Scientific Computing

Wednesday, April 29

Announcements

* Final Exam: Monday, 5/4,
  1pm - 3pm
  Johnston Hall 417
* Final NN project due Monday night, 11:59pm

Office Hours:

Mon, 9:30 - 10:30
Fri, 2:00 - 3:00

Cudahy 307

Train/Test Split

* Use most of your data to train your network.

* Reserve some to test your network.

A reasonable split is 80% train / 20% test.

Never use the test data for training!

↳ Shuffle your data before you split!

If your NN is overfitting, you'll see great (low) loss on the training data, but bad (high) loss on the test set.

overfitting = over training or memorization

Loss / Accuracy / RMSE

Loss = measure of how close actual ouput is to expected output

80% Train Loss, Test Loss 20%

Training process = make loss go down

More interpretable to see if the NN is working on test data

Classification: "Accuracy" = How many predicted classes (the one with the highest prob.) are the right one?

Regression: "Root Mean Squared Error": Just the square root of MSE. Matches the units of the output, unlike MSE which is scaled.

$(\hat{y} - y)^2$

$\sqrt{\$^2}$ unit $\$$

Training Loop:

"Epoch" = feed all training data through and learn from it

train input

train output

test input / output

```
def train(self, X, y, loss_obj, epochs, learning_rate, batch_size=None, shuffle=True,
          verbose=False, X_val=None, y_val=None, metric=None):
    """
    Train the network using gradient descent.
    X: input data of shape (features, samples)
    y: true labels of shape (outputs, samples)
    loss_obj: instance of a loss class with forward_pass and backward_pass
    epochs: number of epochs to train
    learning_rate: step size for parameter updates
    batch_size: number of samples per batch (None for full-batch)
    shuffle: whether to shuffle data each epoch
    verbose: whether to print progress
    X_val: validation data of shape (features, samples), optional
    y_val: validation labels of shape (outputs, samples), optional
    metric: string, either "accuracy" or "rmse", to compute on validation data, optional

    Training can be interrupted at any time with Ctrl+C, and the method will return gracefully.
    """
```

train data

test data

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

Features
- Date + Time
- Season
- Holiday
- Workday
- Weather
- Temperature
- "Feels like" temp.
- Humidity
- Wind Speed

From Kaggle

Regression

rentals per day (registered / casual)

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

Data Fields

datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - whether the day is considered a holiday
workingday - whether the day is neither a weekend nor holiday
weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp - temperature in Celsius
atemp - "feels like" temperature in Celsius
humidity - relative humidity
windspeed - wind speed
casual - number of non-registered user rentals initiated
registered - number of registered user rentals initiated
count - number of total rentals

How do we set these up as numerical inputs?

All inputs should be in the range $[-1, 1]$ (or at least close)

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

* Some data is already numeric (takes values in a range)

Scale data down into the range $[-1, 1]$

Option 1: Scale it linearly.

Option 2: Scale by $\frac{x - \text{mean}}{\text{st. dev.}}$

Data Fields

datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - whether the day is considered a holiday
workingday - whether the day is neither a weekend nor holiday
weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp - temperature in Celsius
atemp - "feels like" temperature in Celsius
humidity - relative humidity
windspeed - wind speed
casual - number of non-registered user rentals initiated
registered - number of registered user rentals initiated
count - number of total rentals = redundant

} output

Scale output too! just make sure you unscale when looking at your predictions later

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

binary variables: holiday is yes or no
workingday is yes or no

Each gets one neuron,
0 input = no
1 input = yes

Data Fields

datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - whether the day is considered a holiday
workingday - whether the day is neither a weekend nor holiday
weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp - temperature in Celsius
atemp - "feels like" temperature in Celsius
humidity - relative humidity
windspeed - wind speed
casual - number of non-registered user rentals initiated
registered - number of registered user rentals initiated
count - number of total rentals

} output

= redundant

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

Season + Weather are encoded as #s, but not #s that mean anything.

Bad: divide "weather" by 4 to get 1 neuron in [0,1].

Good: Use four neurons, one for each option. The correct option is a 1, the rest are 0.

one-hot encoding

Data Fields

datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - whether the day is considered a holiday
workingday - whether the day is neither a weekend nor holiday
weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp - temperature in Celsius
atemp - "feels like" temperature in Celsius
humidity - relative humidity
windspeed - wind speed
casual - number of non-registered user rentals initiated
registered - number of registered user rentals initiated
count - number of total rentals

} output

← redundant

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

First split into month of the year, 1-12, day of the week, 0-6, hour of the day 0-23

Again, we don't want to use these as #s. Hour 0 and 23 should be "close", right?

"Cyclical Encoding"

Data Fields

datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - whether the day is considered a holiday
workingday - whether the day is neither a weekend nor holiday
weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp - temperature in Celsius
atemp - "feels like" temperature in Celsius
humidity - relative humidity
windspeed - wind speed
casual - number of non-registered user rentals initiated
registered - number of registered user rentals initiated
count - number of total rentals

} output

← redundant

(3) Bike sharing (there is a "bike-share-explanation.md" markdown file

Data from a bike rental service in Washington, D.C.

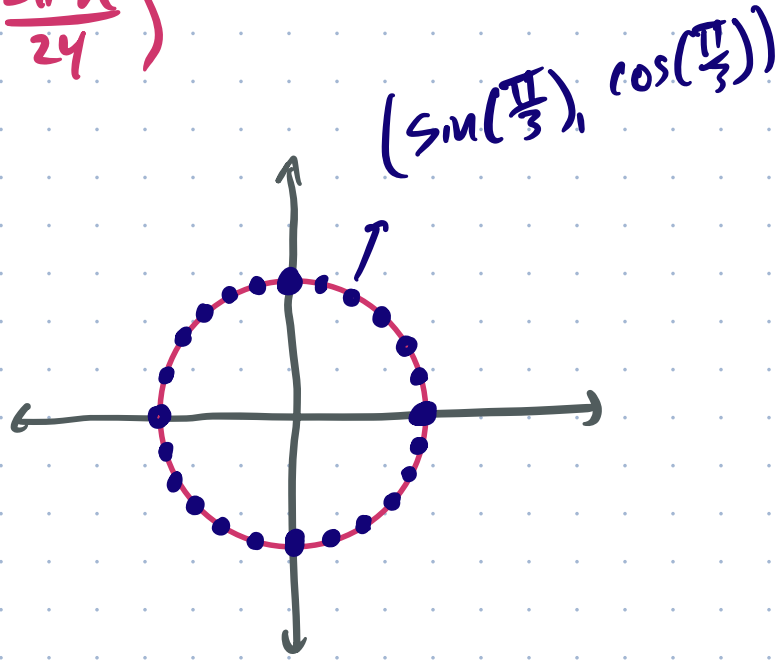
Cyclical Encoding Let h = hour of the day. Encode with two neurons. $n_1 = \sin\left(\frac{2\pi \cdot h}{24}\right)$

$n_2 = \cos\left(\frac{2\pi \cdot h}{24}\right)$

Data Fields

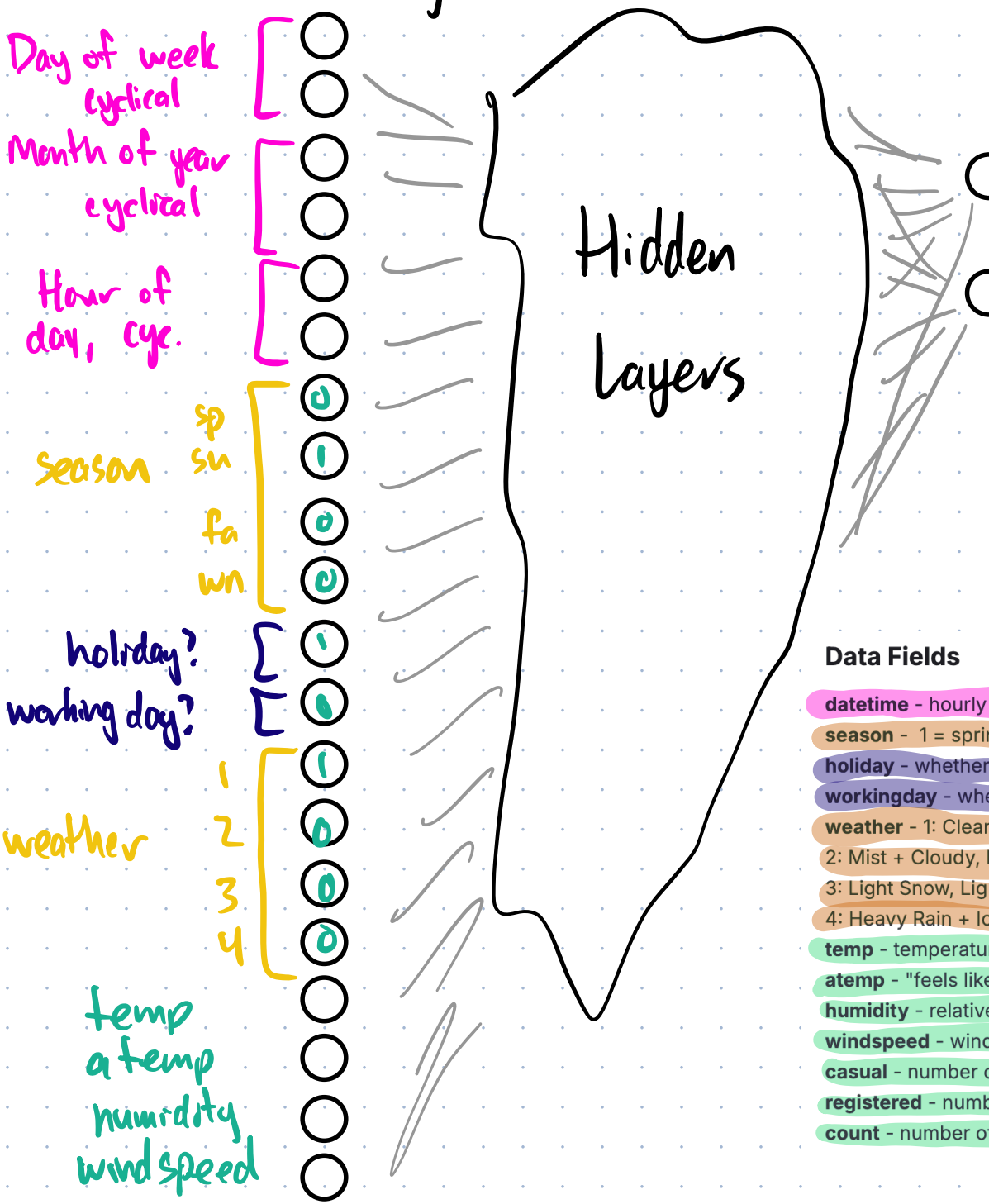
datetime - hourly date + timestamp
season - 1 = spring, 2 = summer, 3 = fall, 4 = winter
holiday - whether the day is considered a holiday
workingday - whether the day is neither a weekend nor holiday
weather - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp - temperature in Celsius
atemp - "feels like" temperature in Celsius
humidity - relative humidity
windspeed - wind speed
casual - number of non-registered user rentals initiated
registered - number of registered user rentals initiated
count - number of total rentals } output

= redundant



Imparts a cyclical feel. 0 and 23 are close now

[3] Bike sharing



Data Fields

- **datetime** - hourly date + timestamp
- **season** - 1 = spring, 2 = summer, 3 = fall, 4 = winter
- **holiday** - whether the day is considered a holiday
- **workingday** - whether the day is neither a weekend nor holiday
- **weather** - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- **temp** - temperature in Celsius
- **atemp** - "feels like" temperature in Celsius
- **humidity** - relative humidity
- **windspeed** - wind speed
- **casual** - number of non-registered user rentals initiated
- **registered** - number of registered user rentals initiated
- **count** - number of total rentals ← redundant

} output

13) Bike sharing "bike-share-explanation.md"

- Demo
- There are some numpy parts that look complicated but are doing simple things. Use Claude or ChatGPT as your coach!

(4) Diabetes prediction

- Demo, investigate and run on your own!
- "diabetes-explanation.md"

Field	Description	Type/Range	Preprocessing Approach
HighBP	High blood pressure (0 = no, 1 = yes)	Binary (0/1)	Use as-is
HighChol	High cholesterol (0 = no, 1 = yes)	Binary (0/1)	Use as-is
CholCheck	Cholesterol check in last 5 years (0 = no, 1 = yes)	Binary (0/1)	Use as-is
BMI	Body Mass Index	Continuous	Standardize (zero mean, unit variance)
Smoker	Smoked at least 100 cigarettes (0 = no, 1 = yes)	Binary (0/1)	Use as-is
Stroke	Ever told had a stroke (0 = no, 1 = yes)	Binary (0/1)	Use as-is
HeartDisease	Coronary heart disease or MI (0 = no, 1 = yes)	Binary (0/1)	Use as-is
PhysActivity	Physical activity in past 30 days (0 = no, 1 = yes)	Binary (0/1)	Use as-is
Fruits	Consume fruit 1+ times/day (0 = no, 1 = yes)	Binary (0/1)	Use as-is
Veggies	Consume vegetables 1+ times/day (0 = no, 1 = yes)	Binary (0/1)	Use as-is
HvyAlcohol	Heavy drinker (0 = no, 1 = yes)	Binary (0/1)	Use as-is
AnyHealthcare	Any health care coverage (0 = no, 1 = yes)	Binary (0/1)	Use as-is
NoDocbcCost	Could not see doctor due to cost (0 = no, 1 = yes)	Binary (0/1)	Use as-is
GenHlth	General health (1 = excellent, 5 = poor)	Ordinal (1–5)	Standardize
MentHlth	Days mental health not good (1–30)	Ordinal (1–30)	Standardize
PhysHlth	Days physical health not good (1–30)	Ordinal (1–30)	Standardize
DiffWalk	Difficulty walking/climbing stairs (0 = no, 1 = yes)	Binary (0/1)	Use as-is
Sex	0 = female, 1 = male	Binary (0/1)	Use as-is
Age	Age group (1 = 18–24, ..., 13 = 80+)	Ordinal (1–13)	Standardize
Education	Education level (1 = none/kindergarten, ..., 6 = college grad)	Ordinal (1–6)	Standardize
Income	Income scale (1 = < $10k, ..., 8 = $75k+)	Ordinal (1–8)	Standardize

You now have all the fundamental knowledge of how Neural Networks work!

There are many more things to learn about them.

Ex: Specialized networks for some kinds of tasks

Convolutional Neural Networks → Grid Data like images

Graph Neural Networks → Drug Design

Ex: Preventing overfitting, which is when the NN memorizes the training data in a way that doesn't extrapolate to the test data.

Ex: Preventing overfitting, which is when the NN memorizes the training data in a way that doesn't extrapolate to the test data.

* Use a smaller network.

* Dropout - each training loop, randomly turn off 20% of neurons (they don't pass data forward and don't get adjusted by gradient descent)

* Regularization - Try to stop the weights from getting too big (2, 5, 10, etc).

  Adds a component to the loss function to penalize for big weights.