CS 453X: Class 3

Jacob Whitehill
• A **column vector** is a \((n \times 1)\) matrix.

• A **row vector** is a \((1 \times n)\) matrix.

• The **transpose** of \((n \times k)\) matrix \(A\), denoted \(A^T\), is \((k \times n)\).

• Multiplication of matrices \(A\) and \(B\):
  - Only possible when: \(A\) is \((n \times k)\) and \(B\) is \((k \times m)\)
  - Result: \((n \times m)\)

• The **inner product** between two column vectors (same length) \(x, y\) can be written as: \(x^T y\)

• The **Hadamard** (element-wise) **product** between two matrices \(A\) and \(B\) is written as \(A \odot B\).
Weakness of our feature set

• So far, the feature we have considered are very weak:
  
  • Is pixel \((r_1,c_1)\) brighter than pixel \((r_2,c_2)\)?

• We can’t even express simple relationships such as:
  
  • “\((r_1,c_1)\) is at least 5 bigger than \((r_2,c_2)\)”
  
  • “2 times \((r_1,c_1)\) is bigger than \((r_2,c_2)\)”
  
  • “2 times \((r_1,c_1)\) plus 4 times \((r_2,c_2)\) is larger than \((r_3,c_3)\)”.
Linear regression

- We can harness these more complex relationships using linear regression.

- Let’s switch back to the age estimation problem…
Linear regression

• Linear regression is built as a linear combination of all the inputs $\mathbf{x}$:

$$\hat{y} = g(\mathbf{x}; \mathbf{w}) = \sum_{j=1}^{m} x_j w_j = \mathbf{x}^\top \mathbf{w}$$

• Here, we treat the image $\mathbf{x}$ as a vector (even though it represents a 2-d image).
Linear regression

- Linear regression is built as a linear combination of all the inputs $\mathbf{x}$:

$$\hat{y} = g(\mathbf{x}; \mathbf{w}) = \sum_{j=1}^{m} x_j w_j = \mathbf{x}^\top \mathbf{w}$$

- Vector $\mathbf{w}$ represents an “overlay image” that weights the different pixel intensities of $\mathbf{x}$.
Linear regression

- Image a 2x2 pixel “image” $\mathbf{x}$ and a weight matrix $\mathbf{w}$:

  $\begin{bmatrix}
  2 & 5 \\
  0 & 3 \\
  \end{bmatrix}$  $\begin{bmatrix}
  1 & 3 \\
  2 & 4 \\
  \end{bmatrix}$

- Then $\hat{y} = 2*1 + 5*3 + 0*2 + 3*4 = 22$
Linear regression

• How should we choose each “weight” $w_j$?

• Let’s define the **loss** function that we seek to minimize:

$$f_{\text{MSE}}(y, \hat{y}; w) = \frac{1}{2n} \sum_{i=1}^{n} \left( g(x^{(i)}; w) - y^{(i)} \right)^2$$

$$= \frac{1}{2n} \sum_{i=1}^{n} \left( x^{(i)\top} w - y^{(i)} \right)^2$$

The 2 in the denominator will slightly simplify the algebra later...
Linear regression

•\( \mathbf{w} \) is an unconstrained real-valued vector; hence, we can use differential calculus to find the minimum of \( f_{\text{MSE}} \).

• Just derive the gradient of \( f_{\text{MSE}} \) w.r.t. \( \mathbf{w} \), set to 0, and solve.

• Since \( f_{\text{MSE}} \) is a convex function, we are guaranteed that this critical point is a global minimum.
Matrix/vector calculus

- For a real-valued function $f : \mathbb{R}^m \rightarrow \mathbb{R}$, we define the gradient w.r.t. $\mathbf{w}$ as:

$$\nabla_{\mathbf{w}} f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \vdots \\ \frac{\partial f}{\partial w_m} \end{bmatrix}$$

- In other words, the gradient is a column vector containing all first partial derivatives w.r.t. $\mathbf{w}$. 
### Solving for w

- The gradient of $f_{\text{MSE}}$ is thus:

$$ \nabla_w f_{\text{MSE}}(y, \hat{y}; w) = \nabla_w \left[ \frac{1}{2n} \sum_{i=1}^{n} \left( \mathbf{x}^{(i)^\top} w - y^{(i)} \right)^2 \right] $$

$$ = \frac{1}{2n} \sum_{i=1}^{n} \nabla_w \left[ \left( \mathbf{x}^{(i)^\top} w - y^{(i)} \right)^2 \right] $$

$$ = ? $$

1. $$ \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}^{(i)} \left( \mathbf{x}^{(i)^\top} w - y^{(i)} \right) $$

2. $$ \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{x}^{(i)^\top} w - y^{(i)} \right) \mathbf{x}^{(i)} $$

3. $$ \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{x}^{(i)^\top} \mathbf{w} \mathbf{x}^{(i)} - \mathbf{x}^{(i)} y^{(i)} \right) $$

4. $$ \frac{1}{n} \sum_{i=1}^{n} \left( \mathbf{x}^{(i)^\top} w - y^{(i)} \right) \mathbf{x}^{(i)^\top} $$
Solving for $w$

- The gradient of $f_{\text{MSE}}$ is thus:

$$
\nabla_w f_{\text{MSE}}(y, \hat{y}; w) = \nabla_w \left[ \frac{1}{2n} \sum_{i=1}^{n} \left( x^{(i) \top} w - y^{(i)} \right)^2 \right] \\
= \frac{1}{2n} \sum_{i=1}^{n} \nabla_w \left[ \left( x^{(i) \top} w - y^{(i)} \right)^2 \right] \\
= \frac{1}{n} \sum_{i=1}^{n} x^{(i)} \left( x^{(i) \top} w - y^{(i)} \right)
$$

1. $\frac{1}{n} \sum_{i=1}^{n} x^{(i)} \left( x^{(i) \top} w - y^{(i)} \right)$ Correct
Solving for $w$

- By setting to 0, splitting the sum apart, and solving, we reach the solution:
Solving for \( w \)

- By setting to 0, splitting the sum apart, and solving, we reach the solution:

\[
\frac{1}{n} \sum_{i=1}^{n} x^{(i)} \left( x^{(i)^T} w - y^{(i)} \right) = 0
\]

\[
\sum_{i} x^{(i)} x^{(i)^T} w = \sum_{i} x^{(i)} y^{(i)}
\]

\[
w = \left( \sum_{i} x^{(i)} x^{(i)^T} \right)^{-1} \sum_{i} x^{(i)} y^{(i)}
\]
Matrix notation

- Let’s define a matrix $X$ to contain all the training images:

$$X = \begin{bmatrix} x^{(1)} & \ldots & x^{(n)} \end{bmatrix}$$

- In statistics, $X$ is called the **design matrix**.

- Let’s define vector $y$ to contain all the training labels:

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$
Matrix notation

- Using summation notation, we derived:

\[ w = \left( \sum_{i=1}^{n} x^{(i)} x^{(i)\top} \right)^{-1} \left( \sum_{i=1}^{n} x^{(i)} y^{(i)} \right) \]

- Using matrix notation, we can write the solution as:

\[
\begin{align*}
\text{where } x &= \begin{bmatrix} x^{(1)} & \ldots & x^{(n)} \end{bmatrix} \\
1. \quad w &= (X^\top X)^{-1} X^\top y \\
2. \quad w &= (X \odot X)^{-1} \odot X^\top y \\
3. \quad w &= (XX^\top)^{-1} Xy
\end{align*}
\]
Matrix notation

- The solution for the optimal $w$ can be re-written as:

$$w = (XX^\top)^{-1} Xy$$

- To compute this, do **not** use `np.linalg.inv`.

- Instead, use `np.linalg.solve`, which avoids explicitly computing the matrix inverse.

- Show demo.