Lecture 2:
Vectors + Matrices
We ultimately need to use training data to learn the "best" weight vector. I.e., we want \( \hat{y}_i = \langle w, x_i \rangle \approx y_i \) for all \( i = 1, \ldots, n \).

Our loss function will measure how far each \( y_i \) is from \( \hat{y}_i \).

We can write this objective more simply. Define:

- label vector \( \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} \) and feature matrix \( \mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n \times p} \) (real matrix with \( n \) rows, \( p \) columns)

- \( x_{ij} \) is the feature of \( i^{th} \) sample of \( X \) - \( p \) features of \( i^{th} \) sample = \( x_i^\top \)
- \( j^{th} \) col of \( X \) - feature \( j \) for all \( n \) samples

Then we can write our model as

\[
\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \mathbf{X}w \quad \leftarrow \text{linear model for all } n \text{ samples in one equation}
\]

Computing \( \mathbf{X}w \) means taking the inner product of each row of \( \mathbf{X} \) with \( w \) and storing the results in a vector \( \hat{y} \).
Note that dimensions should always match:
\[ \hat{y} = X \mathbf{w}, \quad \hat{y} \in \mathbb{R}^n, \quad \mathbf{w} \in \mathbb{R}^p, \quad X \in \mathbb{R}^{n \times p} \]
- \# rows of \( X \) = length of \( \hat{y} \)
- \# cols of \( X \) = length of \( \mathbf{w} \)

**Example**

\[ X = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 0 & 3 \end{bmatrix} \quad \text{2 features} \quad \text{3 training samples} \]

\[ X_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad X_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad X_3 = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \]

\[ \mathbf{w} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \]

\[ X \mathbf{w} = \begin{bmatrix} 2 \cdot 1 + 4 \cdot 0 \\ 2 \cdot 2 + 4 \cdot 0 \\ 2 \cdot 0 + 3 \cdot 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} \]

Another perspective: \( X \mathbf{w} \) is a weighted sum of the columns of \( X \), where \( \mathbf{w} \) gives the weights

\[ X \mathbf{w} = 2 \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + 4 \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} \]
Example

This doesn't look like a straight line, but linear models can still help!

Let \( X = \begin{bmatrix}
1 & z_1 & z_1^2 & z_1^3 \\
1 & z_2 & z_2^2 & z_2^3 \\
\vdots & \vdots & \vdots & \vdots \\
1 & z_n & z_n^2 & z_n^3
\end{bmatrix} \Rightarrow \hat{y} = Xw \) implies

\[
\hat{y}_i = w_0 + w_1 z_i + w_2 z_i^2 + w_3 z_i^3
\]

= cubic polynomial that fits training samples perfectly!

Matrices with this special structure are called Vandermonde matrices.

Observe \((z_i, y_i)\) for \(i = 1, \ldots, n\), where \(z_i, y_i\) are both scalars and make feature vector \(x_i = \begin{bmatrix} 1 & z_i & z_i^2 & z_i^3 \end{bmatrix}^T\)
We've looked at multiplying matrices by vectors, but to find good $W$ that fit training data, we'll also need to be able to multiply two matrices together. Matrix products are also interesting in their own right.

**Example: Recommender System**

<table>
<thead>
<tr>
<th></th>
<th>Becca</th>
<th>Michael</th>
<th>Bo</th>
<th>Victor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star Wars</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td><em>Pride and Prejudice</em></td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Oppenheimer</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Barbie</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td><em>Exorcist</em></td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

Let's write $X$ as the product of two matrices $U$ and $V$.

$$X \in \mathbb{R}^{n \times p} = UV \in \mathbb{R}^{n \times p}$$
Think of

\[ U = \text{taste profiles of } r \text{ representative customers} \]

\[ V = \text{weights on each representative profile (1 set of weights for each customer)} \]

Example:

\[ U = \begin{bmatrix} 8 & 3 \\ 0 & 10 \\ 10 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 4/10 \\ 6/10 \end{bmatrix} \implies UV = \begin{bmatrix} 4 \\ 6 \\ 5 \end{bmatrix} \]

\[ \text{expected preferences of customer who is 60\% action lover and 40\% romance lover} \]
Matrix $V$ will contain a weight vector column for each customer.

Example:

$$UV = \begin{bmatrix} 8 & 3 \\ 0 & 10 \\ 10 & 0 \\ 3 & 8 \\ 7 & 2 \end{bmatrix} \begin{bmatrix} 6 & 2 & 8 & 4 \end{bmatrix} \frac{1}{10}$$

= product of two matrices.

If $X = UV$, then $X_{ij} = \langle i^{th} \text{ row of } U, j^{th} \text{ col of } V \rangle$
What is a column of $X$?

$X_i = j^{th}$ col of $X = \text{weighted sum of columns of } U$, where $j^{th}$ column of $V$ tells us the weights

$= U v_j = \text{expected tastes of } j^{th} \text{ customer}$

What is a row of $X$?

$x_i = i^{th}$ row of $X = u_i^T V = \text{weighted sum of rows of } V$, where $i^{th}$ row of $U$ tells us the weights

$= \text{how much we expect each customer to like } i^{th} \text{ show}$
Inner product representation:

\[ U V = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_n^T \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_p \end{bmatrix} = \begin{bmatrix} u_1^T V_1 & u_1^T V_2 & \cdots & u_1^T V_p \\ u_2^T V_1 & u_2^T V_2 & \cdots & u_2^T V_p \\ \vdots & \vdots & \ddots & \vdots \\ u_n^T V_1 & u_n^T V_2 & \cdots & u_n^T V_p \end{bmatrix} \]

Outer product representation:

\[ U V = \begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_r \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \\ \vdots \\ V_r^T \end{bmatrix} \]

\[ (U V)_{ij} = \sum_{k=1}^c u_{ik} v_{kj} \]
Given a matrix $X \in \mathbb{R}^{n \times p}$, what is the smallest $r$ such that we can find $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{r \times p}$?

**Example.** Let $X = \begin{bmatrix} 1 & 3 & 4 \\ 2 & 6 & 8 \\ 3 & 9 & 12 \\ 4 & 12 & 16 \end{bmatrix}$

We could write $X = UV$ with $U = \begin{bmatrix} 1 & 3 & 4 \\ 2 & 6 & 8 \\ 3 & 9 & 12 \\ 4 & 12 & 16 \end{bmatrix}$ and $V = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\Rightarrow r = 3$ works.

Can we find $U, V$ with smaller $r$?

Consider: $U = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$ and $V = \begin{bmatrix} 1 & 3 & 4 \end{bmatrix}$

$\Rightarrow$ Smallest $r = 1$!

This smallest value of $r$ is the RANK of the matrix $X$.

In the context of our recommender system example, $r$ is the minimum number of representative taste profiles we need to accurately represent everyone's movie ratings.