Linear Models

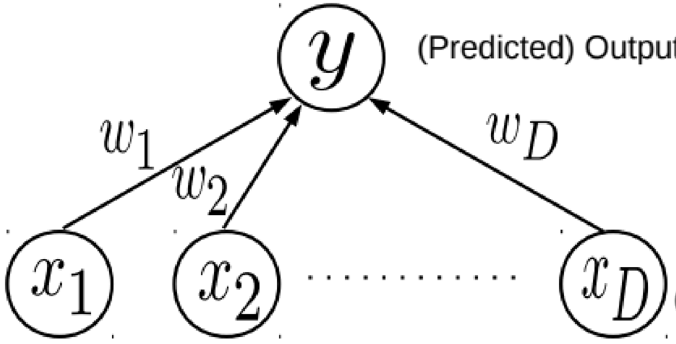
CS771: Introduction to Machine Learning

Piyush Rai

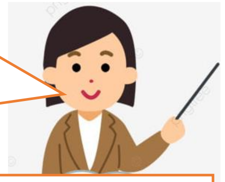
Linear Models

- Consider learning to map an input $\boldsymbol{x} \in \mathbb{R}^D$ to the corresponding (say real-valued) output $\boldsymbol{y}$
- Assume the output to be a linear weighted combination of the $\boldsymbol{D}$ input features

$$y = \sum_{d=1}^{D} w_d x_d = \boldsymbol{w}^\top \boldsymbol{x}$$



This defines a linear model with $\boldsymbol{D}$ parameters given by a "weight vector" $\boldsymbol{w} = [w_1, w_2, \ldots, w_D]$



Each of these weights have a simple interpretation: $\boldsymbol{w_d}$ is the "weight" or importance of the $\boldsymbol{d^{th}}$ feature in making this prediction

The "optimal" weights are unknown and have to be learned by solving an optimization problem, using some **training data**

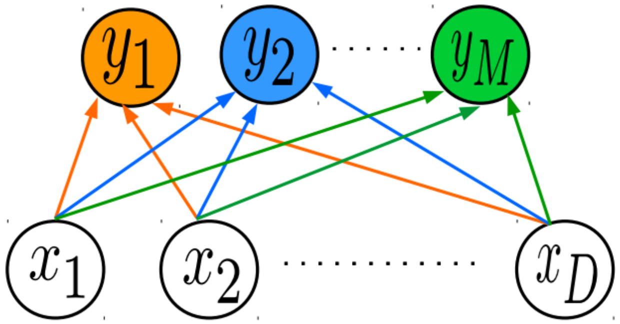
- This simple model can be used for Linear Regression
- This simple model can also be used as a "building block" for more complex models
  - Even classification (binary/multiclass/multi-output/multi-label) and various other ML/deep learning models
  - Even unsupervised learning problems (e.g., dimensionality reduction models)

Simple Linear Models as Building Blocks

- In some regression problems, each output itself is a real-valued vector $\boldsymbol{y} \in \mathbb{R}^M$
  - Example: Given a full body image of a person, predict height, weight, hand size, and leg size $(M = 4)$

- Such problems are commonly known as multi-output regression

- We can assume a separate linear model for each of the M outputs $y_m \ (m = 1,2, \ldots, M)$

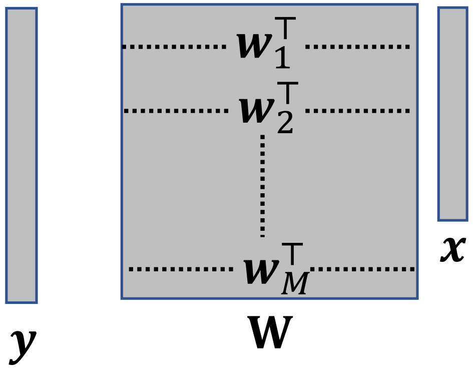


$$y_m = \boldsymbol{w}_m^\top \boldsymbol{x}$$

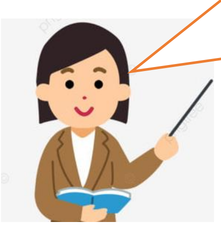
Now each $\boldsymbol{w}_m \in \mathbb{R}^D$ is a D-dim weight vector for predicting the m^{th} output

$$\boldsymbol{y} = \mathbf{W}\boldsymbol{x}$$

Here $\mathbf{W}$ is an *MxD* weight matrix with its m^{th} row containing $\boldsymbol{w}_m$



Learning this model will require us to learn this weight matrix (or equivalently, the M weight vectors)

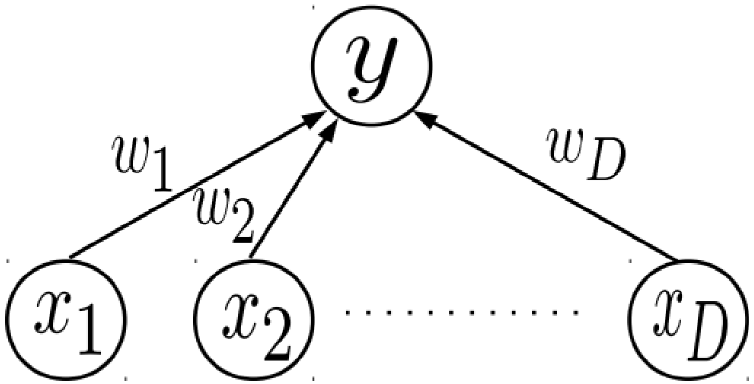


Note: Learning M separate models may not be ideal these multiple outputs are somewhat correlated with each other. But this model can be extended to handle such situation (techniques are a bit advanced to be discussed right now – but if curious, you may look up more about multitask learning techniques)

Simple Linear Models as Building Blocks

- A linear model $\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{x}$ can also be used in classification problems
- For binary classfn, can treat $\boldsymbol{w}^\top \boldsymbol{x}$ as the "score" of input $\boldsymbol{x}$ and threshold to get binary label

$$y = +1 \quad \text{if} \quad \boldsymbol{w}^\top \boldsymbol{x} \geq 0$$
$$y = -1 \quad \text{if} \quad \boldsymbol{w}^\top \boldsymbol{x} < 0$$
$$y = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x})$$



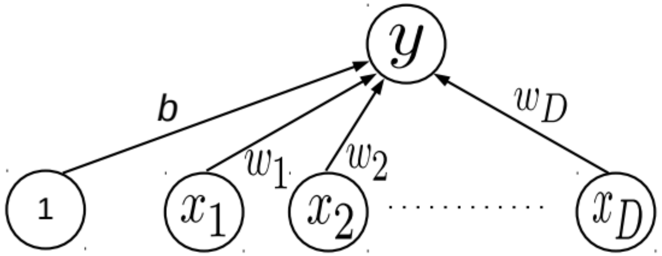
Recall that the LwP model can also be seen as a linear model (although it wasn't formulated like this)

Don't worry. Can easily fold-in the bias term $\boldsymbol{b}$ here as shown in the figure below

Wait – when discussing LwP, wasn't the linear model of the form $\boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b}$? Where did the "bias" term $\boldsymbol{b}$ go?

Can append a constant feature "1" for each input and rewrite as $\boldsymbol{y} = \boldsymbol{w}^\top \boldsymbol{x}$ where now both $\boldsymbol{x}$ and $\boldsymbol{w} \in \mathbb{R}^{D+1}$

We will assume the same and omit the explicit bias for simplicity of notation



$$y = \sum_{d=1}^{D} w_d x_d + b = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Simple Linear Models as Building Blocks

- Linear models are also used in multiclass classification problems

- Assuming K classes, we can assume the following model

$$y = \operatorname{argmax}_{k \in \{1,2,\ldots,K\}} \boldsymbol{w}_k^\top \boldsymbol{x}$$

- Can think of $\boldsymbol{w}_k^\top \boldsymbol{x}$ as the score of the input for the k^{th} class

- Once learned (using some optimization technique), these K weight vectors (one for each class) can sometimes have nice interpretations, especially when the inputs are images

The learned weight vectors of each of the 4 classes visualized as images – they kind of look like a "template" of what the images from that class should look like

$\boldsymbol{w}_{car}$ $\boldsymbol{w}_{frog}$ $\boldsymbol{w}_{horse}$ $\boldsymbol{w}_{cat}$

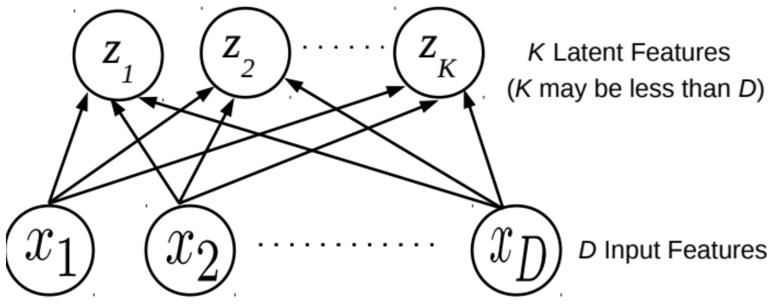
That's why the dot product of each of these weight vectors with an image from the correct class will be expected to be the largest

These images sort of look like class prototypes if I were using LwP ☺

Yeah, "sort of". ☺
No wonder why LwP (with Euclidean distances) acts like a linear model. ☺

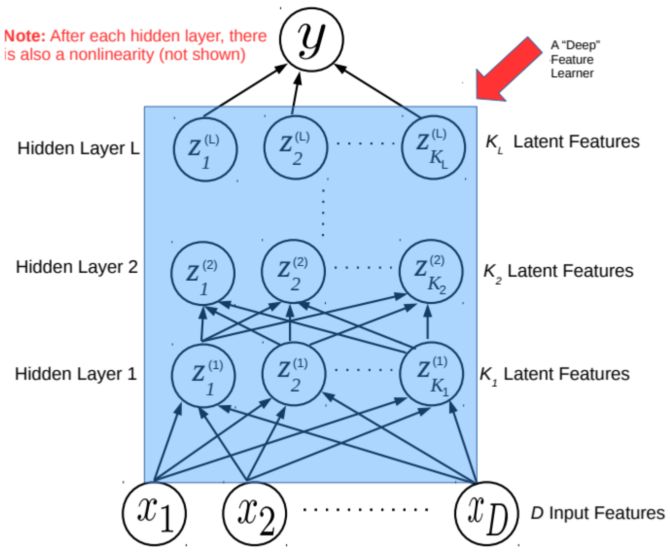
Simple Linear Models as Building Blocks

- Linear models are building blocks for dimensionality reduction methods like PCA



This looks very similar to the multi-output model, except that the values of the K latent features are not known and have to be learned

- Linear models are building blocks for even deep learning model (each layer is like a multi-output linear model, followed by a nonlinearity)

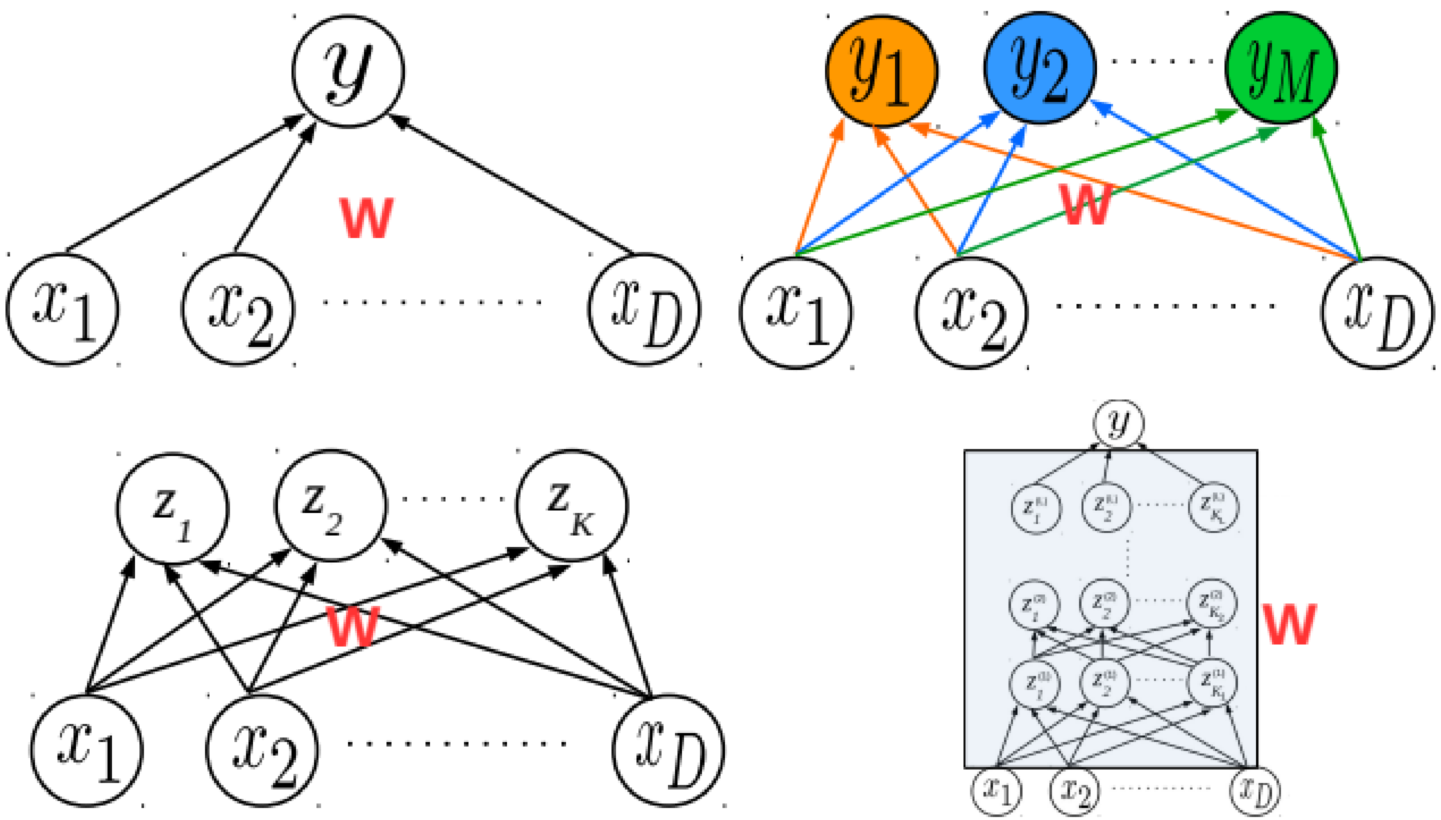


In a deep learning model, each layer learns a latent feature representation of the inputs using a model like a multi-output linear model, followed by a nonlinearity

The last (output) layer can have one or more outputs

More on this when we discuss deep learning later

Learning Linear Models



Linear Models are ubiquitous!
How do we <u>learn</u> them from data?

For linear models, learning = Learning the model parameters (the weights)

We will formulate learning as an optimization problem w.r.t. these parameters

Next Lecture

- Linear Regression