Today's Quiz

1. Define what is an **orthogonal** matrix.

2. **True or False.** If a matrix is non-singular, it is also non-invertible.

3. What are some of the most useful insights you gathered from the article “A Few Useful Things to Know About Machine Learning”?
Types of machine learning

- Supervised learning
  - Classification
  - Regression
- Unsupervised learning
- Reinforcement learning

Terminology

- Columns are called input variables or features or attributes.
- The columns we are trying to predict (outcome and time) are called output variables or targets.
- Rows in the table are training examples or instances.
- The whole table is called a data set.

<table>
<thead>
<tr>
<th>tumor size</th>
<th>texture</th>
<th>perimeter</th>
<th>...</th>
<th>outcome</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.02</td>
<td>27.6</td>
<td>117.5</td>
<td></td>
<td>N</td>
<td>31</td>
</tr>
<tr>
<td>17.99</td>
<td>10.38</td>
<td>122.8</td>
<td></td>
<td>N</td>
<td>61</td>
</tr>
<tr>
<td>20.29</td>
<td>14.34</td>
<td>135.1</td>
<td></td>
<td>R</td>
<td>27</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Supervised learning - Classification

**Goal:** Learning a function for a *categorical* output.

E.g.: Spam filtering. *The output (“Spam?”) is binary.*

<table>
<thead>
<tr>
<th>Sender in address book?</th>
<th>Header keyword</th>
<th>Word 1</th>
<th>Word 2</th>
<th>...</th>
<th>Spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1 Yes</td>
<td>Schedule</td>
<td>Hi</td>
<td>Profesor</td>
<td>...</td>
<td>No</td>
</tr>
<tr>
<td>x2 Yes</td>
<td>meeting</td>
<td>Joelle</td>
<td>I</td>
<td>...</td>
<td>No</td>
</tr>
<tr>
<td>x3 No</td>
<td>urgent</td>
<td>Unsecured</td>
<td>Business</td>
<td>...</td>
<td>Yes</td>
</tr>
<tr>
<td>x4 No</td>
<td>offer</td>
<td>Hello</td>
<td>I</td>
<td>...</td>
<td>Yes</td>
</tr>
<tr>
<td>x5 No</td>
<td>cash</td>
<td>We’ll Help</td>
<td>...</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>x6 No</td>
<td>comp-598</td>
<td>Dear</td>
<td>Professor</td>
<td>...</td>
<td>No</td>
</tr>
</tbody>
</table>

Supervised learning - Regression

**Goal:** Learning a function for a *continuous* output.

E.g.: Self-driving car speed control. *The output (“speed”) is continuous.*
Unsupervised learning

**Goal:** Learning a function over the input alone.

E.g. Organizing data into classes. Inferring distances between data points.

Reinforcement learning

**Goal:** Learning a sequence of actions that optimizes costs/rewards.

E.g.: Balancing an inverted pendulum.
Supervised learning

- Given a set of training examples: \( x_i = \langle x_{i1}, x_{i2}, x_{i3}, \ldots, x_{inn}, y_i \rangle \)
  - \( x_{ij} \) is the \( j \)th feature of the \( i \)th example
  - \( y_i \) is the desired output (or target) for the \( i \)th example.
  - \( X_j \) denotes the \( j \)th feature.
- We want to learn a function \( f: X_1 \times X_2 \times \ldots \times X_n \rightarrow Y \)
  which maps the input variables onto the output domain.

<table>
<thead>
<tr>
<th>tumor size</th>
<th>texture</th>
<th>perimeter</th>
<th>...</th>
<th>outcome</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.02</td>
<td>27.6</td>
<td>117.5</td>
<td></td>
<td>N</td>
<td>31</td>
</tr>
<tr>
<td>17.99</td>
<td>10.38</td>
<td>122.8</td>
<td></td>
<td>N</td>
<td>61</td>
</tr>
<tr>
<td>20.29</td>
<td>14.34</td>
<td>135.1</td>
<td></td>
<td>R</td>
<td>27</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Supervised learning

- Given a dataset \( X \times Y \), find a function: \( f: X \rightarrow Y \) such that \( f(x) \) is a good predictor for the value of \( y \).
- Formally, \( f \) is called the hypothesis.

- Output \( Y \) can have many types:
  - If \( Y = \mathbb{R} \), this problem is called regression.
  - If \( Y \) is a finite discrete set, the problem is called classification.
  - If \( Y \) has 2 elements, the problem is called binary classification.
Prediction problems

- The problem of predicting tumour recurrence is called: **classification**
- The problem of predicting the time of recurrence is called: **regression**
- Treat them as two separate supervised learning problems.

<table>
<thead>
<tr>
<th>tumor size</th>
<th>texture</th>
<th>perimeter</th>
<th>...</th>
<th>outcome</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>18.02</td>
<td>27.8</td>
<td>117.5</td>
<td></td>
<td>N</td>
<td>31</td>
</tr>
<tr>
<td>17.99</td>
<td>10.38</td>
<td>122.8</td>
<td></td>
<td>N</td>
<td>61</td>
</tr>
<tr>
<td>20.29</td>
<td>14.34</td>
<td>135.1</td>
<td></td>
<td>R</td>
<td>27</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Variable types

- **Quantitative**, often real number measurements.
  - Assumes that similar measurements are similar in nature.
- **Qualitative**, from a set (categorical, discrete).
  - E.g. {Spam, Not-spam}
- **Ordinal**, also from a discrete set, without metric relation, but that allows ranking.
  - E.g. {first, second, third}
The i.i.d. assumption

- In supervised learning, the examples $x_i$ in the training set are assumed to be independently and identically distributed.
  - Independently: Every $x_i$ is freshly sampled according to some probability distribution $D$ over the data domain $X$.
  - Identically: The distribution $D$ is the same for all examples.

- Why?

Empirical risk minimization

For a given function class $F$ and training sample $S$,

- Define a notion of error (left intentionally vague for now):
  \[ L_S(f) = \text{# mistakes made on the sample } S \]

- Define the Empirical Risk Minimization (ERM):
  \[ \text{ERM}_F(S) = \arg\min_{f \in F} L_S(f) \]
  where $\arg\min$ returns the function $f$ (or set of functions) that achieves the minimum loss on the training sample.

- Easier to minimize the error with i.i.d. assumption.
A regression problem

- What hypothesis class should we pick?

<table>
<thead>
<tr>
<th>Observe</th>
<th>Predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>0.86</td>
<td>2.49</td>
</tr>
<tr>
<td>0.09</td>
<td>0.83</td>
</tr>
<tr>
<td>-0.85</td>
<td>-0.25</td>
</tr>
<tr>
<td>0.87</td>
<td>3.10</td>
</tr>
<tr>
<td>-0.44</td>
<td>0.87</td>
</tr>
<tr>
<td>-0.43</td>
<td>0.02</td>
</tr>
<tr>
<td>-1.1</td>
<td>-0.12</td>
</tr>
<tr>
<td>0.40</td>
<td>1.81</td>
</tr>
<tr>
<td>-0.96</td>
<td>-0.83</td>
</tr>
<tr>
<td>0.17</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Linear hypothesis

- Suppose $Y$ is a linear function of $X$:

$$f_w(X) = w_0 + w_1 x_1 + ... + w_m x_m$$

- The $w_j$ are called parameters or weights.

- To simplify notation, we add an attribute $x_0=1$ to the $m$ other attributes (also called bias term or intercept).

How should we pick the weights?
Least-squares solution method

- The linear regression problem: \( f_w(x) = w_0 + \sum_{j=1}^m w_j x_j \)
  where \( m \) = the dimension of observation space, i.e. number of features.

- **Goal**: Find the **best** linear model given the data.

- Many different possible **evaluation** criteria!

- Most common choice is to find the \( w \) that minimizes:

  \[
  \text{Err}(w) = \sum_{i=1}^n (y_i - w^T x_i)^2
  \]

  (A note on notation: Here \( w \) and \( x \) are column vectors of size \( m+1 \).)

---

Least-squares solution for \( X \in \mathbb{R}^2 \)

![Image of linear regression model](image-url)
Least-squares solution method

• Re-write in matrix notation:
  
  \( f_w(X) = Xw \)

  \( \text{Err}(w) = (Y - Xw)^T(Y - Xw) \)

  where \( X \) is the \( n \times m \) matrix of input data,
  
  \( Y \) is the \( n \times 1 \) vector of output data,
  
  \( w \) is the \( m \times 1 \) vector of weights.

• To minimize, take the derivative w.r.t. \( w \):
  
  \( \frac{\partial \text{Err}(w)}{\partial w} = -2X^T(Y - Xw) \)

  You get a system of \( m \) equations with \( m \) unknowns.

• Set these equations to 0:
  
  \( X^T(Y - Xw) = 0 \)

Least-squares solution method

• We want to solve for \( w \):
  
  \( X^T(Y - Xw) = 0 \)

• Try a little algebra:
  
  \( X^TY = X^TXw \)

  \( \hat{w} = (X^TX)^{-1}X^TY \)

  (\( \hat{w} \) denotes the estimated weights)

• The fitted data:
  
  \( \hat{Y} = X\hat{w} = X(X^TX)^{-1}X^TY \)

• To predict new data \( X' \rightarrow Y' \):
  
  \( Y' = X'\hat{w} = X'(X^TX)^{-1}X^TY \)
Example of linear regression

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.86</td>
<td>2.49</td>
</tr>
<tr>
<td>0.09</td>
<td>0.83</td>
</tr>
<tr>
<td>-0.85</td>
<td>-0.25</td>
</tr>
<tr>
<td>0.87</td>
<td>3.10</td>
</tr>
<tr>
<td>-0.44</td>
<td>0.87</td>
</tr>
<tr>
<td>-0.43</td>
<td>0.02</td>
</tr>
<tr>
<td>-1.10</td>
<td>-0.12</td>
</tr>
<tr>
<td>0.40</td>
<td>1.81</td>
</tr>
<tr>
<td>-0.96</td>
<td>-0.83</td>
</tr>
<tr>
<td>0.17</td>
<td>0.43</td>
</tr>
</tbody>
</table>

What is a plausible estimate of $w$? **Try it!**

Data matrices

$$X^T X =$$

$$
\begin{bmatrix}
0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\times
\begin{bmatrix}
0.86 & 1 \\
0.09 & 1 \\
-0.85 & 1 \\
0.87 & 1 \\
-0.44 & 1 \\
-0.43 & 1 \\
-1.10 & 1 \\
0.40 & 1 \\
-0.96 & 1 \\
0.17 & 1
\end{bmatrix}
$$

$$= \begin{bmatrix}
4.95 & -1.39 \\
-1.39 & 10
\end{bmatrix}$$
Data matrices

\[
X^TY = \begin{bmatrix}
0.86 & 0.09 & -0.85 & 0.87 & -0.44 & -0.43 & -1.10 & 0.40 & -0.96 & 0.17 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
2.49 \\
0.83 \\
-0.25 \\
3.10 \\
0.87 \\
0.02 \\
-0.12 \\
1.81 \\
-0.83 \\
0.43
\end{bmatrix}
\]

\[
= \begin{bmatrix}
6.49 \\
8.34
\end{bmatrix}
\]

Solving the problem

\[
w = (X^TX)^{-1}X^TY = \begin{bmatrix}
4.95 & -1.39 \\
-1.39 & 10
\end{bmatrix}^{-1} \begin{bmatrix}
6.49 \\
8.34
\end{bmatrix} = \begin{bmatrix}
1.60 \\
1.05
\end{bmatrix}
\]

So the best fit line is \( y = 1.60x + 1.05 \).
Interpreting the solution

- Linear fit for a prostate cancer dataset
  - Features $\mathbf{X} = \{lcavol, lweight, age, lbph, svi, lcp, gleason, pgg45\}$
  - Output $\mathbf{y} = \text{level of PSA (an enzyme which is elevated with cancer)}$.
  - High coefficient weight (in absolute value) = important for prediction.

<table>
<thead>
<tr>
<th>Term</th>
<th>Coefficient</th>
<th>Std. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>2.46</td>
<td>0.09</td>
</tr>
<tr>
<td>lcavol</td>
<td>0.68</td>
<td>0.13</td>
</tr>
<tr>
<td>lweight</td>
<td>0.26</td>
<td>0.10</td>
</tr>
<tr>
<td>age</td>
<td>−0.14</td>
<td>0.10</td>
</tr>
<tr>
<td>lbph</td>
<td>0.21</td>
<td>0.10</td>
</tr>
<tr>
<td>svi</td>
<td>0.31</td>
<td>0.12</td>
</tr>
<tr>
<td>lcp</td>
<td>−0.29</td>
<td>0.15</td>
</tr>
<tr>
<td>gleason</td>
<td>−0.02</td>
<td>0.15</td>
</tr>
<tr>
<td>pgg45</td>
<td>0.27</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Computational cost of linear regression

- What operations are necessary?
  - Matrix multiplication:
    - For $A^{nxm} \times B^{mxp}$: We need $nmp$ operations.
  - Matrix inversion:
    - For $A^{nxn}$: We need $m^3$ operations.
  - In total: 1 matrix inversion + 3 matrix multiplications

- So we can do linear regression in polynomial time, but handling very large datasets (many examples, many features) can be problematic.
An alternative for minimizing mean-squared error (MSE)

• Recall the least-square solution: \( \hat{w} = (X^T X)^{-1} X^T Y \)

• What if \( X \) is too big to compute this explicitly (e.g. \( m \sim 10^6 \))?

• Go back to the gradient step:
  \[
  \text{Err}(w) = (Y - Xw)^T (Y - Xw)
  \]
  \[
  \frac{\partial \text{Err}(w)}{\partial w} = -2X^T(Y - Xw)
  \]
  \[
  \frac{\partial \text{Err}(w)}{\partial w} = 2(X^T Xw - X^T Y)
  \]

Gradient-descent solution for MSE

• Consider the error function:

• The gradient of the error is a vector indicating the direction to the minimum point.

• Instead of directly finding that minimum (using the closed-form equation), we can take small steps towards the minimum.
Gradient-descent solution for MSE

- We want to produce a sequence of weight solutions, \( w_0, w_1, w_2, \ldots \) such that: \( \text{Err}(w_0) > \text{Err}(w_1) > \text{Err}(w_2) > \ldots \)

- The algorithm:

  Given an initial weight vector \( w_0 \)
  
  Do for \( k=1, 2, \ldots \)
  
  \[
  w_{k+1} = w_k - \alpha_k \frac{\partial \text{Err}(w_k)}{\partial w_k}
  \]
  
  End when \( |w_{k+1} - w_k| < \varepsilon \)

- Parameter \( \alpha_k > 0 \) is the step-size (or learning rate) for iteration \( k \).

Convergence

- Convergence depends in part on the \( \alpha_k \).

- If steps are too large: the \( w_k \) may oscillate forever.
  - This suggests that \( \alpha_k \to 0 \) as \( k \to \infty \).

- If steps are too small: the \( w_k \) may not move far enough to reach a local minimum.
Robbins-Monroe conditions

- The $\alpha_k$ are a Robbins-Monroe sequence if:
  \[
  \sum_{k=0}^{\infty} \alpha_k = \infty \\
  \sum_{k=0}^{\infty} \alpha_k^2 < \infty
  \]
- These conditions are sufficient to ensure convergence of the $w_k$ to a local minimum of the error function.

  E.g. $\alpha_k = 1/(k+1)$ (averaging)
  E.g. $\alpha_k = 1/2$ for $k = 1, \ldots, T$
  $\alpha_k = 1/2^2$ for $k = T+1, \ldots, (T+1)+2T$
  etc.

Local minima

- Convergence is NOT to a global minimum, only to local minimum.

- The blue line represents the error function. There is no guarantee regarding the amount of error of the weight vector found by gradient descent, compared to the globally optimal solution.
Local minima

- Convergence is **NOT** to a global minimum, only to local minimum.

- For linear function approximations using Least-Mean Squares (LMS) error, this is not an issue: **only ONE global minimum!**
  - Local minima affects many other function approximators.

- **Repeated random restarts** can help (in all cases of gradient search).
A 3rd optimization method: QR decomposition

- Consider the usual criteria: \( X^T(Y - Xw) = 0 \)
- Assume \( X \) can be decomposed: \( X = QR \)
  where \( Q \) is an \( nxm \) orthogonal matrix (i.e. \( Q^TQ = I \)), and \( R \) is an \( m \times m \) upper triangular matrix.
- Replace \( X \) in equation above: \( (QR)^T Y = (QR)^T(QR)w \)
- Distribute the transpose: \( R^TQ^TY = R^TQ^TQRw \)
- Let \( Q^TQ = I \) and multiply by \( (R^T)^{-1} \): \( Q^TY = Rw \)
- Solution: \( \hat{w} = R^{-1}Q^TY \) The fitted outputs are: \( \hat{Y} = QQ^TY \)
- This method is more numerically stable than others, and \( R^{-1} \) is fast to compute because upper triangular.
- Alternately, we can use singular value decomposition.

What you should know

- Definition and characteristics of a supervised learning problem.
- Linear regression (hypothesis class, cost function, algorithm).
- Closed-form least-squares solution method (algorithm, computational complexity, stability issues).
- Gradient descent method (algorithm, properties).
To-do

- Reproduce the linear regression example (slide 21), solving it by hand and using the software of your choice.

- Suggested complementary readings:
  - Ch.2 (Sec. 2.1-2.4, 2.9) of Hastie et al.
  - Ch.3 of Bishop.
  - Ch.9 of Shalev-Schwartz et al.