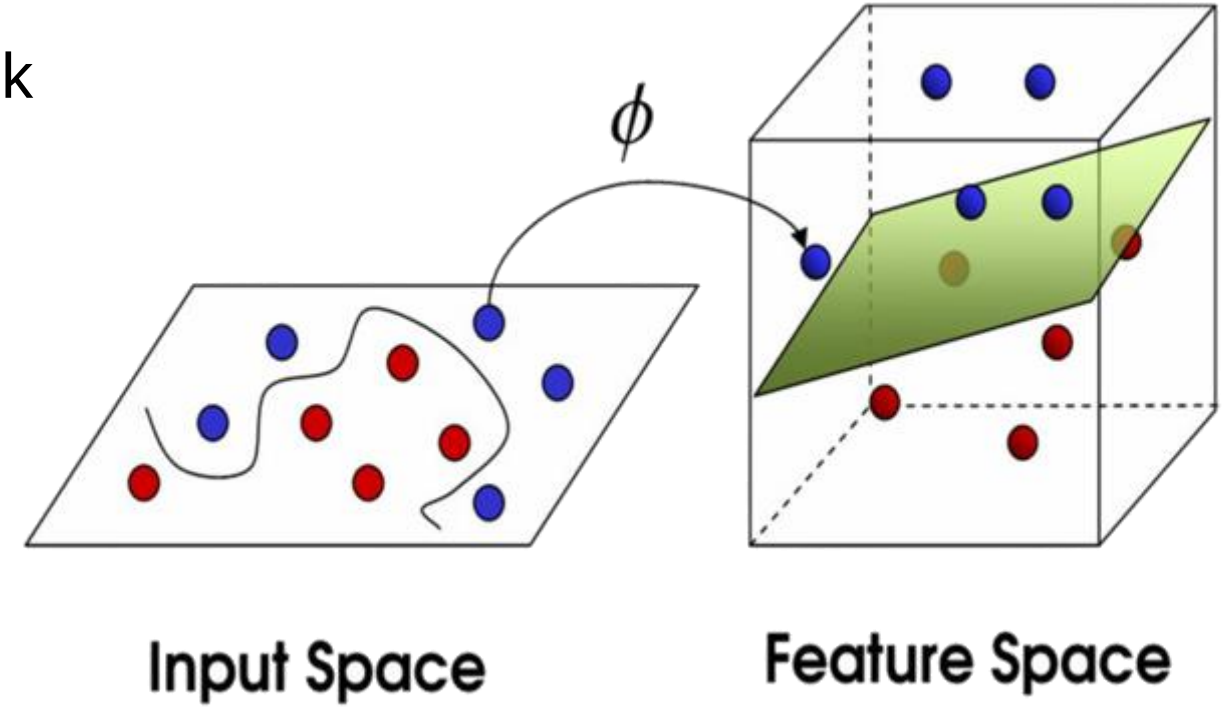
```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(learning_rate=0.01, loss= 'hinge',
max_iter=1000)
sgd.fit(X_train, y_train)
```

```
from sklearn.svm import LinearSVC

sgd = LinearSVC(solver='lbfgs',)
sgd.fit(X_train, y_train)
```

- **loss{*'hinge', 'squared_hinge'*}, *default='squared_hinge'***

Week 7
Support Vector Machine (SVM)

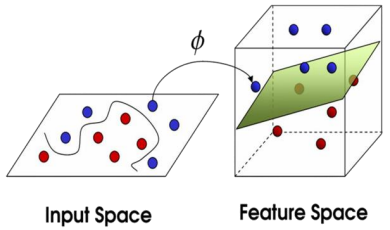
- Classifying Linear Non-Separable Data
  - Kernel Trick



Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

- Classifying Linear Non-Separable Data / Kernel Trick



- **The motto:** instead of tweaking the definition of SVC to accommodate non-linear decision boundaries, we map the data into a feature space in which the classes are linearly separable (or nearly separable):
- Apply transform $\phi: \mathbb{R}^J \rightarrow \mathbb{R}^{J'}$ on training data

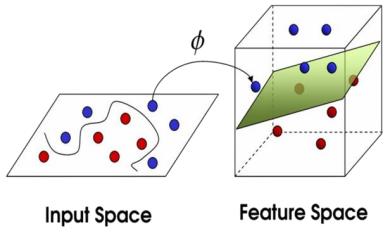
$$x_n \rightarrow \phi(x_n)$$

  where typically J' is much larger than J.
- Train an SVC on the transformed data
- $$\{(\phi(x_1), y_1), (\phi(x_2), y_2), \ldots, (\phi(x_N), y_N)\}$$

Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

- Classifying Linear Non-Separable Data / Kernel Trick



Definition

Given a transformation $\phi : \mathbb{R}^J \to \mathbb{R}^{J'}$, from input space $\mathbb{R}^J$ to feature space $\mathbb{R}^{J'}$, the function $K : \mathbb{R}^J \times \mathbb{R}^J \to \mathbb{R}$ defined by

$$K(x_n, x_m) = \phi(x_n)^\top \phi(x_m), \quad x_n, x_m \in \mathbb{R}^J$$

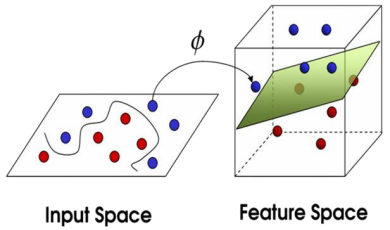
is called the ***kernel function*** of ϕ.

Generally, ***kernel function*** may refer to any function $K : \mathbb{R}^J \times \mathbb{R}^J \to \mathbb{R}$ that measure the similarity of vectors in $\mathbb{R}^J$, without explicitly defining a transform ϕ.

Week 7
Support Vector Machine (SVM)

University of Salford MANCHESTER

- Classifying Linear Non-Separable Data / Kernel Trick



For a choice of kernel *K*,

$$K(x_n, x_m) = \phi(x_n)^\top \phi(x_m)$$

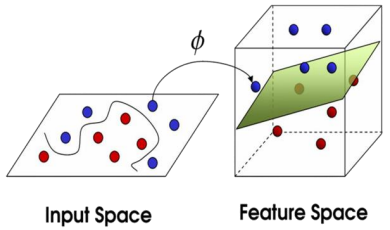
we train an SVC by solving

$$\max_{\alpha_n \geq 0,\ \sum_n \alpha_n y_n = 0} \sum_n \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} y_n y_m \alpha_n \alpha_m K(x_n, x_m)$$

Computing $K(x_n, x_m)$ can be done without computing the mappings $\phi(x_n)$, $\phi(x_m)$.

This way of training a SVC in feature space without explicitly working with the mapping ϕ is called ***the kernel trick***.

Week 7
Support Vector Machine (SVM)



Common kernel functions include:

- **Polynomial Kernel** (kernel='poly')

$$K(x_1, x_2) = (x_1^\top x_2 + 1)^d$$

where *d* is a hyperparameter.

- **Radial Basis Function Kernel** (kernel='rbf')

$$K(x_1, x_2) = \exp\left\{-\frac{\|x_1 - x_2\|^2}{2\sigma^2}\right\}$$

where σ is a hyperparameter.

- **Sigmoid Kernel** (kernel='sigmoid')

$$K(x_1, x_2) = \tanh(\kappa x_1^\top x_2 + \theta)$$

where κ and θ are hyperparameters.

Week 7
Logistic Regression

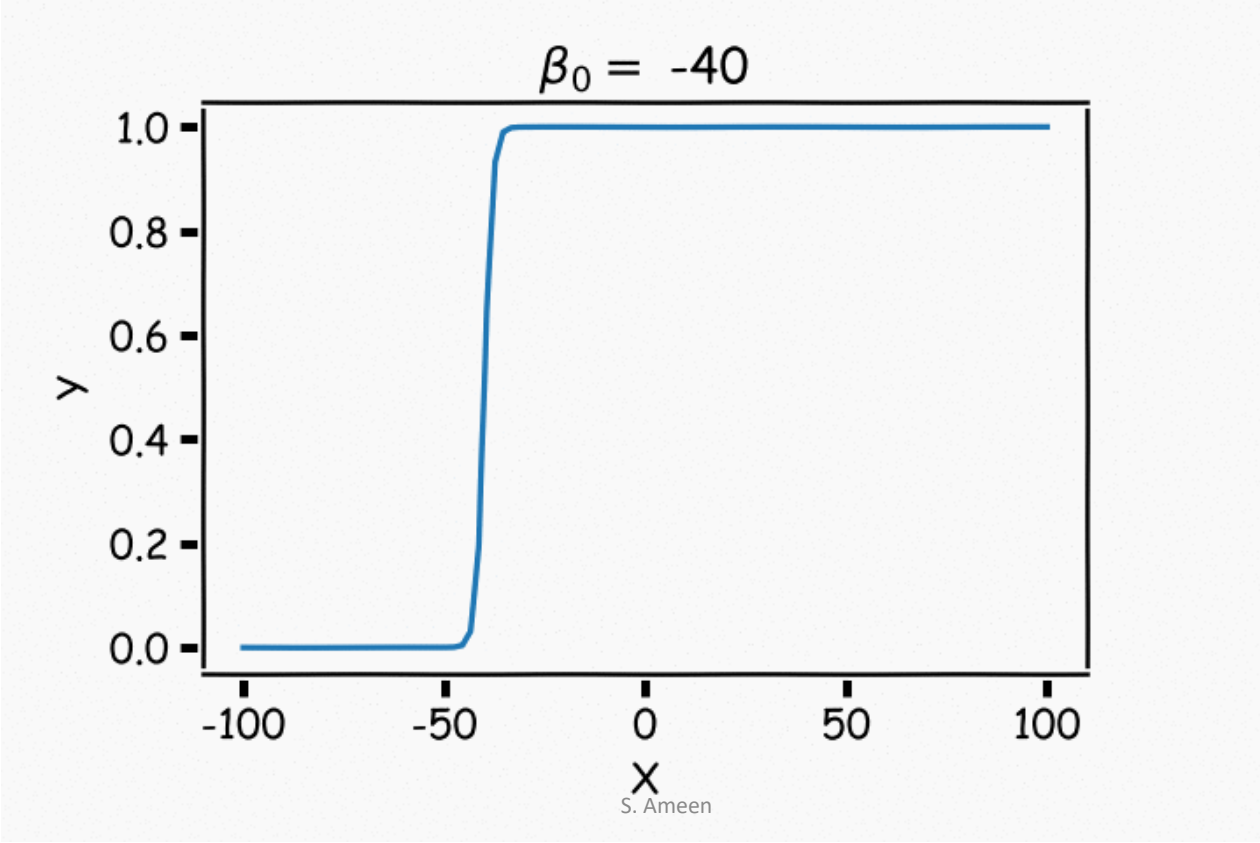
University of Salford MANCHESTER

- Logistic Regression addresses the problem of estimating a probability, $P(y = 1)$, to be outside the range of $[0,1]$. The logistic regression model uses a function, called the ***logistic*** function, to model $P(y = 1)$:

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

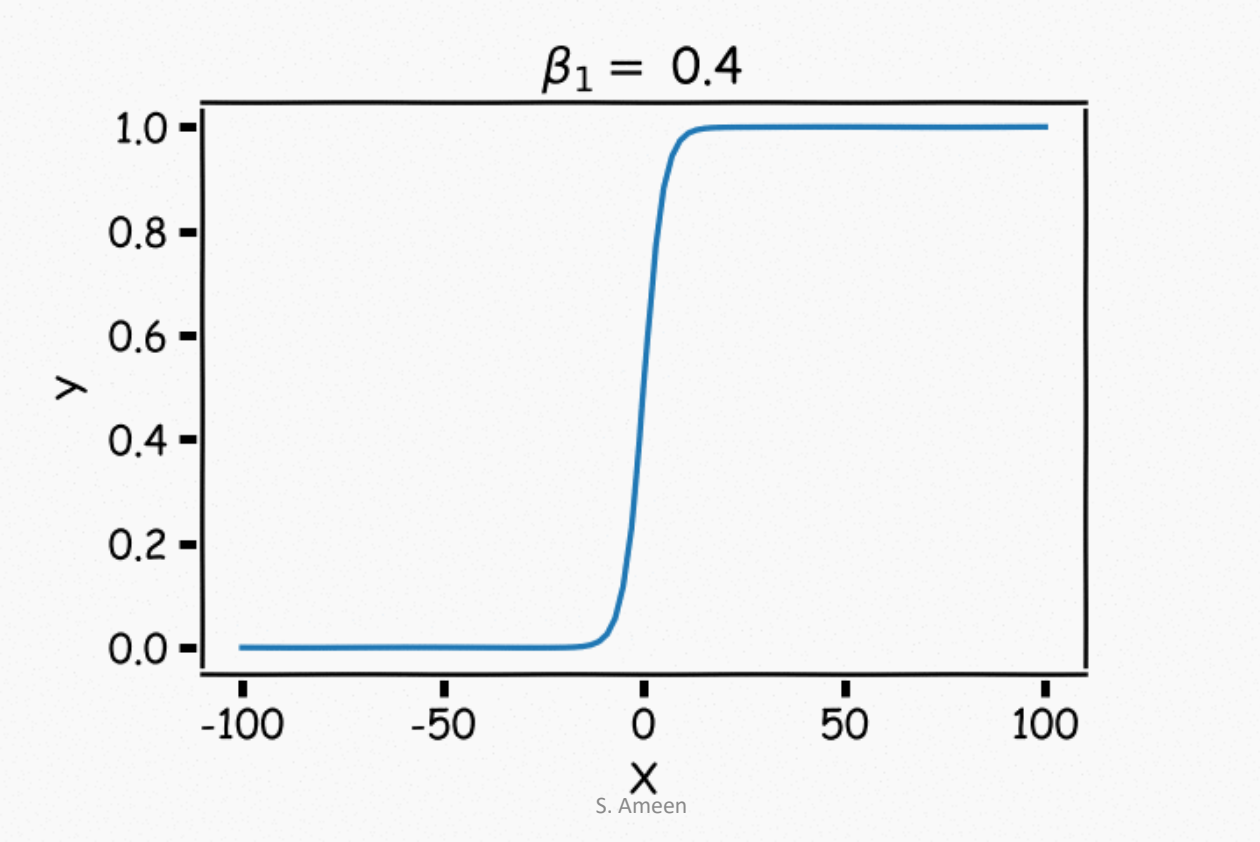
Week 7
Logistic Regression

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$



Week 7
Logistic Regression

$$P(Y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$



Week 7
Logistic Regression

University of Salford MANCHESTER

- With a little bit of algebraic work, the logistic model can be rewritten as:

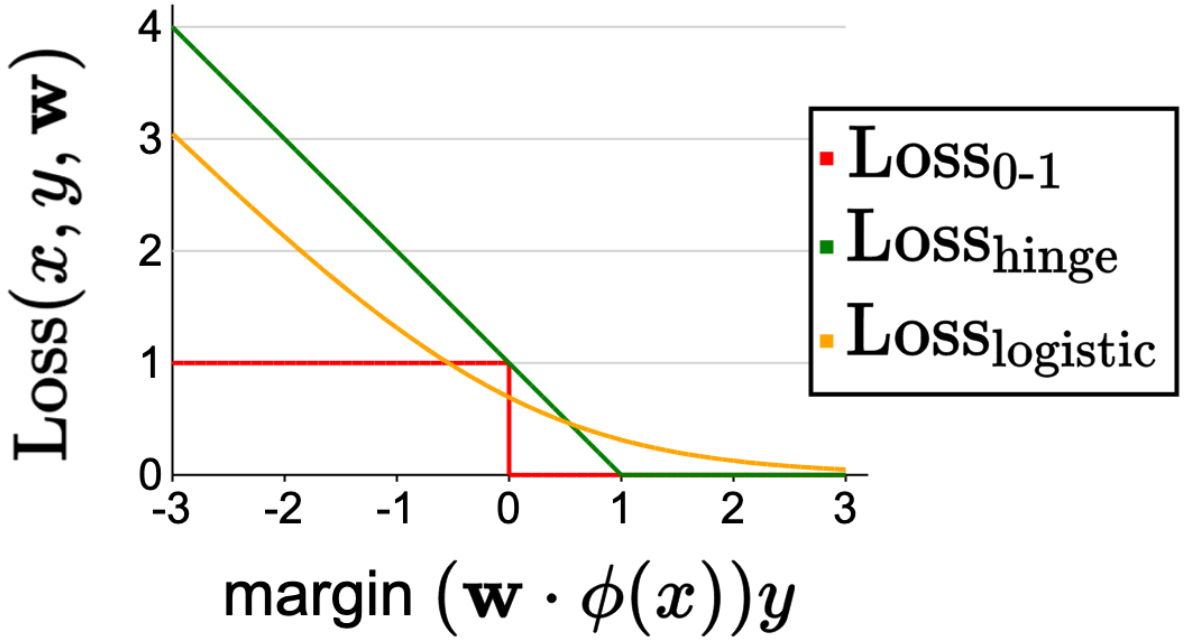
$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = \beta_0 + \beta_1 X.$$

- The value inside the natural log function $\frac{P(Y=1)}{1-P(Y=1)}$, is called the ***odds***, thus logistic regression is said to model the ***log-odds*** with a linear function of the predictors or features, X. This gives us the natural interpretation of the estimates similar to linear regression: a one unit change in X is associated with a β_1 change in the log-odds of $Y = 1$; or better yet, a one unit change in X is associated with an e^{β_1} change in the odds that $Y = 1$.

Week 7 (From week 3 Linear Model) Logistic Regression

University of Salford MANCHESTER

$$\text{Loss}_{\text{logistic}}(x, y, \mathbf{w}) = \log(1 + e^{-(\mathbf{w}\cdot\phi(x))y})$$



Intuition: Try to increase margin even when it already exceeds 1

Week 7 (From week 3 Linear Model) Logistic Regression

```
from sklearn.linear_model import SGDClassifier

sgd = SGDClassifier(learning_rate=0.01, loss= 'log',
max_iter=1000)
sgd.fit(X_train, y_train)
```

```
from sklearn.linear_model import LogisticRegression

sgd = LogisticRegression(solver='lbfgs',)
sgd.fit(X_train, y_train)
```

- 'newton-cg' - ['l2', 'none']
- 'lbfgs' - ['l2', 'none']
- 'liblinear' - ['l1', 'l2']
- 'sag' - ['l2', 'none']
- 'saga' - ['elasticnet', 'l1', 'l2', 'none']

Week 7
Naive Bayes classifier

University of Salford MANCHESTER

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	NO
sunny	hot	high	true	NO
overcast	hot	high	false	YES
rainy	mild	high	false	YES
rainy	cool	normal	false	YES
rainy	cool	normal	true	NO
overcast	cool	normal	true	YES
sunny	mild	high	false	NO
sunny	cool	normal	false	YES
rainy	mild	normal	false	YES
sunny	mild	normal	true	YES
overcast	mild	high	true	YES
overcast	hot	normal	false	YES
rainy	mild	high	true	NO

Week 7
Naive Bayes classifier

University of Salford MANCHESTER

Outlook			Temperature			Humidity			Windy			Play	
	yes	no		yes	no		yes	no		yes	no	yes	no
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overca-st	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cold	3	1								

Week 7
Naive Bayes classifier

University of Salford MANCHESTER

Outlook			Temperature			Humidity			Windy			Play	
	yes	no		yes	no		yes	no		yes	no	yes	no
Sunny	$\frac{2}{9}$	$\frac{3}{5}$	Hot	$\frac{2}{9}$	$\frac{2}{5}$	High	$\frac{3}{9}$	$\frac{4}{5}$	False	$\frac{6}{9}$	$\frac{2}{5}$	$\frac{9}{14}$	$\frac{5}{14}$
Overca-st	$\frac{4}{9}$	0	Mild	$\frac{4}{9}$	$\frac{2}{5}$	Normal	$\frac{6}{9}$	$\frac{1}{5}$	True	$\frac{3}{9}$	$\frac{3}{5}$		
Rainy	$\frac{3}{9}$	$\frac{2}{5}$	Cold	$\frac{3}{9}$	$\frac{1}{5}$								

Week 7
Naive Bayes classifier

Outlook			Temperature			Humidity			Windy			Play	
	yes	no		yes	no		yes	no		yes	no	yes	no
Sunny	$\frac{2}{9}$	$\frac{3}{5}$	Hot	$\frac{2}{9}$	$\frac{2}{5}$	High	$\frac{3}{9}$	$\frac{4}{5}$	False	$\frac{6}{9}$	$\frac{2}{5}$	$\frac{9}{14}$	$\frac{5}{14}$
Overca-st	$\frac{4}{9}$	0	Mild	$\frac{4}{9}$	$\frac{2}{5}$	Normal	$\frac{6}{9}$	$\frac{1}{5}$	True	$\frac{3}{9}$	$\frac{3}{5}$		
Rainy	$\frac{3}{9}$	$\frac{2}{5}$	Cold	$\frac{3}{9}$	$\frac{1}{5}$								

QUESTION

What is the probability that the outlook is sunny, the **temperature** is considered **cold**, the **humidity** is **high** and that it is **windy** and play is either '***yes***' or '***no***'?

Week 7
Naive Bayes classifier

University of Salford MANCHESTER

We get for a sunny outlook, a cool temperature, high humidity, windy weather and play equal to yes:

- $\frac{2}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} \approx 0.0053.$

We get for no sunny outlook, no cool temperature, no high humidity, no windy weather and play equal to no:

- $\frac{3}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} \approx 0.0206.$

So, transforming this to probabilities by normalizing the above values, we get

- $P('yes') = \frac{0.0053}{0.0053 + 0.0206} = 0.205 = 20.5\ \%.$
- $P('no') = \frac{0.0206}{0.0053 + 0.0206} = 0.795 = 79.5\ \%.$

Week 7

University of Salford MANCHESTER

Summary

1. Ensemble Methods
2. SVM
3. Logistic Regression
4. Naïve Bayes

Week 7 – Next Lecture
Deep Learning