Lecture 2: Vectors and Matrices in Machine Learning

Mathematical Foundations of Machine Learning
University of Chicago

Learning we observe training data $(\underline{x}_i, y_i)$ for $i = 1, \ldots, n$

where $\underline{x}_i \in \mathbb{R}^p$ is a vector of p real numbers called the feature

and $y_i \in \mathbb{R}$ or $y_i \in \{-1, +1\}$ or $y_i \in \{0, 1\}$ is the label

Our goal learn a model that predicts a label $\hat{y}$ given a feature vector $\underline{x}_0$.

Ex linear model $\hat{y} = w_1 x_{01} + w_2 x_{02} + \cdots + w_p x_{0p}$

$w_1, \ldots, w_p$ = weights to be learned from data

Let $\underline{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix} \in \mathbb{R}^p$ be the weight vector

Let $\underline{x}_0 = \begin{bmatrix} x_{01} \\ x_{01} \\ \vdots \\ x_{0p} \end{bmatrix} \in \mathbb{R}^p$ be the feature vector

Then our linear model can be equivalently written as

$$\hat{y} = \langle \underline{w}, \underline{x}_0 \rangle = \underline{w}^T \underline{x}_0 = [w_1, \ldots, w_p] \begin{bmatrix} x_{01} \\ x_{01} \\ \vdots \\ x_{0p} \end{bmatrix} = \underline{x}_0^T \underline{w} = \langle x_0, w \rangle$$

Inner product of two vectors

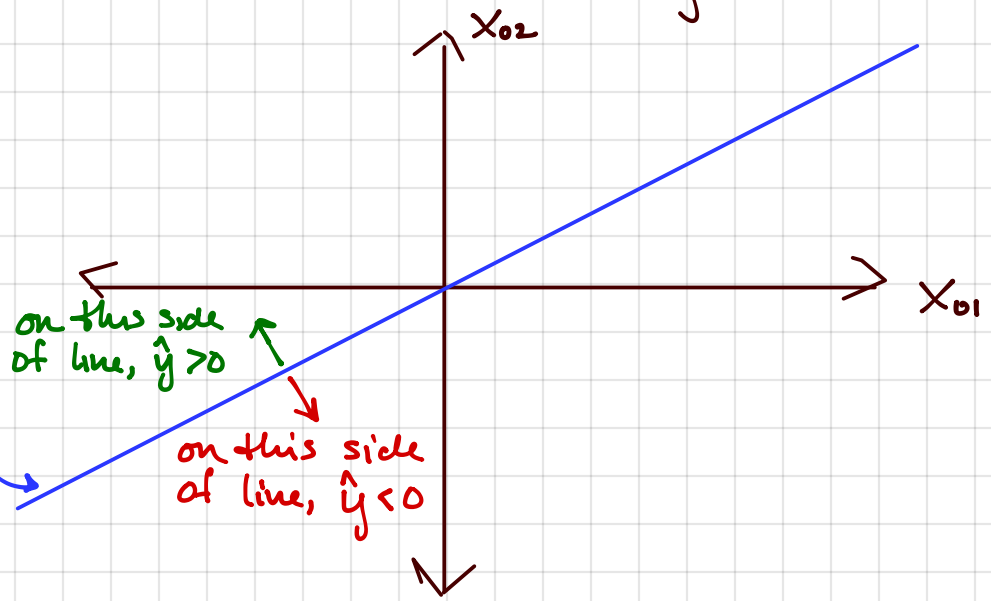
vector transpose

Ex $p=2$ $\underline{w} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ When is $\hat{y} = \langle \underline{w}, \underline{x}_0 \rangle > 0$ and when is $\hat{y} < 0$?

$$\langle \underline{w}, \underline{x}_0 \rangle = w_1 x_{01} + w_2 x_{02} > 0$$

$$\Rightarrow -2x_{01} + x_{02} > 0$$

$$\Rightarrow x_{01} < x_{02}/2$$



Line = set of $\underline{x}_0$ where $\hat{y} = 0$

(we might also consider $\hat{y} = w_0 + w_1 x_{01} + w_2 x_{02} + \cdots + w_p x_{0p}$

this can be done implicitly by letting $\underline{x}_0 = \begin{bmatrix} 1 \\ x_{01} \\ x_{02} \\ \\ x_{0p} \end{bmatrix}$, $\underline{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \\ w_p \end{bmatrix} \in \mathbb{R}^{p+1}$

now our model is $\hat{y} = \langle \underline{w}, \underline{x}_0 \rangle$, same as before!)

We ultimately need to use training data to learn the "best" weight vector. I.e., we want $\hat{y}_i = \langle \underline{w}, \underline{x}_i \rangle \approx y_i$ 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 $\underline{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$ and feature matrix (a design matrix) $X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ & & & \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \in \mathbb{R}^{n \times p}$

↑ real matrix w/ n rows, p columns

x_{ij} = j^{th} feature of i^{th} sample

i^{th} row of X = p features of i^{th} sample = $\underline{x}_i^T$

j^{th} col of X = feature j for all n samples

Then we can write our model as

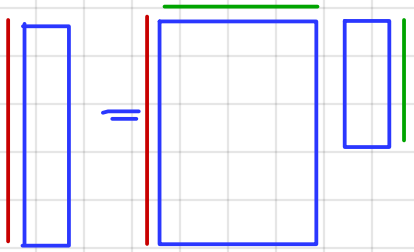
$$\hat{\underline{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = X\underline{w}$$ ⟸ Linear model for all n samples in one equation

Computing $X\underline{w}$ means taking the inner product of each row of X with $\underline{w}$ and storing the results in a vector $\hat{\underline{y}}$

Note that dimensions should always match

$$\hat{y} = X\underline{w}, \quad \hat{y} \in \mathbb{R}^n, \quad \underline{w} \in \mathbb{R}^p, \quad X \in \mathbb{R}^{n \times p}$$

- # rows of X = length of $\hat{y}$
- # cols of X = length of $\underline{w}$



Example

$$X = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 0 & 3 \end{bmatrix}$$

2 features
3 training samples

$$\underline{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \underline{x}_2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \underline{x}_3 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

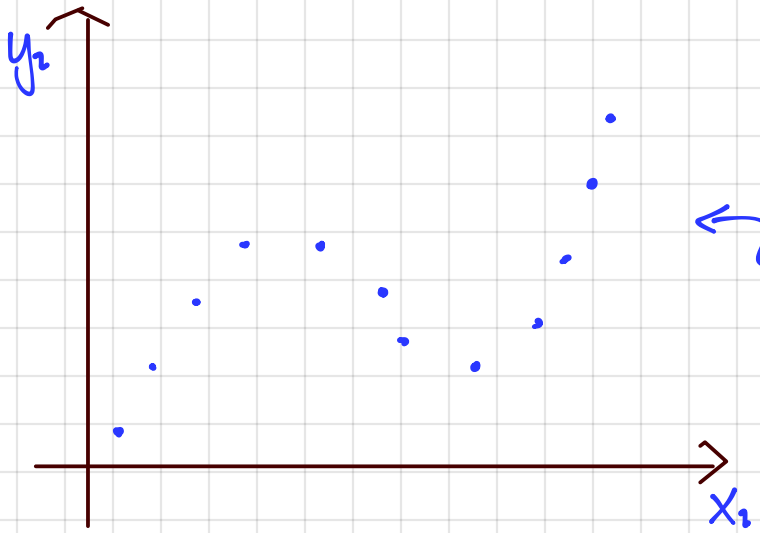
$$\underline{w} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}$$

$$X\underline{w} = \begin{bmatrix} 2\,1+4\,0 \\ 2\,2+4\,0 \\ 2\,0+4\,3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}$$

Another perspective $X\underline{w}$ is a weighted sum of the columns of X, where $\underline{w}$ gives the weights

$$X\underline{w} = 2\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + 4\begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 12 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix}$$

Example



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

Let $X = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & & \ddots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix} \Rightarrow \hat{y} = X\underline{w}$ implies

$\hat{y}_1 = w_1 1 + w_2 x_1 + w_3 x_1^2 + w_4 x_1^3$

= cubic polynomial that fits training samples perfectly!

matrices with this special structure are called Vandermonde matrices

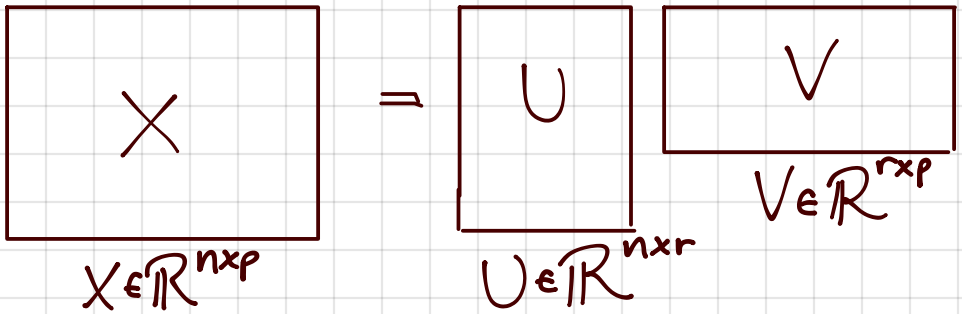
We've looked at multiplying matrices by vectors, but to find good $\underline{W}$ that fit training data, we'll also need to be able to multiply two matrices together Matrix products are also interesting in their own right

Example Recommender system

	Becca	Nathan	Zeus	Chris
Star Wars	10	10	8	5
Pride+Prejudice	10	2	1	10
La la Land	8	3	1	10
Bird Box	1	7	10	2
Marvelous Ms Maisel	1	9	10	4

$= X$

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



Think of

U = taste profiles of r representative customers

V = weights on each representative profile (1 set of weights for each customer)

Example

$$U = \begin{bmatrix} 10 & 4 \\ 1 & 10 \\ 1 & 10 \\ 8 & 1 \\ 10 & 1 \end{bmatrix}, \quad \underline{v} = \begin{bmatrix} 3/8 \\ 5/8 \end{bmatrix} \Longrightarrow U\underline{v} = \begin{bmatrix} 6 & 2 \\ 6 & 6 \\ 6 & 6 \\ 3 & 6 \\ 4 & 3 \end{bmatrix}$$

how much action movie lover likes each show

how much romance movie lover likes each show

expected preferences of customer who is 3/8 action lover and 5/8 romance lover

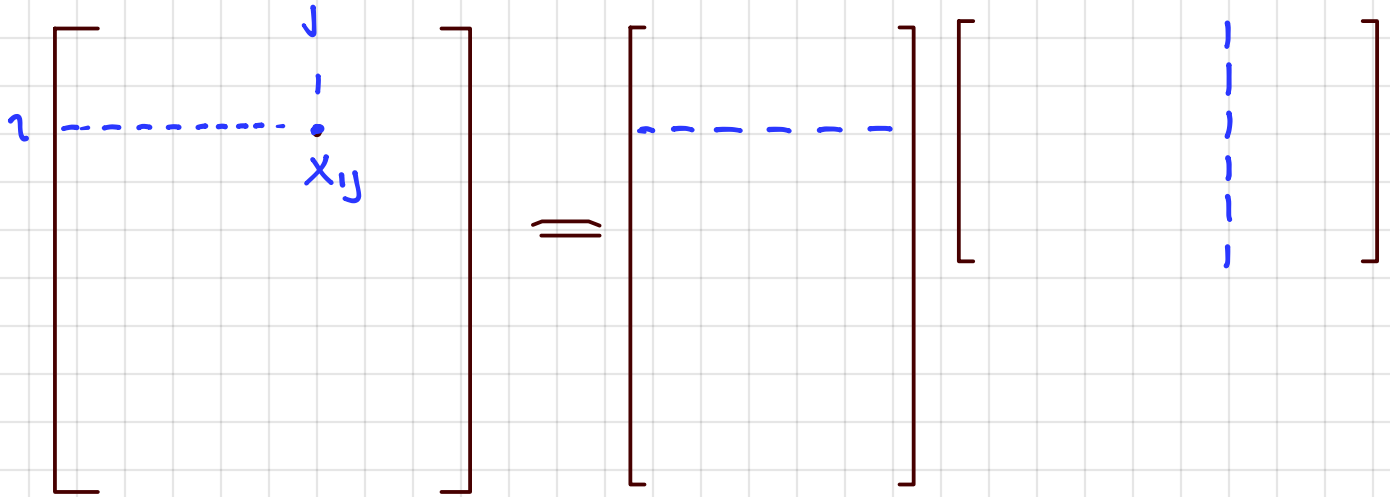
Matrix V will contain a weight vector column for each customer

Example:

$$UV = \begin{bmatrix} 10 & 4 \\ 1 & 10 \\ 1 & 10 \\ 8 & 1 \\ 10 & 1 \end{bmatrix} \begin{bmatrix} 7 & 1 & 8 & 3 \\ 3 & 9 & 2 & 7 \end{bmatrix} / 10$$

= 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?

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

  $= U\underline{v}_j$ = expected tastes of j^{th} customer

- What is a row of X?

  i^{th} row of $X = \underline{u}_i^T V$ = weighted sum of rows of V, where i^{th} row of U tells us the weights

  = how much we expect each customer to like i^{th} show

Inner product representation

$$UV = \begin{bmatrix} \underline{u}_1^T \\ \underline{u}_2^T \\ \vdots \\ \underline{u}_n^T \end{bmatrix} \begin{bmatrix} | & | & & | \\ v_1 & v_2 & & v_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \underline{u}_1^T \underline{v}_1 & \underline{u}_1^T \underline{v}_2 & & \underline{u}_1^T \underline{v}_p \\ & & & \\ \underline{u}_n^T \underline{v}_1 & \underline{u}_n^T \underline{v}_2 & & \underline{u}_n^T \underline{v}_p \end{bmatrix}$$

Outer product representation

$$UV = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & & u_r \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & \underline{v}_1^T & - \\ - & \underline{v}_2^T & - \\ & \vdots & \\ - & \underline{v}_r^T & - \end{bmatrix}$$

$$= u_1 \underline{v}_1^T + u_2 \underline{v}_2^T + \cdots + u_r \underline{v}_r^T$$

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.