Week 6

9/30/25

↳ Quiz

↳ Last class: 3 4 8 1 0 2.4 avg

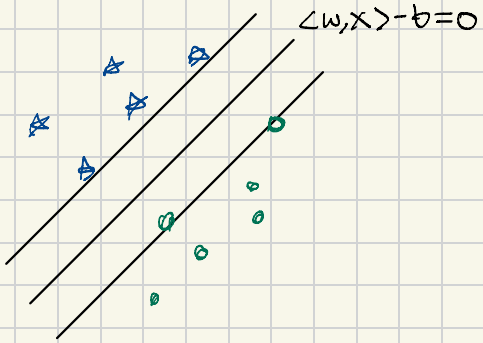
- most mathematical topic
- working on problem helped?

↳ Pset 5 due tonight

↳ Flipped class Thursday

↳ OH Friday 10-11:30, 12:30-1:00

Review



Today: Why is $\langle x^{(i)}, x^{(j)} \rangle$ so useful?

- a new (simple) algorithm
- kernel trick (why dual sum)
- extend trick to other algorithms

Primal

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} \xi_i \quad \text{s.t.} \quad y^{(i)}\left[\langle w, x^{(i)}\rangle - b\right] - 1 \geq -\xi_i, \quad \xi_i \geq 0$$

variables $= d + 1 + n$ // $w \in \mathbb{R}^d$, $b \in \mathbb{R}$, $\xi \in \mathbb{R}^n_+$

Dual

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \boxed{\langle x^{(i)}, x^{(j)} \rangle} \quad \text{s.t.} \quad \alpha_i \geq 0, \ \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

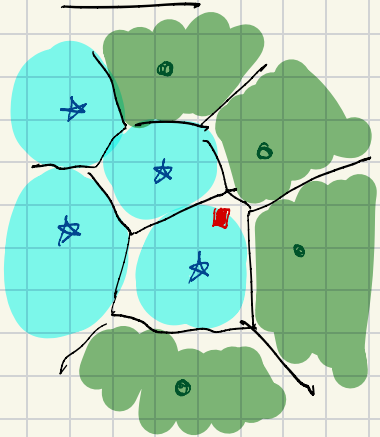
variables $= n$ // $\alpha \in \mathbb{R}^n_+$

K-Nearest Neighbors

On new point x,

- find k points "closest" to x from training points
- aggregate labels
  - ↳ mode for classification
  - ↳ average for regression

Voronoi

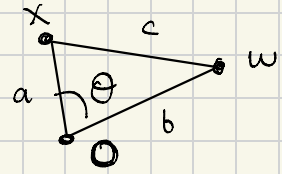


"close" by cosine similarity

$$\cos(\theta) = \frac{\langle x, w \rangle}{\|w\|_2 \|x\|_2}$$

↑ angle between x, w

Proof:



$a = \|x\|_2$

$$\cos\theta = \frac{a^2 + b^2 - c^2}{2ab}$$ by Law of Cosines

$$\cos(\theta) = \frac{\langle x, w \rangle}{\|x\|_2 \|w\|_2}$$

Equivalent:

- large angle θ
- small cosine sim $\cos\theta$
- large distance $\|x - w\|_2$

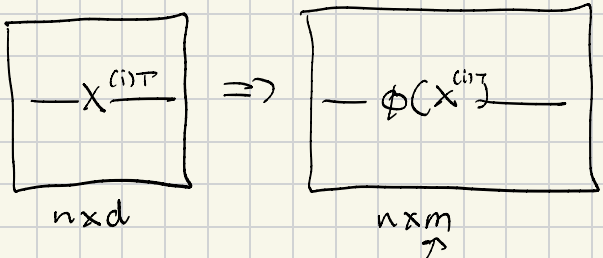
MNIST: 0 1 2 3 4 5 6 7 8 9

x_{new}					
2	2	2	2	2	2
7	7	7	7	7	7
9	9	9	9	4	9

Non-parametric (like Naive Bayes), how can we improve?

Feature Transform

$\phi : \mathbb{R}^d \to \mathbb{R}^m$



Kernel $K(x, x') = \langle \phi(x), \phi(x') \rangle$

$K \in \mathbb{R}^{n \times n} = \underset{n \times m}{\phi(X)} \; \underset{m \times n}{\phi(X)^T}$

Big Q: How is this faster?

Polynomial Kernel

$K(x, x') = (\langle x, x' \rangle + 1)^p$ for integer p

$\uparrow O(d)$ time

$\phi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \sqrt{2}x_3 \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ \sqrt{2}x_1x_3 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_2x_3 \end{bmatrix}$ $\leftarrow O(d^p)$

$d = 3 \quad p = 2$

Gaussian Kernel

$K(x, x') = e^{-\frac{\|x - x'\|_2^2}{2\sigma^2}}$ $\leftarrow O(d)$ time

$\phi(x)$ is infinite series

Reparameterization Trick

What if XX^T doesn't naturally appear?

Replace Xw with XX^Tv

Why is this okay?

WLOG, $w = \sum_{i=1}^{n} v_i x^{(i)} = X^T v$

Proof: Suppose for contradiction not

i.e., $w = v + u$ where $u^T x^{(i)} = 0 \ \forall i$

$Xw = Xv + Xu = Xv$

Logistic Regression

$$\underset{w}{\text{argmin}} \ \mathcal{L}_{CE}(\sigma(Xw), y)$$

vs

$$\underset{v}{\text{argmin}} \ \mathcal{L}_{CE}(\sigma(XX^Tv), y)$$

Linear Regression

$$\underset{w}{\text{argmin}} \ \mathcal{L}_{MSE}(Xw, y)$$

vs

$$\underset{v}{\text{argmin}} \ \mathcal{L}_{MSE}(XX^Tv, y)$$

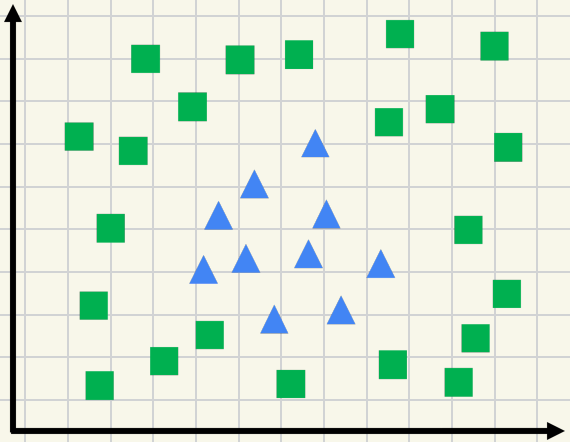
10/2/2025

So far, we have:

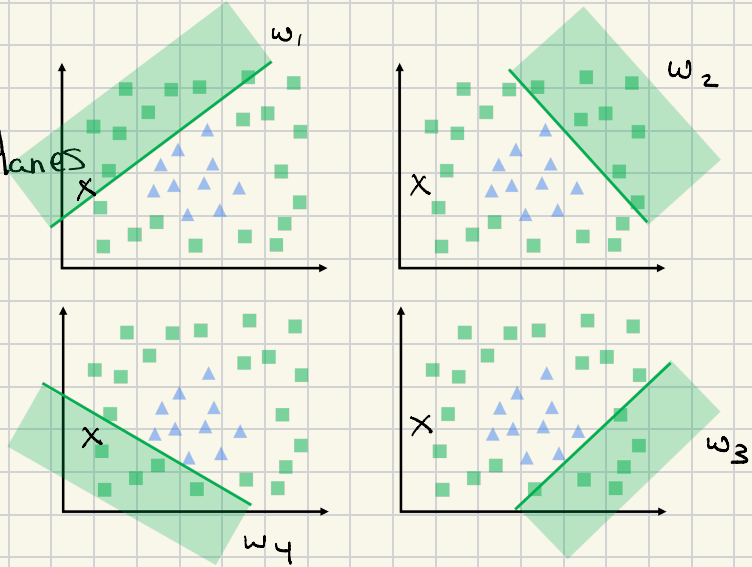
1. Selected features (and transformed them)
2. Trained a model on the features

Neural networks will let us do both simultaneously

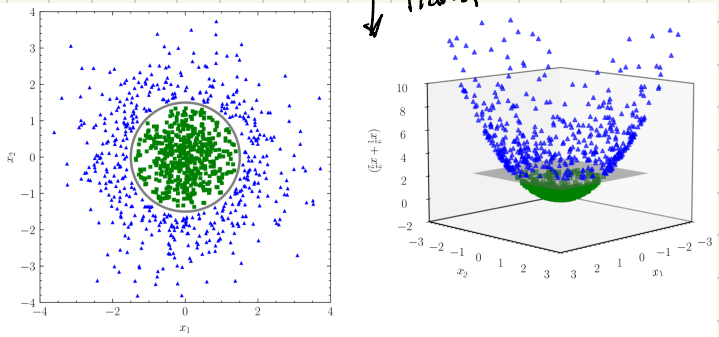
Neural Networks

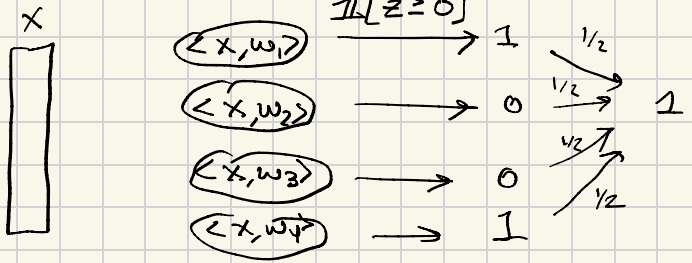


Separating Hyperplanes

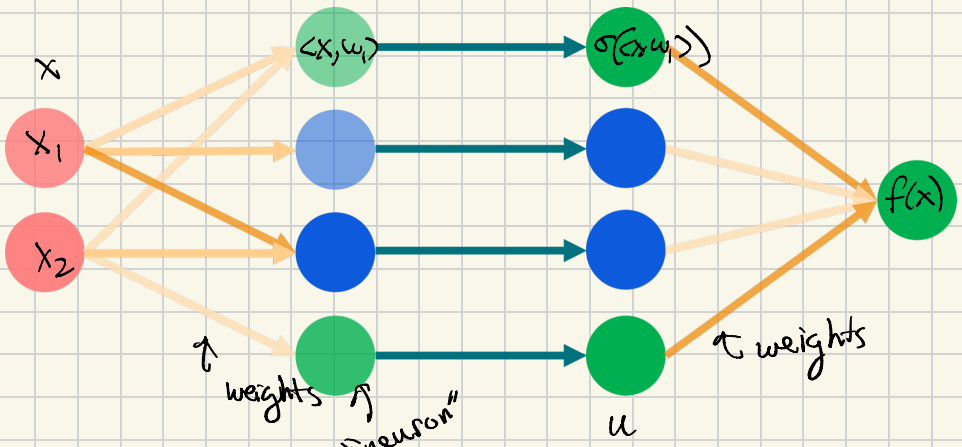


Feature Transformation





Neural Network Architecture



Choices Include:

↳ Activation function
- $\sigma(z) = \max(0, z)$ "ReLU"
- $\sigma(z) = \frac{1}{1+e^{-z}}$ "Sigmoid"

↳ Number of layers

↳ Number of neurons

↳ Loss function
- MSE
- Cross Entropy

Pytorch

`nn.Linear(2,4)` `nn.relu()` `nn.Linear(4,1)`

```
nn.Sequential(
    nn.Linear(2,4), nn.relu(), nn.Linear(4,1)
)
```

Linear Algebra

$W^{(1)} \in \mathbb{R}^{4\times 2}$ $\sigma(W)$ $W^{(2)} \in \mathbb{R}^{1\times 4}$

$W^{(2)} \sigma(W^{(1)} x)$

Optimization

Big Q: How do we compute $\nabla_w \mathcal{L}$ (quickly)?

Chain Rule Review

Single variate

$$\frac{df}{dx} = \lim_{t \to 0} \frac{f(x+t) - f(x)}{t}$$

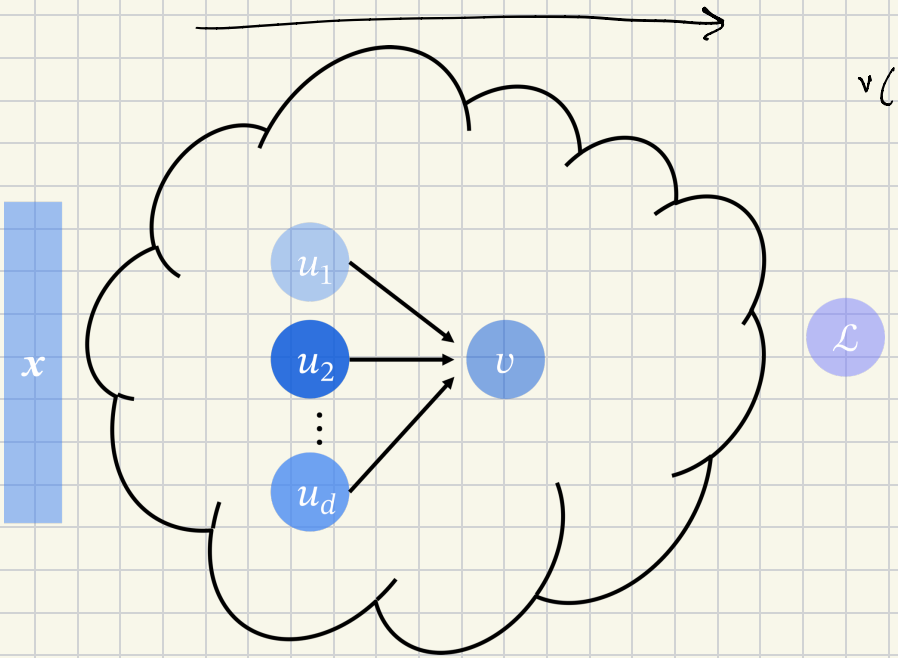
$f : \mathbb{R} \to \mathbb{R} \qquad y : \mathbb{R} \to \mathbb{R}$

$$\frac{d}{dx} f(y(x)) = \lim_{t \to 0} \frac{f(y(x+t)) - f(y(x))}{t} = \lim_{t \to 0} \frac{f(y(x+t)) - f(y(x))}{y(x+t) - y(x)} \cdot \frac{y(x+t) - y(x)}{t} = \frac{df}{dy} \frac{dy}{dx}$$

$f : \mathbb{R}^m \to \mathbb{R} \qquad y_i : \mathbb{R} \to \mathbb{R}$

$$\frac{df}{dx}(y_1(x), \ldots, y_m(x)) = \left(\frac{df}{dy_1} \frac{dy_1}{dx} + \ldots + \frac{df}{y_m} \frac{dy_m}{dx} \right)$$

Forward Pass



$v(u_1, \ldots, u_d)$

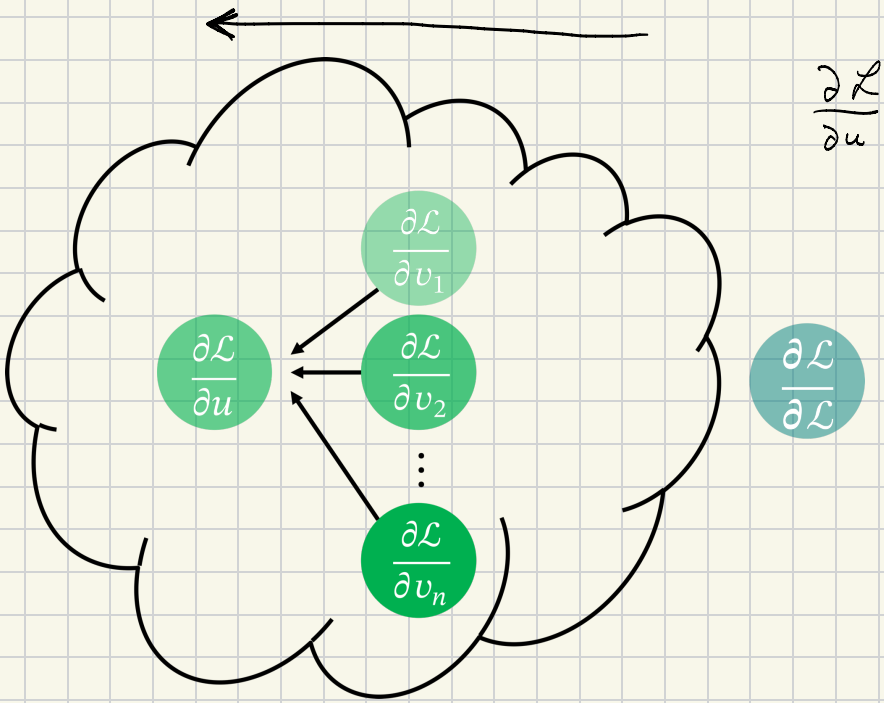
And so on!

Pytorch:

```
pred = model(x)
loss = loss_fn(pred, y)
```

Insight: Only need $u_1, \ldots, u_d$ to compute v

Backward Pass



$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial \mathcal{L}}{\partial v_1}\frac{\partial v_1}{\partial u} + \ldots + \frac{\partial \mathcal{L}}{\partial v_n}\frac{\partial v_n}{\partial u}$$

And so on!

Pytorch:

```
loss.backward()
```

Insight: Only need $\frac{\partial \mathcal{L}}{\partial v_1}, \ldots, \frac{\partial \mathcal{L}}{\partial v_n}$ and $\frac{\partial v_1}{\partial u}, \ldots, \frac{\partial v_n}{\partial u}$ to compute $\frac{\partial \mathcal{L}}{\partial u}$

Linear Algebraic View

Neural Nets invented in 1950s

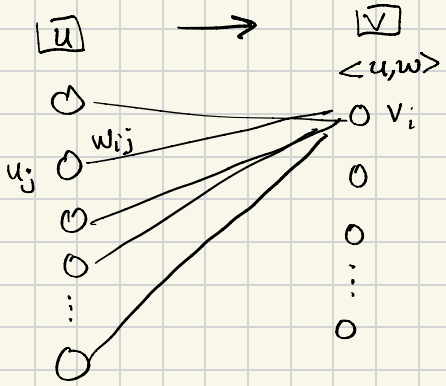
Backprop proposed in 1980s

Q: Why the wait?

A: Only really useful with MANY weights (parameters), which takes storage and compute

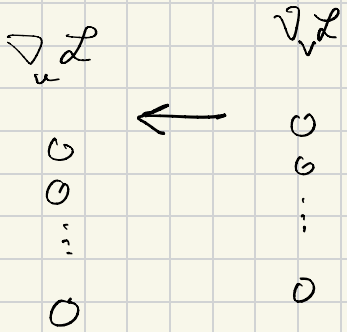
In 2010s, researchers realized GPUs (good at matrix mult for video games) could be used for neural nets!

Forward



$v = Wu \qquad [v]_i = \sum_{j=1}^{d} W_{ij} u_j$

Backward



$\frac{\partial v_i}{\partial u_j} = W_{ij} \qquad \nabla_u \mathcal{L} = W^T \nabla_v \mathcal{L}$

$$\frac{\partial \mathcal{L}}{\partial u_j} = \sum_{i=1}^{m} \frac{\partial \mathcal{L}}{\partial v_i} \frac{\partial v_i}{\partial u_j} = \sum_{i=1}^{n} [W^T]_{j,i} \frac{\partial \mathcal{L}}{\partial v_i}$$

Note: We want $\nabla_w \mathcal{L}$, easy from gradient of neurons

Q: Why would Gradient descent work when function highly non-convex?

- Maybe high dimensional loss functions rarely have local minima
- Not well explained by theory

Next week: Two very import architectures

- Convolutions: How can we make networks efficient?
  ↳ Linear(1000, 1000) has 1m param
- Transformers: How can we process sequential data?
  ↳ eg text, video