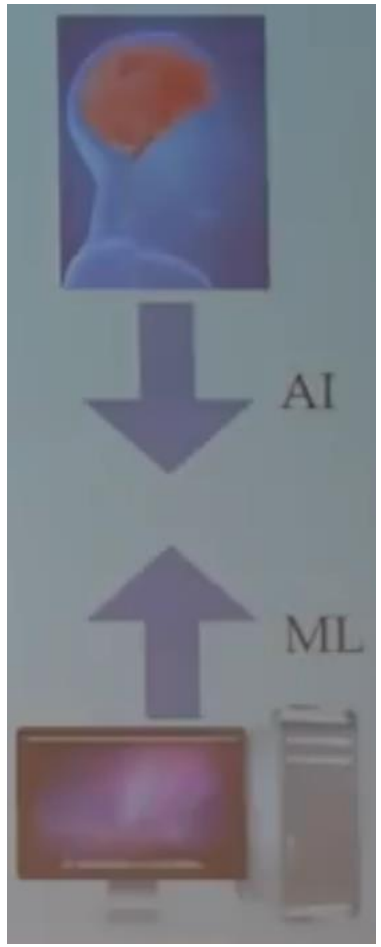
University of Salford MANCHESTER

Artificial Intelligent – Week 5

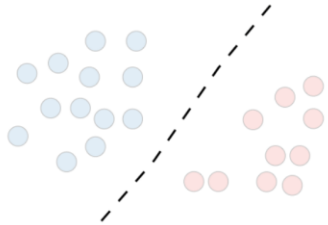
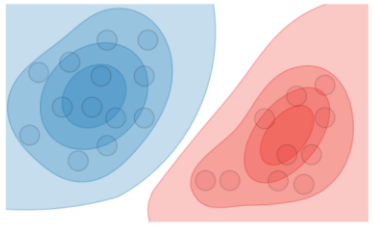
Dr Salem Ameen

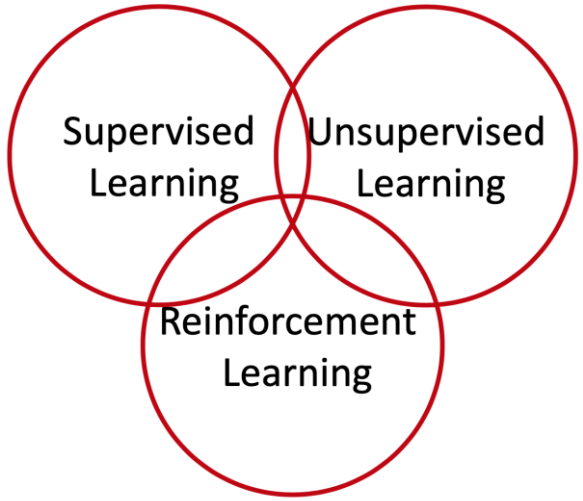
S.A.AMEEN1@SALFORD.AC.UK

Week 5: Artificial Intelligent – Machine Learning



□ **Type of model** — The different models are summed up in the table below:

	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes



Week 5: Artificial Intelligent – Machine Learning

University of Salford MANCHESTER

Given a set of data points $\{x^{(1)}, ..., x^{(m)}\}$ associated to a set of outcomes $\{y^{(1)}, ..., y^{(m)}\}$, we want to build a classifier that learns how to predict y from x.

□ **Type of prediction** — The different types of predictive models are summed up in the table below:

	Regression	Classification
Outcome	Continuous	Class
Examples	Linear regression	Logistic regression, SVM, Naive Bayes

Week 5 - Revision

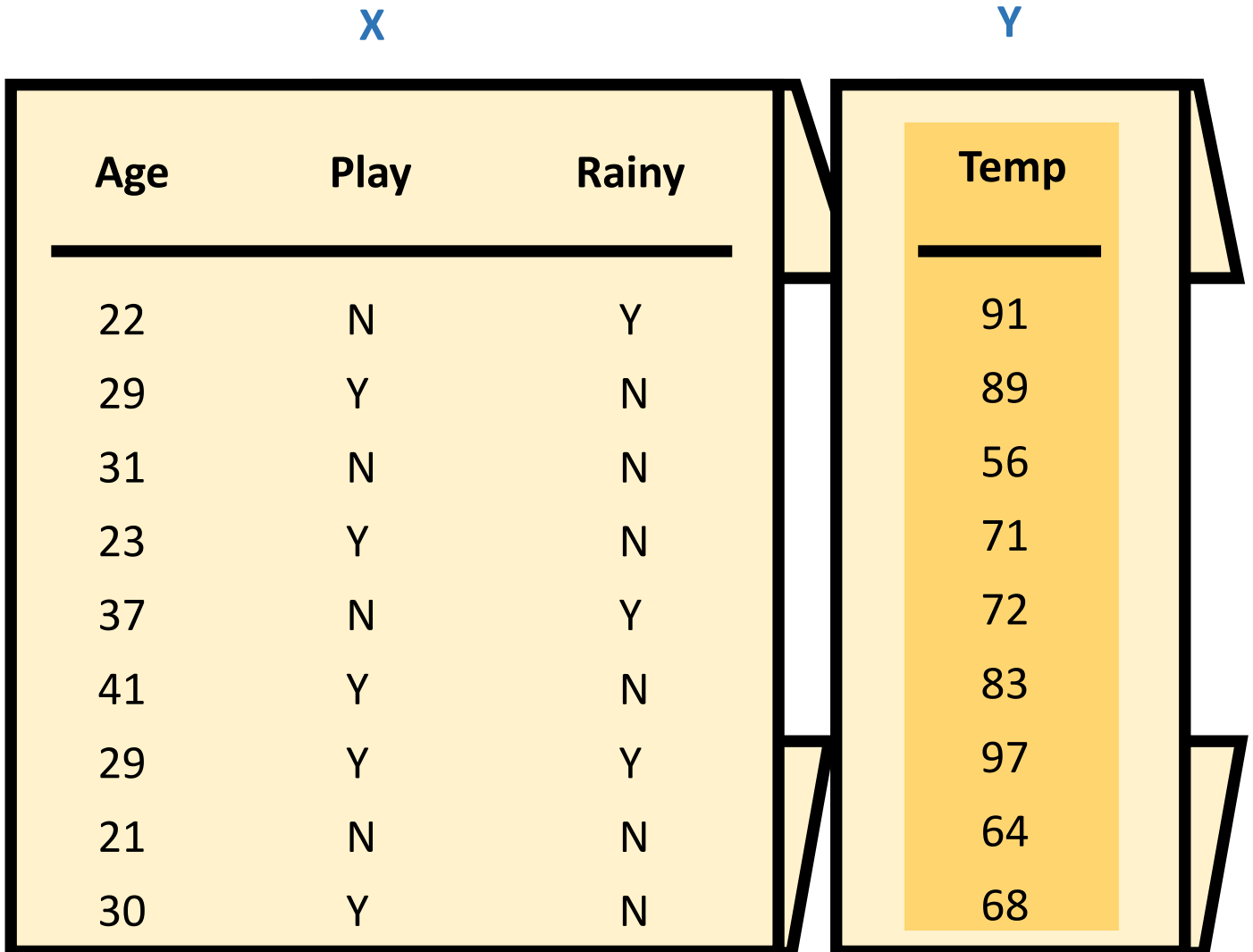
- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)

Your Data X

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Week 5 - Revision

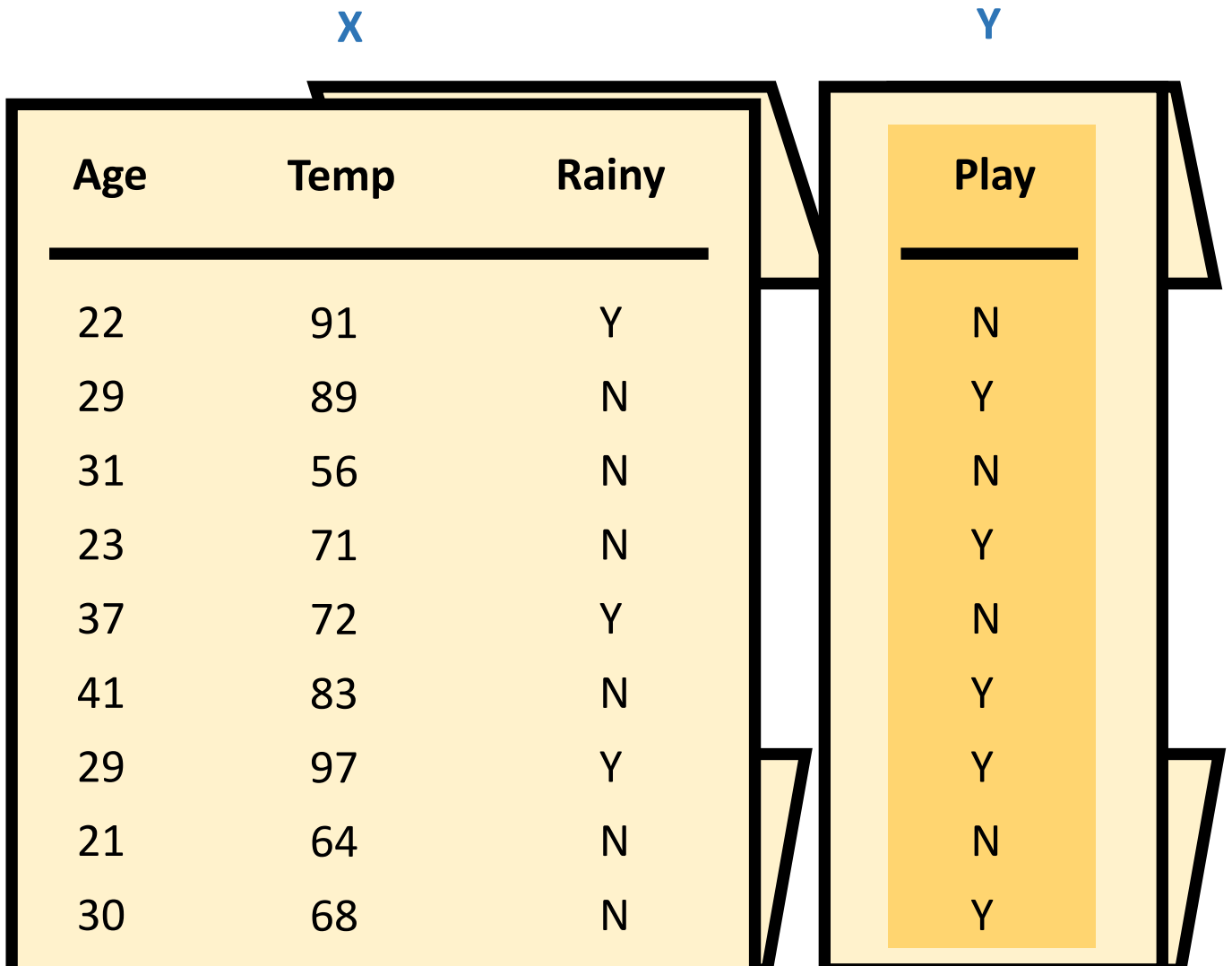
- Let's say this this is our data
- Want a model that is:
  - Supervised
  - Predicts real numbers (regression model)
- **Q**: What model could we use?



X			Y
Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Week 5 - Revision

- Returning to our data, let's again predict if a person will **Play**
- If we do not want to assume anything about how **X** and **Y** relate, we could use a different **supervised** model
- Suppose we do not care to build a decision boundary but merely want to make predictions based on similar data that we saw during training



Week 5 - Revision

University of Salford MANCHESTER

	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**	**Generative** vs **Discriminative**
Linear Regression				
Logistic Regression				

Week 5 - Revision

University of Salford MANCHESTER

	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**	**Generative** vs **Discriminative**
Linear Regression	**Supervised**	**Regression**	**Parametric**	**Discriminative**
Logistic Regression				

Week 5 - Revision

University of Salford MANCHESTER

	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**	**Generative** vs **Discriminative**
Linear Regression	**Supervised**	**Regression**	**Parametric**	**Discriminative**
Logistic Regression	**Supervised**	**Classification**	**Parametric**	**Discriminative**

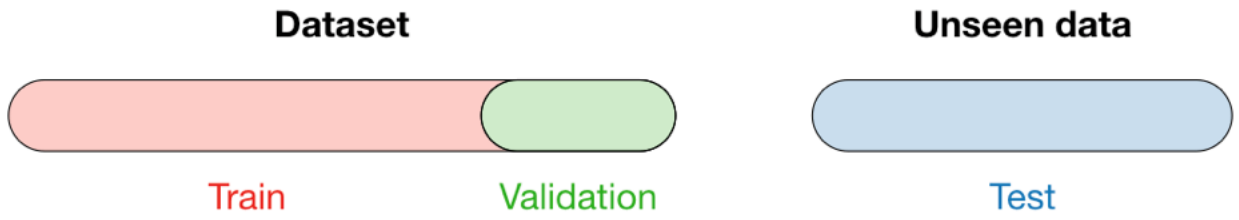
Week 5 – Regression
Last Lecture

University of Salford MANCHESTER

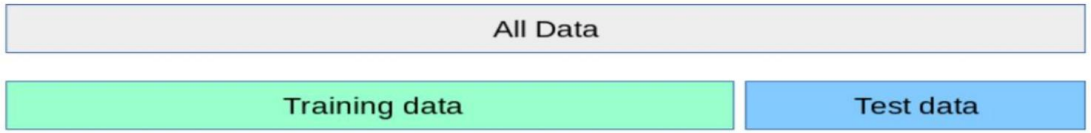
❑Dataset:

Training set	Validation set	Testing set
• Model is trained • Usually 80% of the dataset	• Model is assessed • Usually 20% of the dataset • Also called hold-out or development set	• Model gives predictions • Unseen data

Once the model has been chosen, it is trained on the entire dataset and tested on the unseen test set. These are represented in the figure below:



Week 5: Artificial Intelligent – Machine Learning

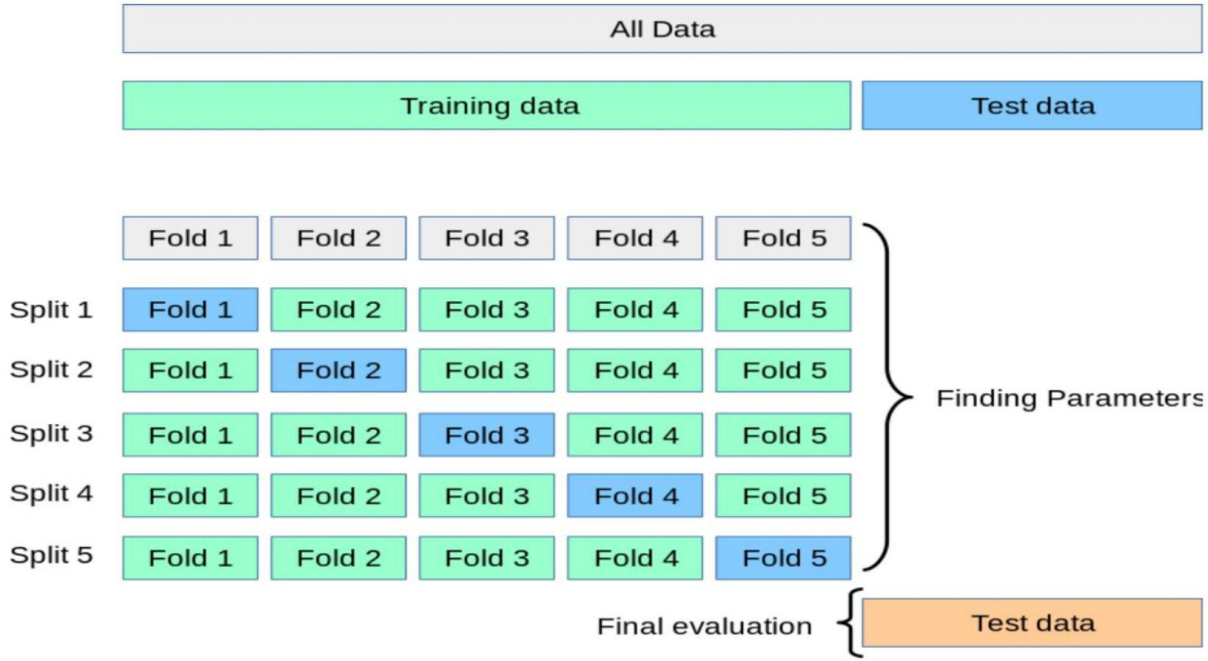



```
from sklearn.model_selection
import train_test_split

X_train, X_test, y_train, y_test =
train_test_split( X, y,
test_size=0.4, random_state=0)
```

Week 5: Artificial Intelligent – Machine Learning

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y,
test_size=0.4, random_state=0)
```

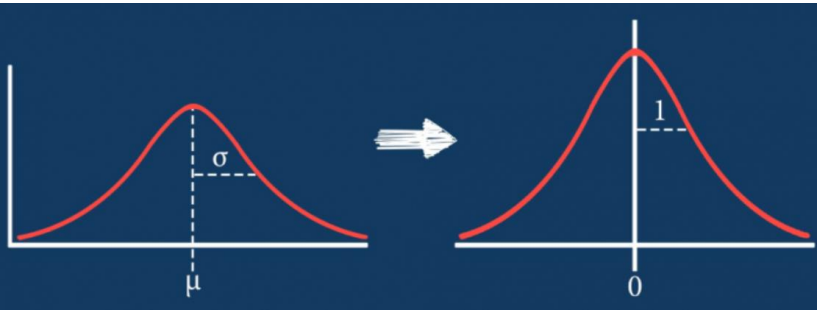
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(Reg, X, y, cv=5)
```



University of Salford MANCHESTER

Feature Scaling

Week 5
Feature Standardization



(22-30)/5

(24-30)/5

31-30

(80-80)/10

70

77

(41-30)/5

Age_S

(22-30)/5 = 1.3

(29 – 30)/5

31

23

37

41

29

21

30

	Temp_S	Rainy
22	(91 – 80)/10	Y 1
29	89	N 0
31	56	N 0
23	71	N 1
37	72	Y 1
41	83	N
29	97	Y
21	64	N
30	68	N

RED 0

RED 0

GREEN 1

B 2

B 1

G 1

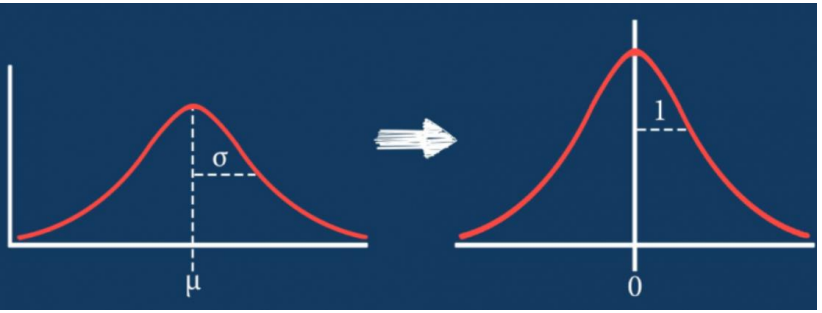
Y 3

Y 3

R 0

W 4

Week 5
Feature Standardization



(22-30)/5

(24-30)/5

31-30

(41-30)/5

(80-80)/10

70

77

Age_S

(22-30)/5 = 1.3

(29 – 30)/5

31

23

37

41

29

21

30

	Temp_S	Rainy_N	Rainy_Y
22	(91 – 80)/10	0	1
29	89	1	0
31	56	1	0
23	71	1	0
37	72	0	1
41	83	N	N
29	97	Y	Y
21	64	N	N
30	68	N	N

Week 5
Feature Scaling

(22 -21)/(41-21)

29

31

23

37

(41-21)/(41-21)

29

(21-21)/(41-21)

30

Age	Temp	Rainy
22	91	Y
29	89	N
31	56	N
23	71	N
37	72	Y
41	83	N
29	97	Y
21	64	N
30	68	N

Week 5 : Preprocessing

University of Salford MANCHESTER

```
# Your should use preprossing before SGD
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import minmax_scale
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import PowerTransformer
```

Week 5 : Preprocessing


```
# Your should use preprossing before SGD
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import minmax_scale
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import PowerTransformer
```

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train_S = scaler.transform(X_train)
X_test_S = scaler.transform(X_test)
```

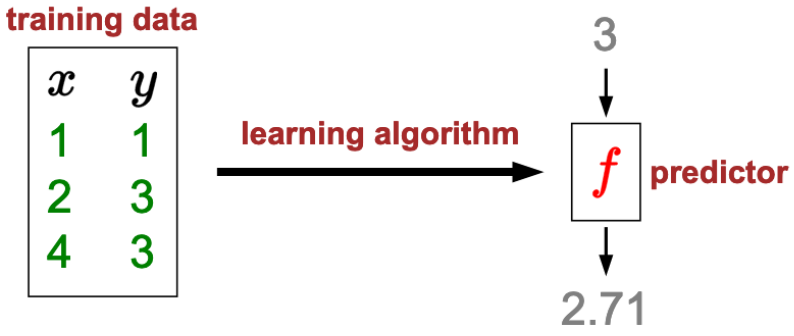
Week 5 – Regression
Last Lecture

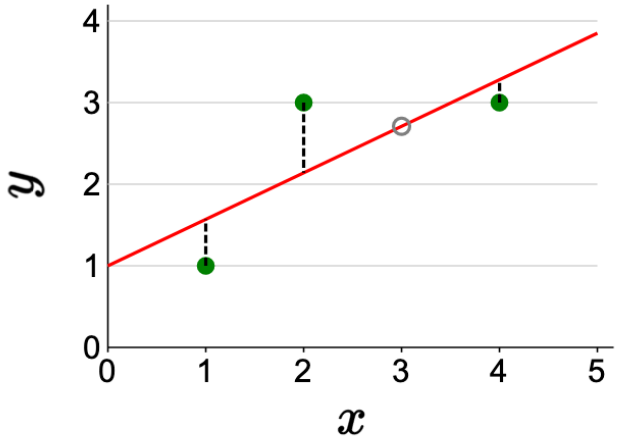
University of Salford MANCHESTER

```python
from sklearn.linear_model import SGDRegressor

scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)


odel = SGDRegressor(penalty="l2", max_iter=1000)
odeleregr = odel.fit(X_train,y_train)
print("r2_score on training dataset",r2_score(y_train, odeleregr.predict(X_train)))
print("\nr2_score on validation dataset",r2_score(y_test, odeleregr.predict(X_test)))
```

```
r2_score on training dataset 0.5532074729050334

r2_score on validation dataset 0.4928109597879584
```

Week 5 – ML - Linear regression
Last Lecture





Which predictors are possible?
Hypothesis class

Linear functions
$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)\}, \phi(x) = [1, x]$

How good is a predictor?
Loss function

Squared loss
$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$

How to compute best predictor?
Optimization algorithm

Gradient descent
$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \text{TrainLoss}(\mathbf{w})$

Lecture - Week 5 – ML - Linear regression
Optimization algorithm: GD

University of Salford MANCHESTER

$$f(x) = 1 + 0.57x$$

$$\text{Loss}(1, 1, [1, 0.57]) = ([1, 0.57] \cdot [1, 1] - 1)^2 = 0.32$$
$$\text{Loss}(2, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 2] - 3)^2 = 0.74$$
$$\text{Loss}(4, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 4] - 3)^2 = 0.08$$

$$\text{TrainLoss}([1, 0.57]) = 0.38$$

Algorithm: gradient descent

Initialize $\mathbf{w} = [0, \ldots, 0]$

For $t = 1, \ldots, T$: epochs

$$\mathbf{w} \leftarrow \mathbf{w} - \underbrace{\eta}_{\text{step size}} \underbrace{\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})}_{\text{gradient}}$$

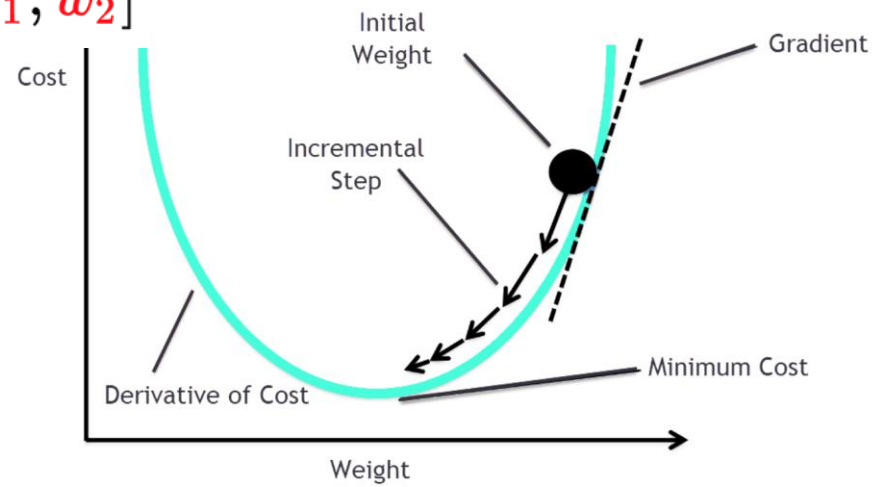
Objective function:

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} (\mathbf{w} \cdot \phi(x) - y)^2$$

Gradient (use chain rule):

$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} 2(\underbrace{\mathbf{w} \cdot \phi(x) - y}_{\text{prediction-target}})\phi(x) = 3.99$$

$[w_1, w_2]$



Lecture - Week 5 – ML - Linear regression
Optimization algorithm: GD

University of Salford MANCHESTER

Demo Code

Regression Metrics

Week 5 – Regression
Last Lecture

- Mean Absolute Error (MAE)
- Root Mean Square Error (RMSE)
- R-Squared

Week 5 : Regression Metrics

University of Salford MANCHESTER

Regression metrics

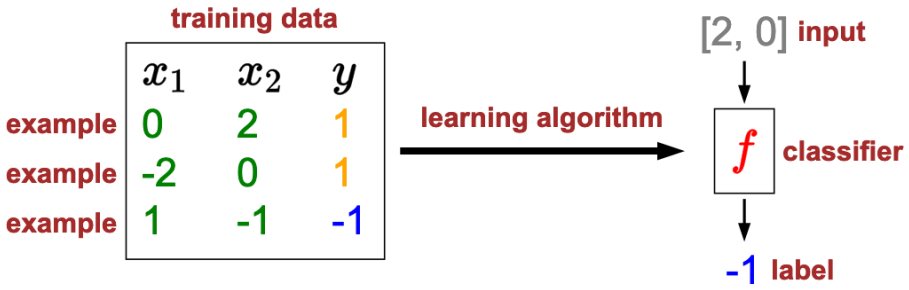
❒ **Basic metrics** — Given a regression model f, the following metrics are commonly used to assess the performance of the model:

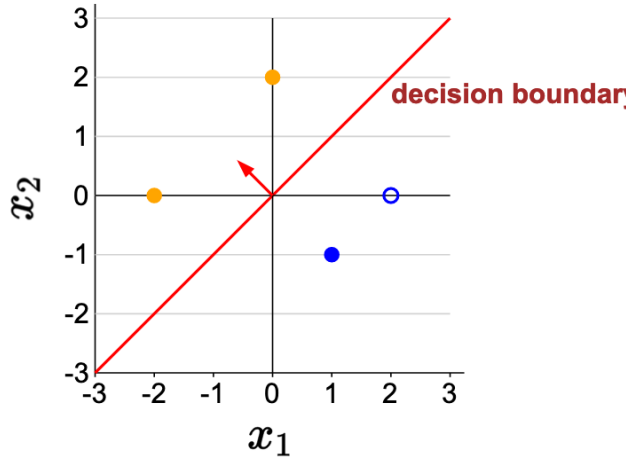
Total sum of squares	Explained sum of squares	Residual sum of squares
$\mathrm{SS}_{\mathrm{tot}} = \sum_{i=1}^{m} (y_i - \overline{y})^2$	$\mathrm{SS}_{\mathrm{reg}} = \sum_{i=1}^{m} (f(x_i) - \overline{y})^2$	$\mathrm{SS}_{\mathrm{res}} = \sum_{i=1}^{m} (y_i - f(x_i))^2$

❒ **Coefficient of determination** — The coefficient of determination, often noted R^2 or r^2, provides a measure of how well the observed outcomes are replicated by the model and is defined as follows:

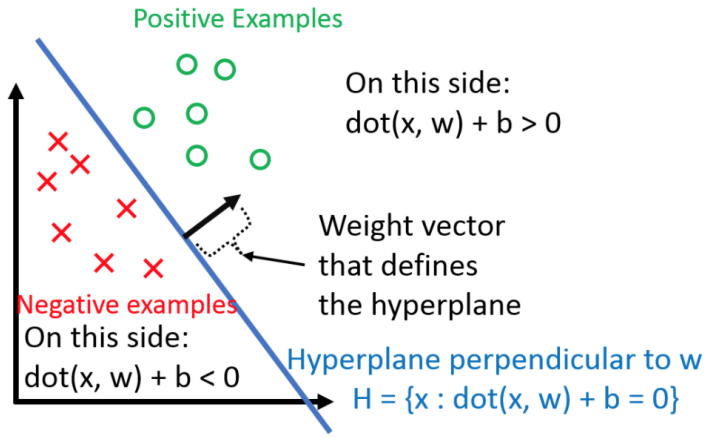
$$R^2 = 1 - \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{SS}_{\mathrm{tot}}}$$

Week 5: Artificial Intelligent – Classification



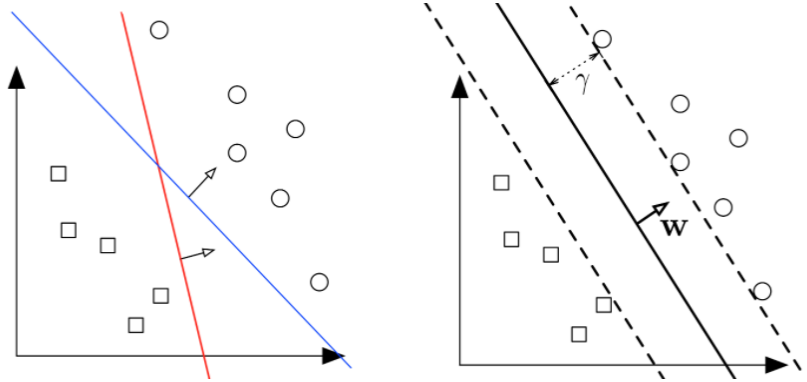


$$h(x_i) = \text{sign}(\mathbf{w}^\top \mathbf{x}_i + b)$$



Design decisions:

- Which classifiers are possible? **hypothesis class**
- How good is a classifier? **loss function**
- How do we compute the best classifier? **optimization algorithm**



Classification Metrics

Week 5 : Classification metrics

❒ **Confusion matrix** — The confusion matrix is used to have a more complete picture when assessing the performance of a model. It is defined as follows:

		Predicted class: +	**Predicted** class: -
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

Week 5 : Classification metrics

❒ **Confusion matrix** — The confusion matrix is used to have a more complete picture when assessing the performance of a model. It is defined as follows:

		Predicted class: +	**Predicted** class: -
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_true, y_pred)
```

```
[[20 32]
 [19 26]]
```

Week 5 : Classification metrics

❒ **Main metrics** — The following metrics are commonly used to assess the performance of classification models:

Metric	Formula	Interpretation
Accuracy	$\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$	Overall performance of model
Precision	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Coverage of actual positive sample
Specificity	$\frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$	Coverage of actual negative sample
F1 score	$\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$	Hybrid metric useful for unbalanced classes

Week 5 : Classification metrics

University of Salford MANCHESTER

```
import metrics


metrics.accuracy_score(y_test, y_pred)

metrics.precision_score(y_test, y_pred)

metrics.recall_score(y_test, y_pred)

metrics.f1_score(y_test, y_pred)
```

Week 5 : Classification metrics

University of Salford MANCHESTER

```
print ("The accuracy on validation dataset of DT: \t", metrics.accuracy_score(y_test, y_pred))
print ("Precision on validation dataset of DT: \t", metrics.precision_score(y_test, y_pred, average= 'weighted'))
print ("Recall on validation dataset of DT : \t", metrics.recall_score(y_test, y_pred, average= 'weighted'))
print ("F1 score on validation dataset of DT: \t", metrics.f1_score(y_test, y_pred, average= 'weighted'))
```

```
The accuracy on validation dataset of DT:        0.5819672131147541
Precision on validation dataset of DT:   0.5819892473118279
Recall on validation dataset of DT :     0.5819672131147541
F1 score on validation dataset of DT:    0.5819391251763757
```

Week 5 : Classification metrics

University of Salford MANCHESTER

```
import metrics

metrics.classification_report(y_test, y_pred)
```

Week 5 : Classification metrics

University of Salford MANCHESTER

import metrics

```
print (metrics.classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.58      0.57      0.58        61
         1.0       0.58      0.59      0.59        61

    accuracy                           0.58       122
   macro avg       0.58      0.58      0.58       122
weighted avg       0.58      0.58      0.58       122
```

Support is the number of actual occurrences of the class in the specified dataset

Week 5 : Classification metrics

Metric	Formula	Interpretation
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$	Overall performance of model
Precision	$\frac{TP}{TP + FP}$	How accurate the positive predictions are
Recall Sensitivity	$\frac{TP}{TP + FN}$	Coverage of actual positive sample
Specificity	$\frac{TN}{TN + FP}$	Coverage of actual negative sample
F1 score	$\frac{2TP}{2TP + FP + FN}$	Hybrid metric useful for unbalanced classes

		Predicted class: +	**Predicted** class: -
Actual class	+	**TP** True Positives	**FN** False Negatives Type II error
	-	**FP** False Positives Type I error	**TN** True Negatives

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Actual		Predicted	
	0		0
1		1	
1		1	
1			0
1		1	
1		1	
	0	1	

Week 5 : Classification metrics

University of Salford MANCHESTER

Actual Class

Predicted Class

True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

F1 score $\quad \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$

Week 5 : Classification metrics

University of Salford MANCHESTER

Actual Class

Predicted Class

True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Precision} = \frac{TP}{TP + FP}$$

Week 5 : Classification metrics

University of Salford MANCHESTER

Actual Class

Predicted Class

True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{1}{1 + 1} = 0.5$$

F1 score	$\frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$

Week 5 : Classification metrics

University of Salford MANCHESTER

Actual Class

Predicted Class

True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Recall} = \frac{TP}{TP + FN}$$

Week 5 : Classification metrics

University of Salford MANCHESTER

Actual Class

Predicted Class

True Positives (TPs): 1	False Positives (FPs): 1
False Negatives (FNs): 8	True Negatives (TNs): 90

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{1}{1 + 8} = 0.11$$

F1 score	$\frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$

Week 5 : Classification metrics

University of Salford MANCHESTER

True Positives (TP): 8	False Positives (FP): 2
False Negatives (FN): 3	True Negatives (TN): 17

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

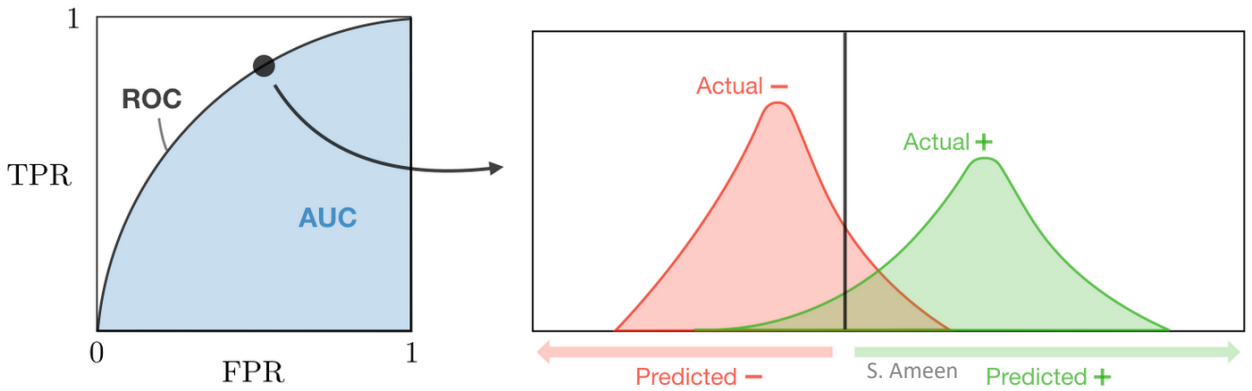
F1 score $\quad \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$

Week 5 : Classification metrics

❒ **ROC** — The receiver operating curve, also noted ROC, is the plot of TPR versus FPR by varying the threshold. These metrics are are summed up in the table below:

Metric	Formula	Equivalent
True Positive Rate TPR	$\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$	Recall, sensitivity
False Positive Rate FPR	$\frac{\mathrm{FP}}{\mathrm{TN}+\mathrm{FP}}$	1-specificity

❒ **AUC** — The area under the receiving operating curve, also noted AUC or AUROC, is the area below the ROC as shown in the following figure:



Week 5 : Classification metrics

University of Salford MANCHESTER

false_positive_rate, true_positive_rate, thresholds

```
1 from sklearn.metrics import accuracy_score, roc_curve, auc
2 roc_curve(y_test, y_pred)
```

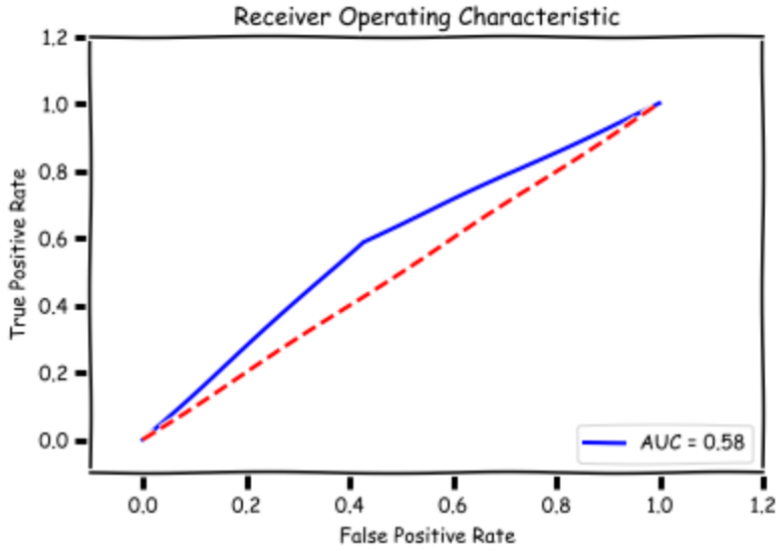
```
(array([0.  , 0.43, 1.  ]), array([0.  , 0.59, 1.  ]), array([2., 1., 0.]))
```

```
1 auc(false_positive_rate, true_positive_rate)
```

```
0.5819672131147541
```

Week 5 : Classification metrics

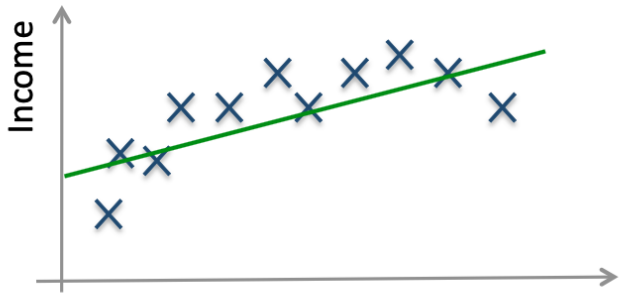
```python
from sklearn.metrics import accuracy_score, roc_curve, auc
false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(false_positive_rate, true_positive_rate)
plt.title('Receiver Operating Characteristic')
plt.plot(false_positive_rate, true_positive_rate, 'b',
label='AUC = %0.2f'% roc_auc)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



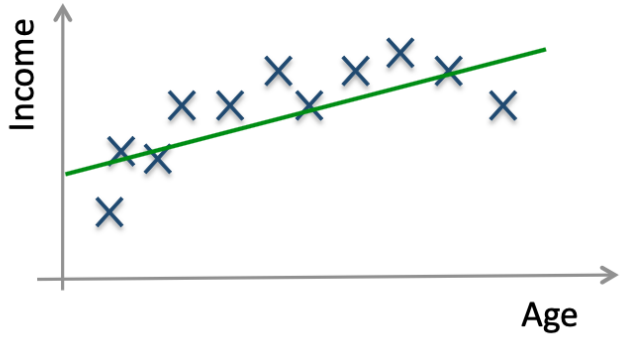
University of Salford MANCHESTER

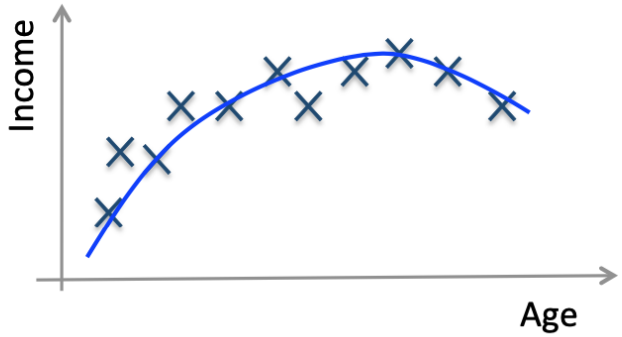
Can we do better!

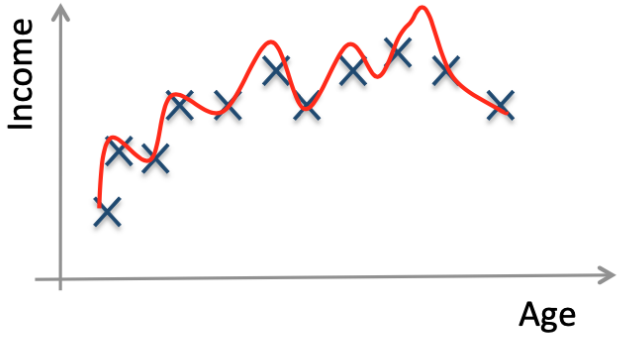
Week 5 – Linear regression
More complex – Features



Week 5 – Linear regression - Polynomial Regression







Week 5 – Linear regression - Polynomial Regression

University of Salford MANCHESTER

- The simplest non-linear model we can consider, for a response *Y* and a predictor *X*, is a polynomial model of degree *M*,

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_M x^M + \epsilon.$$

- Just as in the case of linear regression with cross terms, polynomial regression is a special case of linear regression - we treat each x^m as a separate predictor. Thus, we can write

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_1^M \\ 1 & x_2^1 & \ldots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^M \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

48

Week 5 – Linear regression
Add Features

University of Salford MANCHESTER

Add more complex features

```
]: # Call the models
regr = LinearRegression()
quadratic = PolynomialFeatures(degree=2)
cubic = PolynomialFeatures(degree=3)
```

```
]: # Fit the training dataset
L_quad = quadratic.fit_transform(X_train)
L_cubic = cubic.fit_transform(X_train)
# Fit the validation dataset
L_quadTest = quadratic.fit_transform(X_test)
L_cubicTest = cubic.fit_transform(X_test)
```

Week 5 – Linear regression Features

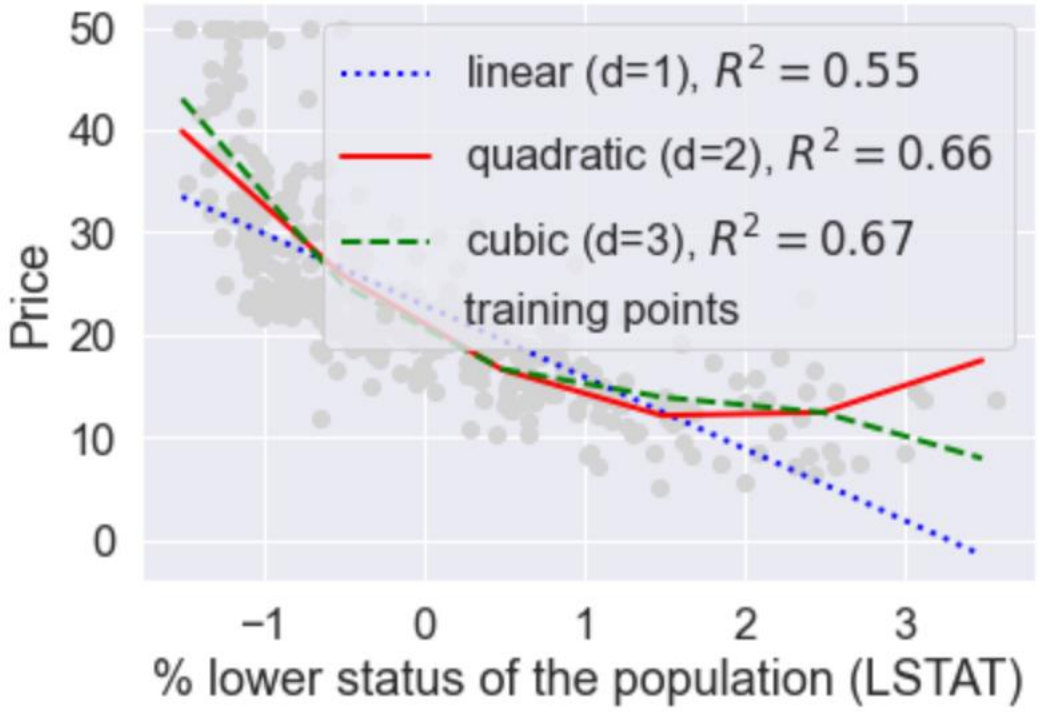
University of Salford MANCHESTER

```
L_fit = np.arange(X_train.min(), X_train.max(), 1)[:, np.newaxis]
L_fitTest = np.arange(X_test.min(), X_test.max(), 1)[:, np.newaxis]

# linear fit
print("Results for Linear features")
regr = regr.fit(X_train,y_train)
y_lin_fit = regr.predict(L_fit)
y_lin_fit_test = regr.predict(L_fitTest)
linear_r2 = r2_score(y_train, regr.predict(X_train))
linear_r2_Test = r2_score(y_test, regr.predict(X_test))
print("r^2 on raining dataset",linear_r2)
print("r^2 on testing dataset",linear_r2_Test)

# quadratic fit
print("\nResults for quadratic features")
regr = regr.fit(L_quad, y_train)
y_quad_fit = regr.predict(quadratic.fit_transform(L_fit))
y_quad_fitTest = regr.predict(quadratic.fit_transform(L_fitTest))

quadratic_r2 = r2_score(y_train, regr.predict(L_quad))
quadratic_r2_Test = r2_score(y_test, regr.predict(L_quadTest))
print("r^2 on raining dataset",quadratic_r2)
print("r^2 on testing dataset",quadratic_r2_Test)
```

```
# cubic fit
print("\nResults for cubic features")
regr = regr.fit(L_cubic, y_train)
y_cubic_fit = regr.predict(cubic.fit_transform(L_fit))
y_cubic_fitTest = regr.predict(cubic.fit_transform(L_fitTest))
cubic_r2 = r2_score(y_train, regr.predict(L_cubic))
cubic_r2_Test = r2_score(y_test, regr.predict(L_cubicTest))
print("r^2 on raining dataset",cubic_r2)
print("r^2 on testing dataset",cubic_r2_Test)
```

```
Results for Linear features
r^2 on raining dataset 0.5532178401407088
r^2 on testing dataset 0.4929340114233606

Results for quadratic features
r^2 on raining dataset 0.6592371636977945
r^2 on testing dataset 0.5555828390062716

Results for cubic features
r^2 on raining dataset 0.6741193529548613
r^2 on testing dataset 0.5835791195458608
```

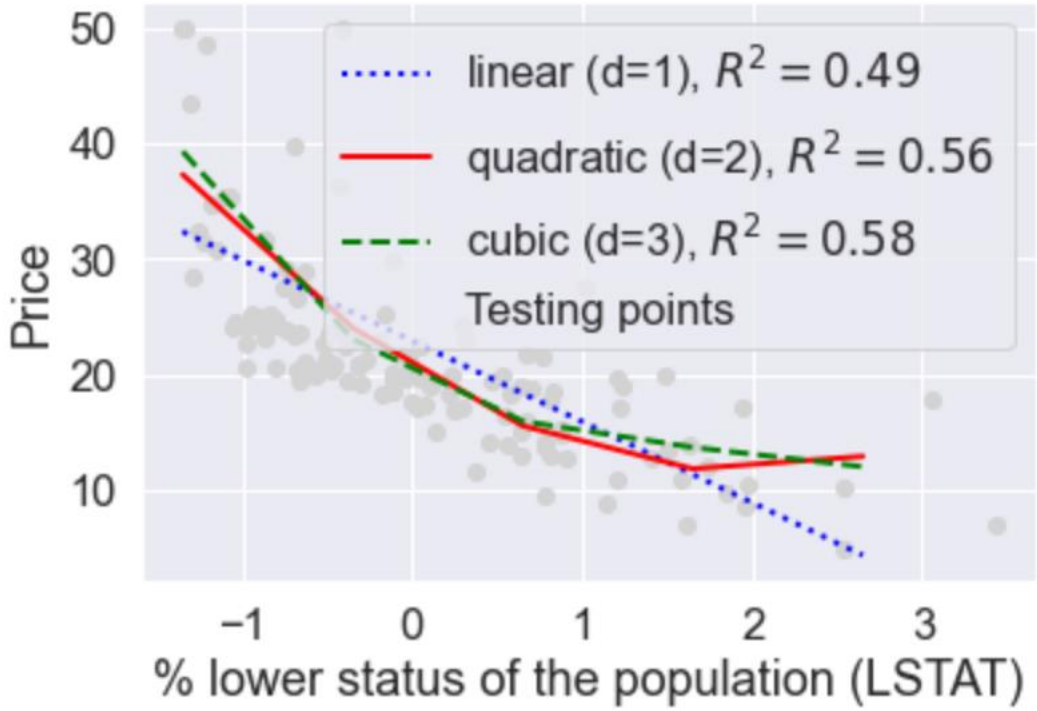
Week 5 – Linear regression Features



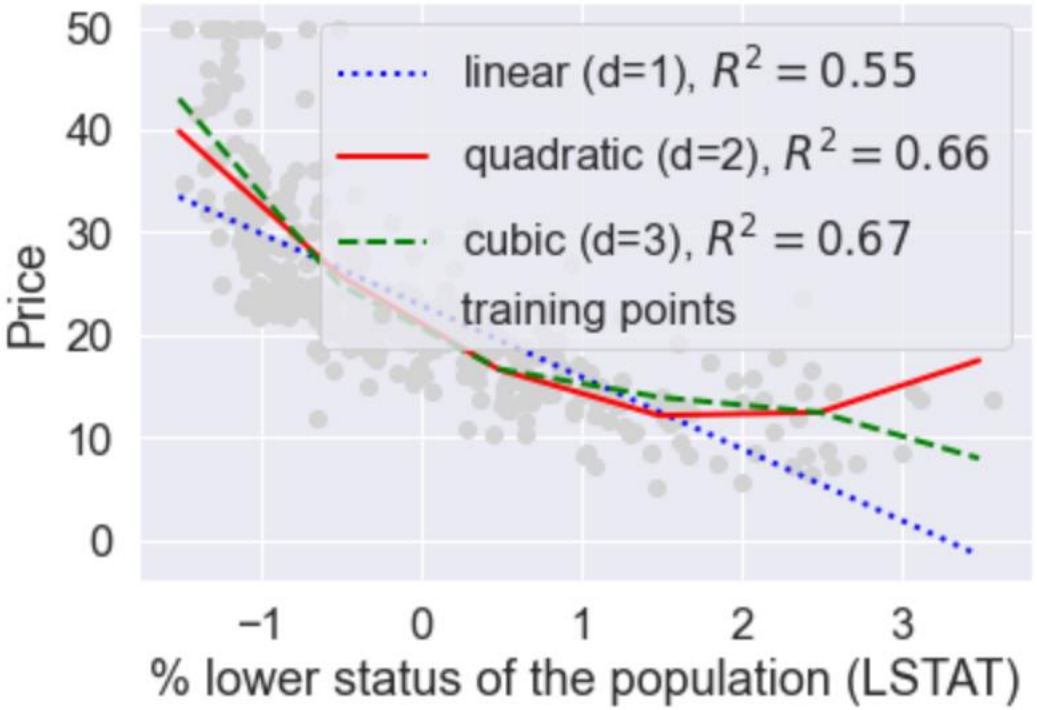

```python
#Plot results on training dataset
plt.scatter(X_train, y_train,
            label='training points',
            color='lightgray')
plt.plot(L_fit, y_lin_fit,
         label='linear (d=1), $R^2=%.2f$'
         % linear_r2,
         color='blue',
         lw=2,
         linestyle=':')
plt.plot(L_fit, y_quad_fit,
         label='quadratic (d=2), $R^2=%.2f$'
         % quadratic_r2,
         color='red',
         lw=2,
         linestyle='-')
plt.plot(L_fit, y_cubic_fit,
         label='cubic (d=3), $R^2=%.2f$'
         % cubic_r2,
         color='green',
         lw=2,
         linestyle='--')
plt.xlabel('% lower status of the population (LSTAT)')
plt.ylabel('Price')
plt.legend(loc='upper right')
plt.show()
```

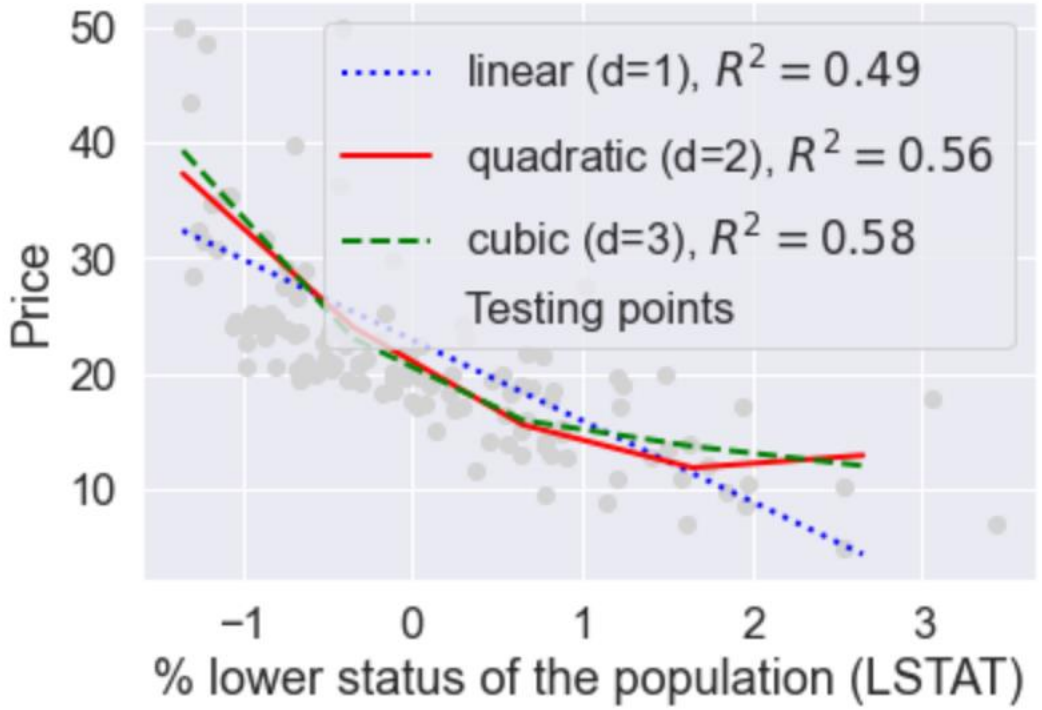
Week 5 – Linear regression Features

```
#Plot results on testing dataset
plt.scatter(X_test, y_test,
            label='Testing points',
            color='lightgray')
plt.plot(L_fitTest, y_lin_fit_test,
         label='linear (d=1), $R^2=%.2f$'
         % linear_r2_Test,
         color='blue',
         lw=2,
         linestyle=':')
plt.plot(L_fitTest, y_quad_fitTest,
         label='quadratic (d=2), $R^2=%.2f$'
         % quadratic_r2_Test,
         color='red',
         lw=2,
         linestyle='-')
plt.plot(L_fitTest, y_cubic_fitTest,
         label='cubic (d=3), $R^2=%.2f$'
         % cubic_r2_Test,
         color='green',
         lw=2,
         linestyle='--')
plt.xlabel('% lower status of the population (LSTAT)')
plt.ylabel('Price')
plt.legend(loc='upper right')
plt.show()
```



Week 5 – Linear regression Features





Week 5 – Linear regression Features

University of Salford MANCHESTER

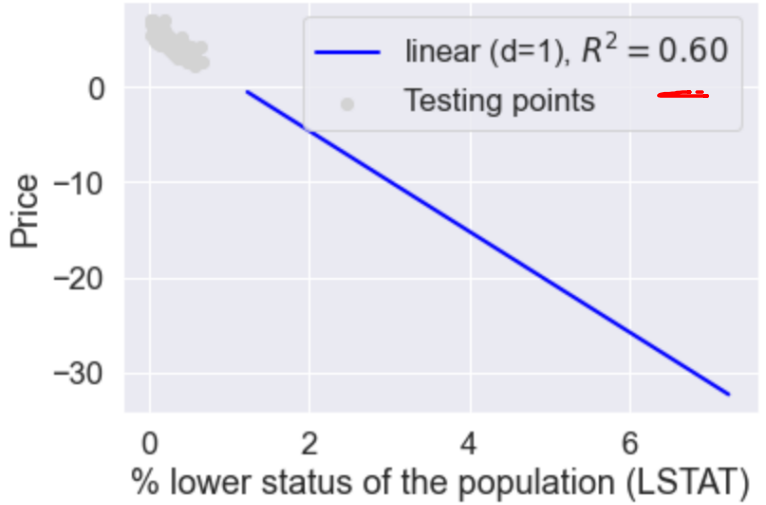
```
X_train, X_test, y_train, y_test = train_test_split(L, y, test_size=0.25, random_state=42)
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Transform features Log
# training
L_log = np.log(X_train+1)
y_sqrt = np.sqrt(y_train)
# testing
L_logTest = np.log(X_test+1)
y_sqrtTest = np.sqrt(y_test)

# Fit features
L_fit = np.arange(L_log.min()-1, L_log.max()+1, 1)[:, np.newaxis]
L_fitTest = np.arange(y_sqrtTest.min()-1, y_sqrtTest.max()+1, 1)[:, np.newaxis]

regr = regr.fit(L_log, y_sqrt)
Logy_lin_fit = regr.predict(L_fit)
Logy_lin_fitTest = regr.predict(L_fitTest)

Loglinear_r2 = r2_score(y_sqrt, regr.predict(L_log))
Loglinear_r2Test = r2_score(y_sqrtTest, regr.predict(L_logTest))
```

Week 5 – Linear regression Features

University of Salford MANCHESTER

```
# Plot
plt.scatter(L_log, y_sqrt,
            label='training points',
            color='lightgray')
plt.plot(L_fit, Logy_lin_fit,
         label='linear (d=1), $R^2=%.2f$'
         % Loglinear_r2,
         color='blue',
         lw=2)

plt.xlabel('% lower status of the population (LSTAT)')
plt.ylabel('Price')
plt.legend(loc='upper right')
plt.show()
```


```
# Plot
plt.scatter(L_logTest, y_sqrtTest,
            label='Testing points',
            color='lightgray')
plt.plot(L_fitTest, Logy_lin_fitTest,
         label='linear (d=1), $R^2=%.2f$'
         % Loglinear_r2Test,
         color='blue',
         lw=2)

plt.xlabel('% lower status of the population (LSTAT)')
plt.ylabel('Price')
plt.legend(loc='upper right')
plt.show()
```

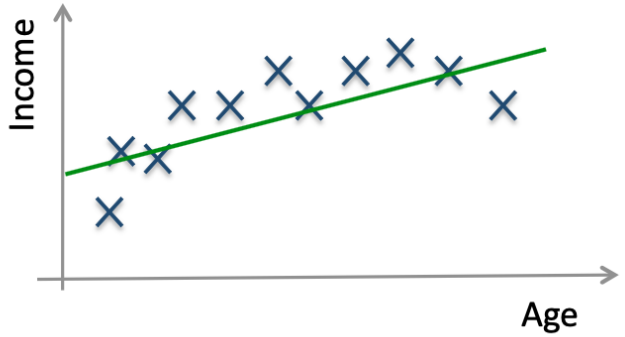


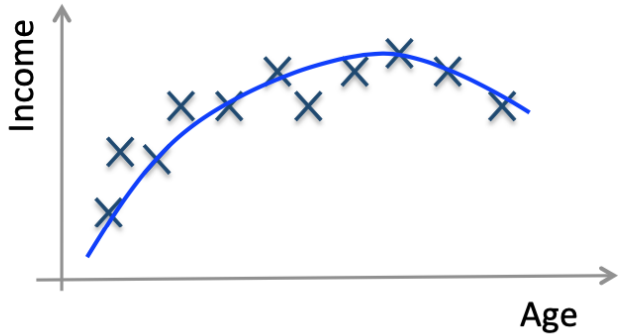
Week 5 – Linear regression Features

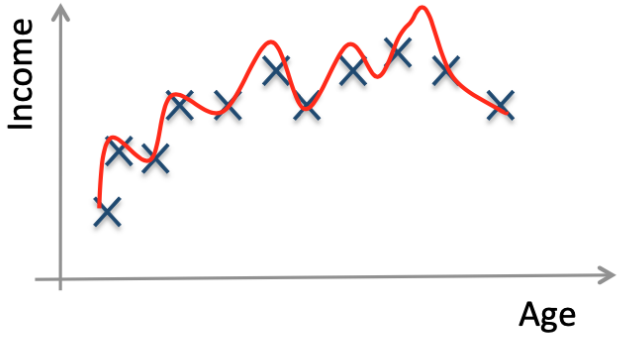
University of Salford MANCHESTER

Demo

Week 5 – Linear regression - Polynomial Regression







Week 5 – Linear regression - Polynomial Regression

- The simplest non-linear model we can consider, for a response *Y* and a predictor *X*, is a polynomial model of degree *M*,

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_M x^M + \epsilon.$$

- Just as in the case of linear regression with cross terms, polynomial regression is a special case of linear regression - we treat each x^m as a separate predictor. Thus, we can write

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_1^M \\ 1 & x_2^1 & \ldots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^M \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

58

Week 5 – Complex Model or Simple Model

University of Salford MANCHESTER

The Bias-Variance Tradeoff

Overfitting VS Underfitting

Week 5 – Complex Model or Simple Model

University of Salford MANCHESTER

IID

Independent and Identically Distributed

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- D={$(x_1,y_1),\ldots,(x_n,y_n)$}, drawn i.i.d. from some distribution P(X,Y)

Expected Label (given $\mathbf{x} \in \mathbb{R}^d$):

$$\bar{y}(\mathbf{x}) = E_{y|\mathbf{x}}[Y] = \int_y y \Pr(y|\mathbf{x}) \partial y.$$

Expected Classifier/ Predictor

$$\bar{h} = E_{D \sim P^n}[h_D] = \int_D h_D \Pr(D) \partial D$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x}) - y\right]^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x})-y\right]^2\right]=E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)+\left(\bar{h}(\mathbf{x})-y\right)\right]^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x})-y\right]^2\right] = E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)+\left(\bar{h}(\mathbf{x})-y\right)\right]^2\right]$$
$$= E_{\mathbf{x},D}\left[\left(\bar{h}_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + 2\,E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x})-y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$= E_{\mathbf{x},D}\left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2\right] + 2\, E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - y\right)^2\right]$$

$$\begin{aligned}
E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] &= E_{\mathbf{x},y}\left[E_D\left[h_D(\mathbf{x}) - \bar{h}(\mathbf{x})\right]\left(\bar{h}(\mathbf{x}) - y\right)\right] \\
&= E_{\mathbf{x},y}\left[\left(E_D\left[h_D(\mathbf{x})\right] - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\
&= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x}) - \bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x}) - y\right)\right] \\
&= E_{\mathbf{x},y}\left[0\right] \\
&= 0
\end{aligned}$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x})-y\right]^2\right] = E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)+\left(\bar{h}(\mathbf{x})-y\right)\right]^2\right]$$

$$= E_{\mathbf{x},D}\left[\left(\bar{h}_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + 2\,E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x})-y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left[h_D(\mathbf{x})-y\right]^2\right] = E_{\mathbf{x},y,D}\left[\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)+\left(\bar{h}(\mathbf{x})-y\right)\right]^2\right]$$

$$= E_{\mathbf{x},D}\left[\left(\bar{h}_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right] + 2\,E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x})-y\right)\right] + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]=\underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}}+E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

$$E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]=E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)+\left(\bar{y}(\mathbf{x})-y\right)^2\right]$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

$$\begin{aligned} E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)+\left(\bar{y}(\mathbf{x})-y\right)^2\right] \\ &= E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right] + E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right] + 2\,E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-y\right)\right] \end{aligned}$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

Bias is the tendency of a statistic to overestimate or underestimate

$$E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right] = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right]$$

$$\begin{aligned} E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)+\left(\bar{y}(\mathbf{x})-y\right)^2\right] \\ &= \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} + 2\,E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-y\right)\right] \end{aligned}$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$
\begin{aligned}
E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-y\right)^2\right] &= E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)+\left(\bar{y}(\mathbf{x})-y\right)^2\right] \\
&= \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2} + 2\,E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-y\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
E_{\mathbf{x},y}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{y}(\mathbf{x})-y\right)\right] &= E_{\mathbf{x}}\left[E_{y|\mathbf{x}}\left[\bar{y}(\mathbf{x})-y\right]\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\right] \\
&= E_{\mathbf{x}}\left[E_{y|\mathbf{x}}\left[\bar{y}(\mathbf{x})-y\right]\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\right] \\
&= E_{\mathbf{x}}\left[\left(\bar{y}(\mathbf{x})-E_{y|\mathbf{x}}\left[y\right]\right)\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\right] \\
&= E_{\mathbf{x}}\left[\left(\bar{y}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)\right] \\
&= E_{\mathbf{x}}\left[0\right] \\
&= 0
\end{aligned}
$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

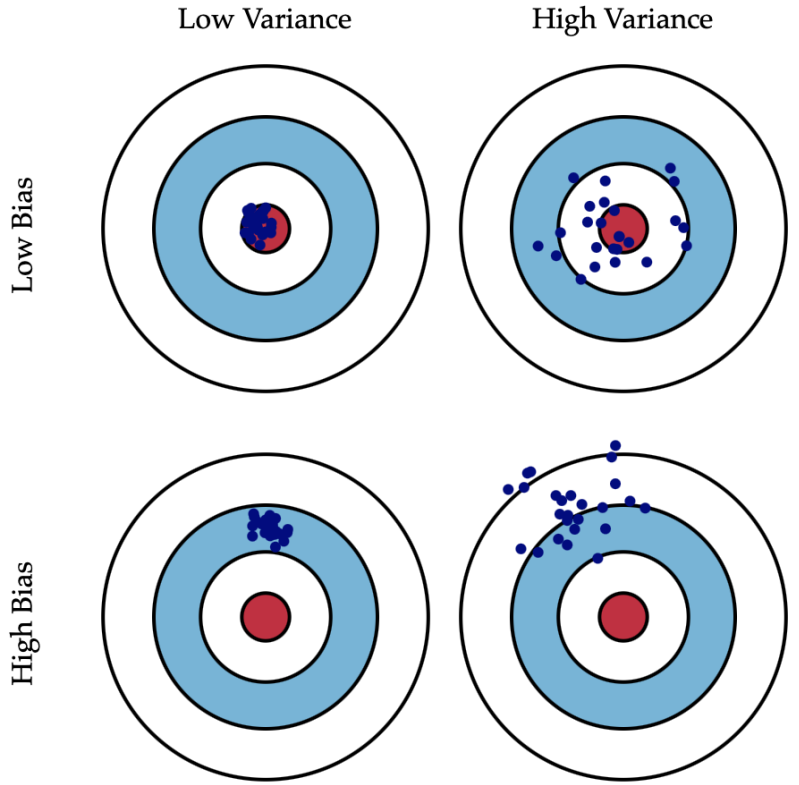
$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$

Week 5
The Bias-Variance Tradeoff

University of Salford MANCHESTER

- **Decomposition of Expected Test Error**

$$\underbrace{E_{\mathbf{x},y,D}\left[\left(h_D(\mathbf{x})-y\right)^2\right]}_{\text{Expected Test Error}} = \underbrace{E_{\mathbf{x},D}\left[\left(h_D(\mathbf{x})-\bar{h}(\mathbf{x})\right)^2\right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y}\left[\left(\bar{y}(\mathbf{x})-y\right)^2\right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}}\left[\left(\bar{h}(\mathbf{x})-\bar{y}(\mathbf{x})\right)^2\right]}_{\text{Bias}^2}$$

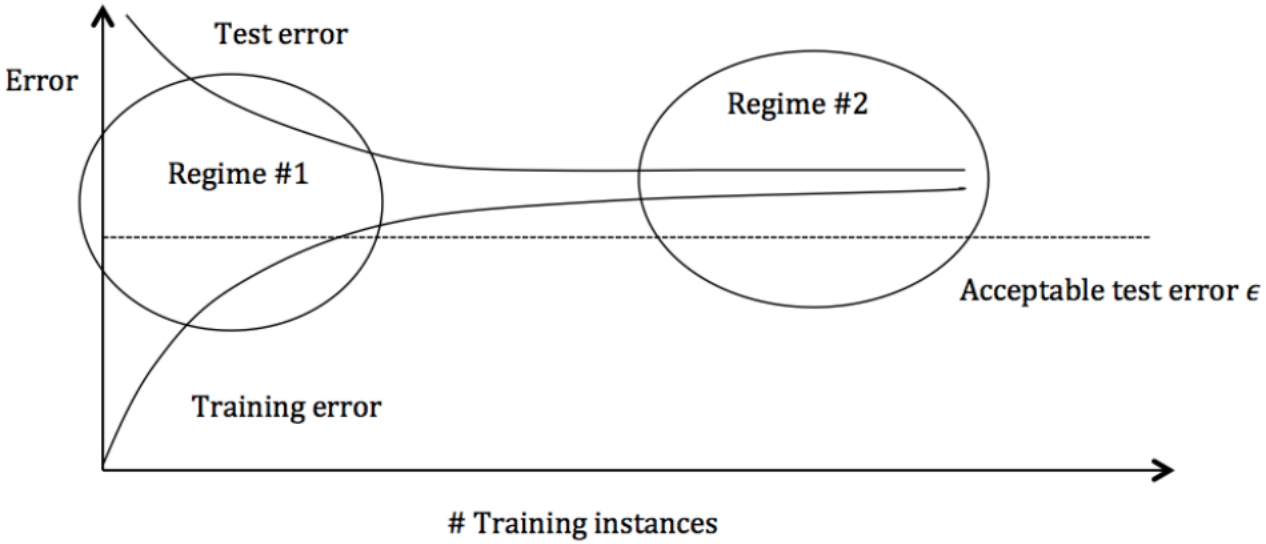


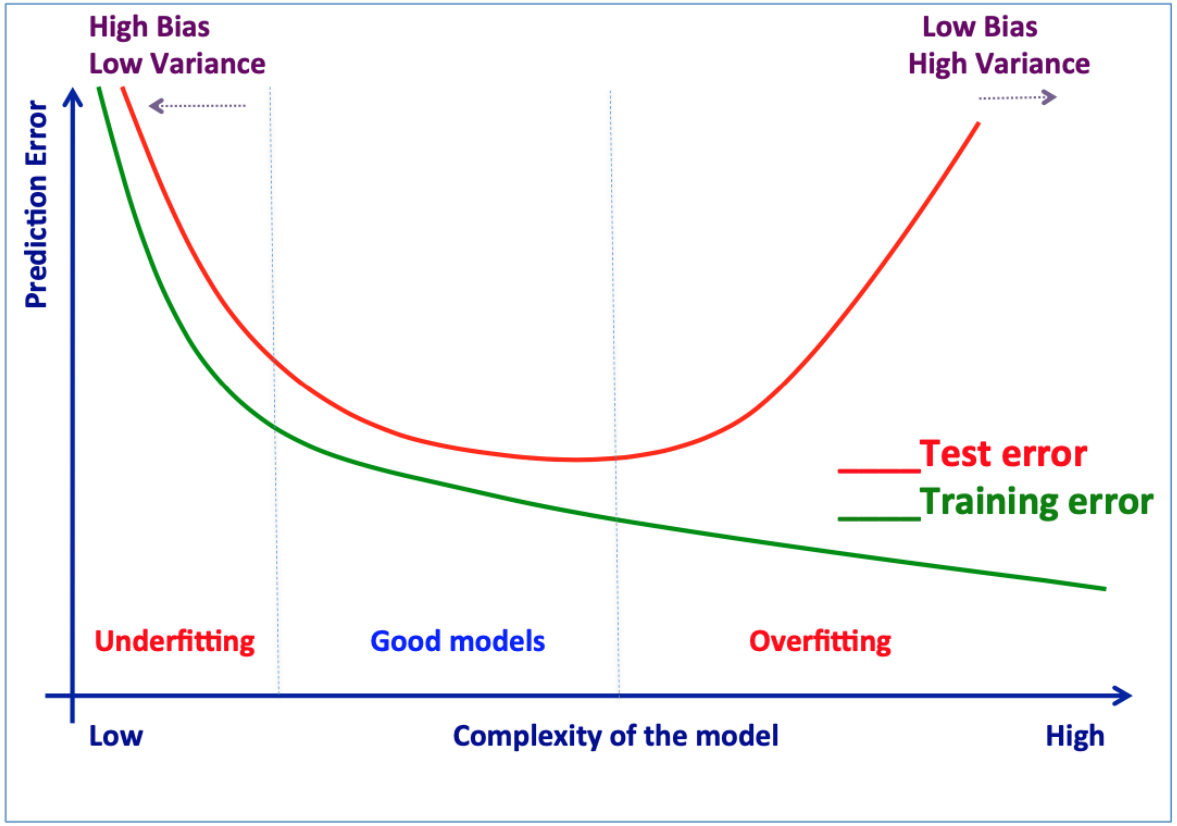
http://scott.fortmann-roe.com/docs/BiasVariance.html

Week 5 – ML Generalization

University of Salford MANCHESTER

- Overfitting VS Underfitting
- The Bias-Variance Tradeoff





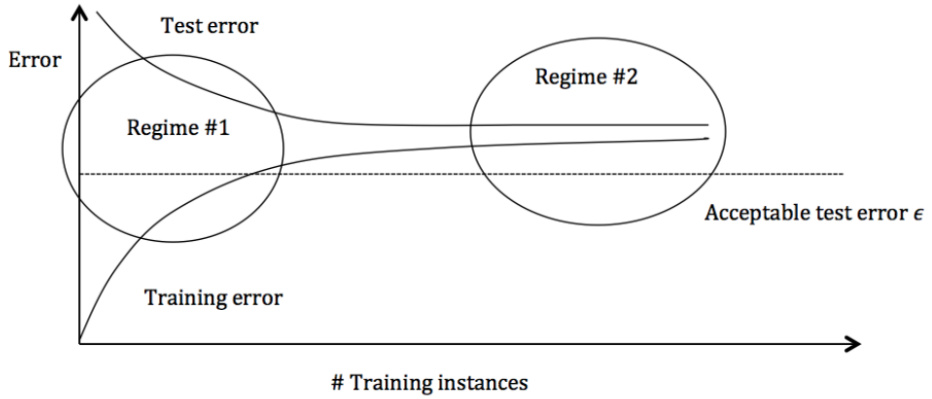
Week 5 – ML Generalization

Regime 1 (High Variance)

In the first regime, the cause of the poor performance is high variance

Symptoms:

1. Training error is much lower than test error
2. Training error is lower than ϵ
3. Test error is above ϵ



Week 5 – ML Generalization

Regime 1 (High Variance)

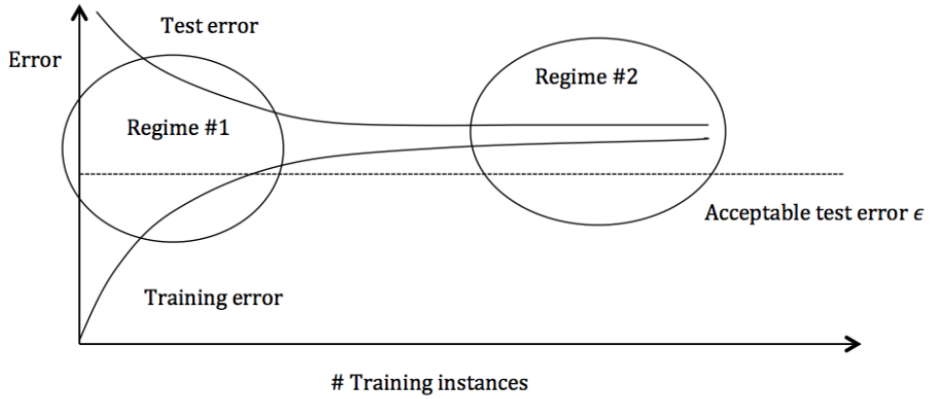
In the first regime, the cause of the poor performance is high variance

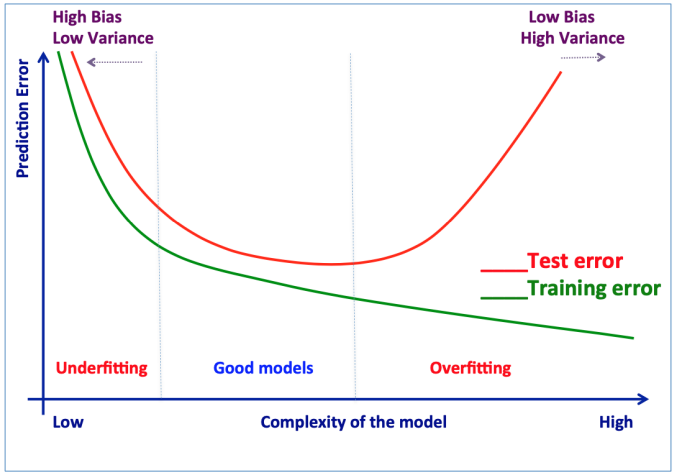
Symptoms:

1. Training error is much lower than test error
2. Training error is lower than ϵ
3. Test error is above ϵ

Remedies:

- Add more training data
- Reduce model complexity -- complex models are prone to high variance
- Bagging (will be covered later in the course)





Week 5 – ML Generalization

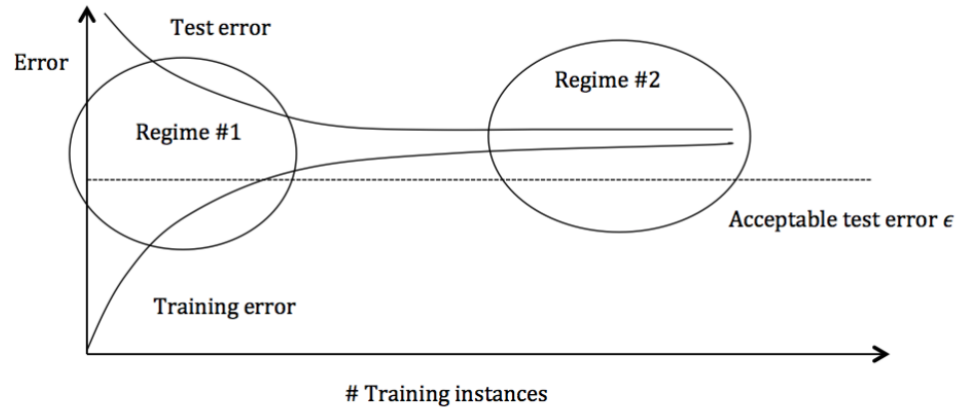
University of Salford MANCHESTER

Regime 2 (High Bias)

Unlike the first regime, the second regime indicates high bias: the model being used is not robust enough to produce an accurate prediction.

Symptoms:

1. Training error is higher than ϵ



Week 5 – ML Generalization

University of Salford MANCHESTER

Regime 2 (High Bias)

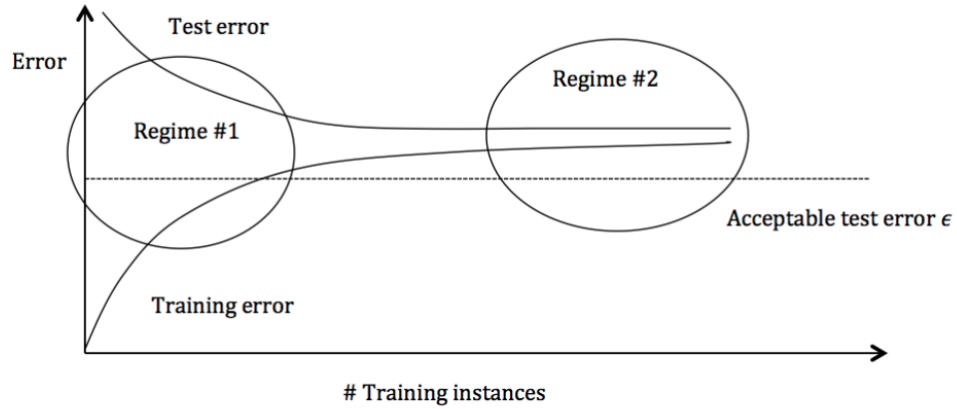
Unlike the first regime, the second regime indicates high bias: the model being used is not robust enough to produce an accurate prediction.

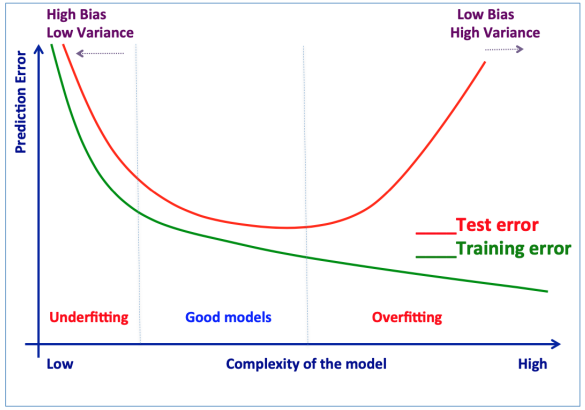
Symptoms:

1. Training error is higher than ϵ

Remedies:

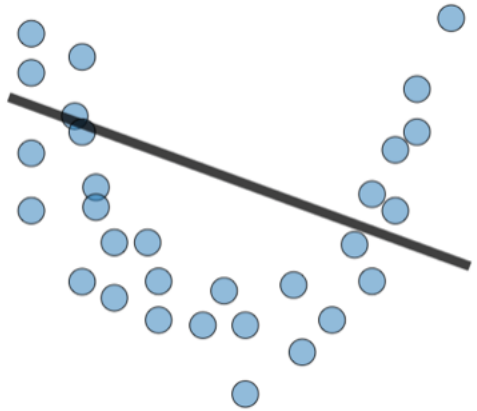
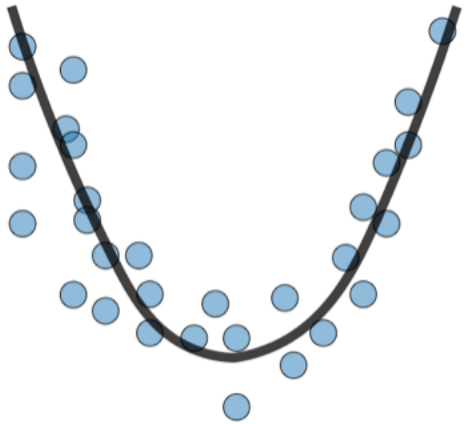
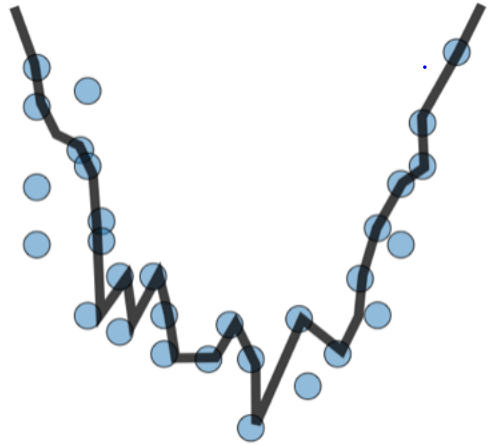
- Use more complex model (e.g. kernelize, use non-linear models)
- Add features
- Boosting (will be covered later in the course)





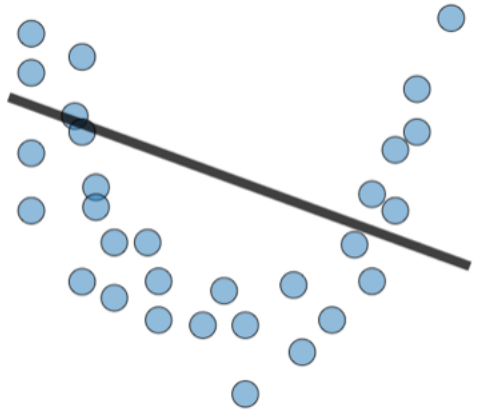
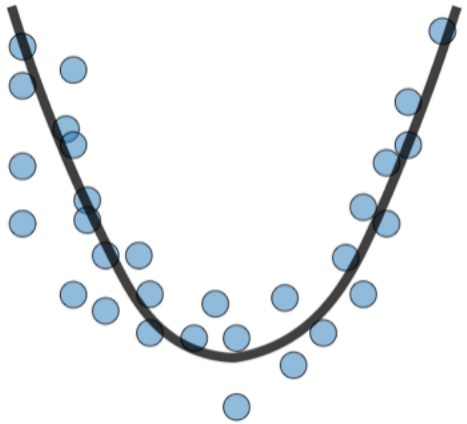
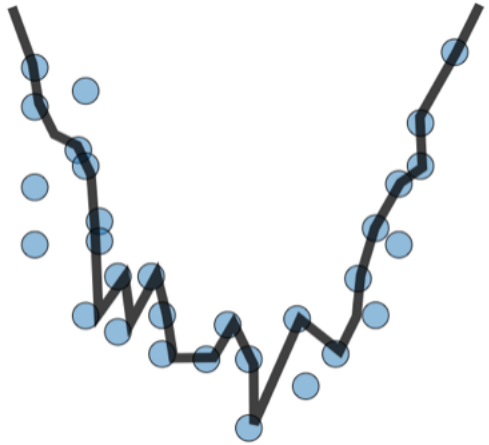
Week 5 – ML Generalization

University of Salford MANCHESTER

Symptoms			
Regression illustration			

Week 5 – ML Generalization

University of Salford MANCHESTER

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			

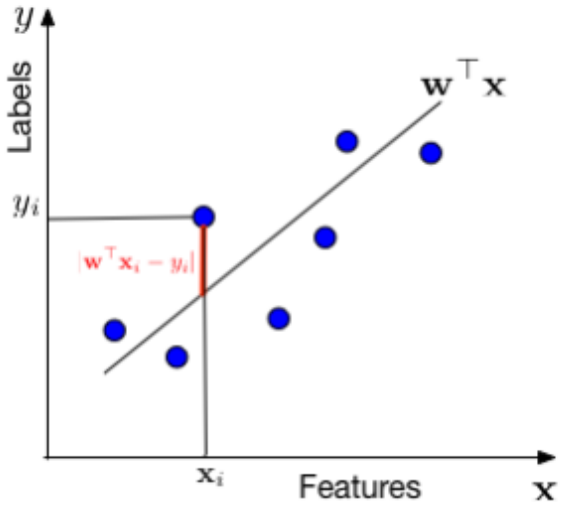
Model complexity

Week 5 – ML
Estimating with MAP

Data Assumption: $y_i \in \mathbb{R}$

Model Assumption: $y_i = \mathbf{w}^\top \mathbf{x}_i + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$

$$\Rightarrow y_i|\mathbf{x}_i \sim N(\mathbf{w}^\top \mathbf{x}_i, \sigma^2) \Rightarrow P(y_i|\mathbf{x}_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mathbf{x}_i^\top \mathbf{w} - y_i)^2}{2\sigma^2}}$$



Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$\mathbf{w} = \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$\begin{aligned}\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)}\end{aligned}$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})
\end{aligned}
$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w})
\end{aligned}
$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w})
\end{aligned}
$$

https://en.wikipedia.org/wiki/Chain_rule_(probability)

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i)\right] P(\mathbf{w})
\end{aligned}
$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i)\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})\right] P(\mathbf{w})
\end{aligned}
$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i)\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) + \log P(\mathbf{w})
\end{aligned}
$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i)\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) + \log P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmin}}\, \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \frac{1}{2\tau^2} \mathbf{w}^\top \mathbf{w}
\end{aligned}
$$

Week 5 – ML
Estimating with MAP

University of Salford MANCHESTER

Additional Model Assumption: $P(\mathbf{w}) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{\mathbf{w}^\top \mathbf{w}}{2\tau^2}}$

$$
\begin{aligned}
\mathbf{w} &= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(\mathbf{w}|y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \frac{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w})}{P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n)} \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, P(y_1, \mathbf{x}_1, \ldots, y_n, \mathbf{x}_n|\mathbf{w})P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i, \mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i|\mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})P(\mathbf{x}_i)\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \left[\prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w})\right] P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmax}}\, \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i, \mathbf{w}) + \log P(\mathbf{w}) \\
&= \underset{\mathbf{w}}{\operatorname{argmin}}\, \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \frac{1}{2\tau^2} \mathbf{w}^\top \mathbf{w} \\
&= \underset{\mathbf{w}}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \lambda ||\mathbf{w}||_2^2
\end{aligned}
$$

$$\lambda = \frac{\sigma^2}{n\tau^2}$$

Week 5 – ML
Ridge Regression

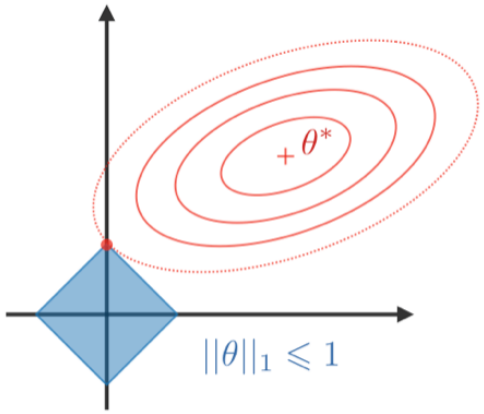
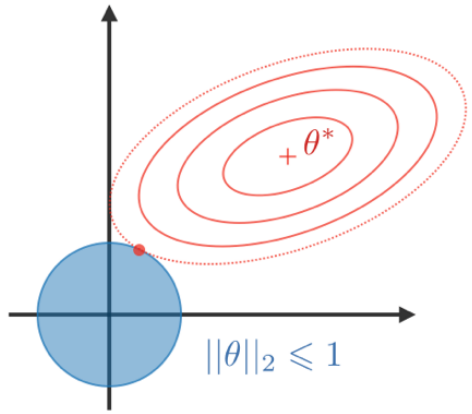
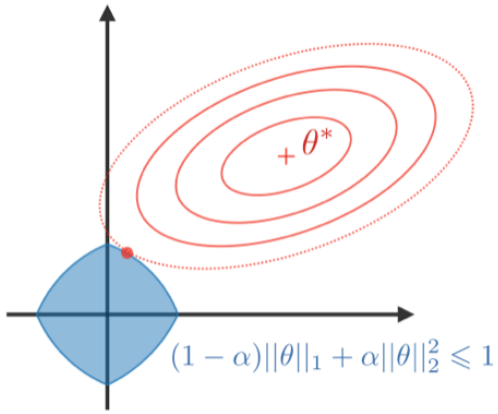
University of Salford MANCHESTER

- $\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \lambda ||\mathbf{w}||_2^2.$
- Squared loss.
- $l2$-regularization.

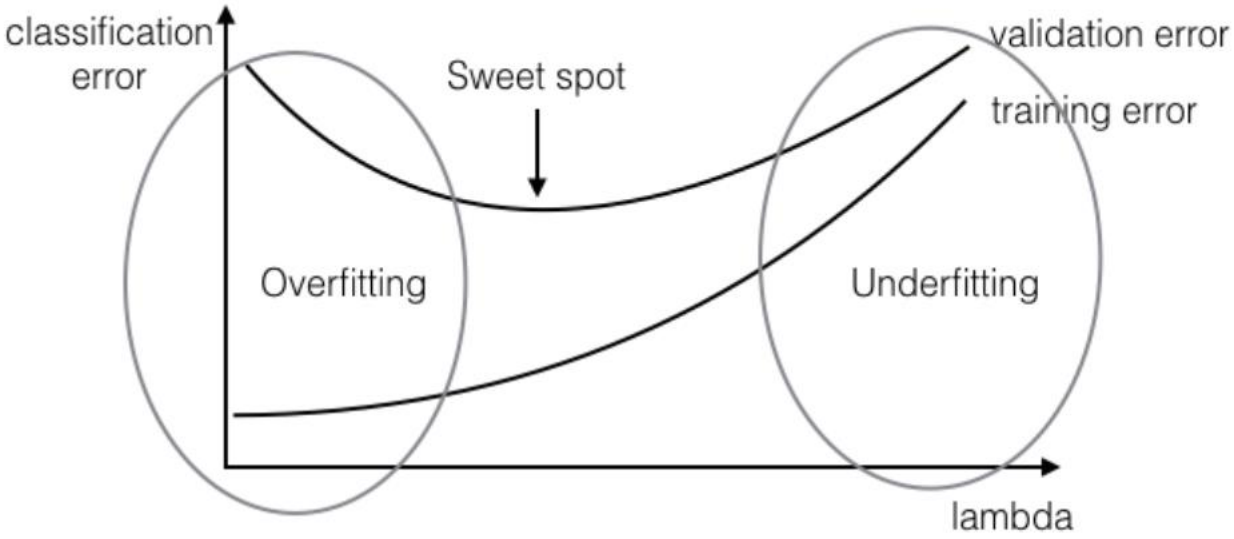
Week 5 – ML
Ridge Regression

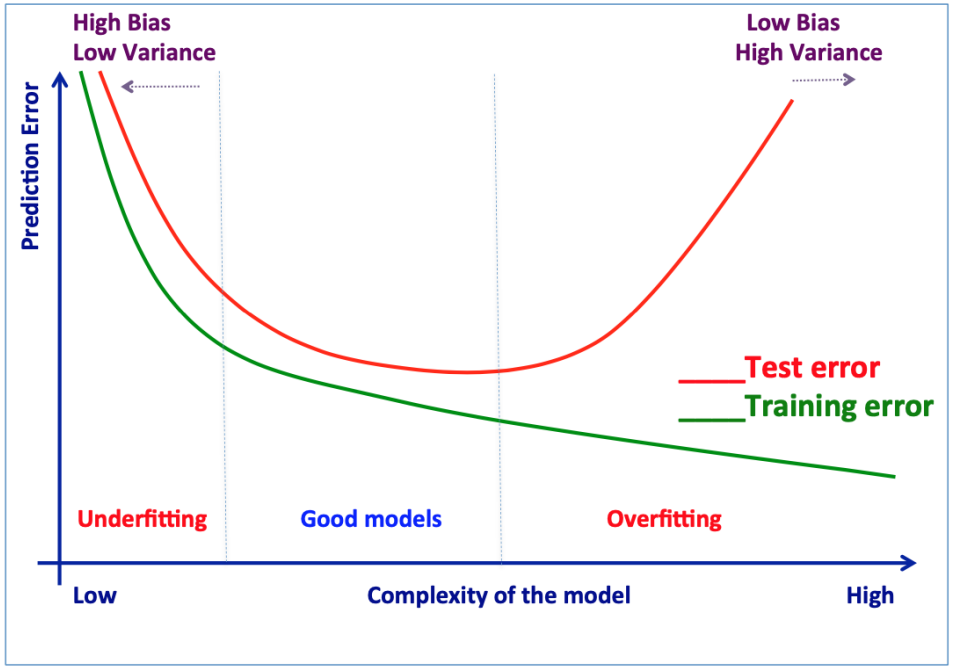
University of Salford MANCHESTER

□ **Regularization** — The regularization procedure aims at avoiding the model to overfit the data and thus deals with high variance issues. The following table sums up the different types of commonly used regularization techniques:

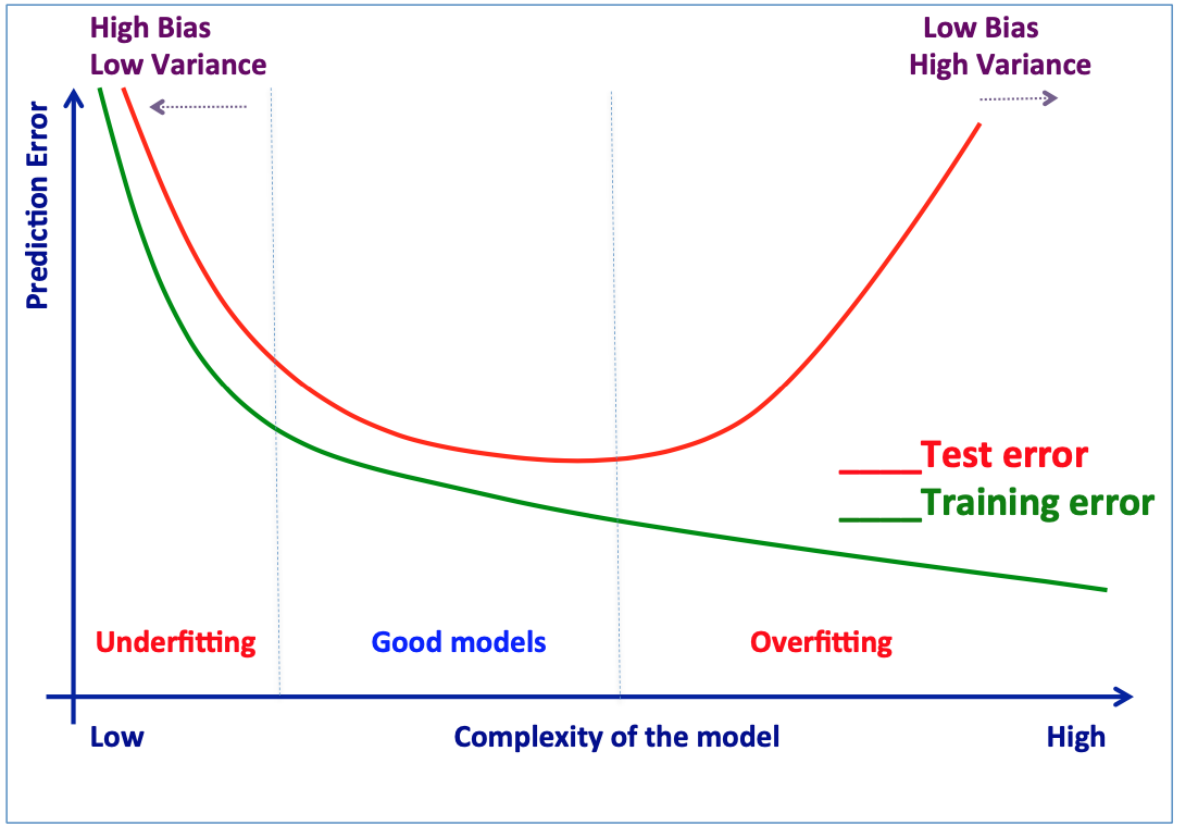
LASSO	Ridge	Elastic Net
• Shrinks coefficients to 0 • Good for variable selection	Makes coefficients smaller	Tradeoff between variable selection and small coefficients
		
$\ldots + \lambda\|\|\theta\|\|_1$ $\lambda \in \mathbb{R}$	$\ldots + \lambda\|\|\theta\|\|_2^2$ $\lambda \in \mathbb{R}$	$\ldots + \lambda\left[(1-\alpha)\|\|\theta\|\|_1 + \alpha\|\|\theta\|\|_2^2\right]$ $\lambda \in \mathbb{R}, \alpha \in [0,1]$

Week 5 – ML
Generalization/ Regularization



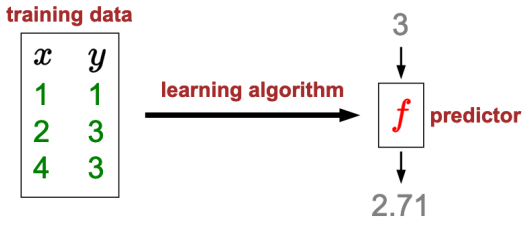


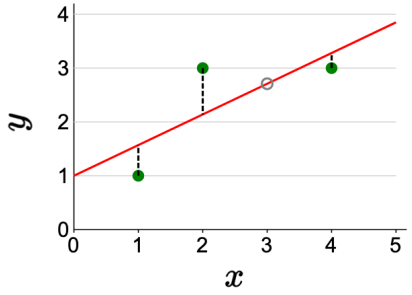
Week 5: Artificial Intelligent – Machine Learning



$$L = \underbrace{\frac{1}{N}\sum_{i} L_i}_{\text{data loss}} + \underbrace{\lambda R(W)}_{\text{regularization loss}}$$

Week 5: Artificial Intelligent – Machine Learning





Which predictors are possible?
Hypothesis class

How good is a predictor?
Loss function

How to compute best predictor?
Optimization algorithm

Linear functions
$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)\}, \phi(x) = [1, x]$

Squared loss
$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$

$$\mathbf{x}^T\mathbf{x}\widehat{\beta} - \mathbf{x}^T\mathbf{y} = 0$$

Loss and Regularizer	Comments
Ordinary Least Squares $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2$	◦ Squared Loss ◦ No Regularization ◦ Closed form solution: ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$ ◦ $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ ◦ $\mathbf{y} = [y_1, \ldots, y_n]$
Ridge Regression $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2 + \lambda\|w\|_2^2$	◦ Squared Loss ◦ l_2-Regularization ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbb{I})^{-1}\mathbf{X}\mathbf{y}^\top$
Lasso $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|_1$	◦ + sparsity inducing (good for feature selection) ◦ + Convex ◦ - Not strictly convex (no unique solution) ◦ - Not differentiable (at 0) ◦ Solve with (sub)-gradient descent or SVEN
Elastic Net $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2$ $\alpha \in [0, 1)$	◦ ADVANTAGE: Strictly convex (i.e. unique solution) ◦ + sparsity inducing (good for feature selection) ◦ + Dual of squared-loss SVM, see SVEN ◦ DISADVANTAGE: - Non-differentiable

Week 5: Artificial Intelligent – Machine Learning

University of Salford MANCHESTER

```python
from sklearn.linear_model import Ridge
reg = Ridge(alpha=1.0).fit(X, y)
```

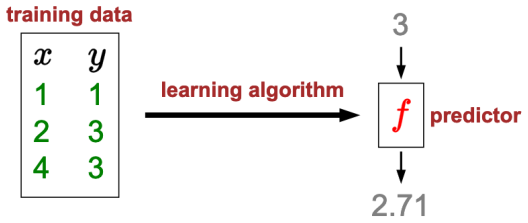
```python
n_samples, n_features = 10, 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
```

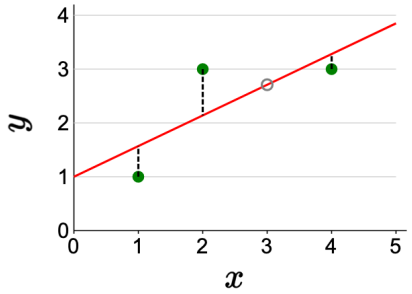
```python
print("score = ",SDS.score(X, y))
print("coef =", SDS.coef_)
print("intercept = ", SDS.intercept_)
print("New data point = ",SDS.predict(np.array([[3,2,1,3, 5]])))
```

```
score =  0.364855050716991 54
coef = [-0.14688728  0.16152898 -0.35137982  0.10289913  0.31422636]
intercept =  [0.50283142]
New data point =  [1.9136769]
```

Loss and Regularizer	Comments
Ordinary Least Squares $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2$	◦ Squared Loss ◦ No Regularization ◦ Closed form solution: ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y}^\top$ ◦ $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ ◦ $\mathbf{y} = [y_1, \ldots, y_n]$
Ridge Regression $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top x_i - y_i)^2 + \lambda\|w\|_2^2$	◦ Squared Loss ◦ l_2-Regularization ◦ $\mathbf{w} = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbb{I})^{-1}\mathbf{X}\mathbf{y}^\top$
Lasso $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \lambda\|\mathbf{w}\|_1$	◦ + sparsity inducing (good for feature selection) ◦ + Convex ◦ - Not strictly convex (no unique solution) ◦ - Not differentiable (at 0) ◦ Solve with (sub)-gradient descent or SVEN
Elastic Net $\min_{\mathbf{w}} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2$ $\alpha \in [0, 1)$	◦ ADVANTAGE: Strictly convex (i.e. unique solution) ◦ + sparsity inducing (good for feature selection) ◦ + Dual of squared-loss SVM, see SVEN ◦ DISADVANTAGE: - Non-differentiable

Week 5: Artificial Intelligent – Machine Learning





Which predictors are possible?
Hypothesis class

Linear functions
$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)\}, \phi(x) = [1, x]$

How good is a predictor?
Loss function

Squared loss
$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$

How to compute best predictor?
Optimization algorithm

Gradient descent
$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \text{TrainLoss}(\mathbf{w})$

```
n_samples, n_features = 10, 5
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
```

```
from sklearn.linear_model import SGDRegressor
SDS = SGDRegressor(max_iter=1000, tol=1e-3).fit(X, y)
```

```
print("score = ",SDS.score(X, y))
print("coef =", SDS.coef_)
print("intercept = ", SDS.intercept_)
print("New data point = ",SDS.predict(np.array([[3,2,1,3, 5]])))
```

```
score =  0.36485505071699154
coef = [-0.14688728  0.16152898 -0.35137982  0.10289913  0.31422636]
intercept =  [0.50283142]
New data point =  [1.9136769]
```

class `sklearn.linear_model.` **`SGDRegressor`**(*loss='squared_loss', *, penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, random_state=None, learning_rate='invscaling', eta0=0.01, power_t=0.25, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, warm_start=False, average=False*) [source]

Week 5: Artificial Intelligent – Machine Learning

University of Salford MANCHESTER

class `sklearn.linear_model.` **`SGDClassifier`**(*loss='hinge', *, penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False*) [source]

```
1 X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
2 Y = np.array([1, 1, 2, 2])
```

```
1 from sklearn.linear_model import SGDClassifier
2 SGDCLASS = SGDClassifier(max_iter=1000, tol=1e-3).fit(X, Y)
```

```
1 print(SGDCLASS.predict([[-0.8, -1]]))
```

```
1 from sklearn.linear_model import LogisticRegression
2 LG = LogisticRegression().fit(X, Y)
3 print(LG.predict([[-0.8, -1]]))
```

$$\mathbf{x}^T\mathbf{x}\widehat{\beta} - \mathbf{x}^T\mathbf{y} = 0$$

Week 5 Summary

University of Salford MANCHESTER

Linear Model

Model complexity

Week 5 Next

University of Salford MANCHESTER

Non Linear Models