Implementing ML Models

Duke MLSS

Alex Lew

Roadmap

- Framing the problem
- Math review: linear algebra basics
- Demo: Manipulating matrices in Python
- Loss functions and optimization
- The computational graph
- Demo: Gradient descent with TensorFlow

What is a model?

$$y = f(x)$$

A model is a **program** we write to make **predictions** about **data**.

A model is an **approximation** of a true function relating **input** and **output**.

A model is a **story** we tell about the **data**-generating process.

What is a model?

$$\textbf{price} = f(\textbf{square footage})$$

A model is a **program** we write to make **predictions** about **data**.

$$y = mx + b$$

1. Begin with some base price $b.
2. Add $m for each square foot.

What is a model?

$$\text{price} = f(\text{square footage})$$

A model is an **approximation** of a true function relating **input** and **output**.

$$y = mx + b$$

We approximate the true function from **square footage** to **price** with a **simple line**: $f(x) = mx + b$.

What is a model?

$$\textbf{price} = f(\textbf{square footage})$$

A model is a **story** we tell about the **data**-generating process.

$$\boldsymbol{y} = \boldsymbol{m}\boldsymbol{x} + \boldsymbol{b}$$

A house is built with some square footage $\boldsymbol{x}$. To **price** it, the developer begins with a base price $$\boldsymbol{b}$, and adds $$\boldsymbol{m}$ per square foot.

Deep learning

A model is an **approximation** of a true function relating **input** and **output**.

$f($  $) =$ **dog**

$f($  $) =$ **seven**

Parameters

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$$

1. Begin with some base price $\$\boldsymbol{\theta_1}$.
2. Add $\$\boldsymbol{\theta_2}$ for each square foot.

like a *template* or *mad libs* model: we leave some *parameters* unspecified

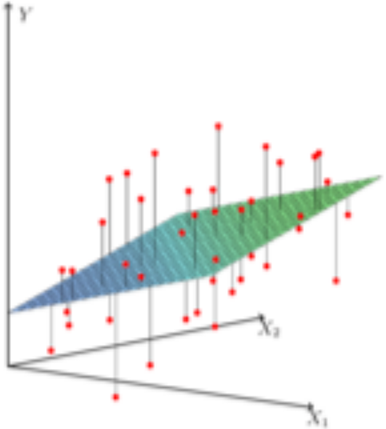
$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\theta_1 x} + \boldsymbol{\theta_2}$$

Multivariate input

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$$

To predict **watch time**:

1. Begin with some base time θ_0
2. Add θ_1 for each like the video has (**x_1**)
3. Add θ_2 for each dislike the video has (**x_2**)
4. Add θ_3 for each minute in the video (**x_3**)

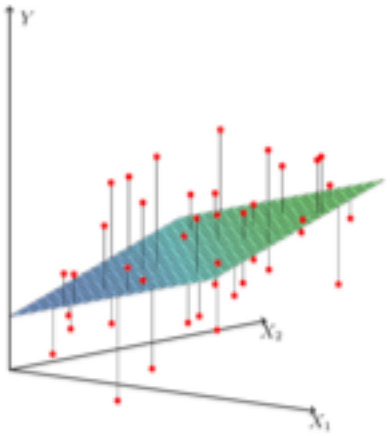


$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}_3 \boldsymbol{x}_3 + \boldsymbol{\theta}_2 \boldsymbol{x}_2 + \boldsymbol{\theta}_1 \boldsymbol{x}_1 + \boldsymbol{\theta}_0$$

Multivariate input: dot product

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$$

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}_3 \boldsymbol{x}_3 + \boldsymbol{\theta}_2 \boldsymbol{x}_2 + \boldsymbol{\theta}_1 \boldsymbol{x}_1 + \boldsymbol{\theta}_0$$



$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

Interpretations of the dot product

Dimensional Analysis

$$\begin{pmatrix} 5 \\ 7 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 3 \\ 4 \\ 2 \end{pmatrix} \quad \begin{array}{l} \textit{apples} \\ \textit{oranges} \\ \textit{bananas} \end{array}$$

lbs of fruit *$ per lb*

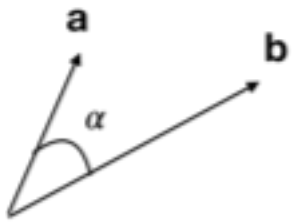
$$5\times 3 + 7\times 4 + 3\times 2$$
$$=$$
$$\$49$$

Total $

Similarity / Closeness

$$\begin{array}{l} \textit{water} \\ \textit{sugar} \\ \textit{fruit} \end{array} \begin{pmatrix} .8 \\ .6 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} .7 \\ .5 \\ .5 \end{pmatrix} = 0.86$$

simple syrup *Gatorade*



|a| |b| cos(α)

$$\arccos(0.86) = \mathbf{30°}$$

Interpretations of the dot product

Dimensional Analysis: Predicting Total Life Expectancy

$$\begin{array}{r} \textit{Height (ft)} \\ \textit{Weight (lbs)} \\ \textit{Birthdate (years since 1950)} \end{array} \begin{pmatrix} 6 \\ 180 \\ 30 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -0.1 \\ 0.3 \end{pmatrix} \begin{array}{l} \textit{Years per ft.} \\ \textit{Years per lb.} \\ \textit{Years per year} \end{array} + 80 \text{ years}$$

$$= \mathbf{77\ years}$$

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

Interpretations of the dot product

Similarity / Closeness: Classifying Images

 $\cdot$  $+ \mathbf{-500} = 12{,}500$

(more than 0,
so: a dog!)

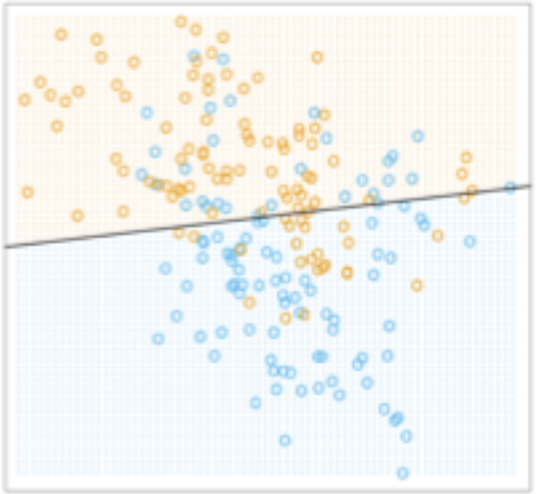
$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w},b) = \boldsymbol{w}\cdot\boldsymbol{x}+b$$

learn a template and threshold

Interpretations of the dot product

Classification: Another View

Linear Regression:
y = mx + b gives a *line* that is uniformly *near* the data.



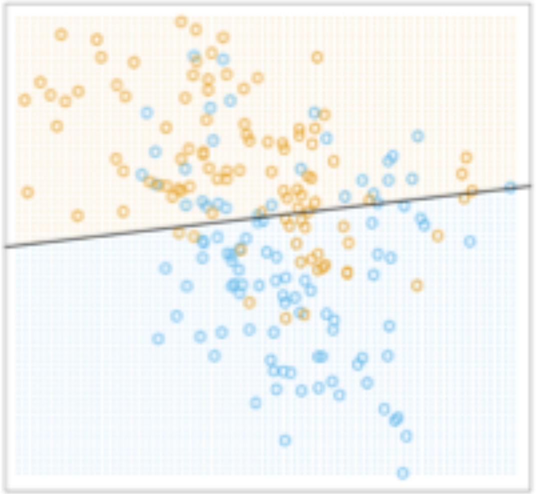
Linear Classification:
y = mx + b gives a *line* that separates the data into classes.

Interpretations of the dot product

Classification: Another View

Linear Regression:
y = mx + b gives a *line* that is uniformly *near* the data.

x2 = mx1 + b

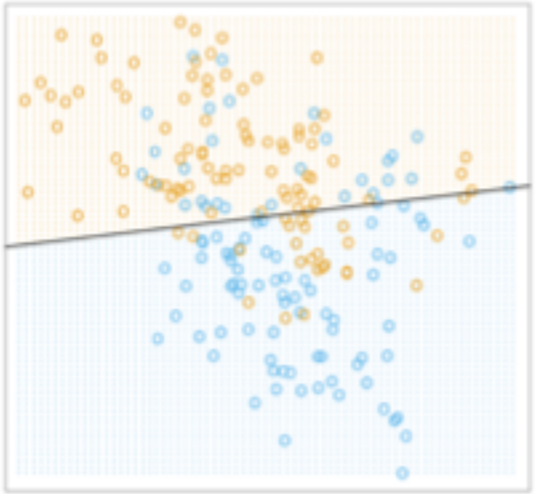


Linear Classification:
x2 = mx1 + b gives a *line* that separates the data into classes.

Interpretations of the dot product

Classification: Another View

$0 = m_1x_1 + m_2x_2 + b$



Linear Regression:

$y = mx + b$ gives a *line* that is uniformly *near* the data.

Linear Classification:

$0 = m_1x_1 + m_2x_2 + b$ gives a *line* that separates the data into classes.

Interpretations of the dot product

Classification: Another View

$3 = m1x1 + m2x2 + b$

$2 = m1x1 + m2x2 + b$

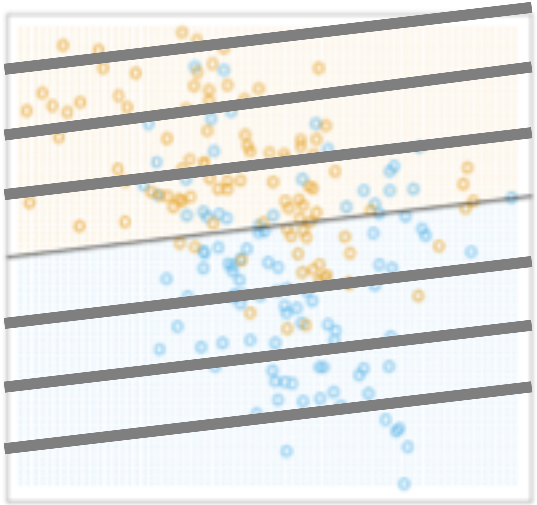
$1 = m1x1 + m2x2 + b$

$0 = m1x1 + m2x2 + b$

$-1 = m1x1 + m2x2 + b$

$-2 = m1x1 + m2x2 + b$

$-3 = m1x1 + m2x2 + b$



Linear Regression:

y = mx + b gives a *line* that is uniformly *near* the data.

Linear Classification:

m1x1 + m2x2 + b gives a *score* that measures how far into the "orange" area a point is.

Interpretations of the dot product

Classification: Another View

$3 = w \cdot x + b$

$2 = w \cdot x + b$

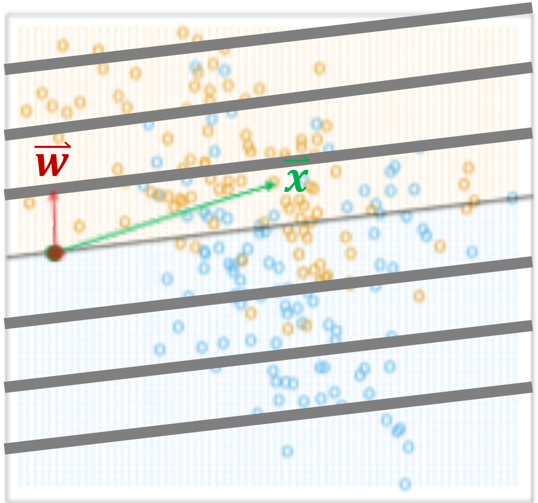
$1 = w \cdot x + b$

$0 = w \cdot x + b$

$-1 = w \cdot x + b$

$-2 = w \cdot x + b$

$-3 = w \cdot x + b$



Linear Regression:

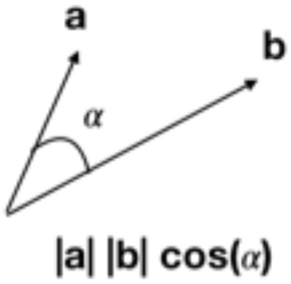
y = mx + b gives a *line* that is uniformly *near* the data.

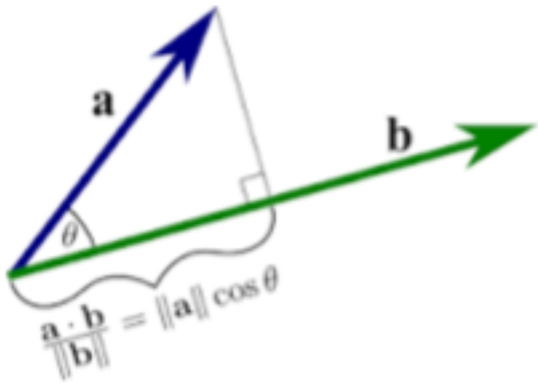
Linear Classification:

$w \cdot x + b$ gives a *score* that measures how far into the "orange" area a point is.

Interpretations of the dot product

Projection





If b is a unit vector, a•b gives the projection of a onto b.

Multivariate output

$$y = f(x; \theta)$$

To predict **life expectancy (y1)**:
Begin with some base life expectancy $\theta_{1,0}$
Add $\theta_{1,1}$ for each year born after 1950 (x_1)
Add $\theta_{1,2}$ for each pound the person weighs (x_2)
Add $\theta_{1,3}$ for each inch tall the person is (x_3)

To predict **current income level (y2)**:
Begin with some base income $\theta_{2,0}$
Add $\theta_{2,1}$ for each year born after 1950 (x_1)
Add $\theta_{2,2}$ for each pound the person weighs (x_2)
Add $\theta_{2,3}$ for each inch tall the person is (x_3)

$$f(x; \theta) = \begin{cases} \theta_{13}x_3 + \theta_{12}x_2 + \theta_{11}x_1 + \theta_{10} \\ \theta_{23}x_3 + \theta_{22}x_2 + \theta_{21}x_1 + \theta_{20} \end{cases}$$

Multivariate output: matrices

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta})$$

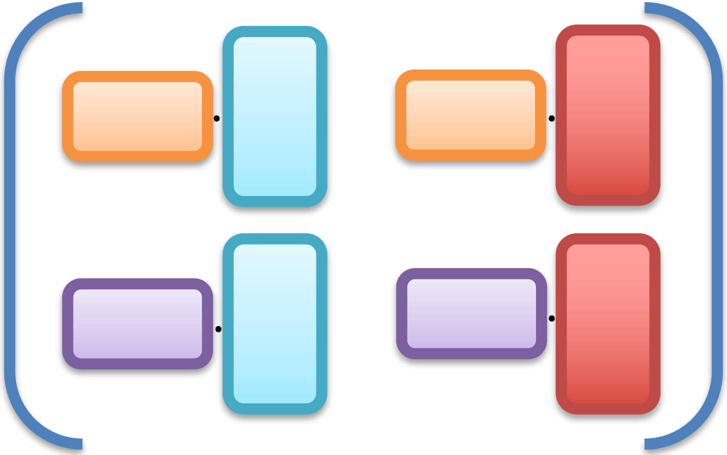
$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \begin{cases} \boldsymbol{\theta}_{13}\boldsymbol{x}_3 + \boldsymbol{\theta}_{12}\boldsymbol{x}_2 + \boldsymbol{\theta}_{11}\boldsymbol{x}_1 + \boldsymbol{\theta}_{10} \\ \boldsymbol{\theta}_{23}\boldsymbol{x}_3 + \boldsymbol{\theta}_{22}\boldsymbol{x}_2 + \boldsymbol{\theta}_{21}\boldsymbol{x}_1 + \boldsymbol{\theta}_{20} \end{cases}$$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{\theta}11 & \boldsymbol{\theta}12 & \boldsymbol{\theta}13 \\ \boldsymbol{\theta}21 & \boldsymbol{\theta}22 & \boldsymbol{\theta}23 \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}1 \\ \boldsymbol{x}2 \\ \boldsymbol{x}3 \end{bmatrix} \qquad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{\theta}10 \\ \boldsymbol{\theta}20 \end{bmatrix}$$

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

Matrix multiplication

$$\underbrace{\begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}}_{X} =$$



WX

$$f(x; W, b) = Wx + b$$

dot the *rows* of W with the *cols* of X

Matrix mult.: dimensional analysis

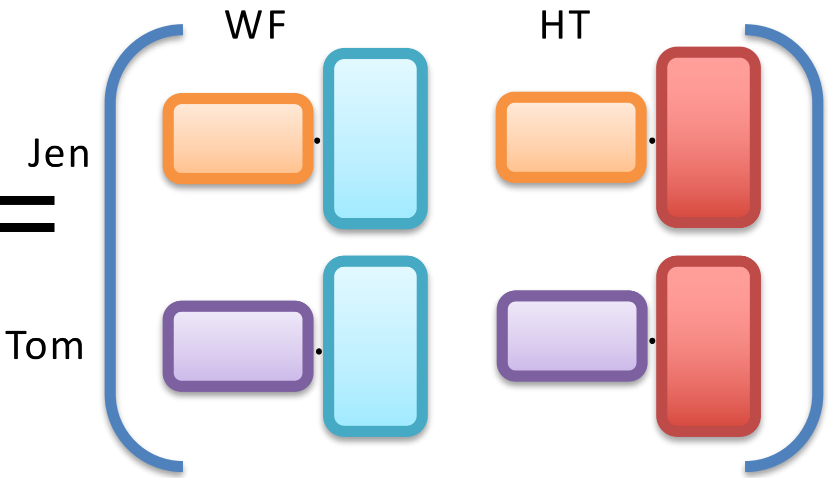
$$\begin{array}{cc} & \begin{array}{ccc}\text{Figs} & \text{Nuts} & \text{Salt}\end{array} \\ \begin{array}{c}\text{Jen}\\ \text{Tom}\end{array} & \begin{bmatrix}3 & 5 & 2\\ 0 & 2 & 10\end{bmatrix}\end{array}$$

$\boldsymbol{W}$

$$\begin{array}{cc} & \begin{array}{cc}\text{WF} & \text{HT}\end{array} \\ \begin{array}{c}\text{Figs}\\ \text{Nuts}\\ \text{Salt}\end{array} & \begin{bmatrix}\$2 & \$3\\ \$3 & \$3\\ \$5 & \$4\end{bmatrix}\end{array}$$

$\boldsymbol{X}$

$=$



$\boldsymbol{W}\boldsymbol{X}$

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{W},\boldsymbol{b}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

dot the *rows* of W with the *cols* of X

In multivariate linear models…

$$\begin{array}{r|ccc} & \text{Age} & \text{Ht} & \text{Wt} \\ \hline \text{Vegan} & & \begin{bmatrix} -3 & 0 & -1 \\ 2 & 1 & 1 \end{bmatrix} & \\ \text{Omnivore} & & & \end{array}$$

$$\begin{array}{r} \\ \text{Vegan} \\ \text{Omnivore} \end{array} \begin{array}{c} \begin{array}{ccc} \text{Age} & \text{Ht} & \text{Wt} \end{array} \\ \begin{bmatrix} -3 & 0 & -1 \\ 2 & 1 & 1 \end{bmatrix} \end{array}$$

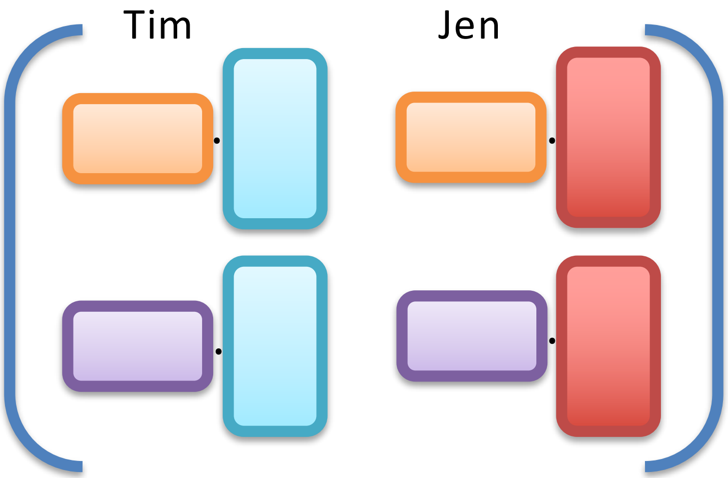
$\boldsymbol{W}$

(rows: coefficients for some classification class)

$$\begin{array}{r} \\ \text{Age} \\ \text{Ht.} \\ \text{Wt.} \end{array} \begin{array}{c} \begin{array}{cc} \text{Tim} & \text{Jen} \end{array} \\ \begin{bmatrix} 32 & 25 \\ 70 & 65 \\ 65 & 43 \end{bmatrix} \end{array} =$$

$\boldsymbol{X}$

(cols: data points)



$\boldsymbol{WX}$

The *other dimension* represents the *features*, and must match across matrices.

In multivariate linear models…

$$\begin{array}{c|ccc} & \text{Age} & \text{Ht} & \text{Wt} \\ \hline \text{Vegan} & & & \\ \text{Omnivore} & & & \end{array} \quad \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \end{bmatrix}$$

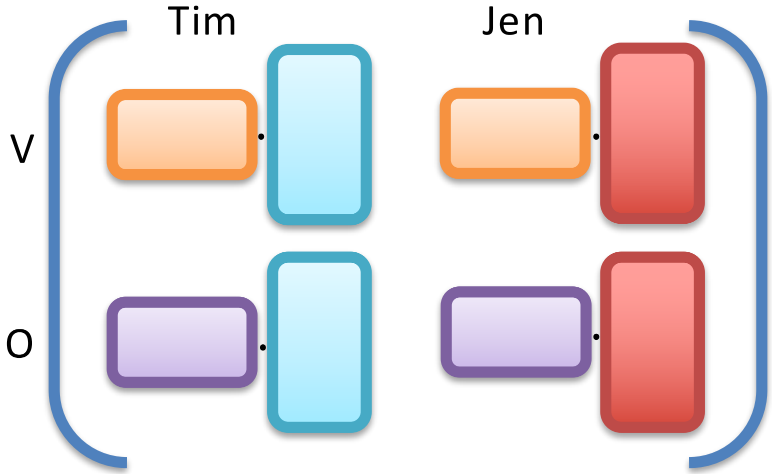
$\boldsymbol{W}$

(rows: coefficients for some classification class)

$$\begin{array}{c} \\ \text{Age} \\ \text{Ht.} \\ \text{Wt.} \end{array} \begin{array}{cc} \text{Tim} & \text{Jen} \\ \left[\begin{matrix} 32 & 25 \\ 70 & 65 \\ 65 & 43 \end{matrix}\right] & \end{array} =$$

$\boldsymbol{X}$

(cols: data points)



$\boldsymbol{W}\boldsymbol{X}$

The *other dimension* represents the *features*, and must match across matrices.

Let's code!
Python and Numpy

Loss functions

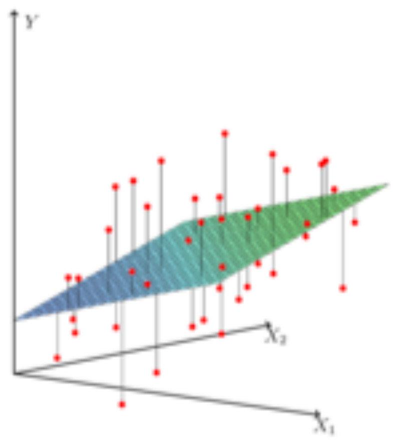
Measure of how *badly* a model fits the data.

$$\boldsymbol{l}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y})$$

We choose model **parameters** to *minimize* loss.

Loss functions

Example: Squared Error



$$f(x;\theta) = \theta_2 x_2 + \theta_1 x_1 + \theta_0$$

$$l(\theta; x, y) = (f(x;\theta) - y)^2$$

We choose model **parameters** to *minimize* total loss.

$$\sum_{i=1}^{N} (f(x;\theta) - y)^2$$

Loss functions

In classification:

penalize for *wrongness* (especially when confident)

reward for *rightness* (especially when confident)

3 = m1x1 + m2x2 + b

2 = m1x1 + m2x2 + b

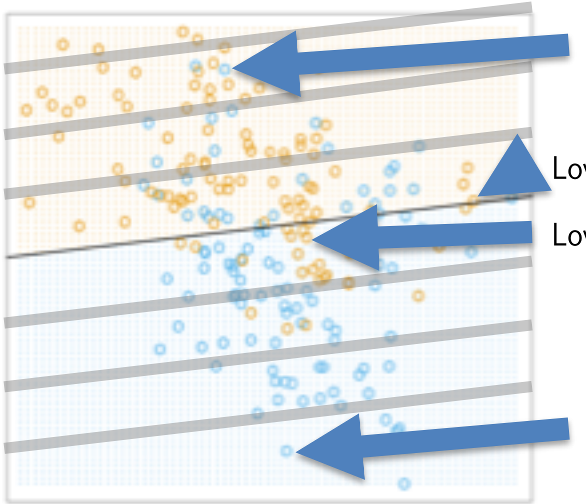
1 = m1x1 + m2x2 + b

0 = m1x1 + m2x2 + b

-1 = m1x1 + m2x2 + b

-2 = m1x1 + m2x2 + b

-3 = m1x1 + m2x2 + b

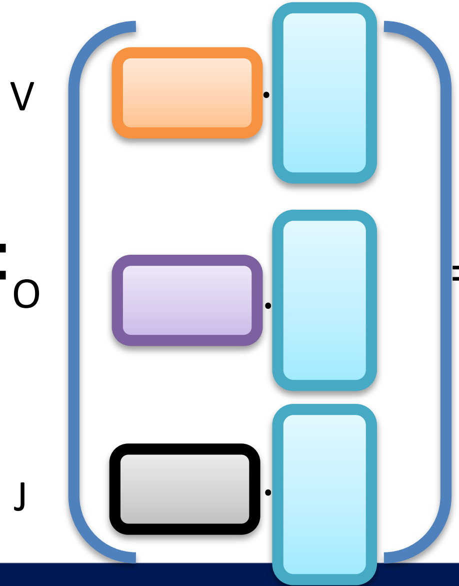


Logistic Regression with Softmax Loss

Step 1 (model): Calculate scores for each class

$$f(x; W, b) = Wx + b$$

$$\begin{array}{r c} & \begin{array}{ccc} \text{Age} & \text{Ht} & \text{Wt} \end{array} \\ \begin{array}{r} \text{Vegan} \\ \text{Omnivore} \\ \text{Juice only} \end{array} & \begin{bmatrix} -.3 & 0 & -.1 \\ .2 & .1 & .1 \\ -.1 & 0 & -.5 \end{bmatrix} \end{array} \begin{array}{r c} \begin{array}{r} \text{Age} \\ \text{Ht.} \\ \text{Wt.} \end{array} & \begin{bmatrix} -10 \\ 3 \\ -3 \end{bmatrix} \end{array} + b =$$



$$= \begin{bmatrix} 3.3 \\ -2.0 \\ 2.5 \end{bmatrix}$$

W

(rows: coefficients for some classification class)

x

Logistic Regression with Softmax Loss

Step 2 (model): Exponentiate so that all scores are positive

Step 3 (model): Divide by sum of scores to get probabilities

$$\begin{array}{cccccc} & \text{Raw} & & \text{Exponentiated} & & \text{Normalized} \\ \begin{array}{r} \text{Vegan} \\ \text{Omnivore} \\ \text{Juice only} \end{array} & \begin{bmatrix} 3.3 \\ -2.0 \\ 2.5 \end{bmatrix} & \rightarrow & \begin{bmatrix} 27.1 \\ 0.13 \\ 12.2 \end{bmatrix} & \rightarrow & \begin{bmatrix} 0.687 \\ 0.003 \\ 0.310 \end{bmatrix} \\ & & & \downarrow & & \\ & & & \text{Sum is } 39.43 & & \end{array}$$

Logistic Regression with Softmax Loss

Designing the loss: we want to reward the model for assigning *high probability* to the training dataset.

$$\begin{array}{rccc} & \text{Raw} & \text{Exponentiated} & \text{Normalized} \\ \begin{array}{r}\text{Vegan}\\ \text{Omnivore}\\ \text{Juice only}\end{array} & \begin{bmatrix}3.3\\-2.0\\2.5\end{bmatrix} \rightarrow & \begin{bmatrix}27.1\\0.13\\12.2\end{bmatrix} \rightarrow & \begin{bmatrix}0.687\\0.003\\0.310\end{bmatrix}\end{array}$$

Sum is 39.43

Reward if Tom is vegan: 0.687

Reward if Tom is an omnivore: 0.003

Reward if Tom is on a juice cleanse: 0.310

Logistic Regression with Softmax Loss

We could actually calculate *the probability the model assigns the dataset* by multiplying the probabilities it assigns to each example.

$$p_{y_i}^{(i)} = \frac{e^{(Wx_i+b)_{y_i}}}{\sum_{c=1}^{C} e^{(Wx_i+b)_c}}$$

$$\text{prob of all data} = \prod_{i=1}^{N} p_{y_i}^{(i)}$$

Logistic Regression with Softmax Loss

But products of *thousands* of small numbers are hard for computers.
So we take a **log**, which converts the product into a sum.

$$p_{y_i}^{(i)} = \frac{e^{(Wx_i+b)_{y_i}}}{\sum_{c=1}^{C} e^{(Wx_i+b)_c}}$$

$$\text{Log prob of all data} = \sum_{i=1}^{N} \log p_{y_i}^{(i)}$$

Logistic Regression with Softmax Loss

The model does *badly* when log probability is *low*.
As a loss, we can use the *negative* log probability.

$$p_{y_i}^{(i)} = \frac{e^{(Wx_i+b)_{y_i}}}{\sum_{c=1}^{C} e^{(Wx_i+b)_c}}$$

$$\mathrm{Loss} = -\sum_{i=1}^{N} \log p_{y_i}^{(i)}$$

Logistic Regression with Softmax Loss

Step 4 (loss): Take the negative log of the probability that the model assigns to the *real* class.

$$\begin{array}{rcccccc} & \text{Raw} & & \text{Exponentiated} & & \text{Normalized} & \\ \begin{array}{r}\text{Vegan}\\ \text{Omnivore}\\ \text{Juice only}\end{array} & \begin{bmatrix} 3.3 \\ -2.0 \\ 2.5 \end{bmatrix} & \rightarrow & \begin{bmatrix} 27.1 \\ 0.13 \\ 12.2 \end{bmatrix} & \rightarrow & \begin{bmatrix} 0.687 \\ 0.003 \\ 0.310 \end{bmatrix} & \end{array}$$

Sum is 39.43

Loss if Tom is vegan: 0.163

Loss if Tom is an omnivore: 2.52

Loss if Tom is on a juice cleanse: 0.51

Logistic Regression with Softmax Loss

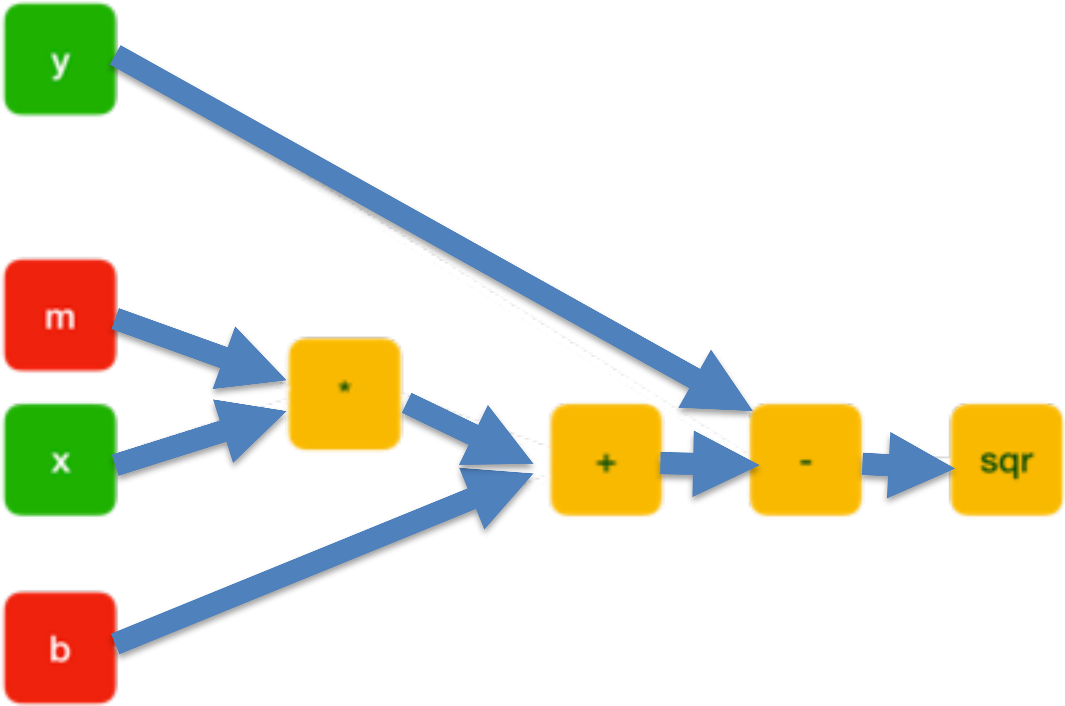
Step 5 (loss): Average losses of all data points to get full model loss

$$l(W, b; X, Y) = -\frac{1}{N}\sum_{i=1}^{N} \log \underbrace{\frac{e^{(Wx_i+b)_{y_i}}}{\sum_{c=1}^{C} e^{(Wx_i+b)_c}}}_{\text{probability assigned to the real class of data point } (x_i, y_i)}$$

Learning with Gradient Descent

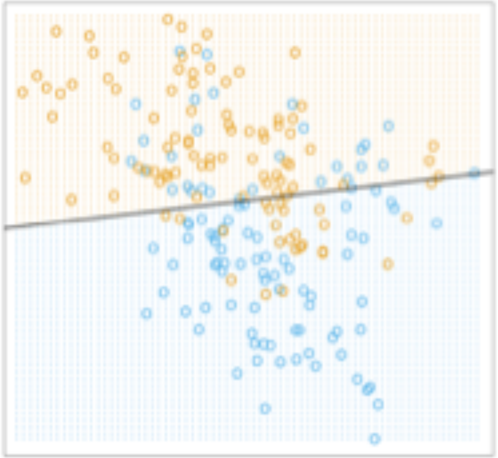
$$\theta_j^{(i)} = \theta_j^{(i-1)} - \alpha \frac{\partial \mathcal{L}}{\partial \theta_j}$$

Computational Graphs

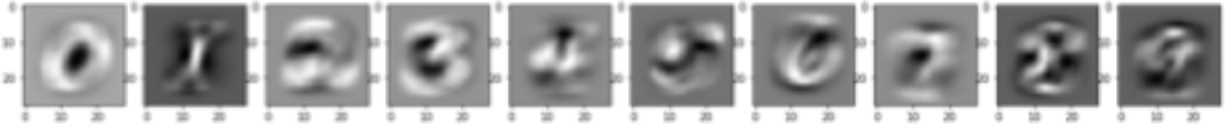


Let's code!
Using TensorFlow

Limitations of logistic classification

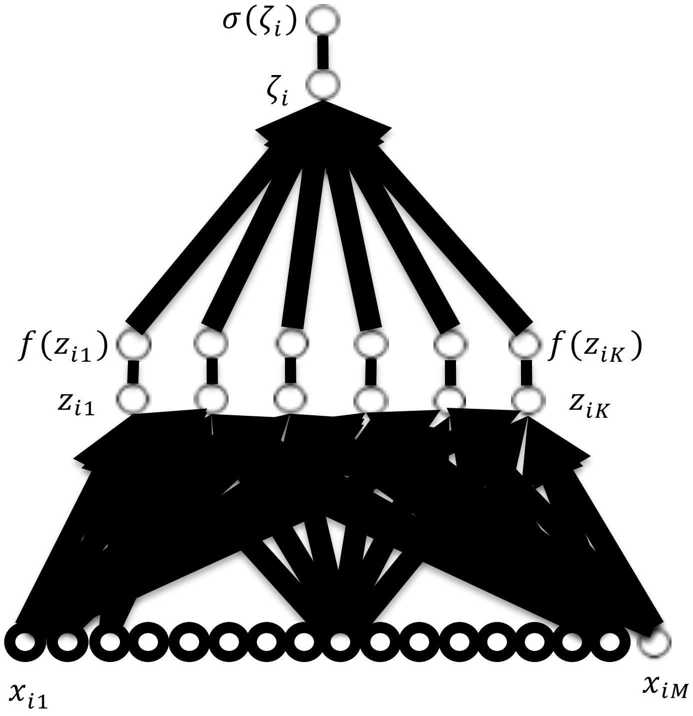


Classification boundary is necessarily *linear*



Necessarily measuring closeness to a single *template image*

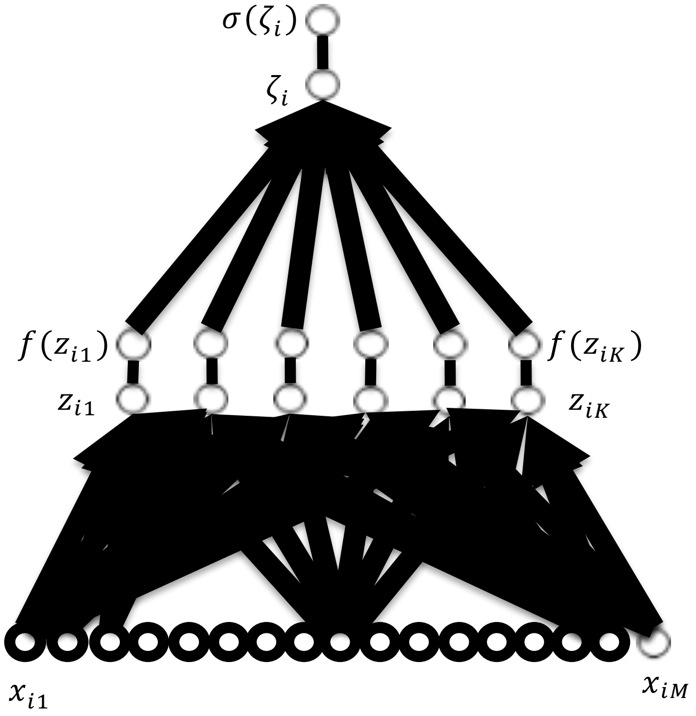
Add hidden layers



Final output

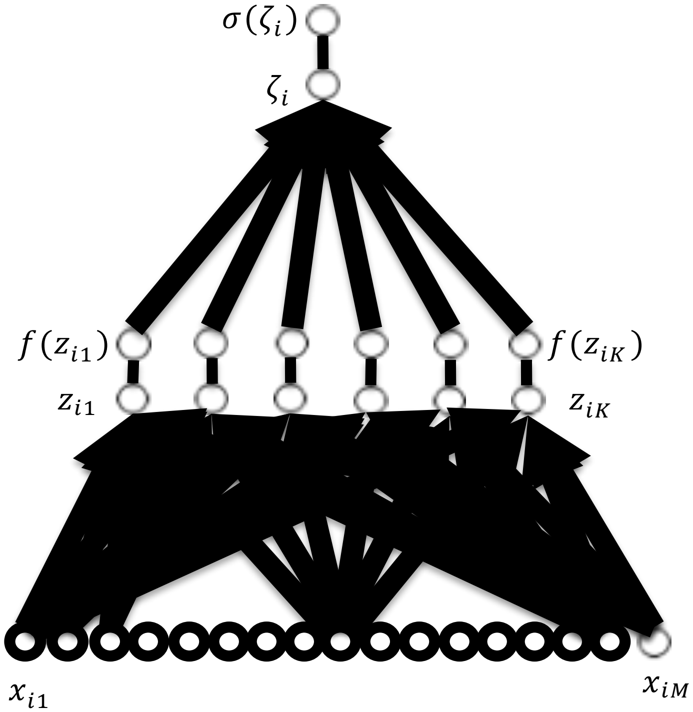
"Scores" of K "hidden" features

Add hidden layers

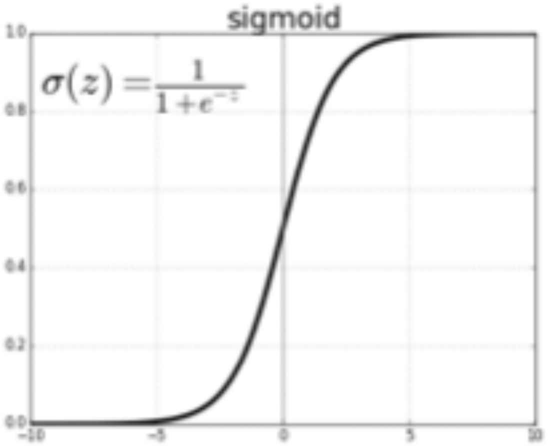


What f to use?
No *need* to be betw.
0 and 1.

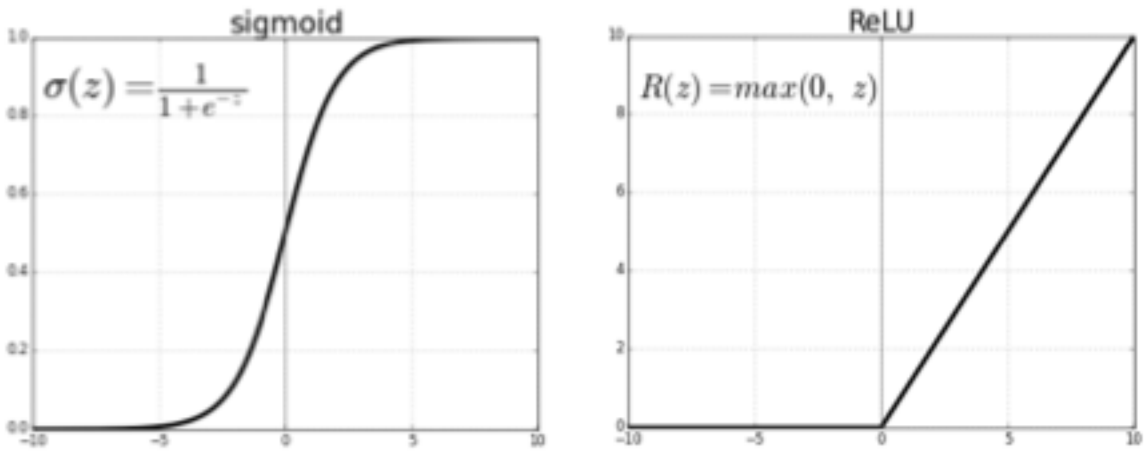
Add hidden layers



What f to use?
Problem with sigmoid:
"vanishing gradients"

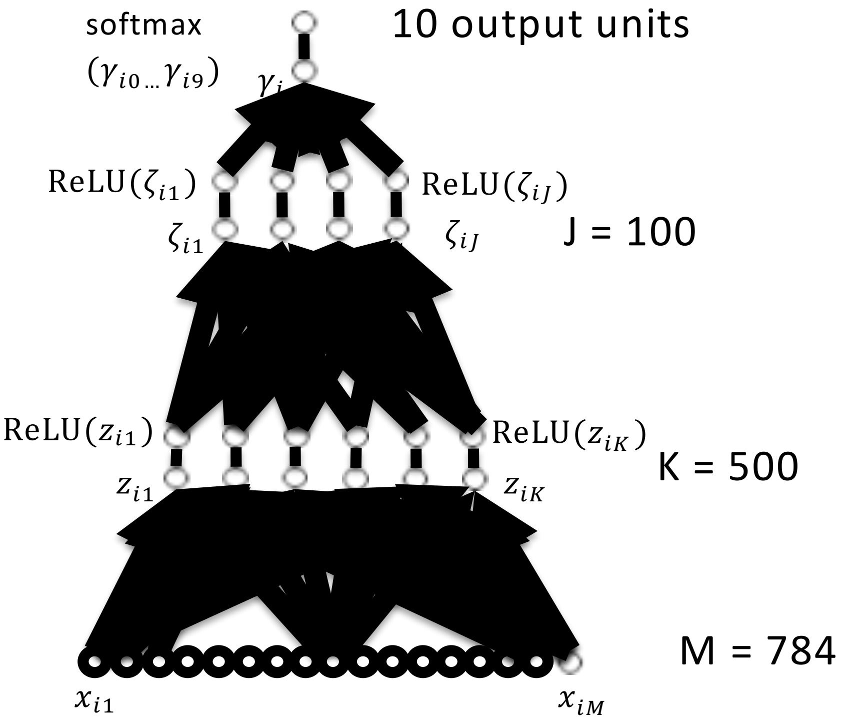


The ReLU Activation Function



https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Homework: MLP



tf.matmul(…, …)
tf.nn.relu(…)
tf.nn.softmax_cross_entropy_with_logits(…)