

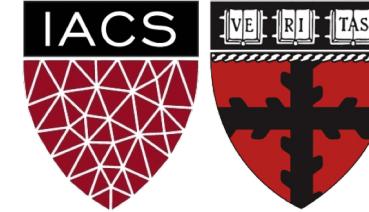
The Roadmap:

a recap of where we've been, where
we're heading, and how it's all related

Harvard IACS

CS109B

Chris Tanner, Pavlos Protopapas, Mark Glickman



Learning Objectives

- Recap models from CS109A and CS109B
- Understand the different categories of models
- Discern similarities/differences between models
- Feel comfortable choosing which model to use
- Understand the limitations of our models thus far
- Feel prepared tackling the remaining course content

Data Science World

- ▶▶ 1 player
- ▶▶ 2 players
- ▶▶ 226 players

Your Data X

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Your Data X

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Your Data X

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Your Data X

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Your Data X

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $Y = f(X) + \varepsilon$

X	Age	Play	Rainy	Y	Temp
	22	N	Y		91
	29	Y	N		89
	31	N	N		56
	23	Y	N		71
	37	N	Y		72
	41	Y	N		83
	29	Y	Y		97
	21	N	N		64
	30	Y	N		68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $Y = f(X) + \varepsilon$
- Want a model f that is:
 - Supervised

X	Age	Play	Rainy	Y	Temp
	22	N	Y		91
	29	Y	N		89
	31	N	N		56
	23	Y	N		71
	37	N	Y		72
	41	Y	N		83
	29	Y	Y		97
	21	N	N		64
	30	Y	N		68

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $Y = f(X) + \varepsilon$
- Want a model f that is:
 - Supervised

X	Age	Play	Rainy	Y	Temp
	22	N	Y		91
	29	Y	N		89
	31	N	N		56
	23	Y	N		71
	37	N	Y		72
	41	Y	N		83
	29	Y	Y		97
	21	N	N		64
	30	Y	N		68

Def:

Supervised models use target data, **Y**, to provide feedback so that your model can learn the relationship between **X** and **Y**.

$$Y = f(X)$$

- Supervised

X	Play	Rainy	Y	Temp
29	N	Y	91	
21	Y	N	89	
30	N	N	56	
	Y	N	71	
	N	Y	72	
	Y	N	83	
	Y	Y	97	
	N	N	64	
	Y	N	68	

- Given some data such that each row corresponds to a distinct i.i.d. observation
- You may be interested in a particular column (e.g. **Temp**)
- Let's divide our data and learn how data **X** is related to data **Y**
- Assert that: $Y = f(X) + \varepsilon$
- Want a model f that is:
 - Supervised**
 - Predicts real numbers (regression model)

X	Age	Play	Rainy	Y	Temp
	22	N	Y		91
	29	Y	N		89
	31	N	N		56
	23	Y	N		71
	37	N	Y		72
	41	Y	N		83
	29	Y	Y		97
	21	N	N		64
	30	Y	N		68

- Given some data such that each

Def:

Regression models are **supervised**

models, whereby **Y** are *continuous values*.

- Predicts real numbers
(regression model)

X	Rainy	Temp
21	Y	91
30	Y	89
N	N	56
N	N	71
N	Y	72
N	N	83
Y	Y	97
N	N	64
N	N	68

- Given some data such that each

Def:

Regression models are **supervised**

models, whereby **Y** are *continuous* values.

Classification models are **supervised**

models, whereby **Y** are *categorical* values.

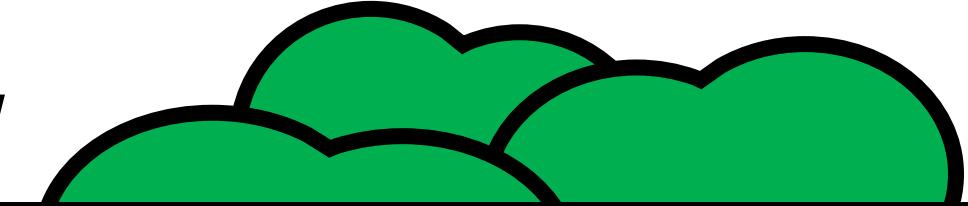
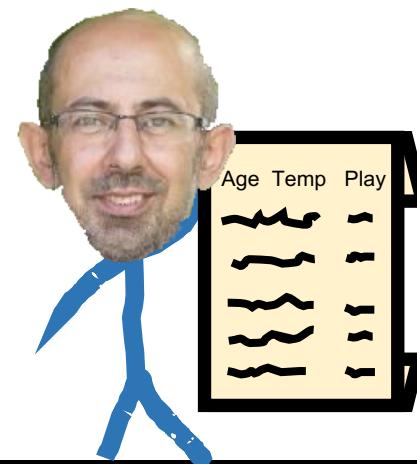
- Predicts real numbers
(regression model)

X	Rainy	Temp
21	Y	91
30	N	89
	N	56
	N	71
	Y	72
	N	83
	Y	97
	N	64
	N	68

- Let's say this is our data
- Want a model that is:
 - Supervised
 - Predicts real numbers (regression model)
- Q: What model could we use?

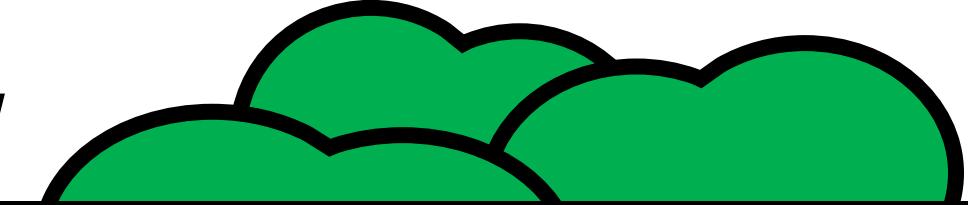
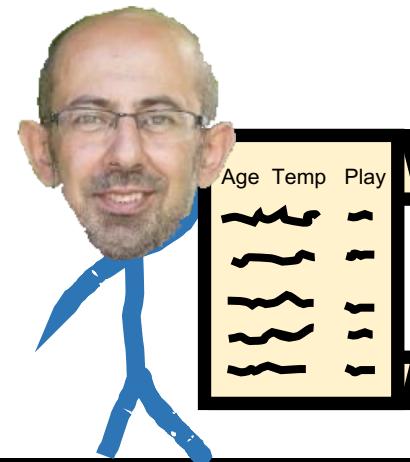
	X	Y		
	Age	Play	Rainy	Temp
	22	N	Y	91
	29	Y	N	89
	31	N	N	56
	23	Y	N	71
	37	N	Y	72
	41	Y	N	83
	29	Y	Y	97
	21	N	N	64
	30	Y	N	68

Playground of Models

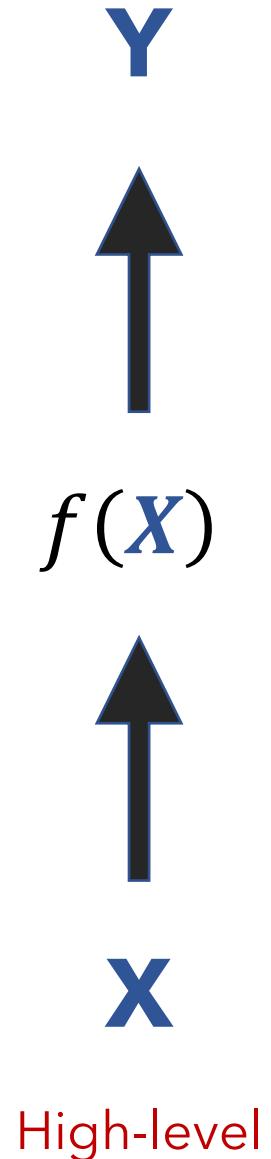


Playground of Models

Linear
Regression



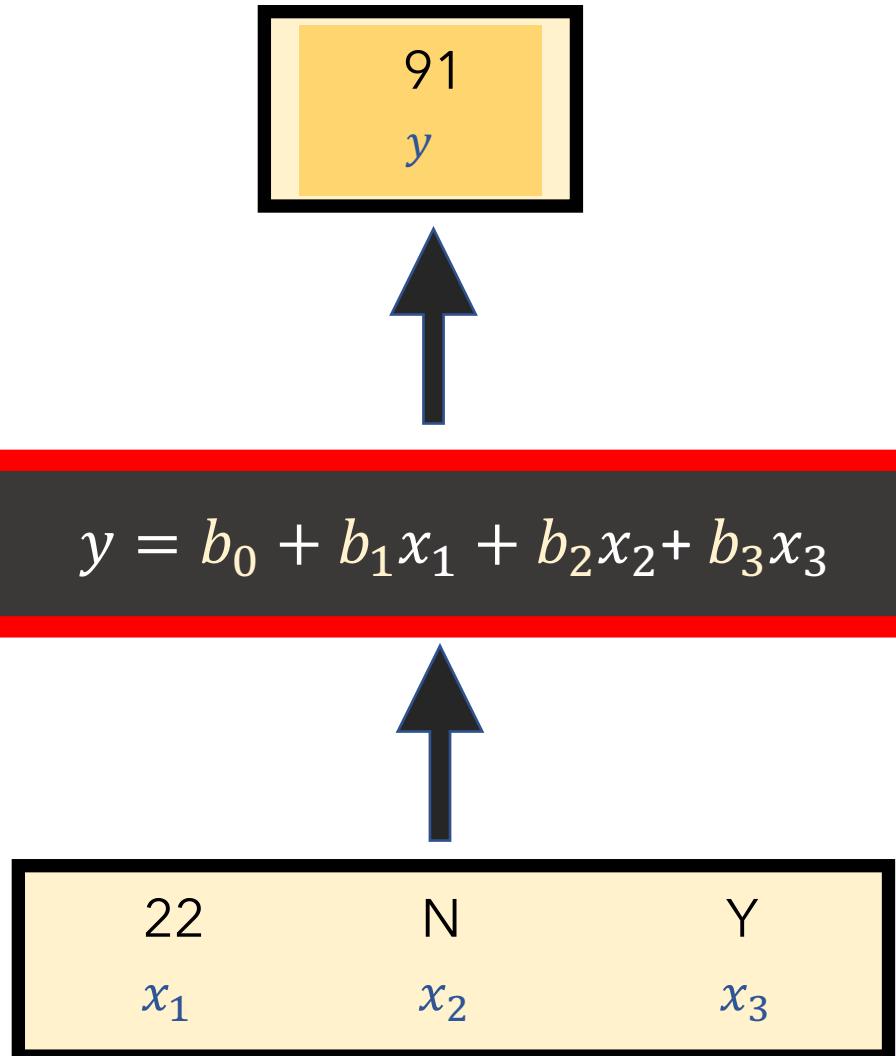
Linear Regression



Linear Regression

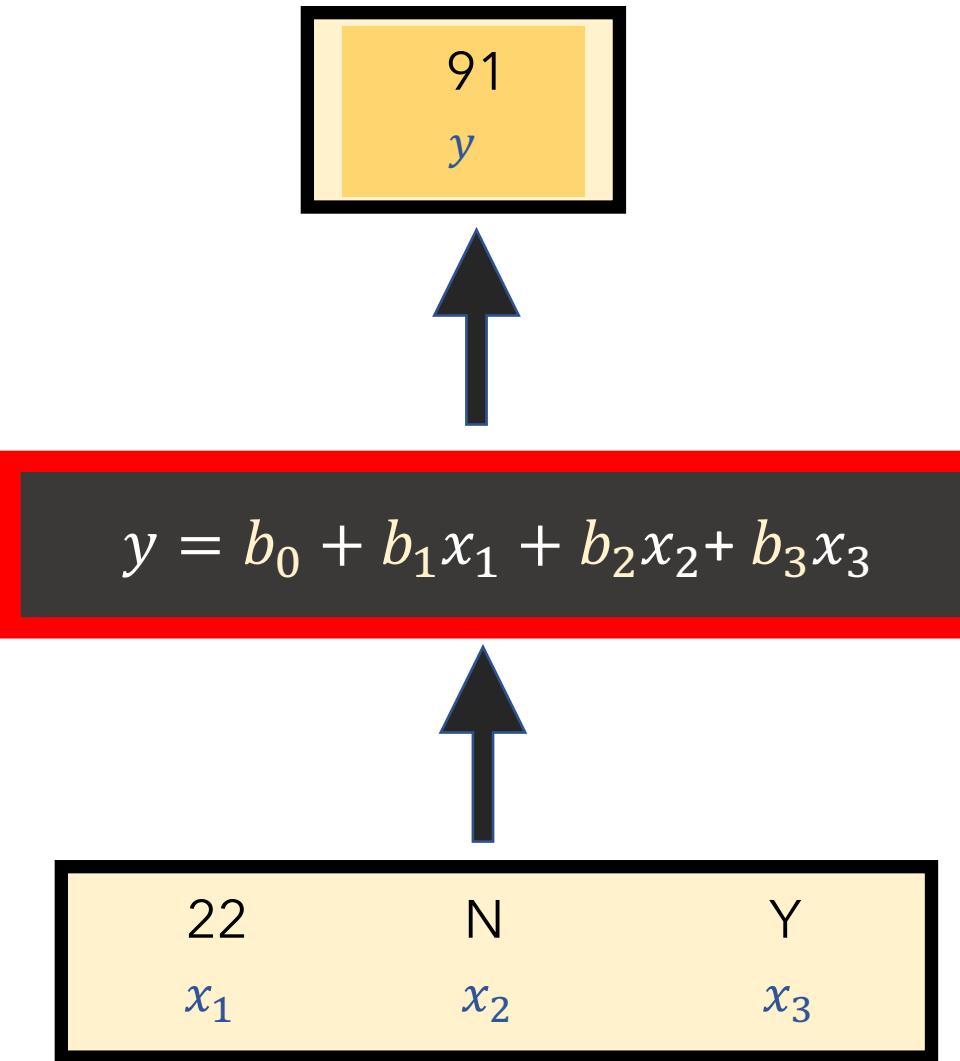
High-level

Y
↑
 $f(\mathbf{X})$
↑
X

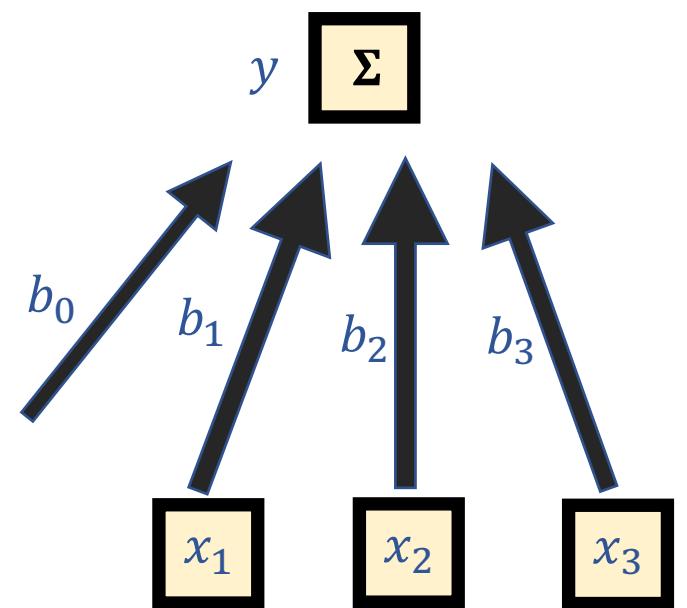


Linear Regression

Y
↑
 $f(X)$
↑
X
High-level



Mathematically



Graphically
(NN format)

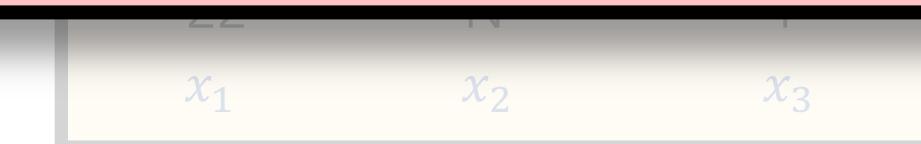
Linear Regression

Y
↑
 $f(X)$
↑
X
High-level

NOTE:

For convenience, in machine learning we tend to let θ represent all of our model's parameters (e.g., $\theta = \{b_0, b_1, b_2, b_3\}$)

Mathematically



Graphically
(NN format)

IMPORTANT

When **training** any supervised model,
be mindful of what you select for:

1. Our **loss function** (aka cost function)

Measures how bad our current
parameters θ are

2. Our **optimization** algorithm?

Determines how we update our parameters θ
so that our model better fits our training data

IMPORTANT

When **training** any supervised model,
be mindful of what you select for:

1. Our **loss function** (aka cost function)

Measures how bad our current
parameters θ are

2. Our **optimization** algorithm?

Determines how we update our parameters θ
so that our model better fits our training data

When **testing** our model's predictions,
be mindful of what you select for:

3. Our **evaluation metric**

Determines our model's performance
(e.g., Mean Squared Error (MSE), R^2 ,
 $F1$ score, etc.)

Linear Regression

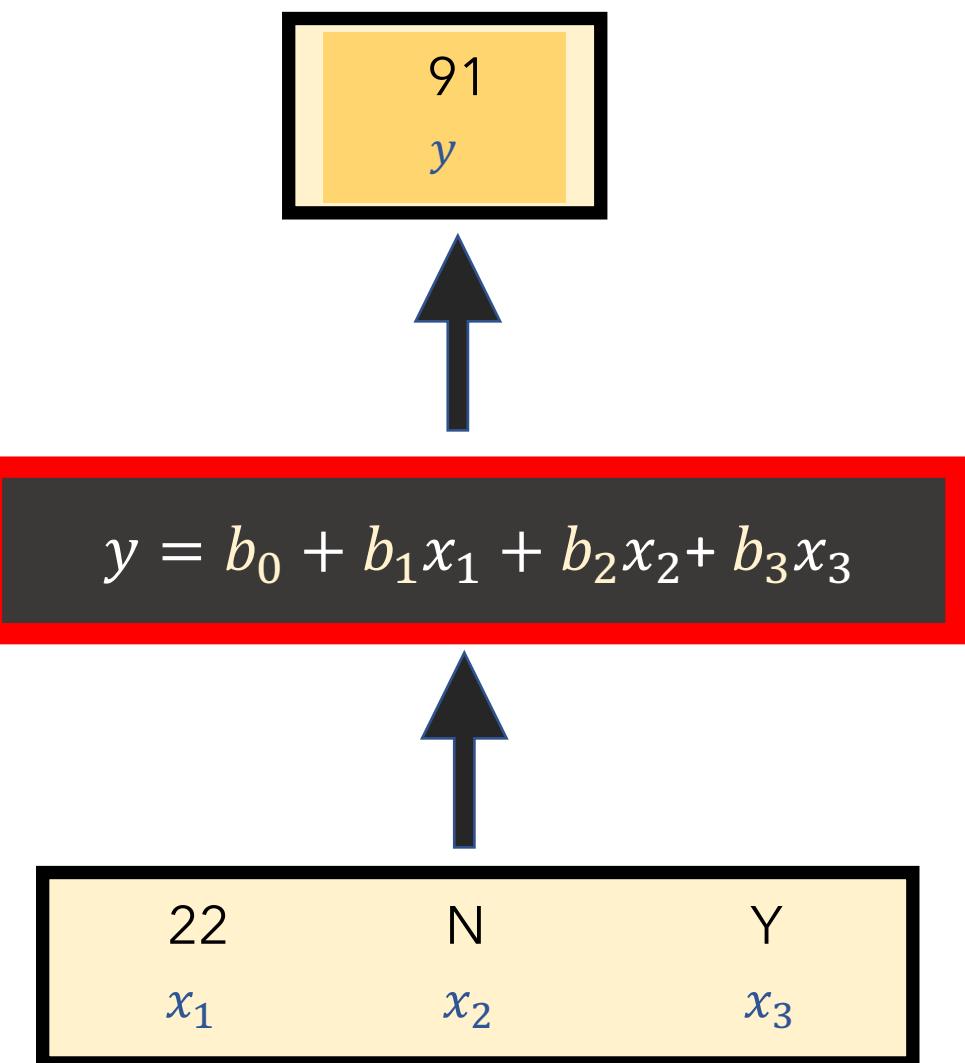
Q1

When training our model, how do we measure its m predictions \hat{y} ?

A1

Cost function $J(\theta) = \frac{1}{2} \sum_{i=1}^m (\hat{y} - y)^2$

"Least Squares"



Mathematically

Linear Regression

Q1

When training our model, how do we measure its m predictions \hat{y} ?

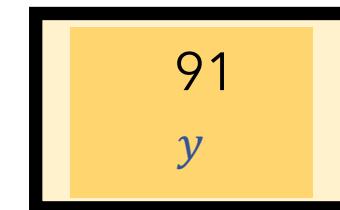
A1

Cost function $J(\theta) = \frac{1}{2} \sum_{i=1}^m (\hat{y} - y)^2$

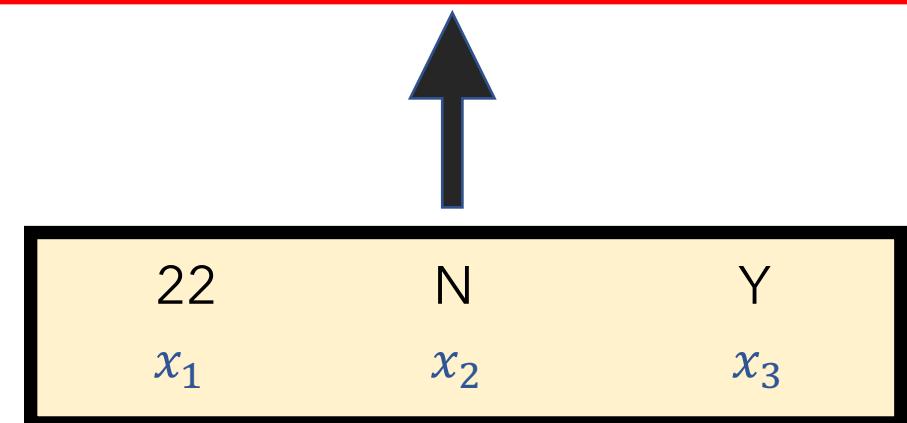
"Least Squares"

• **Q2**

How do we find the optimal θ so that we yield the best predictions?



$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$$



Mathematically

Linear Regression

Q1

When training our model, how do we measure its m predictions \hat{y} ?

A1

Cost function $J(\theta) = \frac{1}{2} \sum_{i=1}^m (\hat{y} - y)^2$

"Least Squares"

• **Q2**

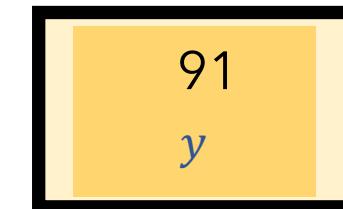
How do we find the optimal θ so that we yield the best predictions?

A2

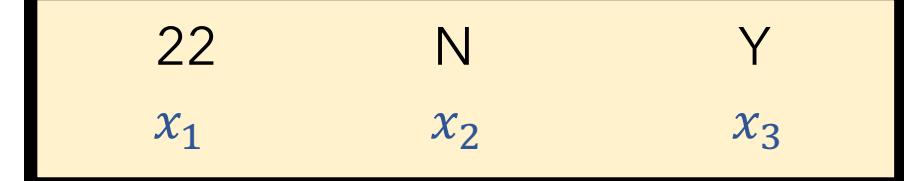
Two optimization algorithm options:

- Gradient Descent (iteratively search)
- Directly (closed-form solution)

$$\theta = (X^T X)^{-1} X^T Y$$



$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$$



Mathematically

Linear Regression

Fitted model example

The plane is chosen to minimize the sum of the squared vertical distances (per our loss function, least squares) between each observation (red dots) and the plane.

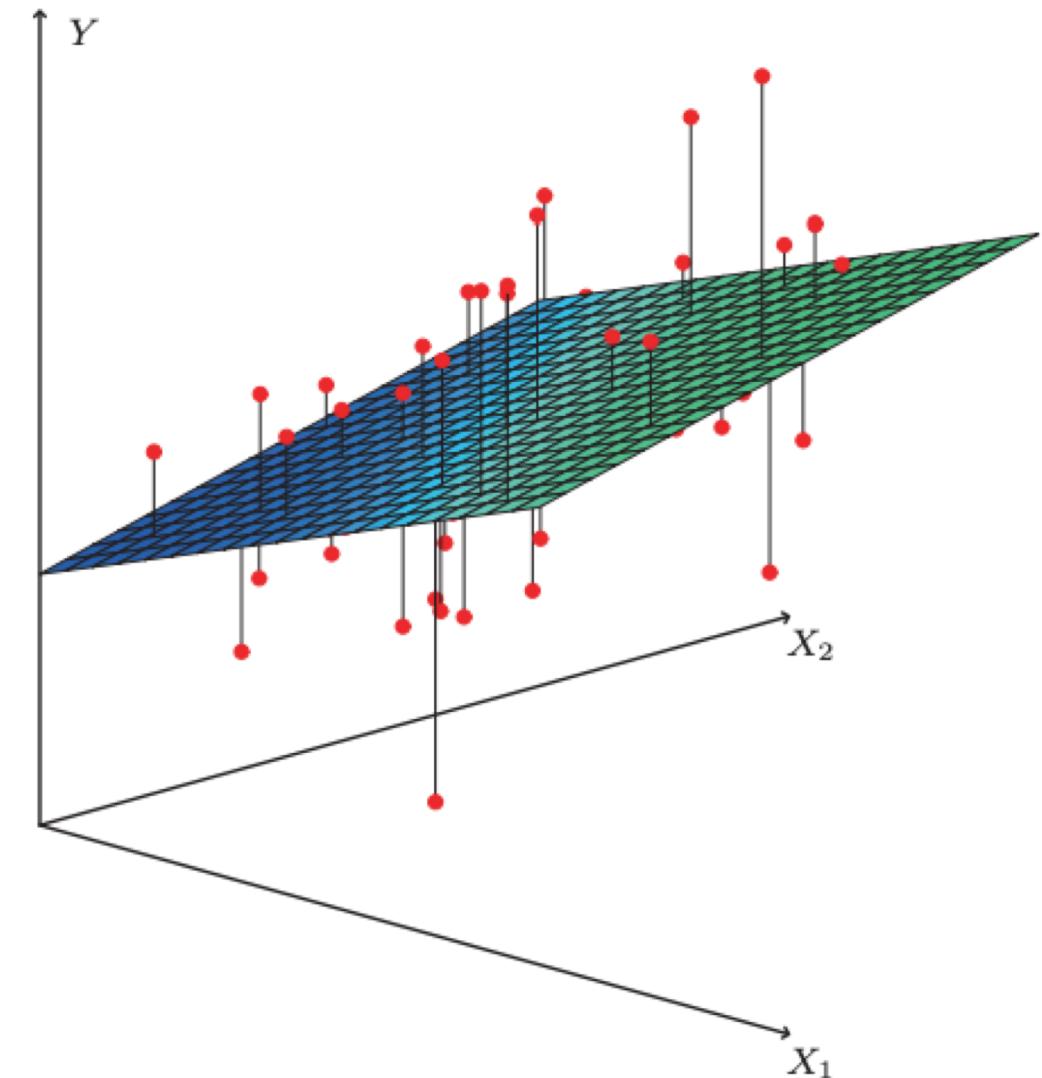


Photo from "An Introduction to Statistical Learning" (James, et al. 2017)

Linear Regression

PROS

- **Simple** and **fast** approach to model linear relationships
- Interpretable results via θ (β coefficients)

CONS

- Can't model **non-linear** relationships
- Vulnerable to **outliers**
- Vulnerable to **collinearity**
- Assumes error terms are **uncorrelated***

* otherwise, we have false feedback during training

**Supervised vs
Unsupervised**

**Regression vs
Classification**

Linear Regression

Supervised

Regression

- Returning to our data, let's model **Play** instead of **Temp**
- Again, we divide our data and learn how data **X** is related to data **Y**
- Again, assert: $\mathbf{Y} = f(\mathbf{X}) + \varepsilon$

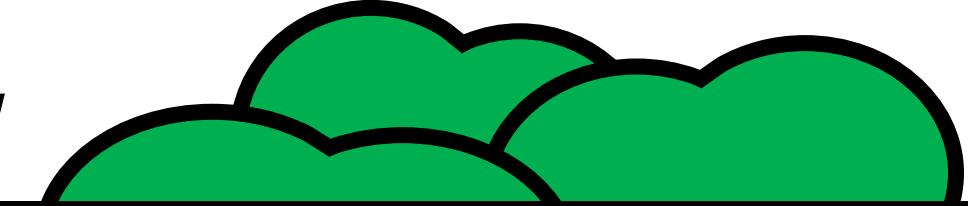
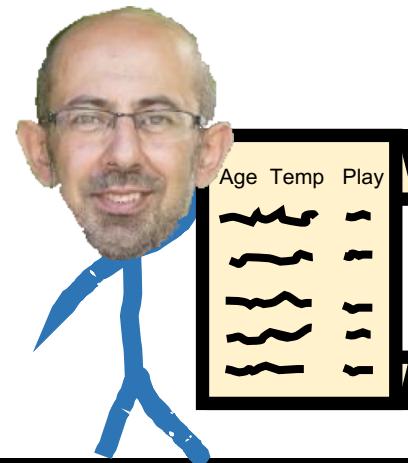
X	Y		
Age	Temp	Rainy	Play
22	91	Y	N
29	89	N	Y
31	56	N	N
23	71	N	Y
37	72	Y	N
41	83	N	Y
29	97	Y	Y
21	64	N	N
30	68	N	Y

- Returning to our data, let's model **Play** instead of **Temp**
- Again, we divide our data and learn how data **X** is related to data **Y**
- Again, assert: $\mathbf{Y} = f(\mathbf{X}) + \varepsilon$
- Want a model that is:
 - Supervised
 - Predicts categories/classes (**classification model**)
- **Q:** What model could we use?

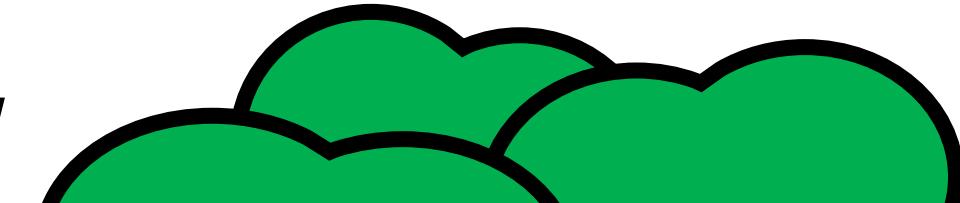
X			
Age	Temp	Rainy	Play
22	91	Y	N
29	89	N	Y
31	56	N	N
23	71	N	Y
37	72	Y	N
41	83	N	Y
29	97	Y	Y
21	64	N	N
30	68	N	Y

Playground of Models

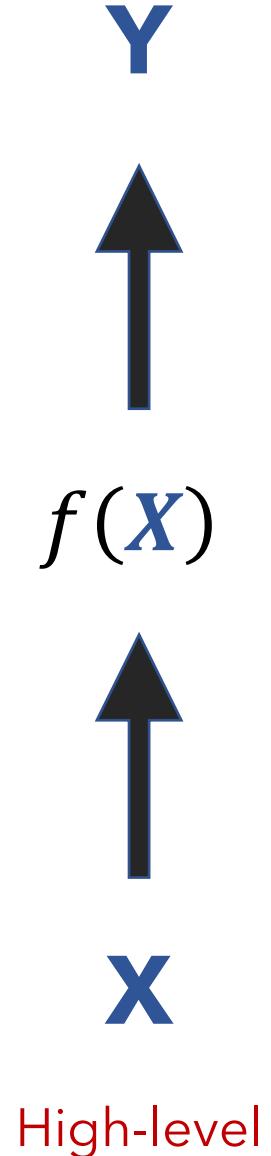
Linear
Regression



Playground of Models

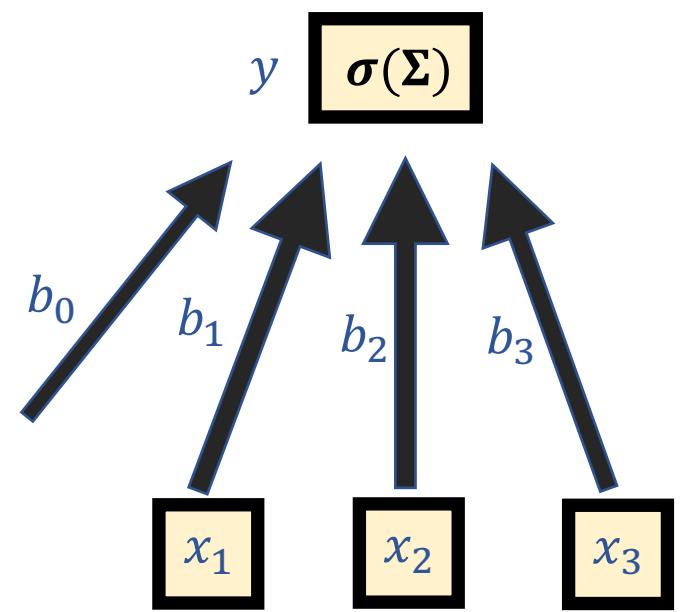
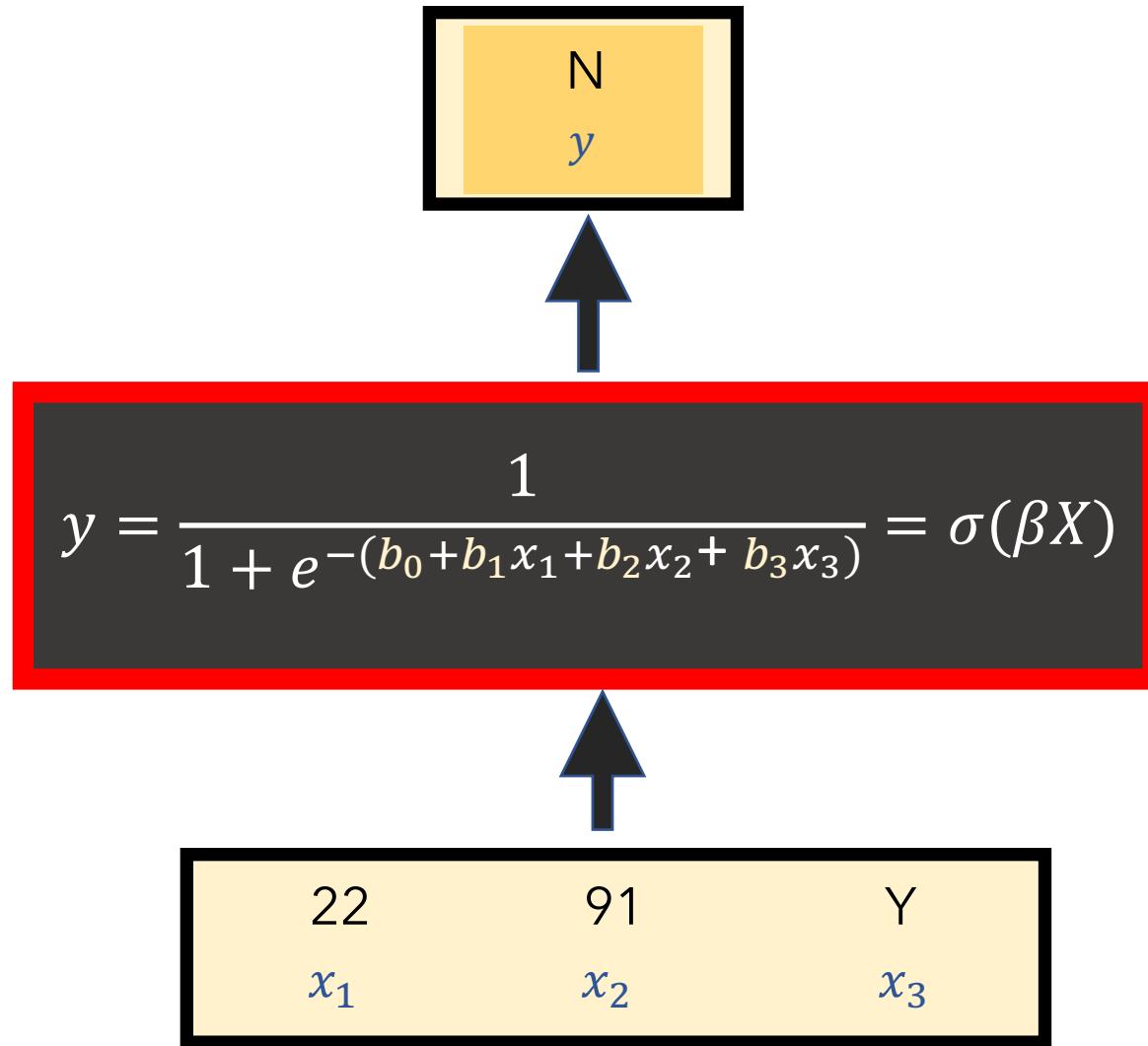


Logistic Regression



Logistic Regression

Y
↑
 $f(X)$
↑
X
High-level



Logistic Regression

This is a **non-linear activation function**, called a **sigmoid**.

Yet, our overall model is still considered **linear** w.r.t. the β coefficients.

It's a **generalized linear model**.

$f(X)$



X

High-level

$$y = \frac{1}{1 + e^{-(b_0 + b_1x_1 + b_2x_2 + b_3x_3)}} = \sigma(\beta X)$$



22
 x_1

91
 x_2

Y
 x_3

Mathematically

Logistic Regression

Q1

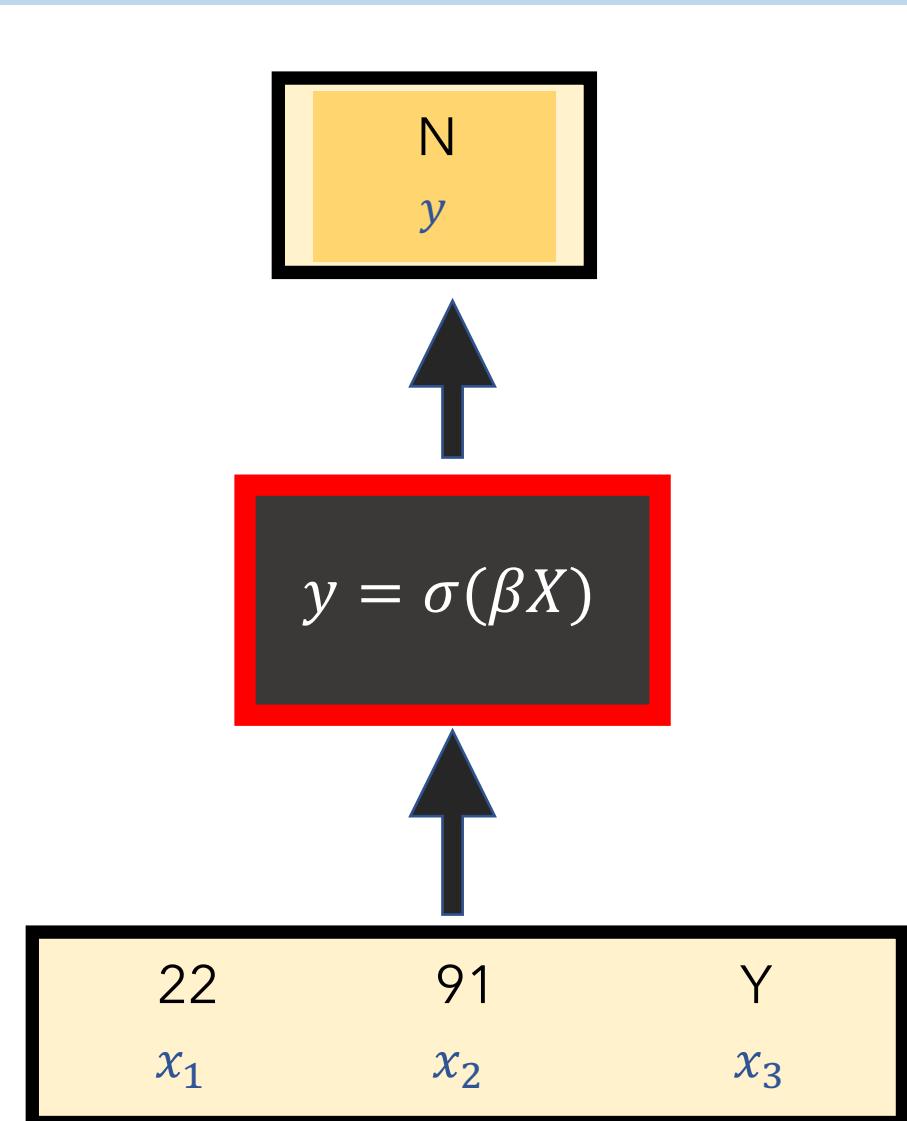
When training our model, how do we measure its m predictions \hat{y} ?

A1

Cost function

$$J(\theta) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

"Cross-Entropy" aka "Log loss"



Logistic Regression

Q1

When training our model, how do we measure its m predictions \hat{y} ?

A1

Cost function

$$J(\theta) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

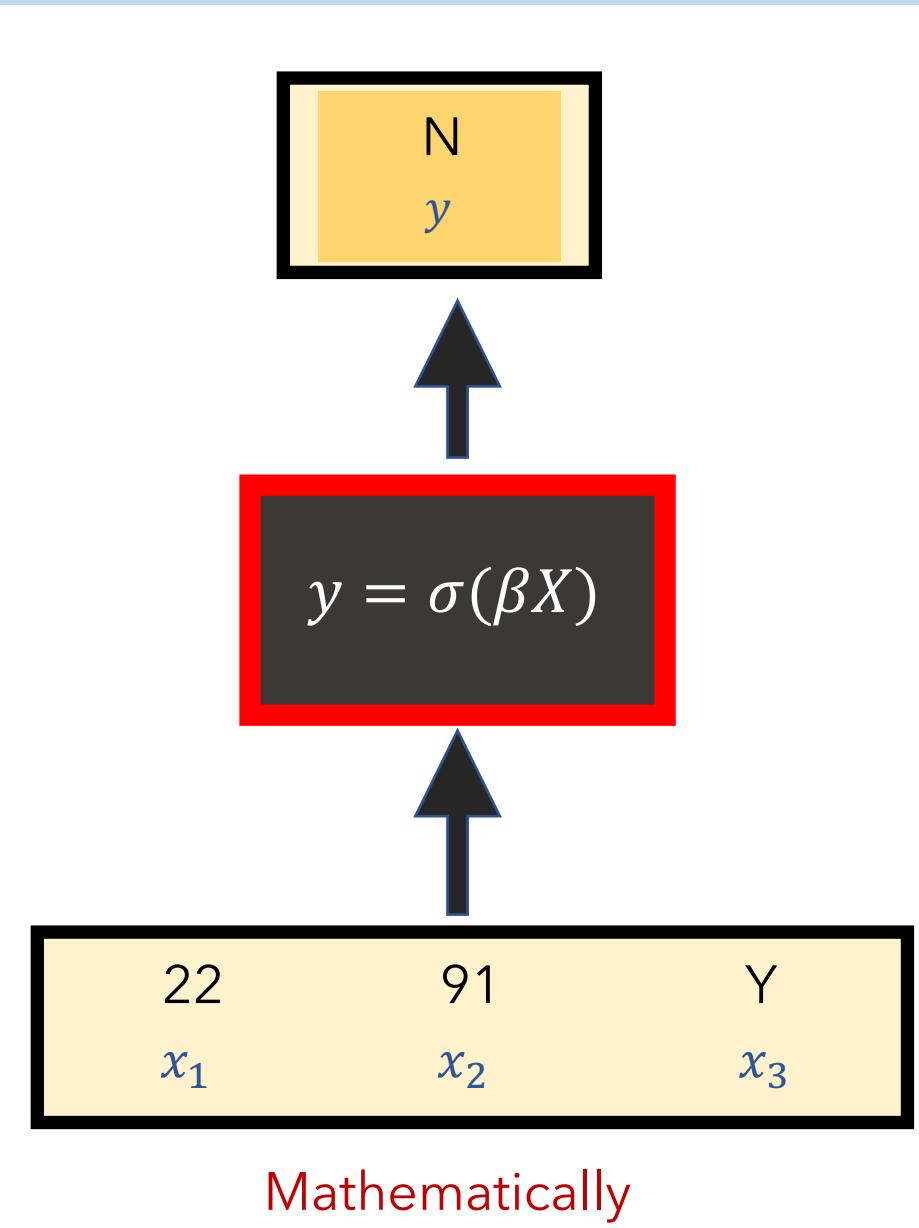
"Cross-Entropy" aka "Log loss"

Q2

How do we find the optimal θ so that we yield the best predictions?

A2

[Scikits](#) has many optimization solvers:
(e.g., liblinear, newton-cg, saga, etc)



Mathematically

Logistic Regression

Fitted model example

The plane is chosen to minimize the error of our class probabilities (per our loss function, **cross-entropy**) and the true labels (mapped to **0** or **1**)

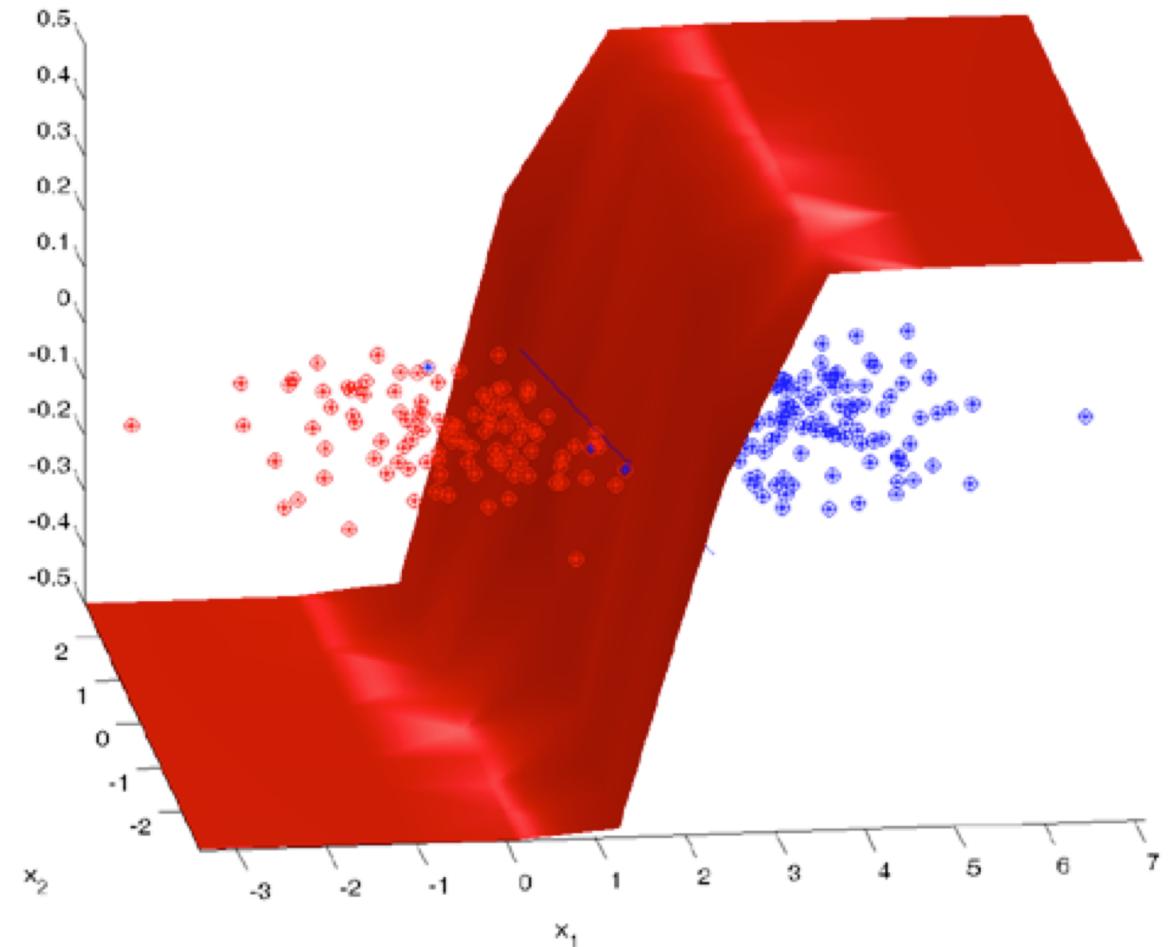


Photo from <http://strijov.com/sources/demoDataGen.php> (Dr. Vadim Strijov)

Parametric Models

- So far, we've assumed our data \mathbf{X} and \mathbf{Y} can be represented by an underlying model \mathbf{f} (i.e., $\mathbf{Y} = \mathbf{f}(\mathbf{X}) + \varepsilon$) that has a particular form (e.g., a linear relationship, hence our using a linear model)
- Next, we aimed to fit the model \mathbf{f} by estimating its parameters $\boldsymbol{\theta}$ (we did so in a **supervised** manner)

Parametric Models

- So far, we've assumed our data \mathbf{X} and \mathbf{Y} can be represented by an underlying model \mathbf{f} (i.e., $\mathbf{Y} = \mathbf{f}(\mathbf{X}) + \varepsilon$) that has a particular form (e.g., a linear relationship, hence our using a linear model)
- Next, we aimed to fit the model \mathbf{f} by estimating its parameters $\boldsymbol{\theta}$ (we did so in a **supervised** manner)
- Parametric models make the above assumptions. Namely, that there exists an underlying model \mathbf{f} that has a fixed number of parameters.

**Supervised vs
Unsupervised**

**Regression vs
Classification**

**Parametric vs
Non-Parametric**

Linear Regression

Supervised

Regression

Parametric

Logistic Regression

Supervised

Classification

Parametric

Non-Parametric Models

Alternatively, what if we make no assumptions about the underlying model f ? Specifically, let's **not assume f** :

- has any particular distribution/shape
(e.g., Gaussian, linear relationship, etc.)
- can be represented by a finite number of parameters.

Non-Parametric Models

Alternatively, what if we make no assumptions about the underlying model f ? Specifically, let's **not assume f** :

- has any particular distribution/shape
(e.g., Gaussian, linear relationship, etc.)
- can be represented by a finite number of parameters.

This would constitute a non-parametric model.

Non-Parametric Models

- Non-parametric models are **allowed to have parameters**; in fact, oftentimes the # of parameters grows as our amount of training data increases
- Since they make no strong assumptions about the form of the function/model, they are free to learn **any functional form** from the training data - *infinitely complex*.

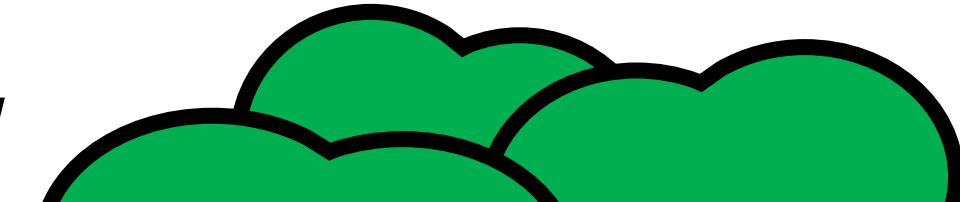
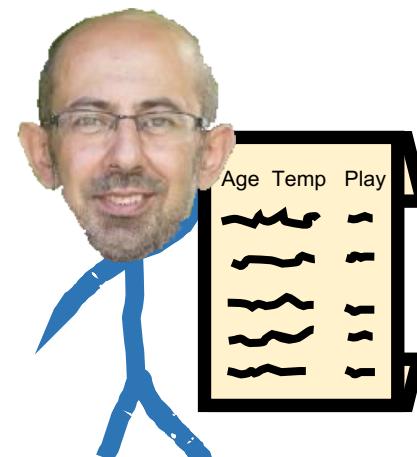
- Returning to our data, let's again predict if a person will **Play**
- If we do not want to assume anything about how **X** and **Y** relate, we could use a different **supervised** model
- Suppose we do not care to build a decision boundary but merely want to make predictions based on similar data that we saw during training

X	Y		
Age	Temp	Rainy	Play
22	91	Y	N
29	89	N	Y
31	56	N	N
23	71	N	Y
37	72	Y	N
41	83	N	Y
29	97	Y	Y
21	64	N	N
30	68	N	Y

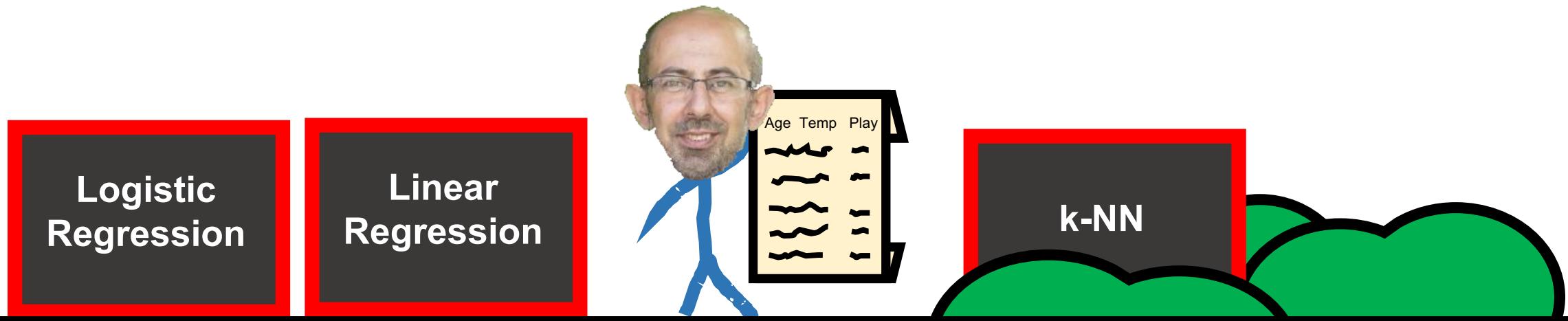
Playground of Models

Logistic
Regression

Linear
Regression

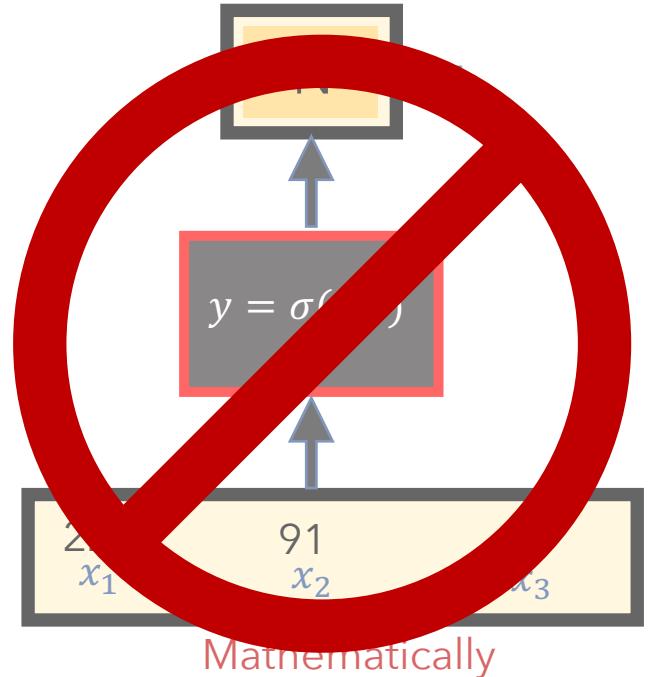


Playground of Models



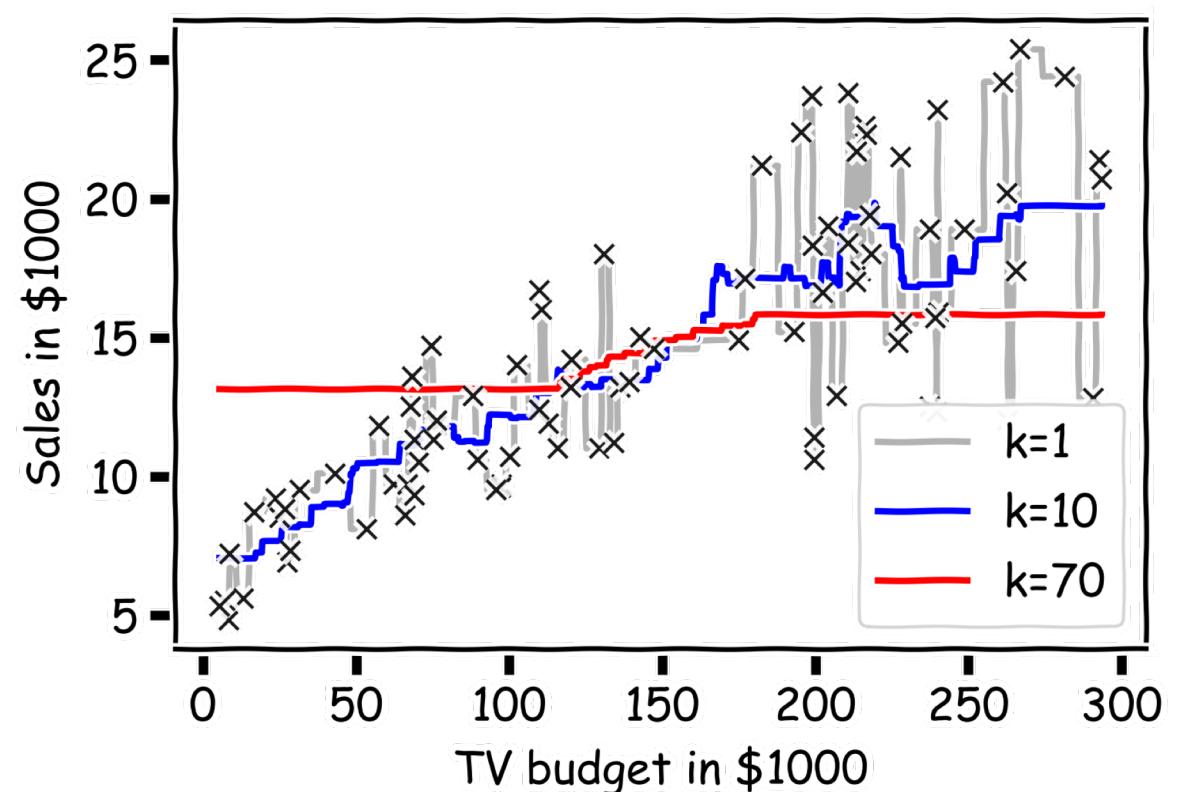
Refresher:

- k-NN doesn't train a model
- One merely specifies a ***k*** value
- At test time, a new piece of data ***a***:
 - must be compared to all other training data ***b***, to determine its k-nearest neighbors, per some distance metric ***d(a, b)***
 - is classified as being the majority class (if categorical) or average (if quantitative) of its k-neighbors



Conclusion:

- k-NN makes no assumptions about the data \mathbf{X} or the form of $f(\mathbf{X})$
- k-NN is a **non-parametric model**



PROS

- **Intuitive** and **simple** approach
- Can model any type of data / places no assumptions on the data
- Fairly robust to missing data
- Good for highly sparse data
(e.g., user data, where the columns are thousands of potential items of interest)

CONS

- Can be very **computationally expensive** if the data is large or high-dimensional
- Should carefully think about features, including scaling them
- Mixing quantitative and categorical data can be tricky
- Interpretation isn't meaningful
- Often, regression models are better, especially with little data

**Supervised vs
Unsupervised****Regression vs
Classification****Parametric vs
Non-Parametric**

Linear Regression

Supervised**Regression****Parametric**

Logistic Regression

Supervised**Classification****Parametric**

k-NN

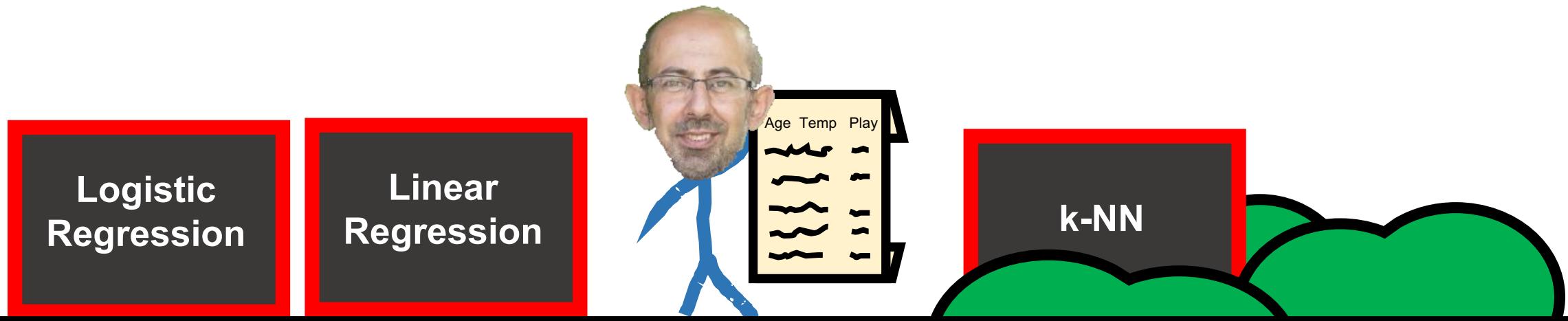
Supervised**either****Non-Parametric**

- Returning to our data yet again, let's predict if a person will **Play**

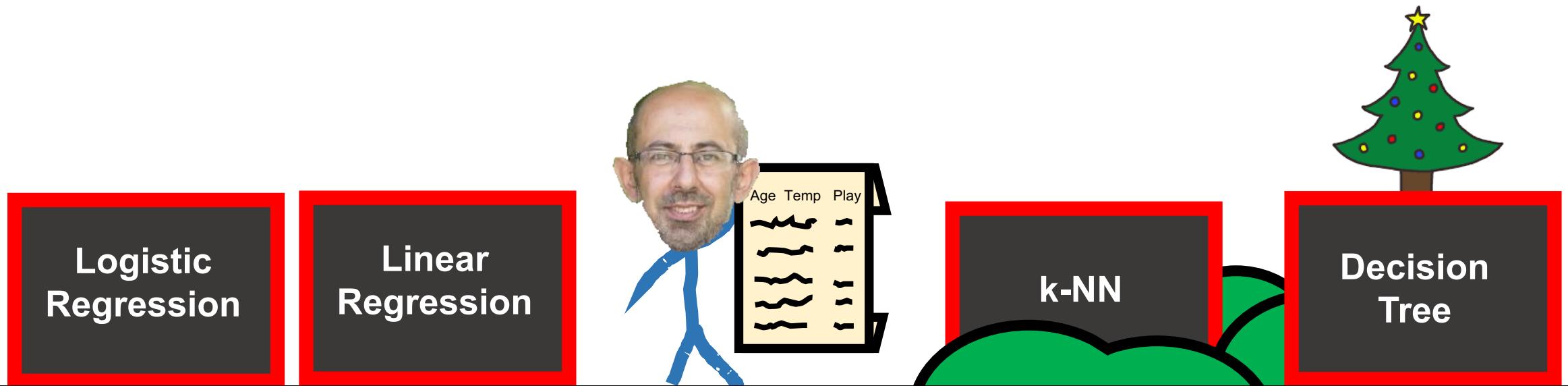
- If we do not want to assume anything about how **X** and **Y** relate, believing that no single equation can model the possibly non-linear relationship
- Suppose we just want our model to have robust decision boundaries with interpretable results

X	Y		
Age	Temp	Rainy	Play
22	91	Y	N
29	89	N	Y
31	56	N	N
23	71	N	Y
37	72	Y	N
41	83	N	Y
29	97	Y	Y
21	64	N	N
30	68	N	Y

Playground of Models



Playground of Models

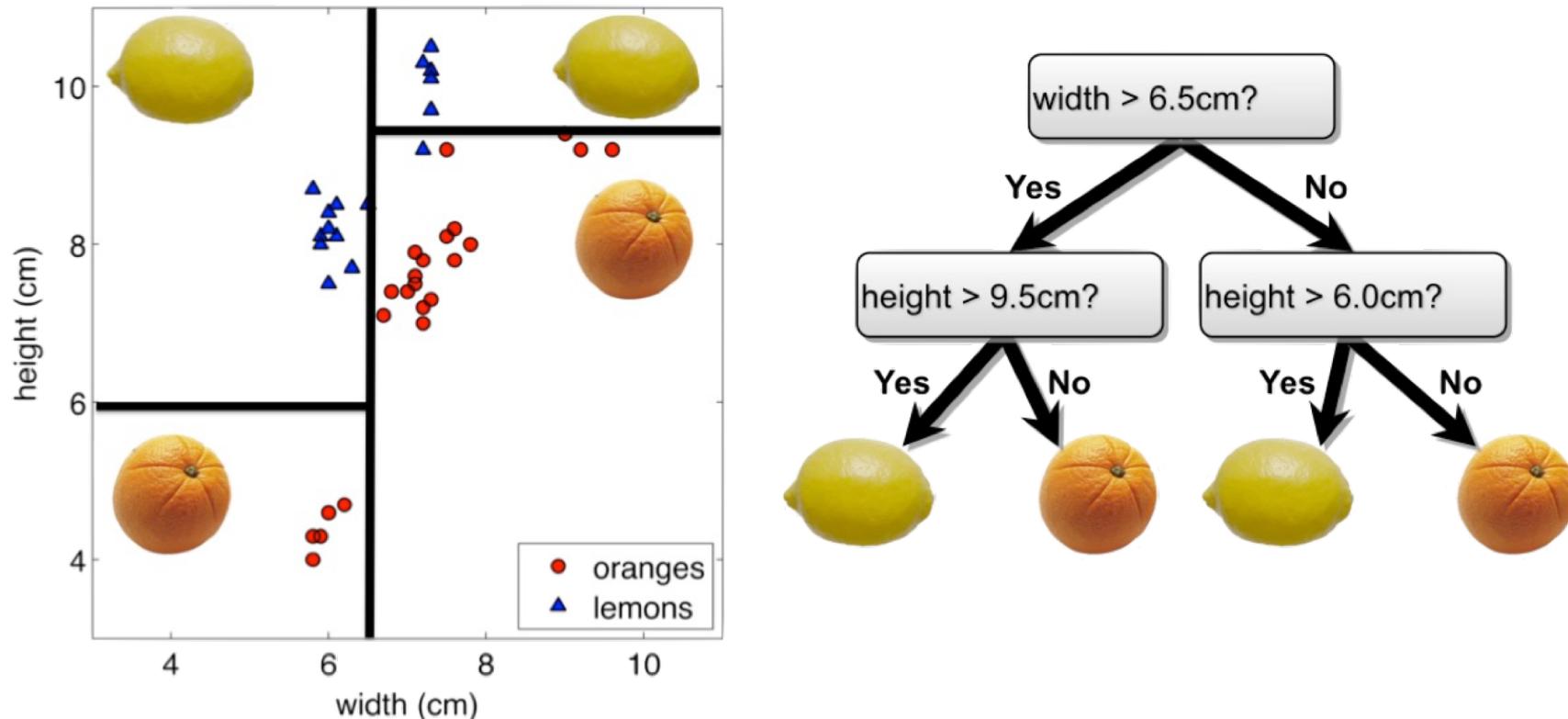


Refresher:

- A **Decision Tree** iteratively determines how to split our data by the best feature value so as to minimize the entropy (uncertainty) of our resulting sets.
- Must specify the:
 - Splitting criterion (e.g., Gini index, Information Gain)
 - Stopping criterion (e.g., tree depth, Information Gain Threshold)

Decision Tree

Refresher: Each comparison and branching represents splitting a region in the feature space on a single feature. Typically, at each iteration, we split once along one dimension (one predictor).



	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric
Linear Regression	Supervised	Regression	Parametric
Logistic Regression	Supervised	Classification	Parametric
k-NN	Supervised	either	Non-Parametric
Decision Tree	?	?	?

Decision Tree

- A **Decision Tree** makes no distributional assumptions about the data.
- The number of parameters / shape of the tree depends entirely on the data (i.e., imagine data that is perfectly separable into disjoint sections by features, vs data that is highly complex with overlapping values)
- Decision Trees make use of the full data (**X** and **Y**) and can handle **Y** values that are categorical or quantitative

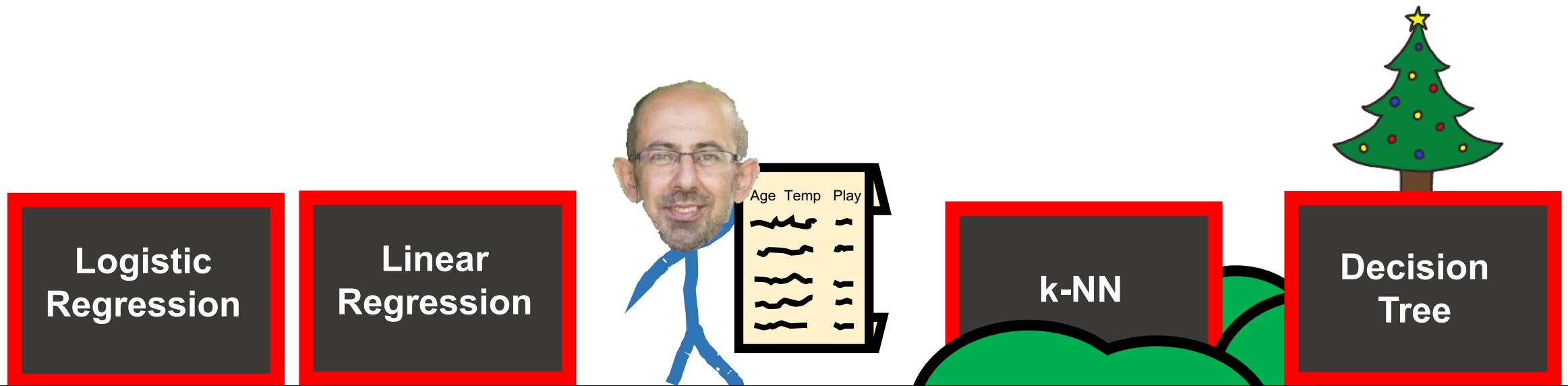
	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric
Linear Regression	Supervised	Regression	Parametric
Logistic Regression	Supervised	Classification	Parametric
k-NN	Supervised	either	Non-Parametric
Decision Tree	Supervised	either	Non-Parametric

Your Data \mathbf{X}

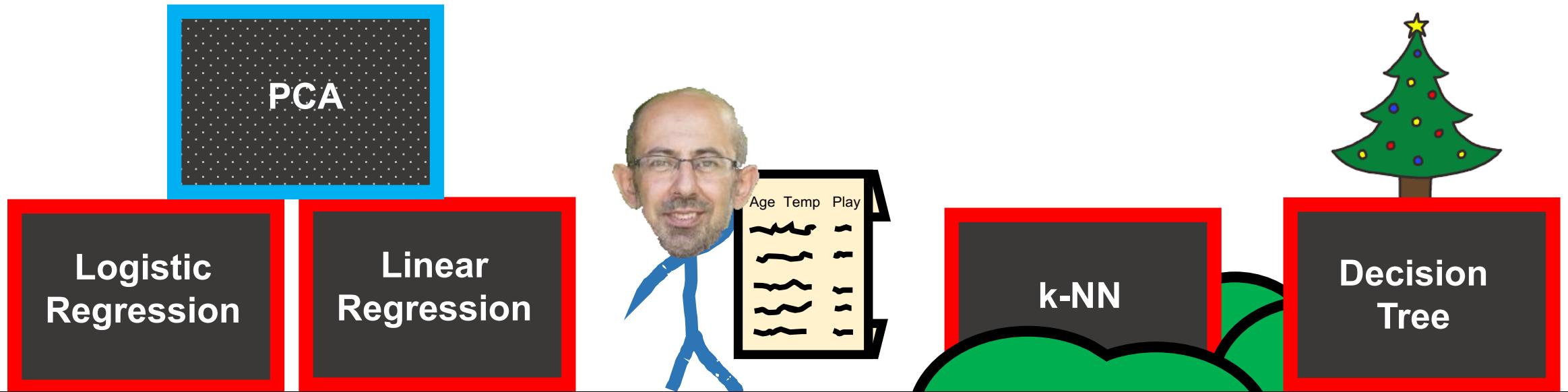
- Returning to our full dataset \mathbf{X} , imagine we do not wish to leverage any particular column \mathbf{Y} , but merely wish to **transform** the data into a smaller, useful representation $\hat{\mathbf{X}} = \mathbf{f}(\mathbf{X})$

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Playground of Models



Playground of Models



Principal Component Analysis (PCA)

Refresher:

- PCA isn't a model per se but is a procedure/technique to transform data, which may have correlated features, into a **new, smaller set of uncorrelated features**
- These new features, by design, are a linear combination of the original features so as to capture the most variance
- Often useful to perform PCA on data before using models that explicitly use data values and distances between them (e.g., clustering)

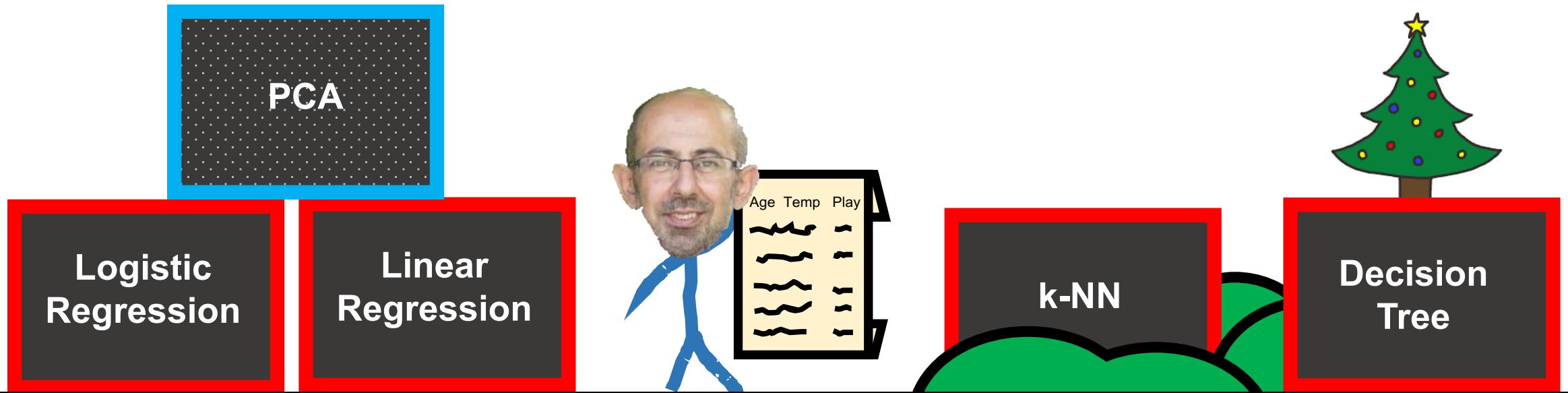
	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric
Linear Regression	Supervised	Regression	Parametric
Logistic Regression	Supervised	Classification	Parametric
k-NN	Supervised	either	Non-Parametric
Decision Tree	Supervised	either	Non-Parametric
PCA	Unsupervised	neither	Non-Parametric

Your Data X

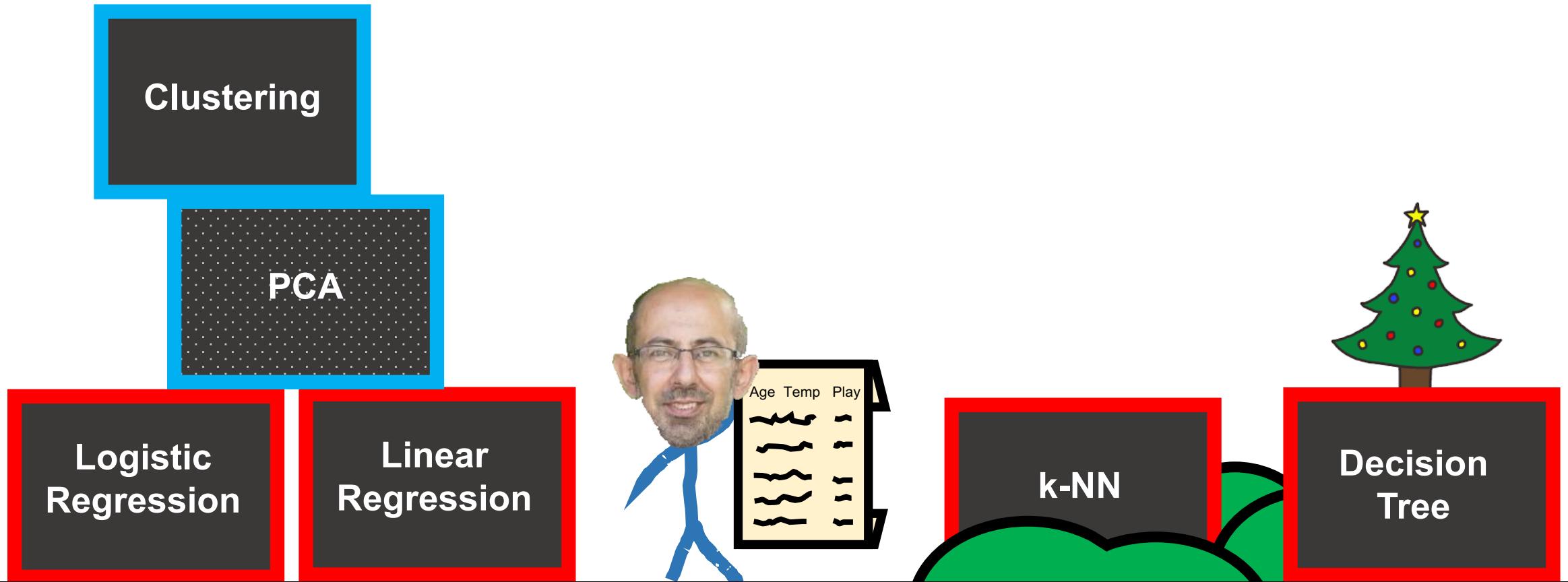
- Returning to our full dataset **X** yet again, imagine we do not wish to leverage any particular column **Y**, but merely wish to discern patterns/groups of similar observations

Age	Play	Rainy	Temp
22	N	Y	91
29	Y	N	89
31	N	N	56
23	Y	N	71
37	N	Y	72
41	Y	N	83
29	Y	Y	97
21	N	N	64
30	Y	N	68

Playground of Models



Playground of Models



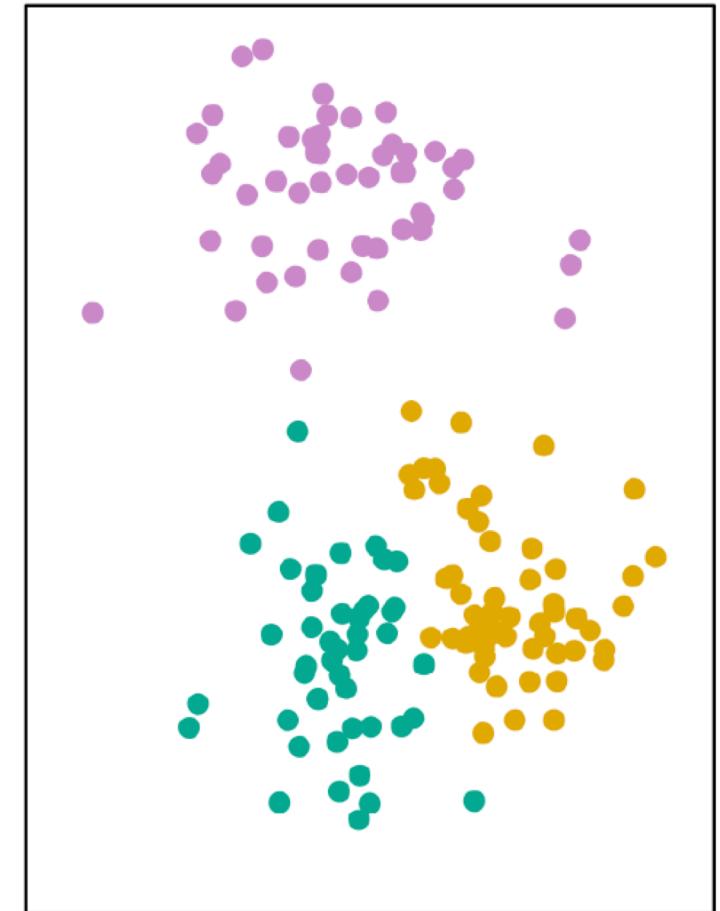
Refresher:

- There are many approaches to clustering
(e.g., k-Means, hierarchical, DBScan)
- Regardless of the approach, we need to specify a distance metric
(e.g., Euclidean, Manhattan)
- **Performance:** we can measure the intra-cluster and outer-cluster fit (i.e., silhouette score), along with an estimate that compares our clustering to the situation had our data been randomly generated (gap statistic)

Clustering

k-Means example:

- Although we are not explicitly using any column \mathbf{Y} , one could imagine that the 3 resulting cluster labels are our \mathbf{Y} 's (labels being class **1**, **2**, and **3**)
- Of course, we do not know these class labels ahead of time, as clustering is an unsupervised model

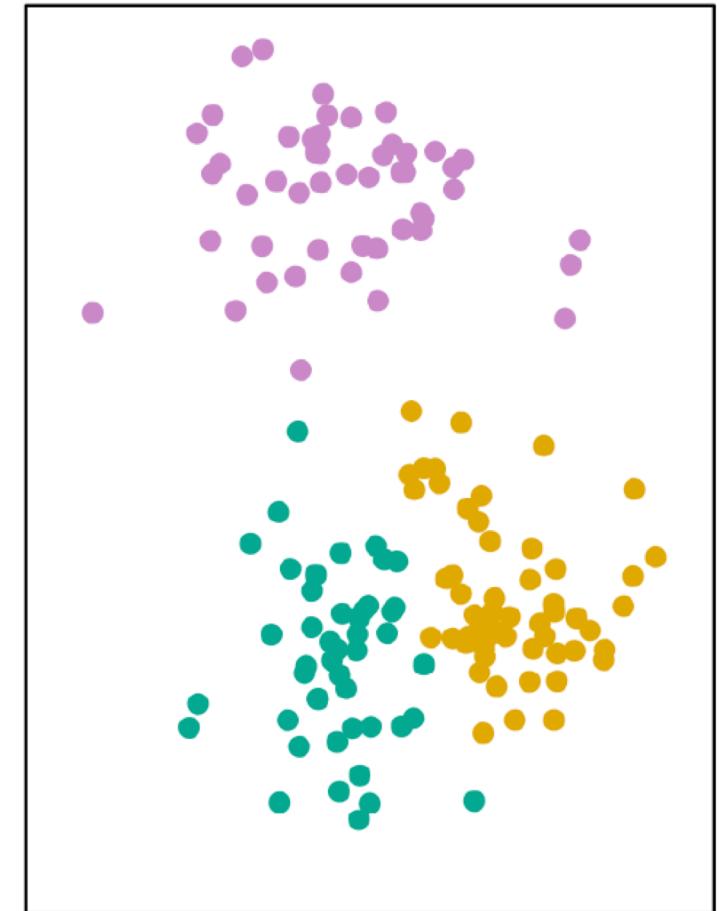


Visual Representation

Clustering

k-Means example:

- Although we are not explicitly using any column \mathbf{Y} , one could imagine that the 3 resulting cluster labels are our \mathbf{Y} 's (labels being class **1**, **2**, and **3**)
- Of course, we do not know these class labels ahead of time, as clustering is an unsupervised model
- Yet, one could imagine a narrative whereby our data points were **generated** by these 3 classes.

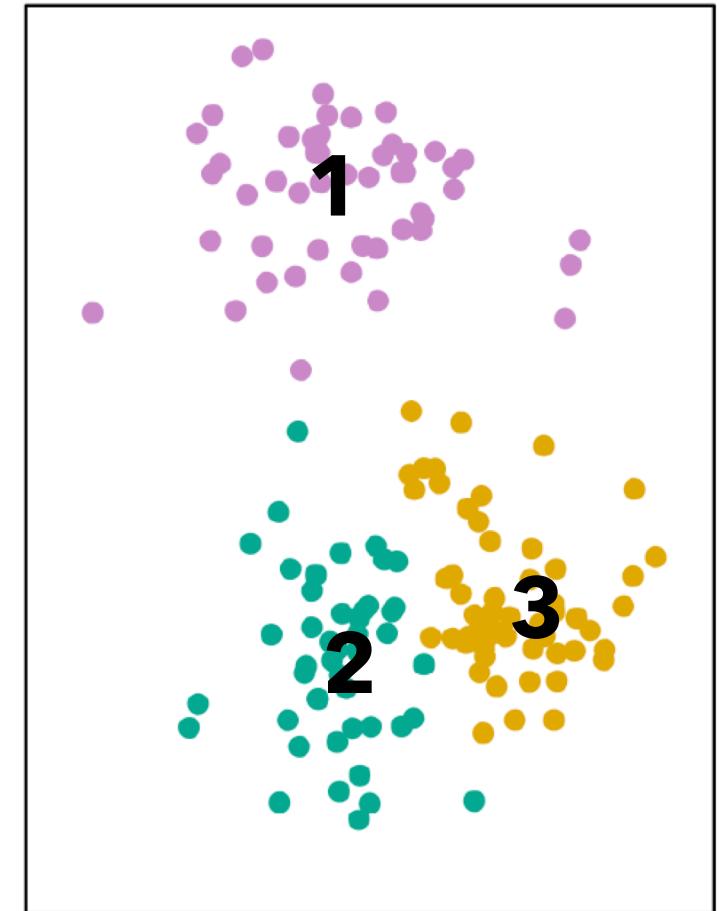


Visual Representation

Clustering

k-Means example:

- That is, we are flipping the modelling process on its head; instead of doing our traditional supervised modelling approach of trying to estimate $P(\mathbf{Y}|\mathbf{X})$:
 - Imagine centroids for each of the 3 clusters \mathbf{Y}_i . We assert that the data \mathbf{X} were generated from \mathbf{Y} .
 - We can estimate the joint probability of $P(\mathbf{Y}, \mathbf{X})$



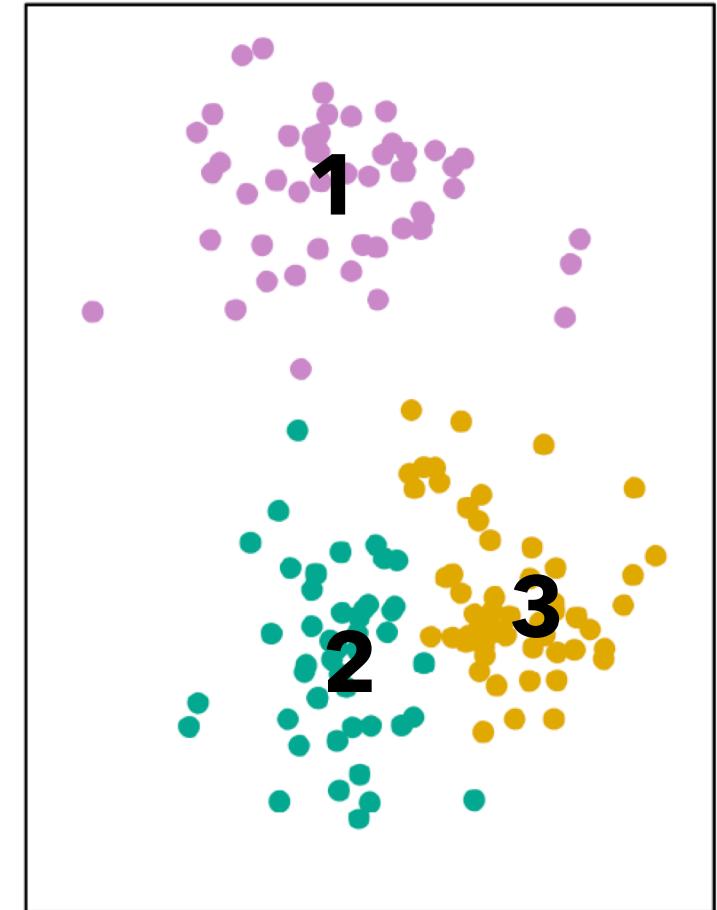
Visual Representation

Clustering

k-Means example:

Assuming our data was **generated** from Gaussians centered at 3 centroids, we can estimate the probability of the current situation - that the data \mathbf{X} exists and has the following class labels \mathbf{Y} . This is a **generative** model.

- We can estimate the joint probability of $P(\mathbf{Y}, \mathbf{X})$



Visual Representation

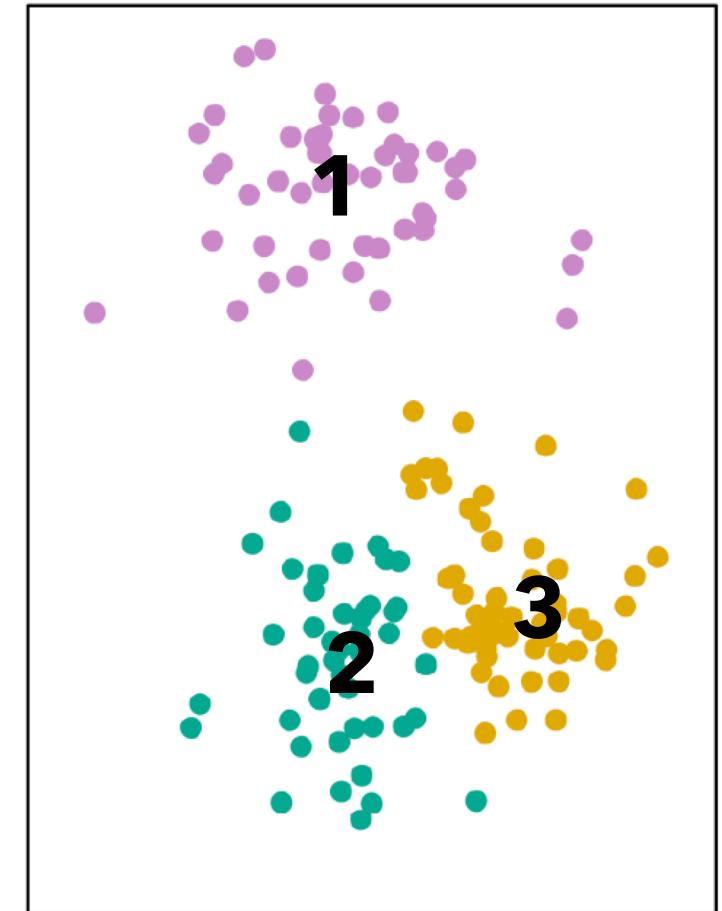
Clustering

k-Means example:

- That is, we are flipping the modelling process on its

Generative models explicitly model the actual distribution of each class (e.g., data and its cluster assignments).

- We can estimate the joint probability of $P(Y, X)$

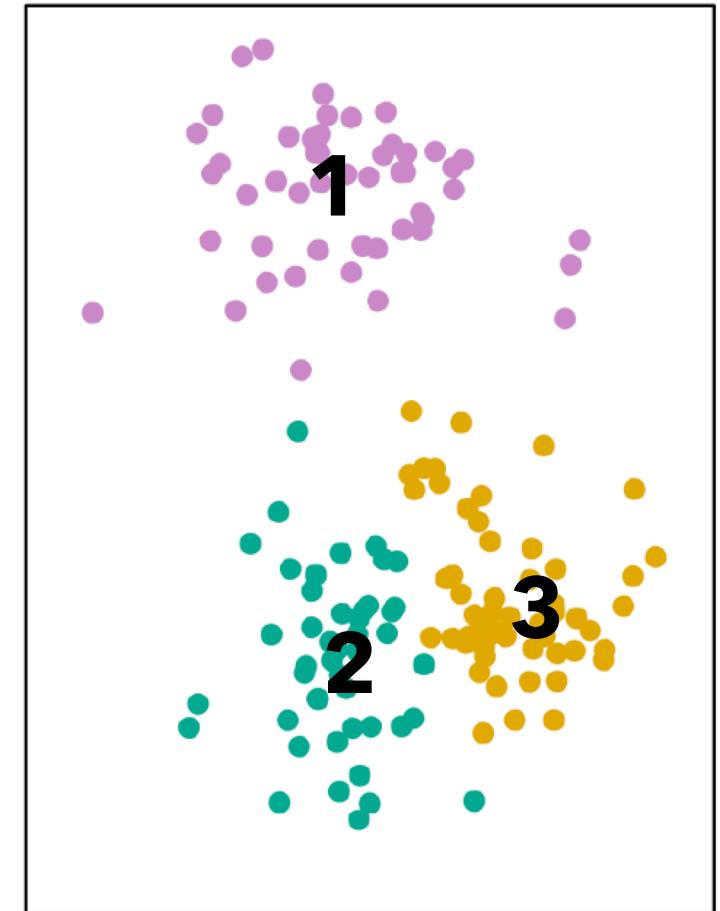


Visual Representation

Clustering

k-Means example:

- That is, we are flipping the modelling process on its head; instead of doing our traditional supervised modelling approach of trying to estimate $P(\mathbf{Y}|\mathbf{X})$:
 - Imagine centroids for each of the 3 clusters \mathbf{Y}_i . We assert that the data \mathbf{X} were generated from \mathbf{Y} .
 - We can estimate the joint probability of $P(\mathbf{Y}, \mathbf{X})$



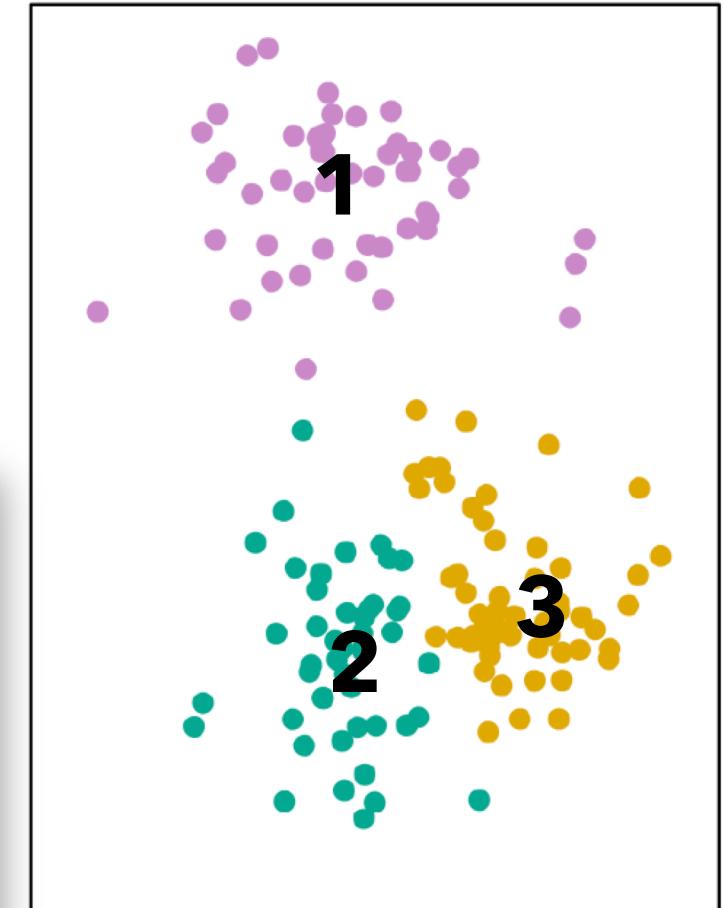
Visual Representation

Clustering

k-Means example:

- That is, we are flipping the modelling process on its head; instead of doing our traditional supervised modelling approach of trying to estimate $P(Y|X)$:

Supervised models are given some data X and want to calculate the probability of Y . They learn to **discriminate** between different values of possible Y 's (learns a **decision boundary**).



Visual Representation

	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
PCA	Unsupervised	neither	Non-Parametric	neither
Clustering	Unsupervised	neither	Non-Parametric	Generative

Supervised vs Unsupervised

Regression vs Classification

Parametric vs Non-Parametric

Generative vs Discriminative

Linear Regression

Supervised

Regression

Parametric

Discriminative

Logistic Regression

Supervised

Classification

Parametric

Discriminative

k-NN

Supervised

Particularly, **k-Means** is
generative, as it can be seen
as a special case of
Gaussian Mixture Models

Decision Tree

Supervised

PCA

Unsupervised

Clustering

Unsupervised

neither

Generative

**Supervised vs
Unsupervised****Regression vs
Classification****Parametric vs
Non-Parametric****Generative vs
Discriminative**

Linear Regression

Supervised
Given training X , learns to
discriminate between possible
 Y values (quantitative)

ic

Discriminative

Logistic Regression

Supervised

ic

Discriminative

k-NN

Supervised

metric

Discriminative

Decision Tree

Supervised

Non-Parametric

Discriminative

PCA

Unsupervised

Non-Parametric

neither

Clustering

Unsupervised

Non-Parametric

Generative

**Supervised vs
Unsupervised****Regression vs
Classification****Parametric vs
Non-Parametric****Generative vs
Discriminative**

Linear Regression

Supervised

Regression

Parametric

Discriminative

Logistic Regression

Supervised

Regression

Parametric

Discriminative

k-NN

Supervised

Regression

Parametric

Discriminative

Decision Tree

Supervised

Regression

Parametric

Discriminative

PCA

Unsupervised

neither

Non-Parametric

neither

Clustering

Unsupervised

neither

Non-Parametric

Generative

Given training X , learns to discriminate between possible Y classes (categorical)

**Supervised vs
Unsupervised****Regression vs
Classification****Parametric vs
Non-Parametric****Generative vs
Discriminative**

Linear Regression

Supervised

Regression

Parametric

Discriminative

Logistic Regression

Supervised

Classification

Parametric

Discriminative

k-NN

Given training X , learns to
discriminate between possible Y
values (quantitative or categorical)

metric

metric

metric

Discriminative

Decision Tree

PCA

Clustering

Unsupervised

neither

Non-Parametric

neither

Generative

	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Given training \mathbf{X} , learns decision boundaries so as to discriminate between possible \mathbf{Y} values (quantitative or categorical)			
PCA				
Clustering				

	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
PCA	<p>PCA is a process, not a model, so it doesn't make sense to consider it as a Discriminate or Generative model</p>			
Clustering				Generative

	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
PCA	Unsupervised	neither	Non-Parametric	neither
Clustering	Unsupervised	neither	Non-Parametric	Generative

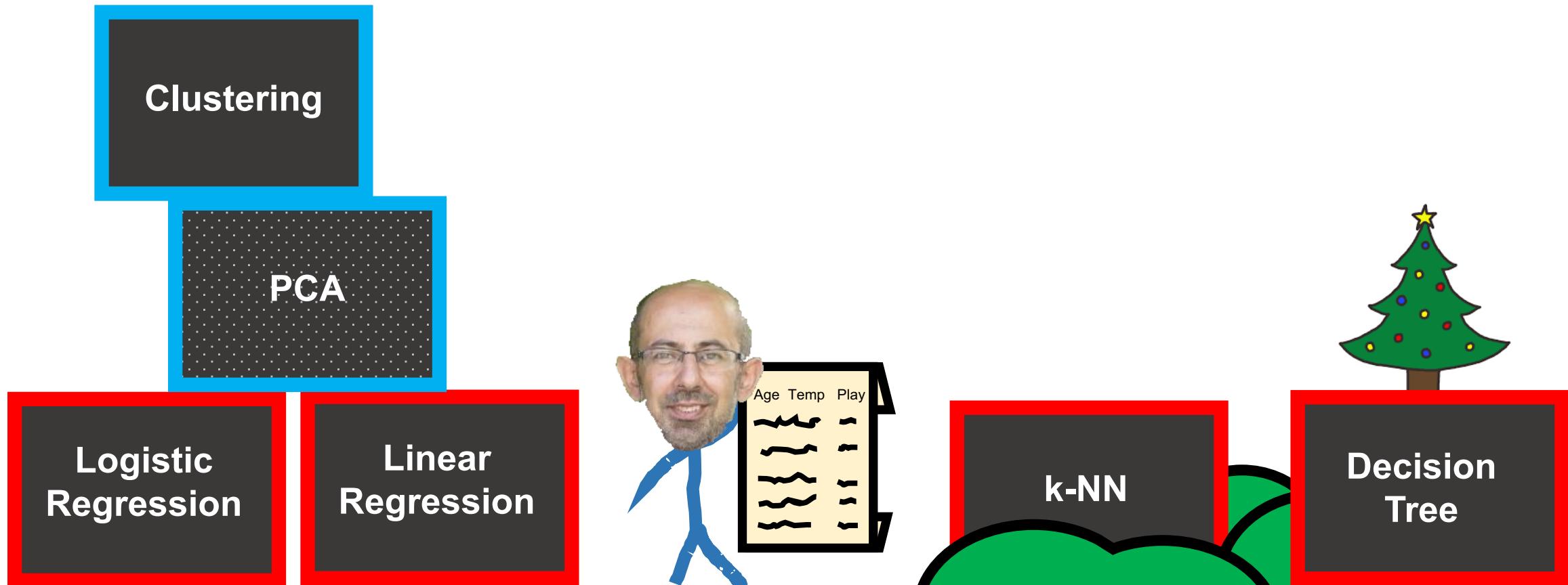
- Returning our data yet again, perhaps we've plotted our data

X and see it's **non-linear**

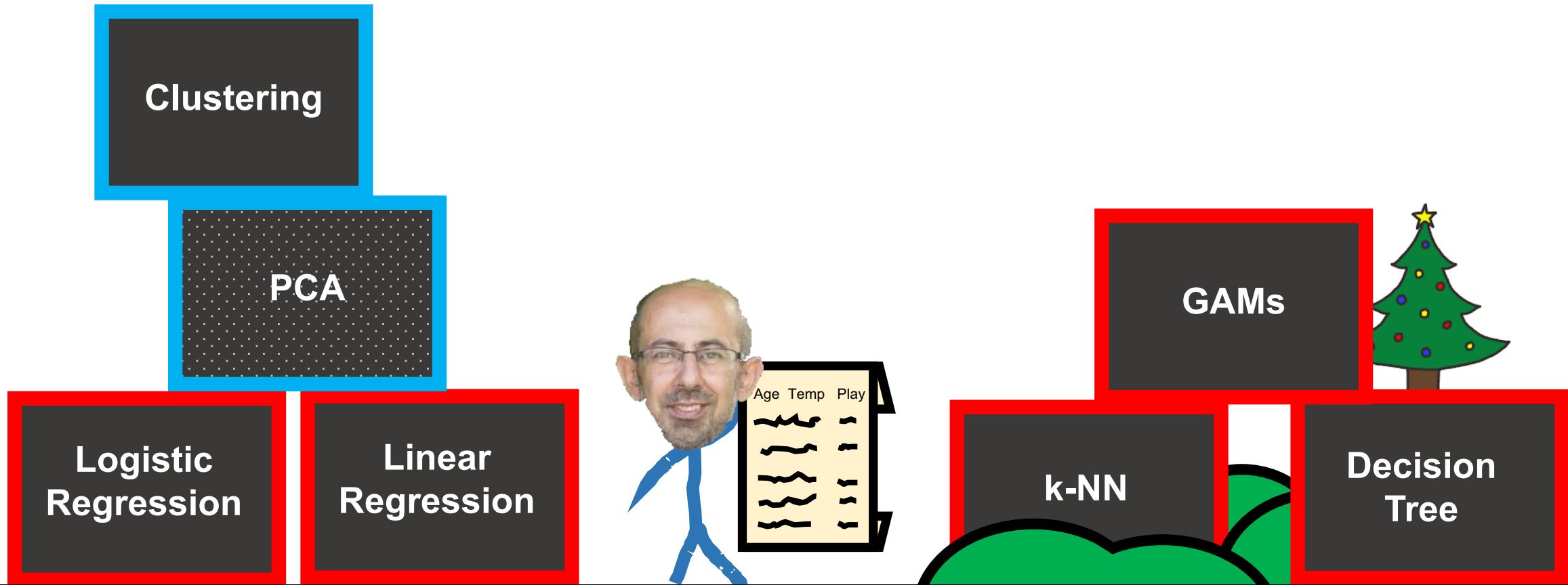
- Knowing how unnatural and finnicky polynomial regression can be, we prefer to let our model learn how to make its own non-linear functions for each feature **x_i**

X	Age	Play	Rainy	Y	Temp
	22	N	Y		91
	29	Y	N		89
	31	N	N		56
	23	Y	N		71
	37	N	Y		72
	41	Y	N		83
	29	Y	Y		97
	21	N	N		64
	30	Y	N		68

Playground of Models



Playground of Models

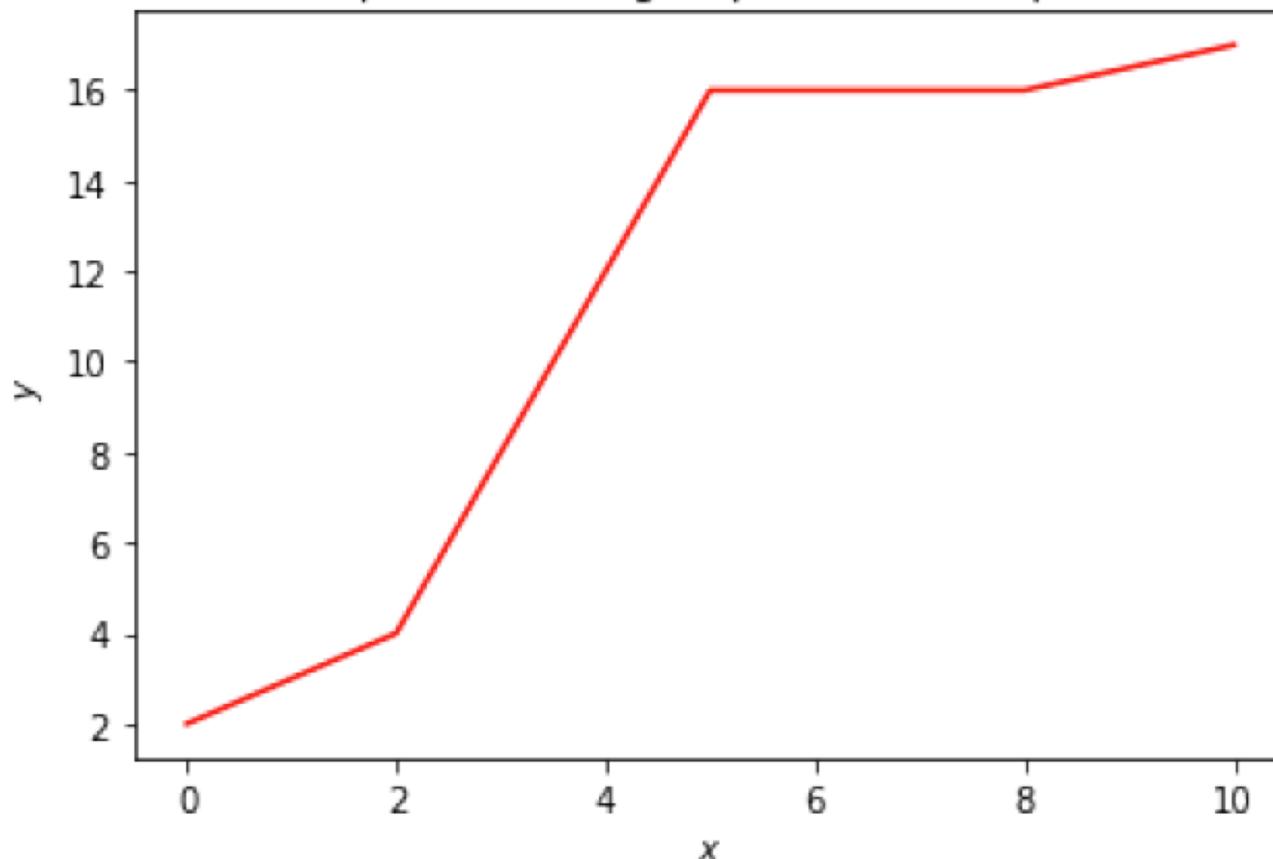


Generalized Additive Models (GAMs)

Refresher:

Not our data, but imagine it's plotting **age** vs **temp**:

Piecewise linear spline with knots at $x=2$, 5 , and 8
plus a starting slope and intercept



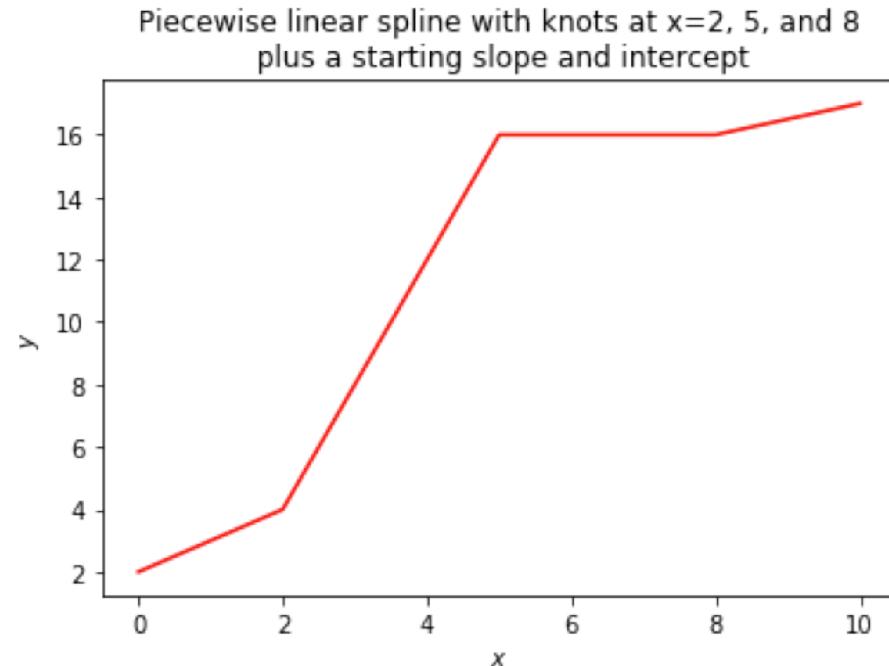
Generalized Additive Models (GAMs)

Refresher:

- We can make the line smoother by using a **cubic spline** or “**B-spline**”
- Imagine having 3 of these models:
 - $f_1(\text{age})$
 - $f_2(\text{play})$
 - $f_3(\text{rainy})$
- We can model **Temp** as:

$$\text{Temp} = \beta_0 + f_1(\text{age}) + f_2(\text{play}) + f_3(\text{rainy})$$

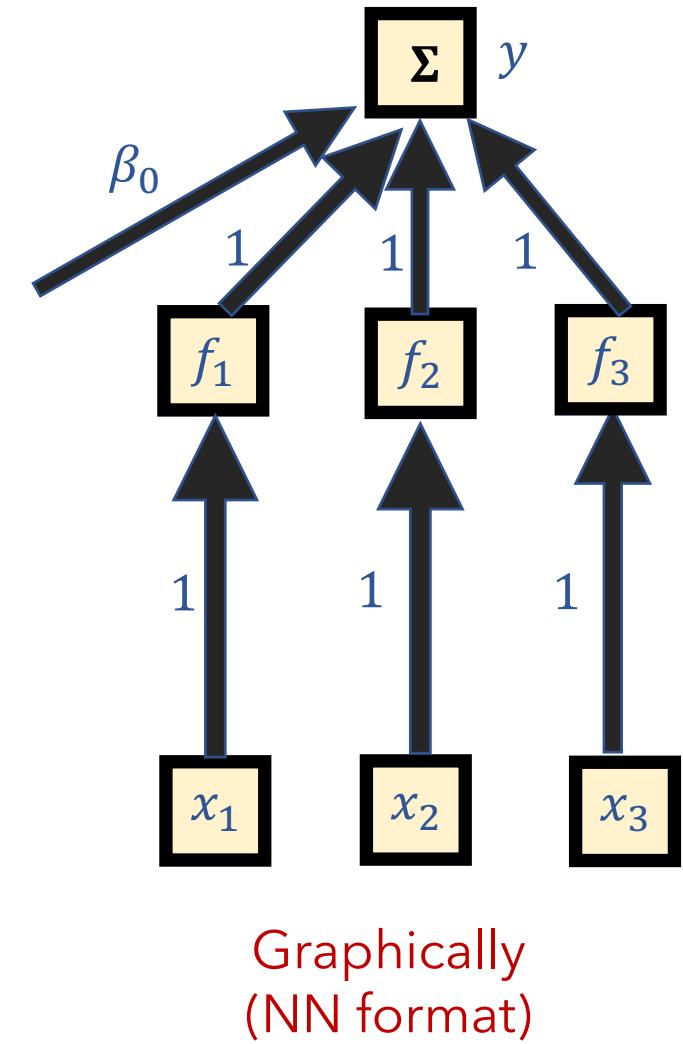
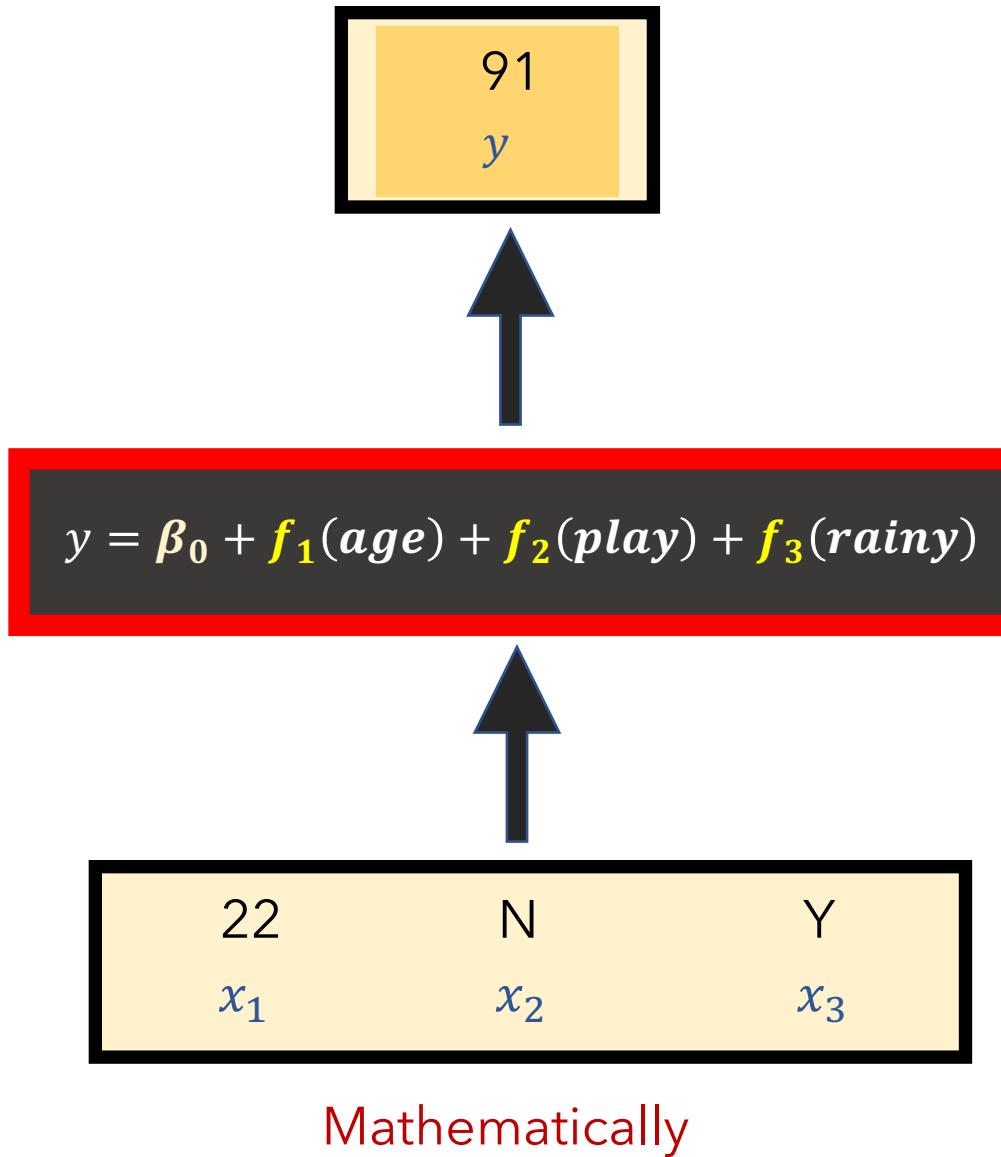
Not our data, but imagine it's plotting **age** vs **Temp**:



Generalized Additive Models (GAMs)

Y
↑
 $f(X)$
↑
X

High-level



Generalized Additive Models (GAMs)

It is called an additive model because we calculate a separate f_i for each x_i , and then add together all of their contributions.

$$f(X)$$

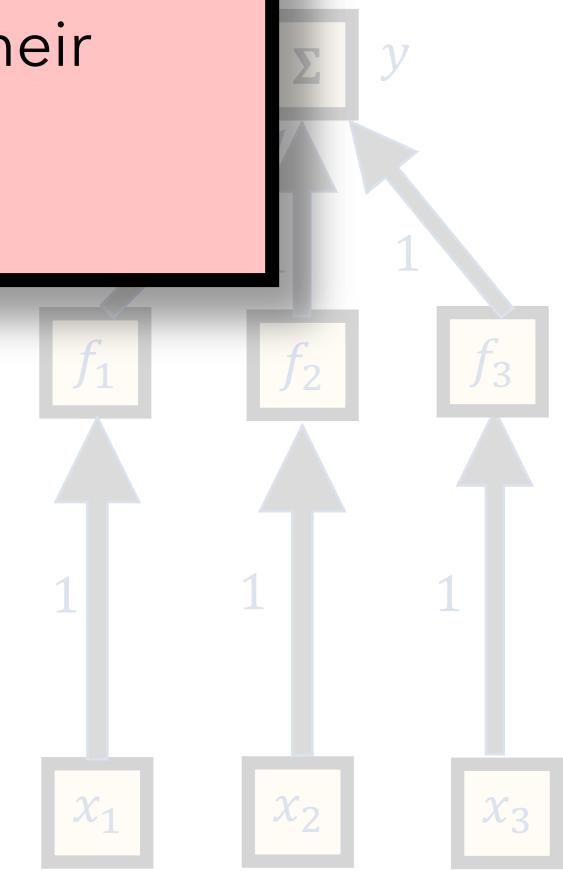
$$y = \beta_0 + f_1(\text{age}) + f_2(\text{play}) + f_3(\text{rainy})$$

$$X$$

High-level



Mathematically



Graphically
(NN format)

Generalized Additive Models (GAMs)

It is called an additive model because we calculate a separate f_i for each x_i , and then add together all of their contributions.

$f(X)$

$$y = \beta_0 + f_1(\text{age}) + f_2(\text{play}) + f_3(\text{rainy})$$

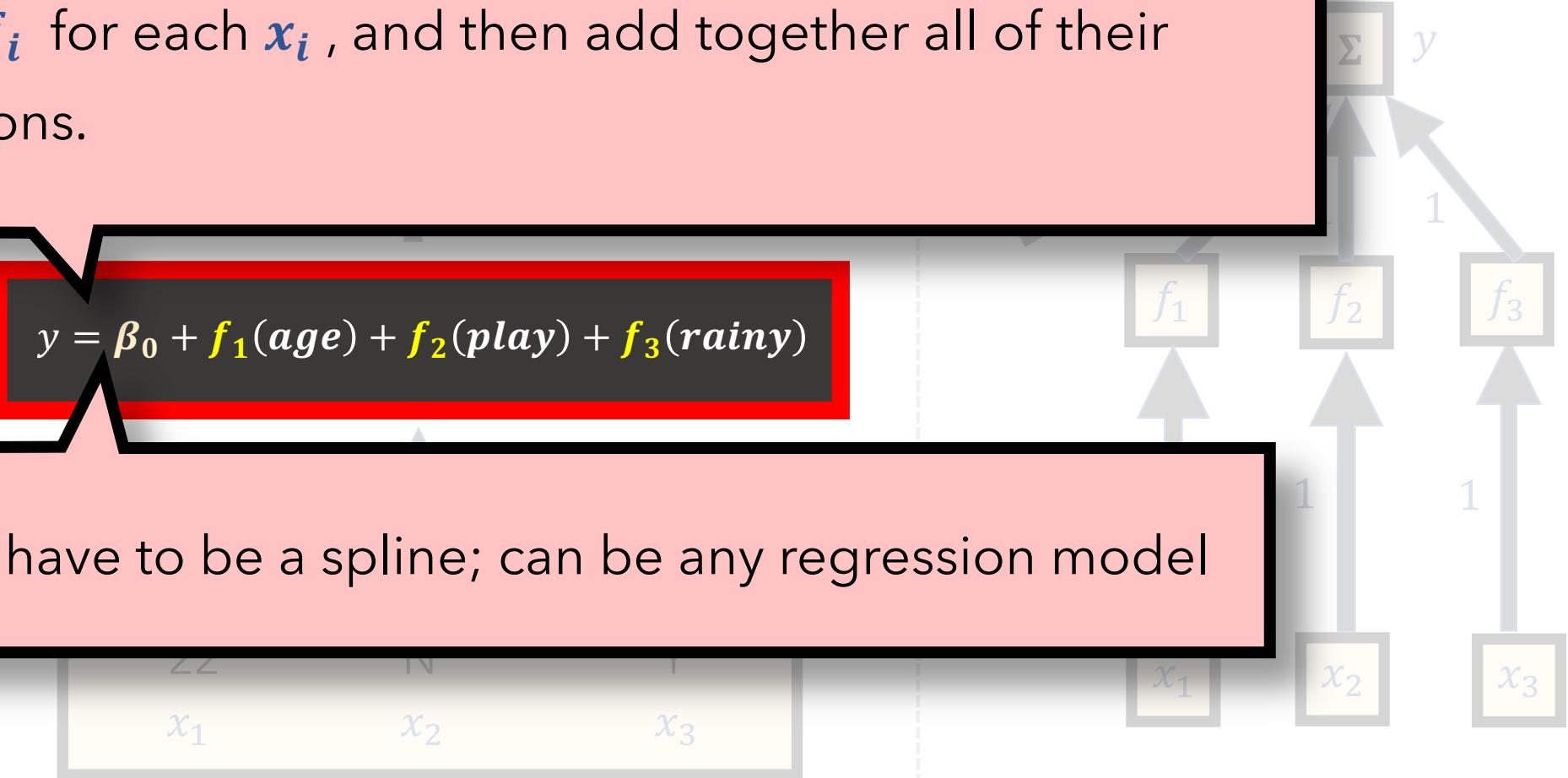
f_i doesn't have to be a spline; can be any regression model

X

High-level

Mathematically

Graphically
(NN format)



Generalized Additive Models (GAMs)

PROS

- Fits a **non-linear** function f_i to each feature x_i
- Much easier than guessing polynomial terms and multinomial interaction terms.
- Model is additive, allowing us to exam the effects of each x_i on y by holding the other features $x_{j \neq i}$ constant
- The smoothness is easy to adjust

CONS

- Restricted to being additive; important interactions may not be captured
- Providing interactions via $f_1(age, rainy)$ can only capture so much, a la multinomial interaction terms

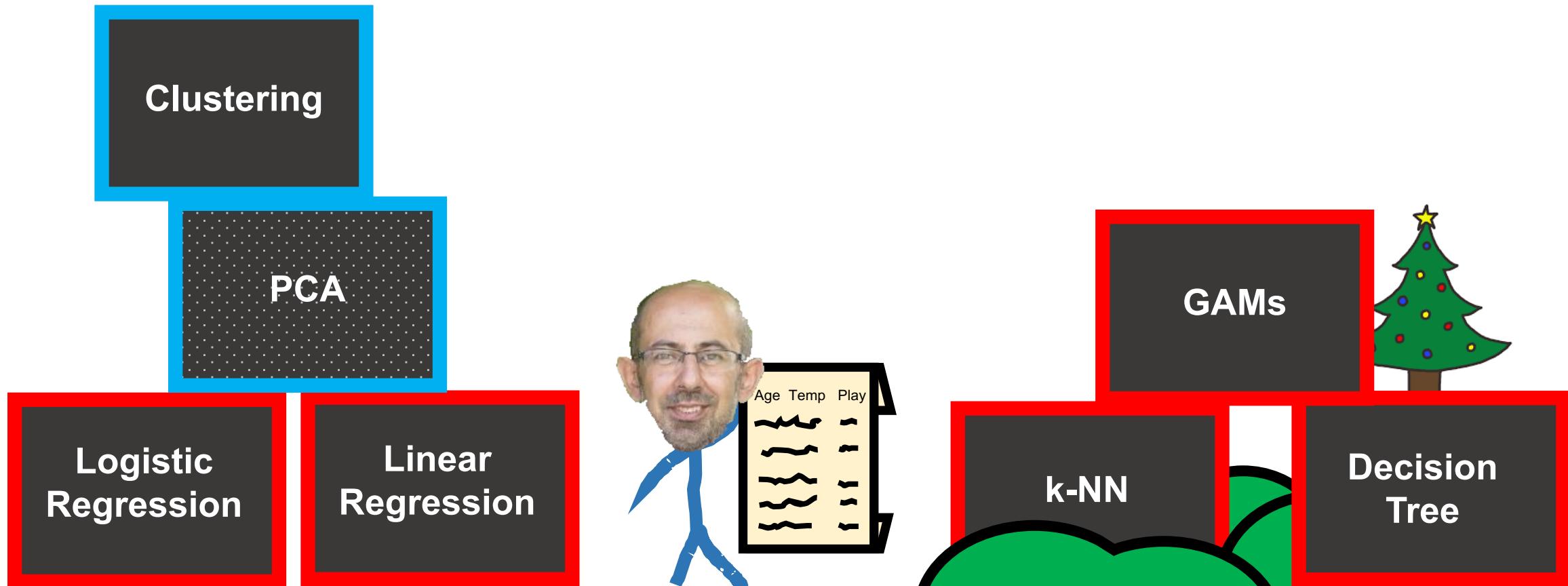
	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
PCA	Unsupervised	neither	Non-Parametric	neither
Clustering	Unsupervised	neither	Non-Parametric	Generative
GAMs	Supervised	either	Parametric	Discriminative

- Returning our data yet again, perhaps we've plotted our data **X** and see it's **non-linear**

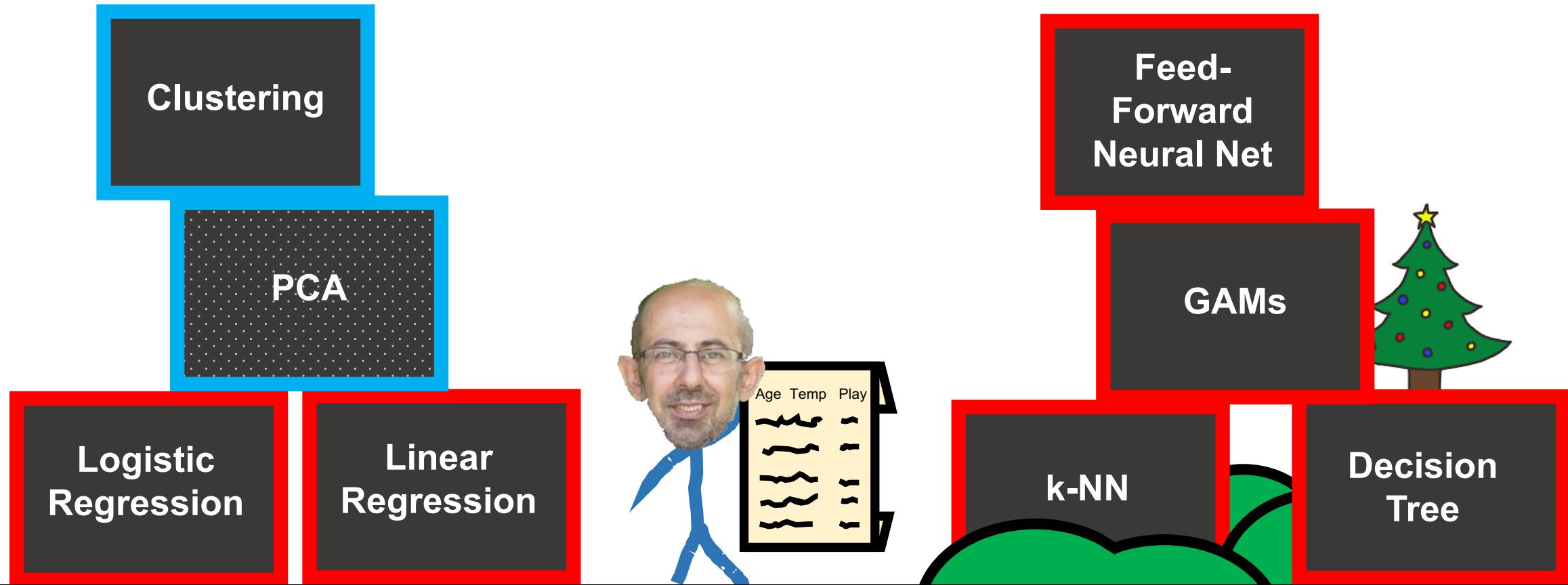
- We further suspect that there are complex interactions that cannot be represented by polynomial regression and GAMs
- We just want great results and don't care about interpretability

X	Y		
Age	Temp	Rainy	Play
22	91	Y	N
29	89	N	Y
31	56	N	N
23	71	N	Y
37	72	Y	N
41	83	N	Y
29	97	Y	Y
21	64	N	N
30	68	N	Y

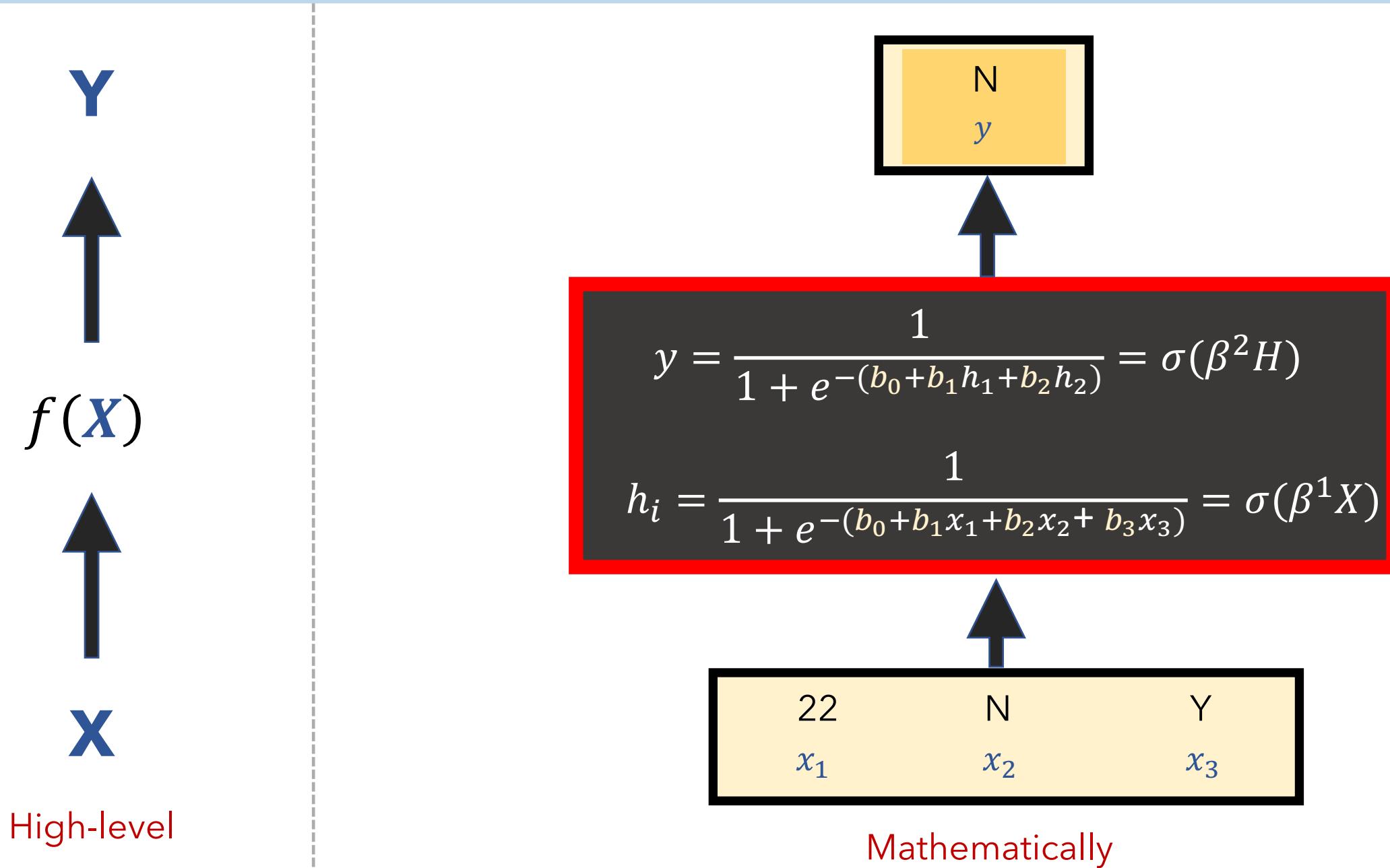
Playground of Models



Playground of Models

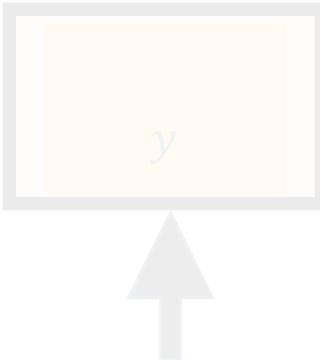


Feed-Forward Neural Network



Feed-Forward Neural Network

Y
↑
 $f(X)$
↑
X
High-level



$$y = \frac{1}{1 + e^{-(b_0 + b_1 h_1 + b_2 h_2)}} = \sigma(\beta^2 H)$$

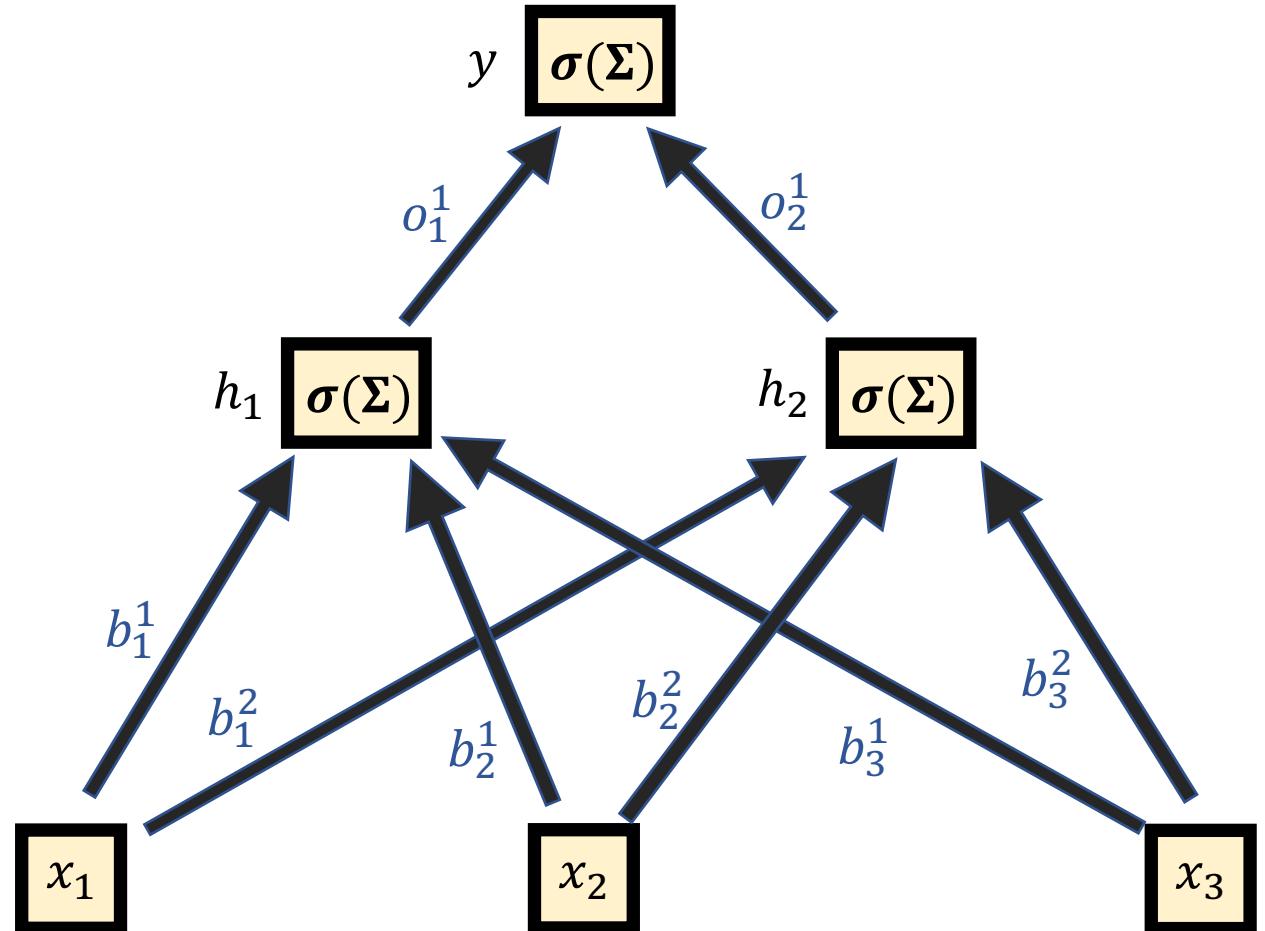
$$h_i = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3)}} = \sigma(\beta^1 X)$$

NOTE: a Neural Network can be viewed as a function $f(X)$, just like all of our past models

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  is a weight, which is multiplied by its input
- Every  is a scalar value
- Parameters $\theta = \{\beta, O\}$ (weights)

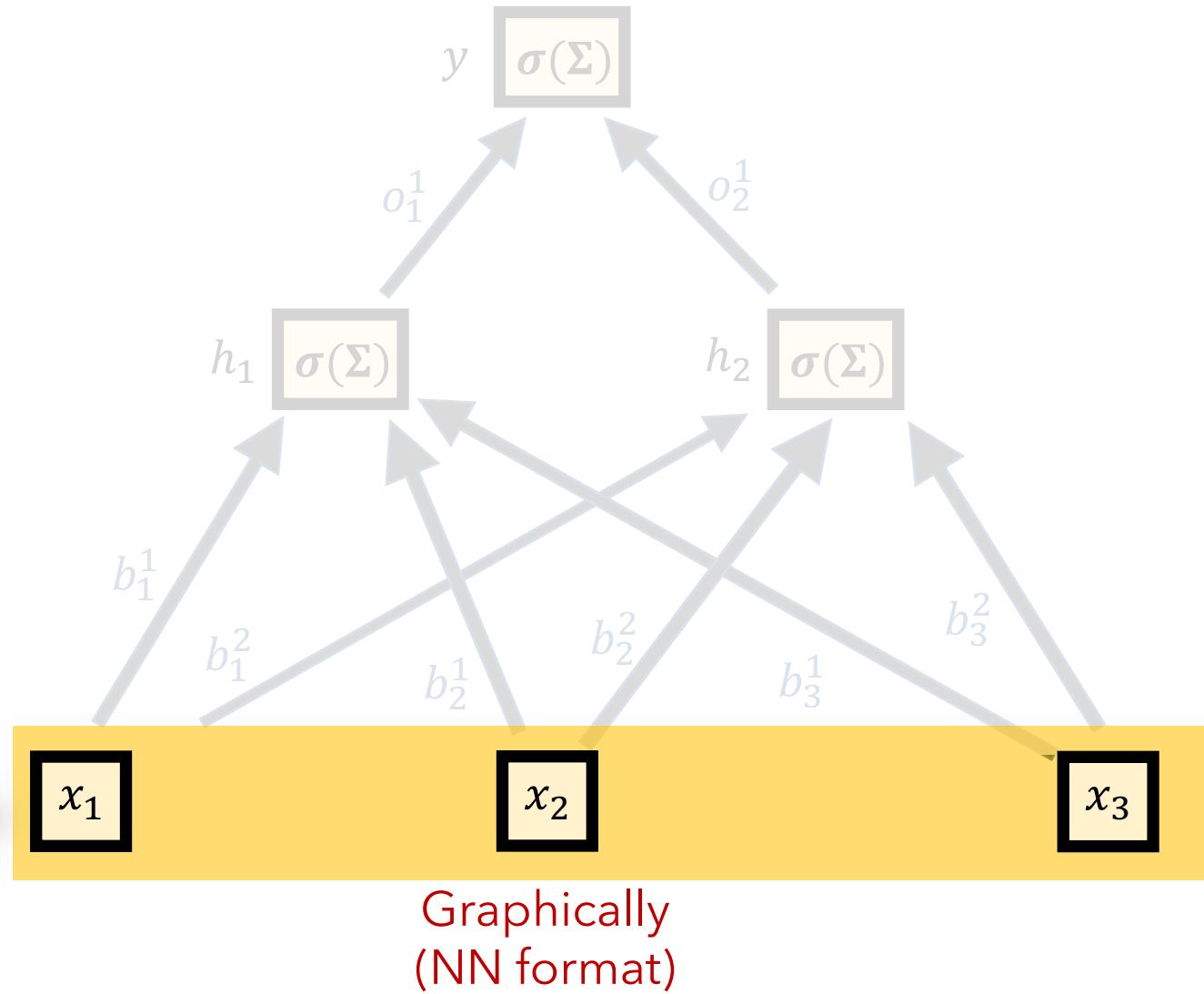
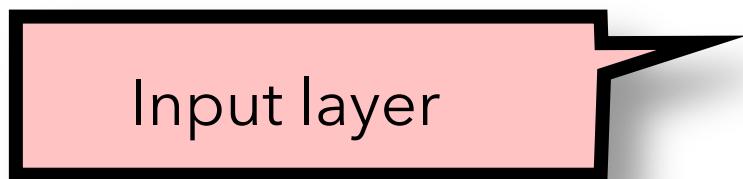


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

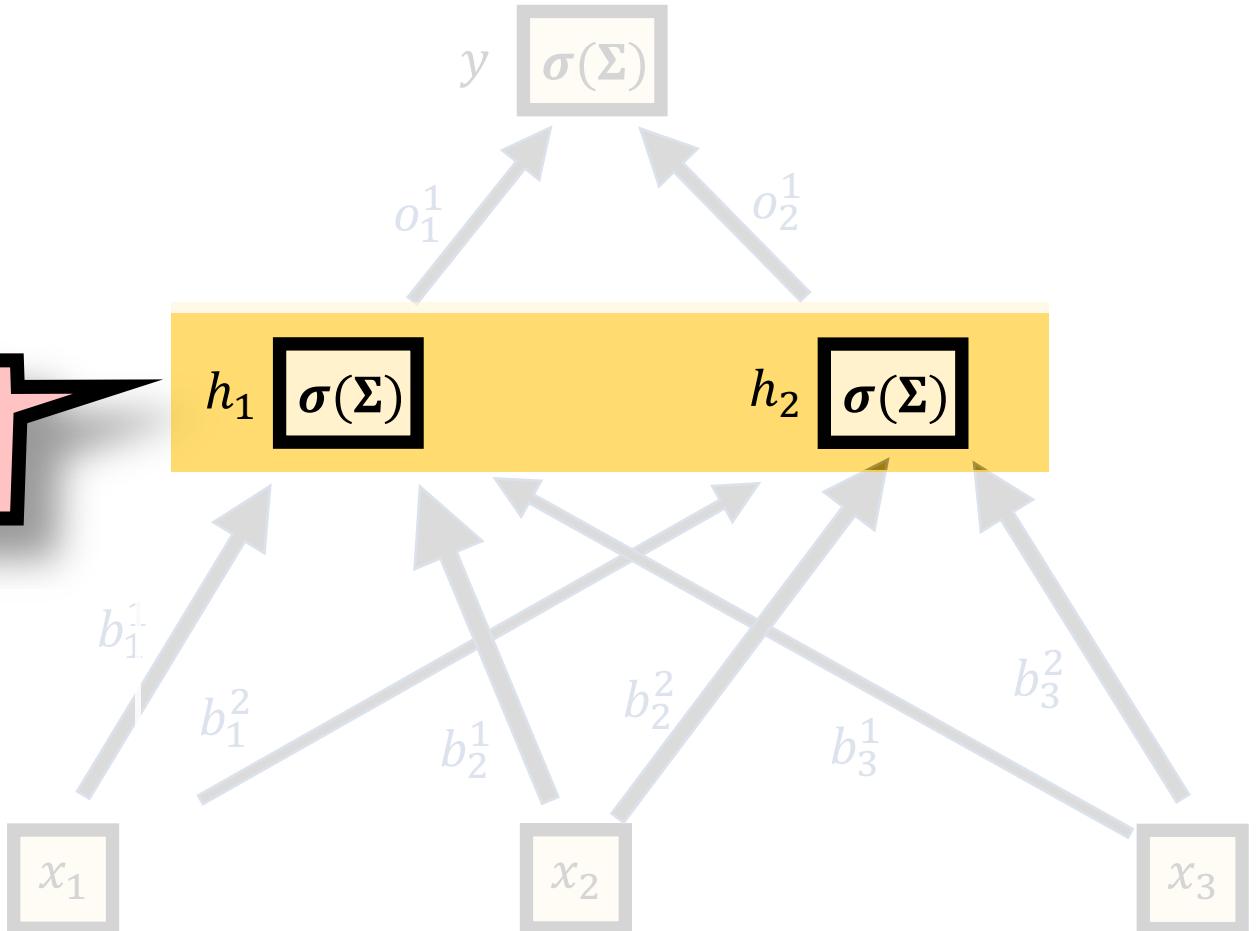
- It's a fully connected network
- Every  is a weight, which is multiplied by its input
- Every  is a scalar value
- Parameters $\theta = \{\beta, O\}$ (weights)



Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  is a weight, which is multiplied by a bias.
- Every  is a $\sigma(\Sigma)$ activation function.
- Parameters $\theta = \{\beta, O\}$ (weights)

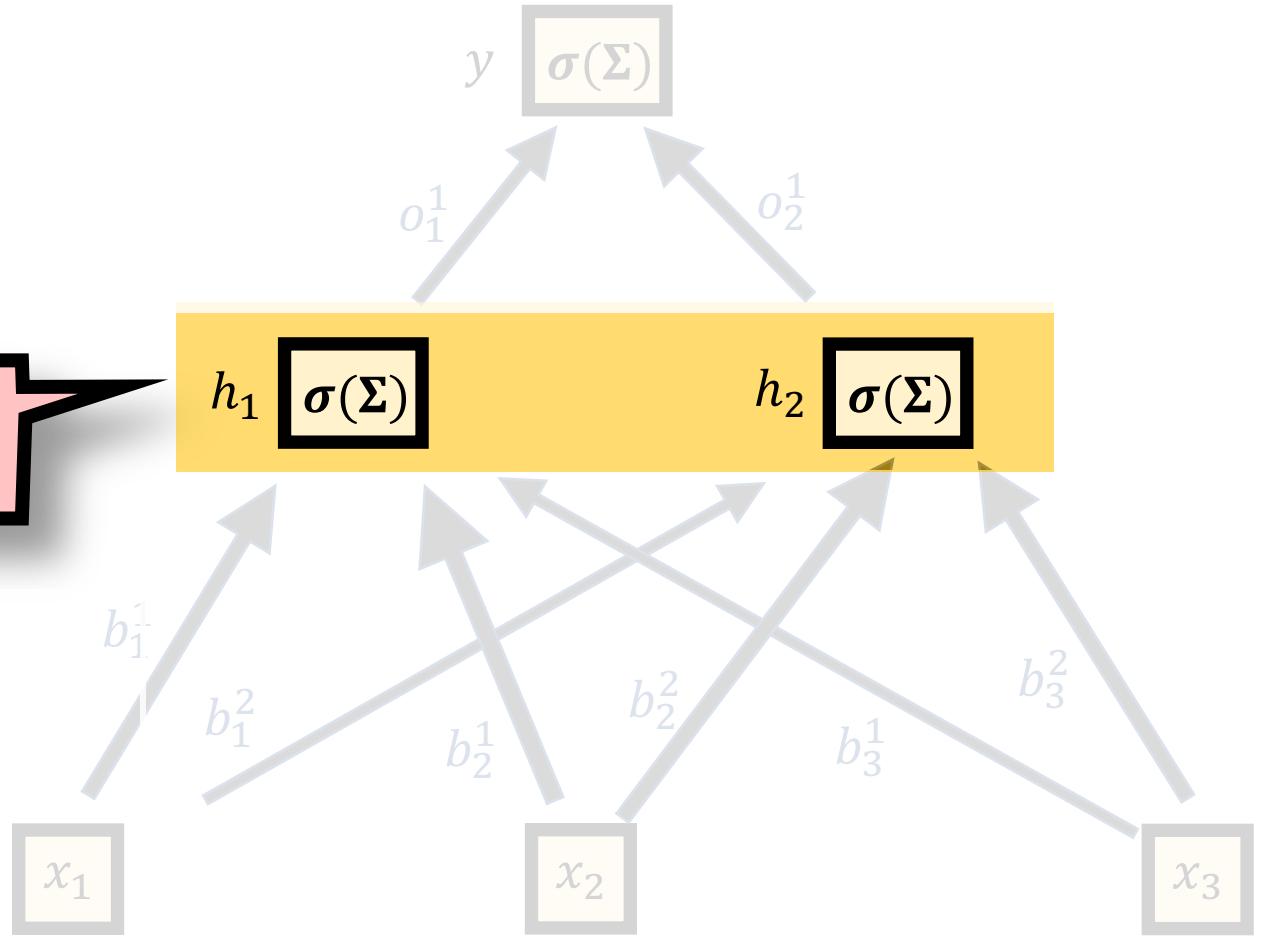


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  is a weight, which is multiplied by a value.
- Every  is a bias.
- Parameters $\theta = \{\beta, O\}$ (weights)

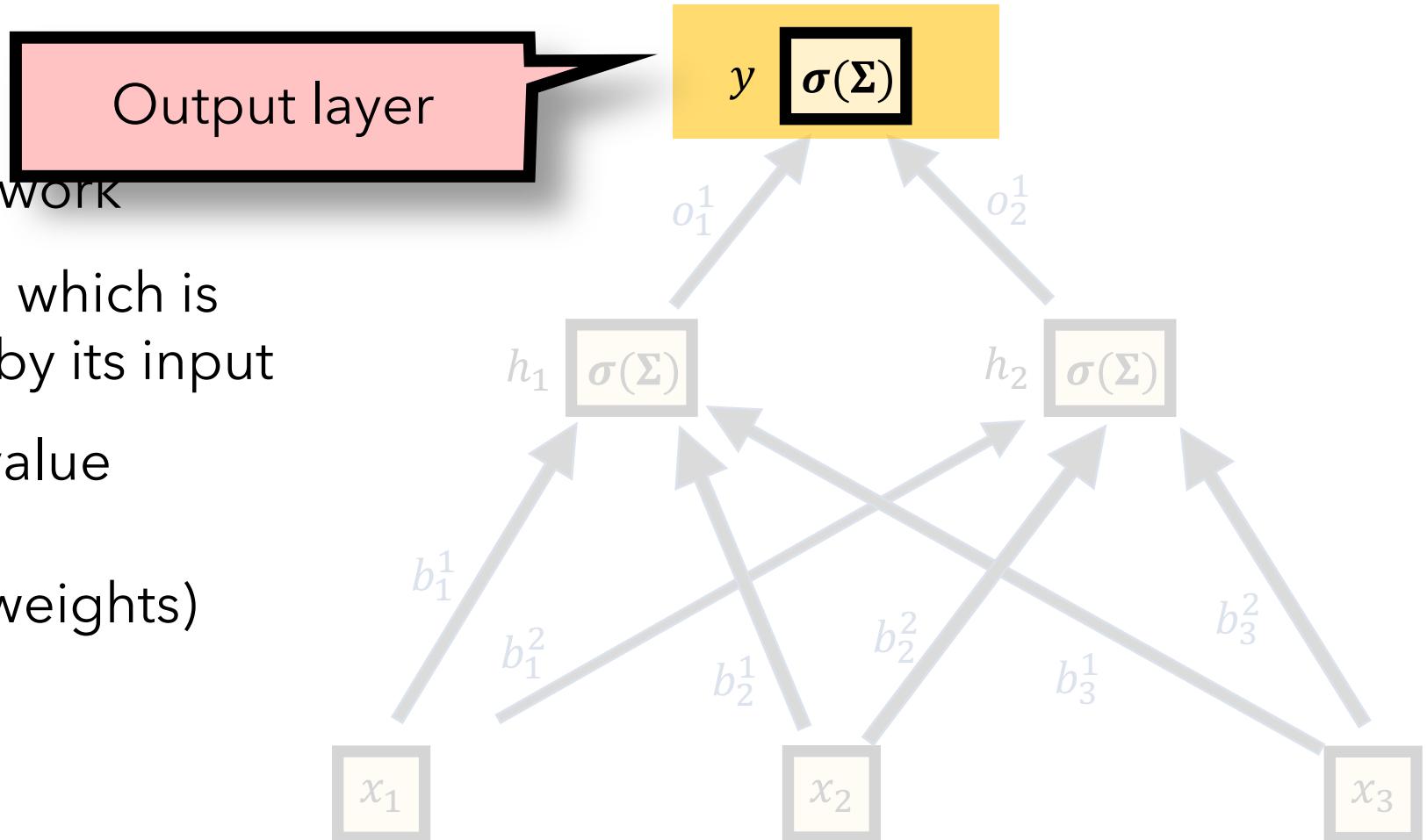


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  is a weight, which is multiplied by its input
- Every  is a scalar value
- Parameters $\theta = \{\beta, O\}$ (weights)

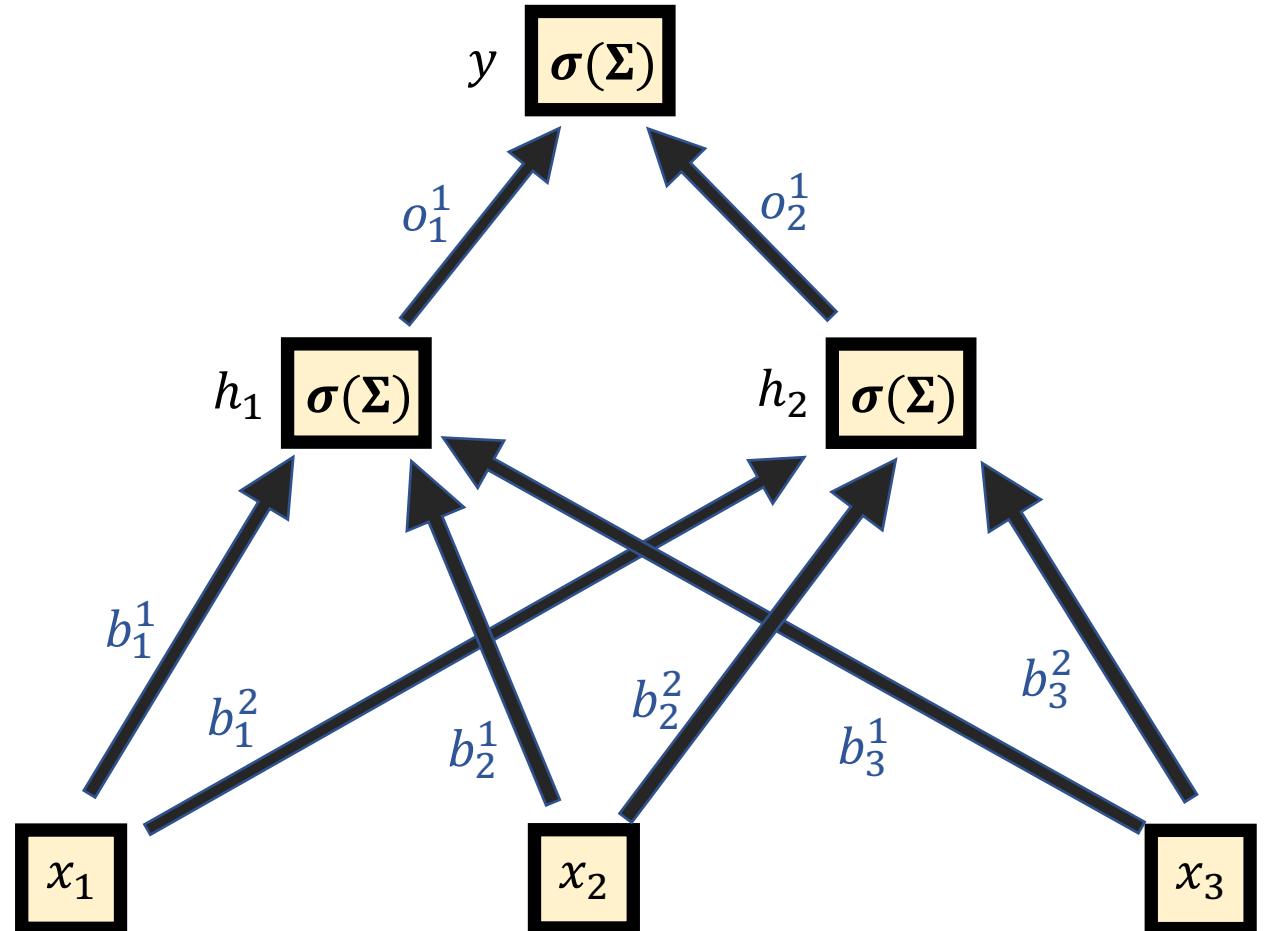


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  is a weight, which is multiplied by its input
- Every  is a scalar value
- Parameters $\theta = \{\beta, O\}$ (weights)

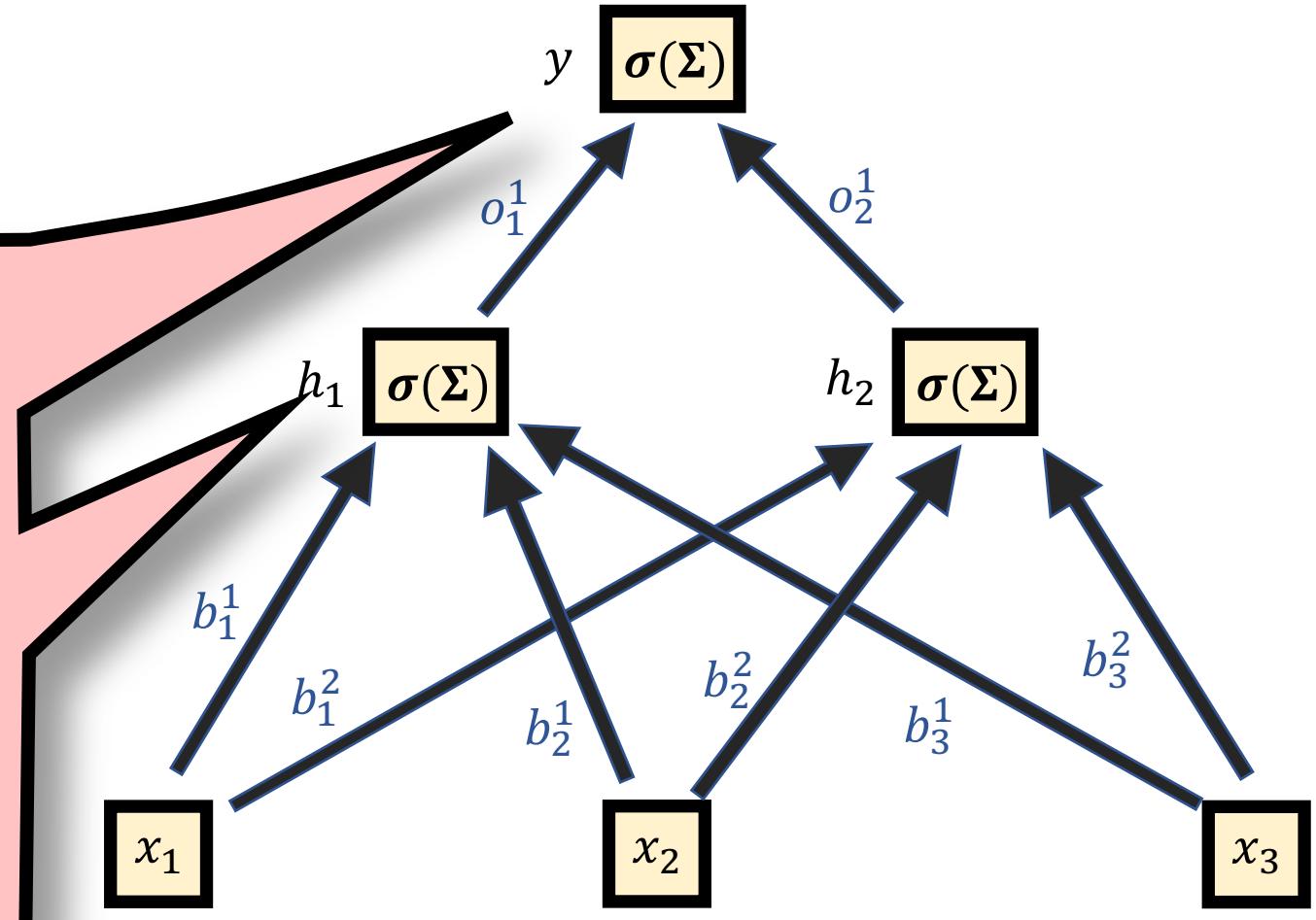


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  , except for the input layer's, is called an **activation function**.
- Parameters They take input(s), apply some aggregate operation(s) -- often a **non-linear transformation** -- and yield a scalar value.

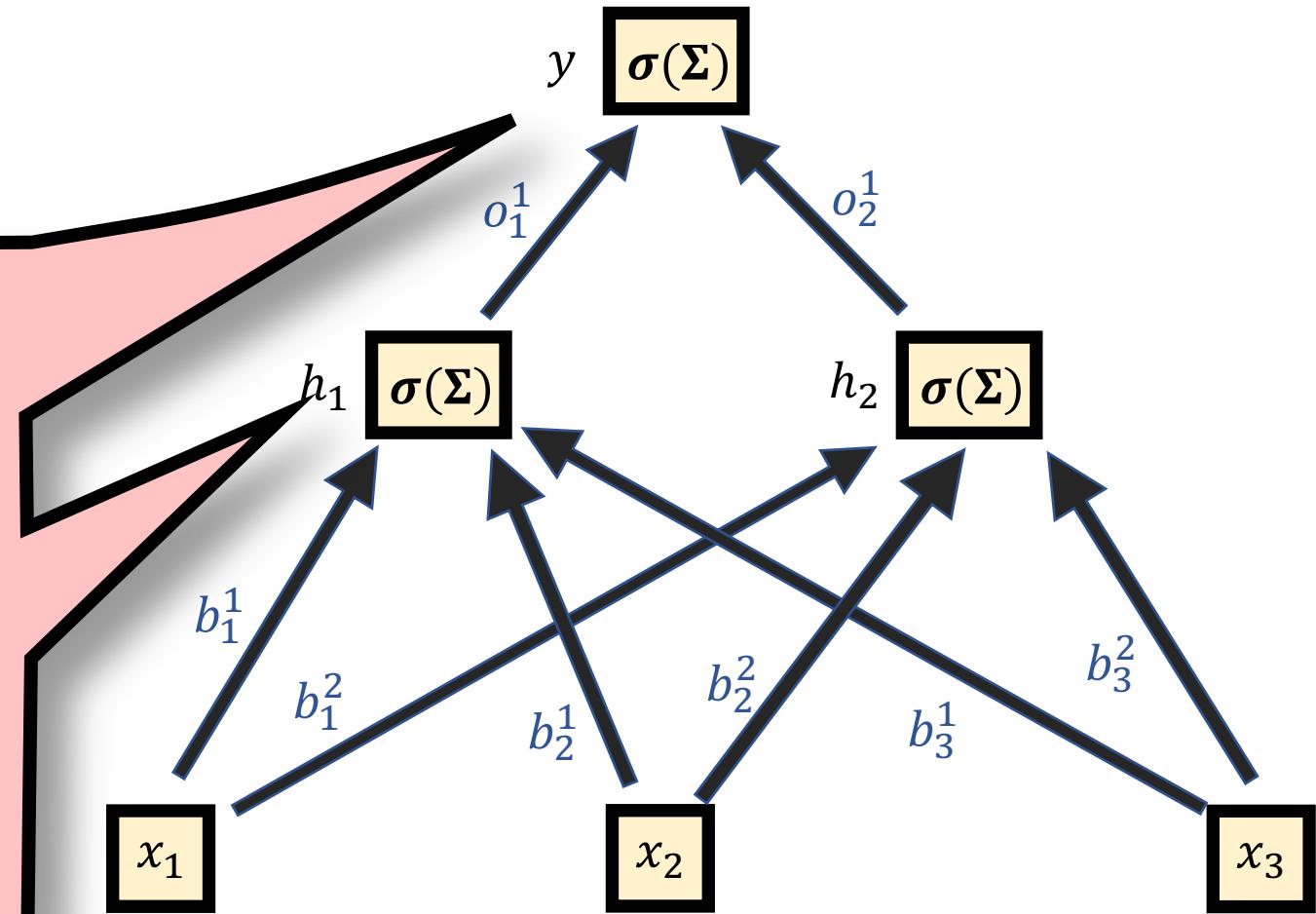


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  , except for the input layer's, is called an **activation function**.
- Par The sigmoid function σ is a common choice and is equivalent to performing **logistic regression** on its given inputs.

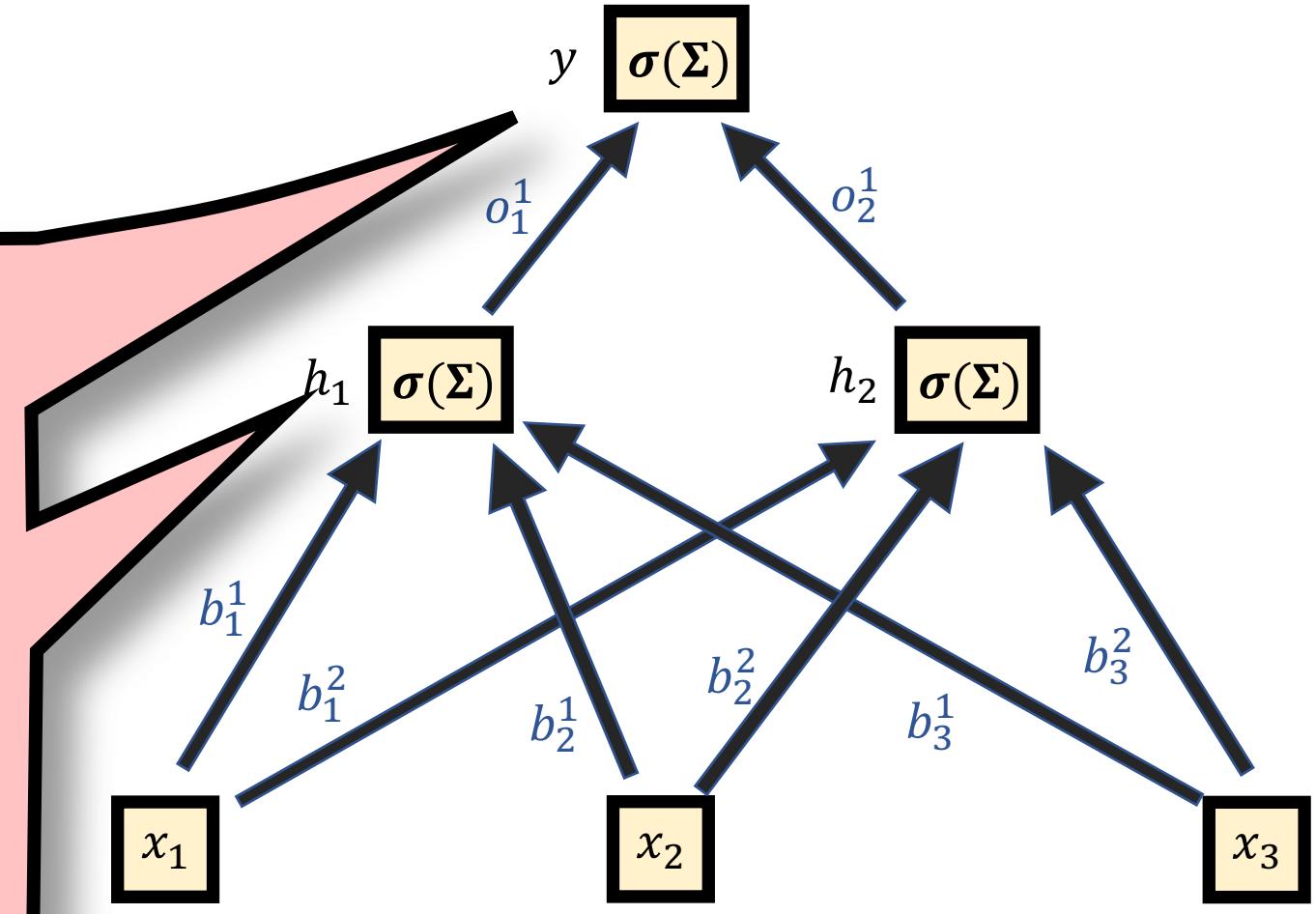


Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every  , except for the input layer's, is called an **activation function**.
- Parameters Thus, neural nets can be viewed as being a fully-connected set of logistic regressions, oftentimes stacked (multiple hidden layers)

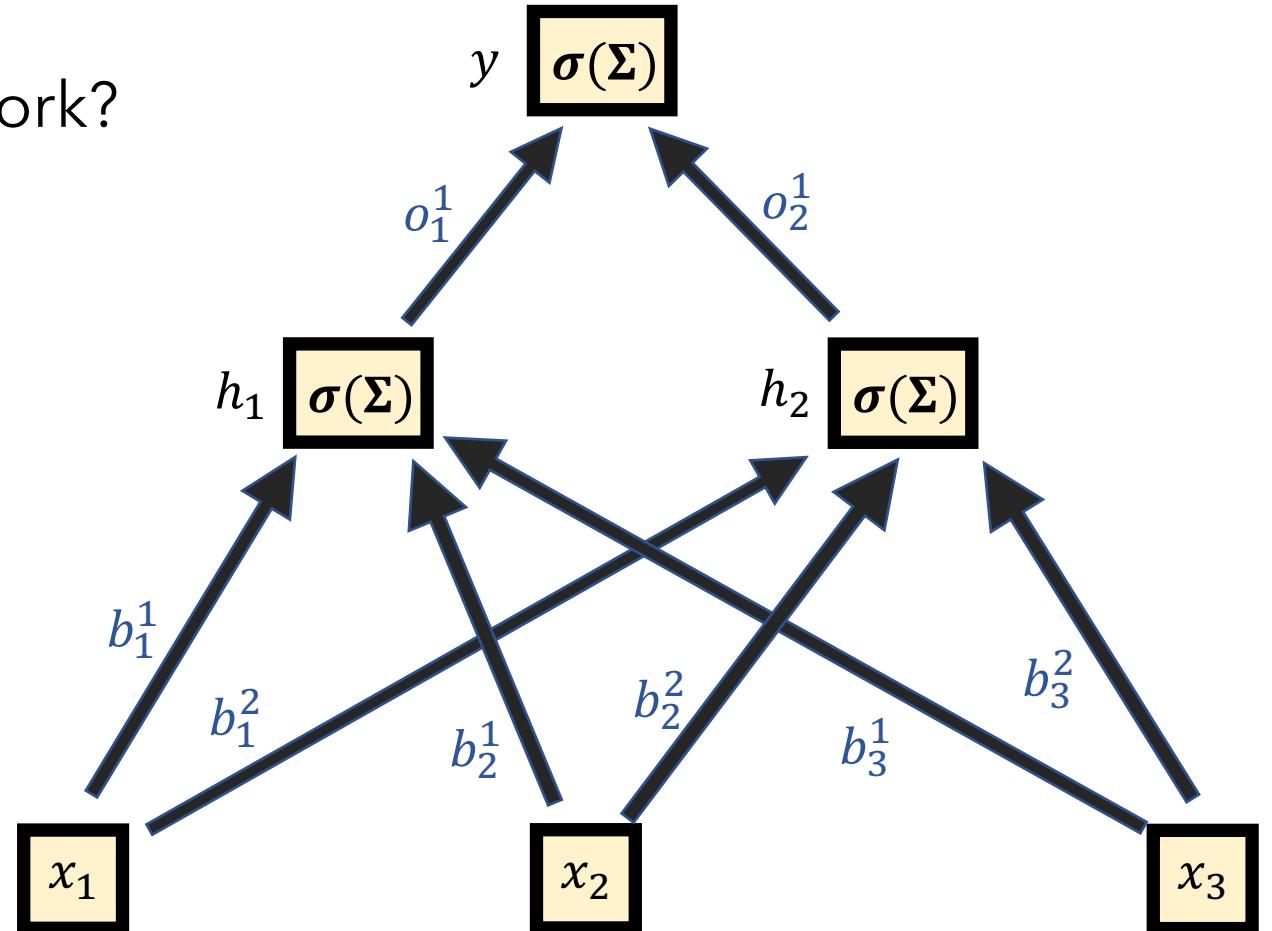


Graphically
(NN format)

Feed-Forward Neural Network

Training:

- **Q1** How do we train a neural network?



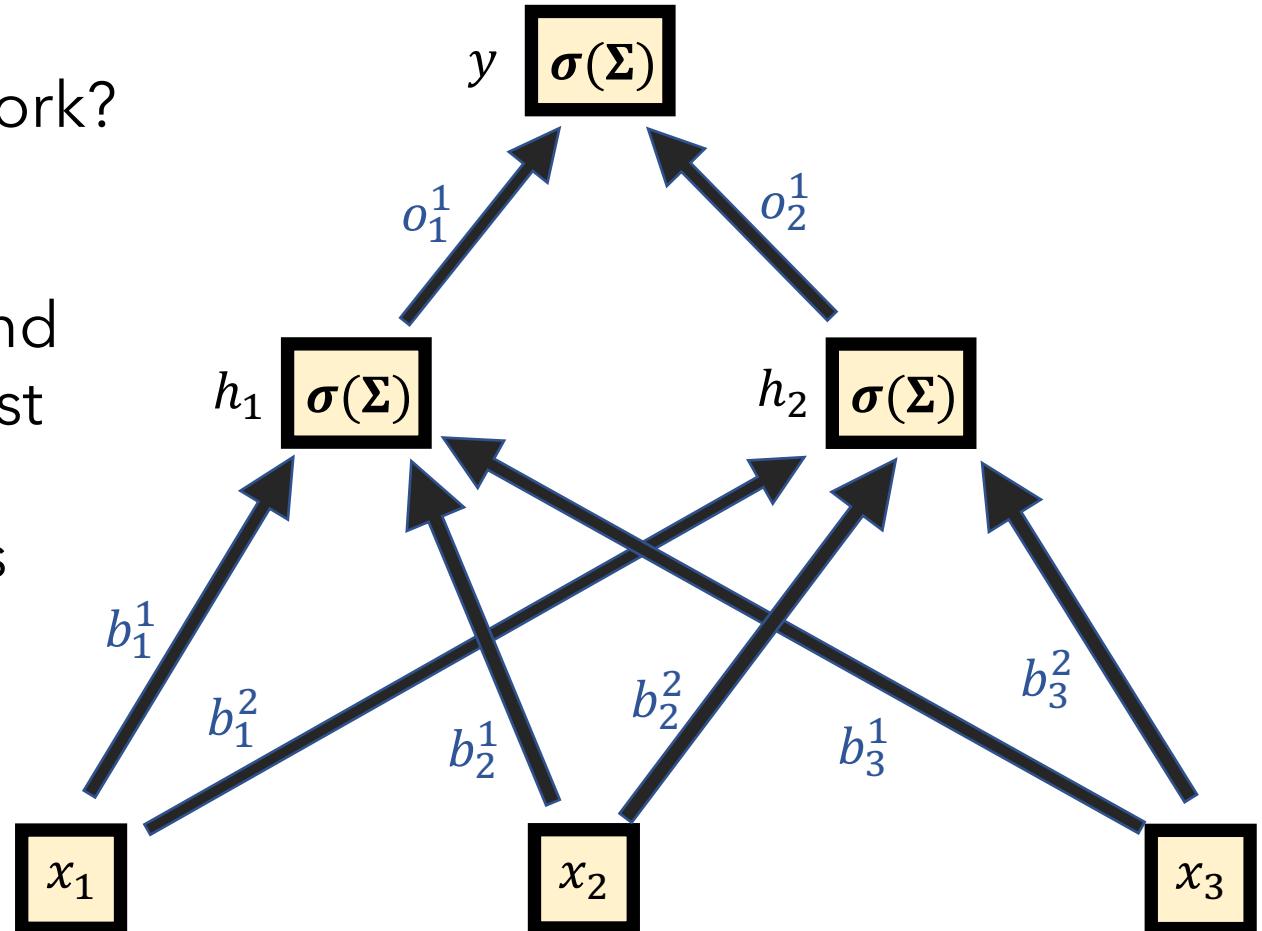
Graphically
(NN format)

Feed-Forward Neural Network

Training:

- **Q1** How do we train a neural network?

A1 First, specify a **cost function** and an **optimization algorithm**, just like we did for our other supervised, parametric models



Graphically
(NN format)

Feed-Forward Neural Network

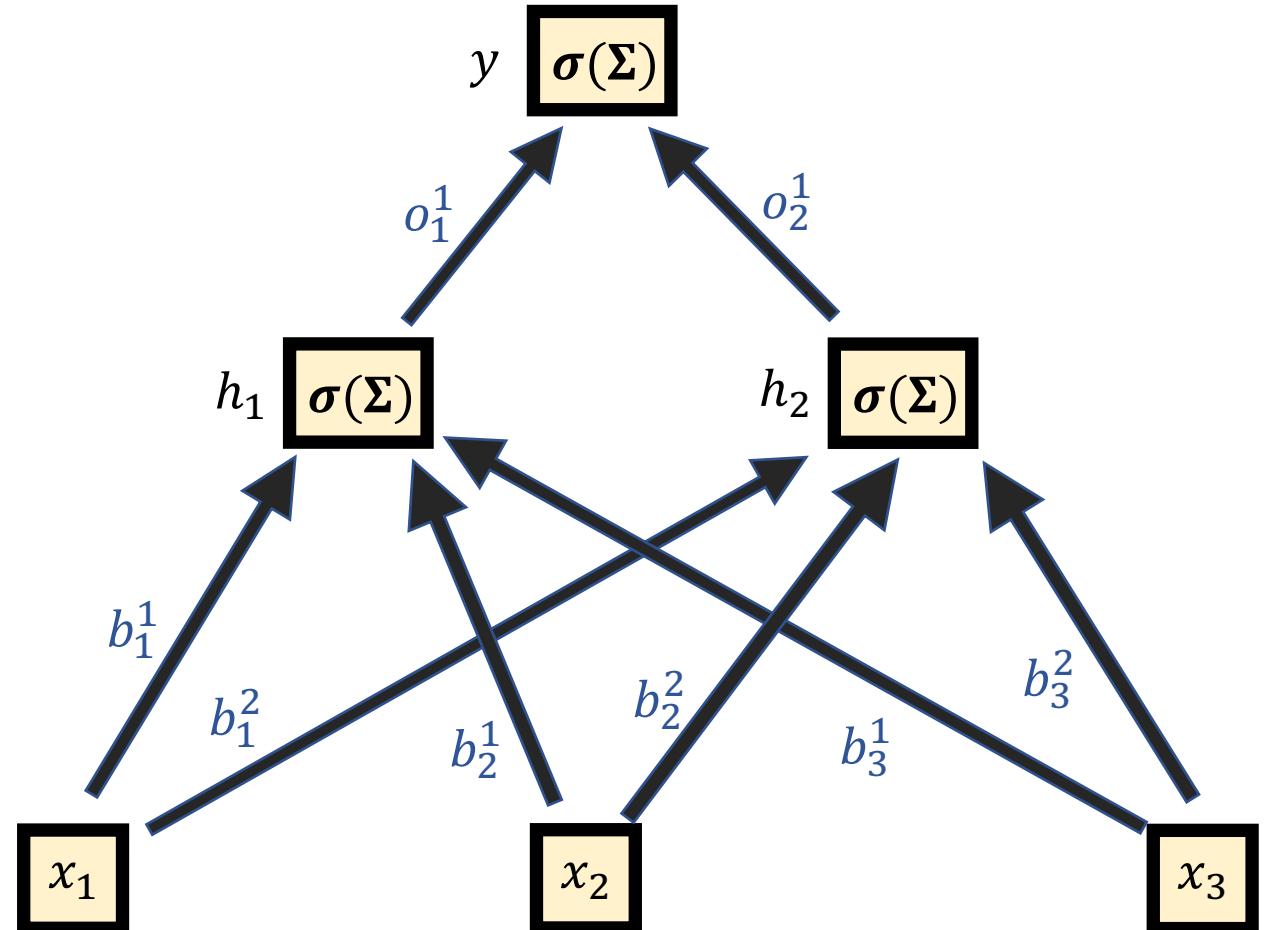
Training:

Cost function

$$J(\theta) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$

"Cross-Entropy" aka "Log loss"

Update the θ via
gradient descent



Graphically
(NN format)

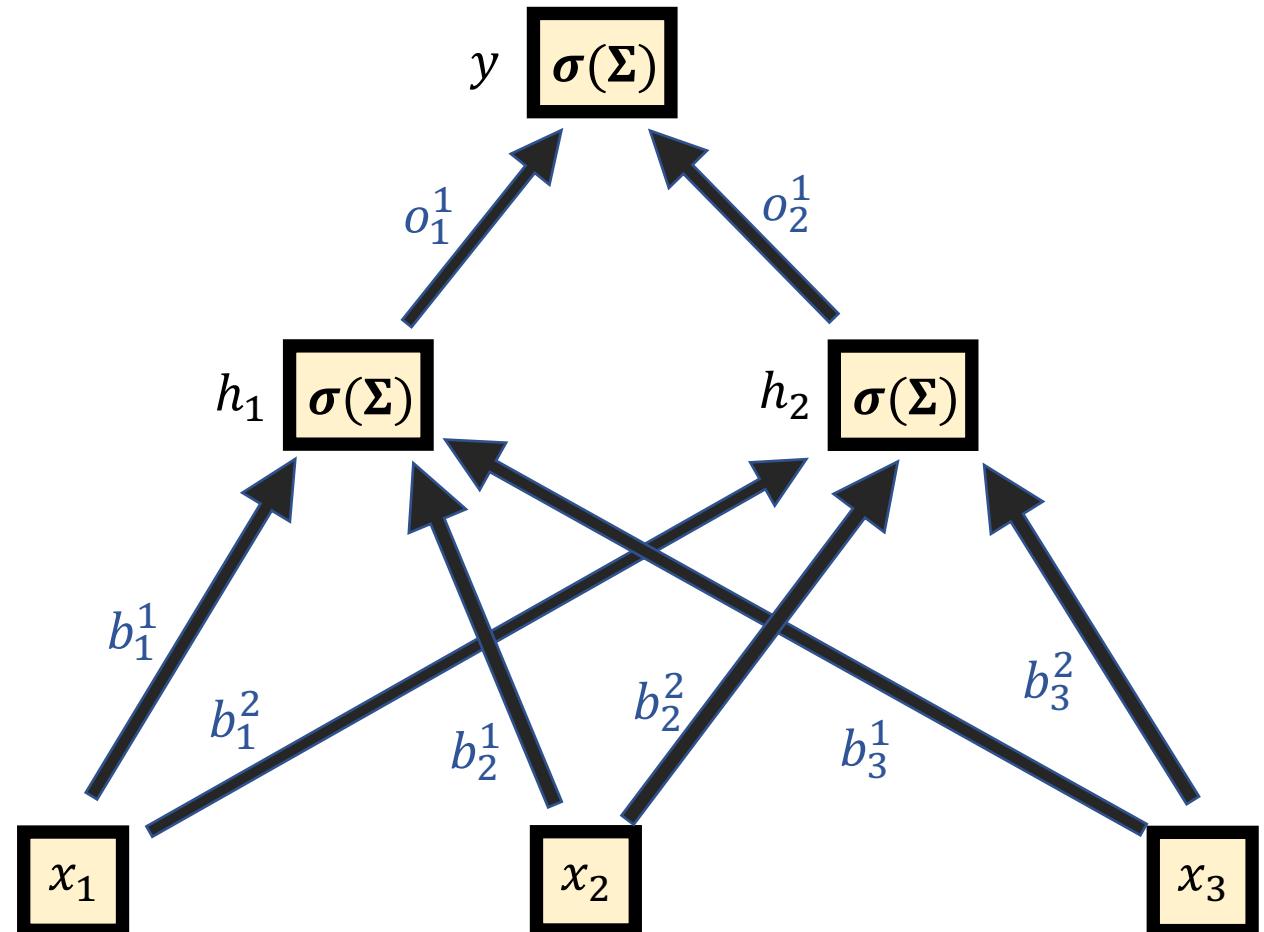
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss
4. Calculate gradients via **backpropagation**
5. Update the weights (aka θ) via **gradient descent**

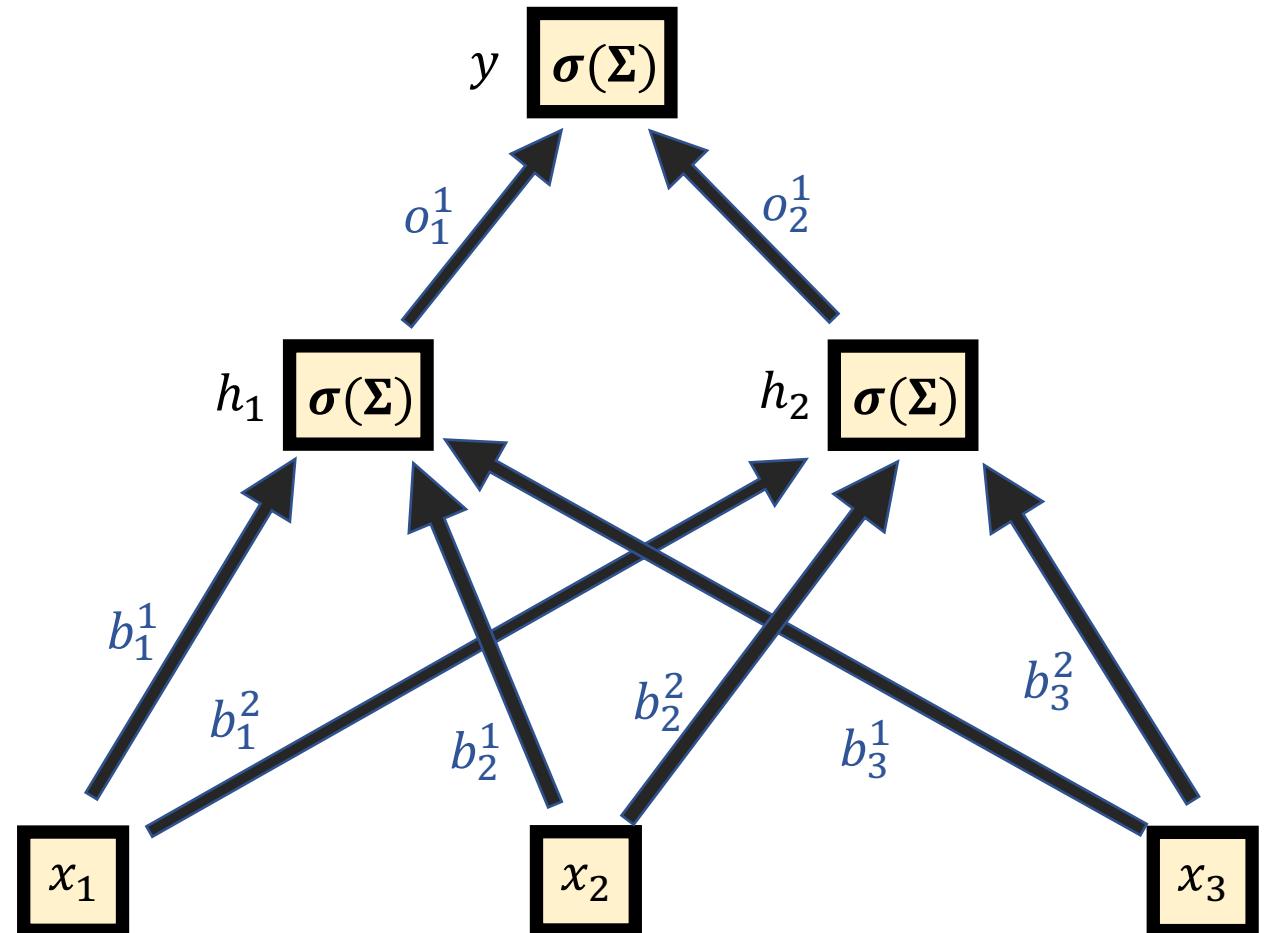


Graphically
(NN format)

Feed-Forward Neural Network

Training:

Initialize θ with random values

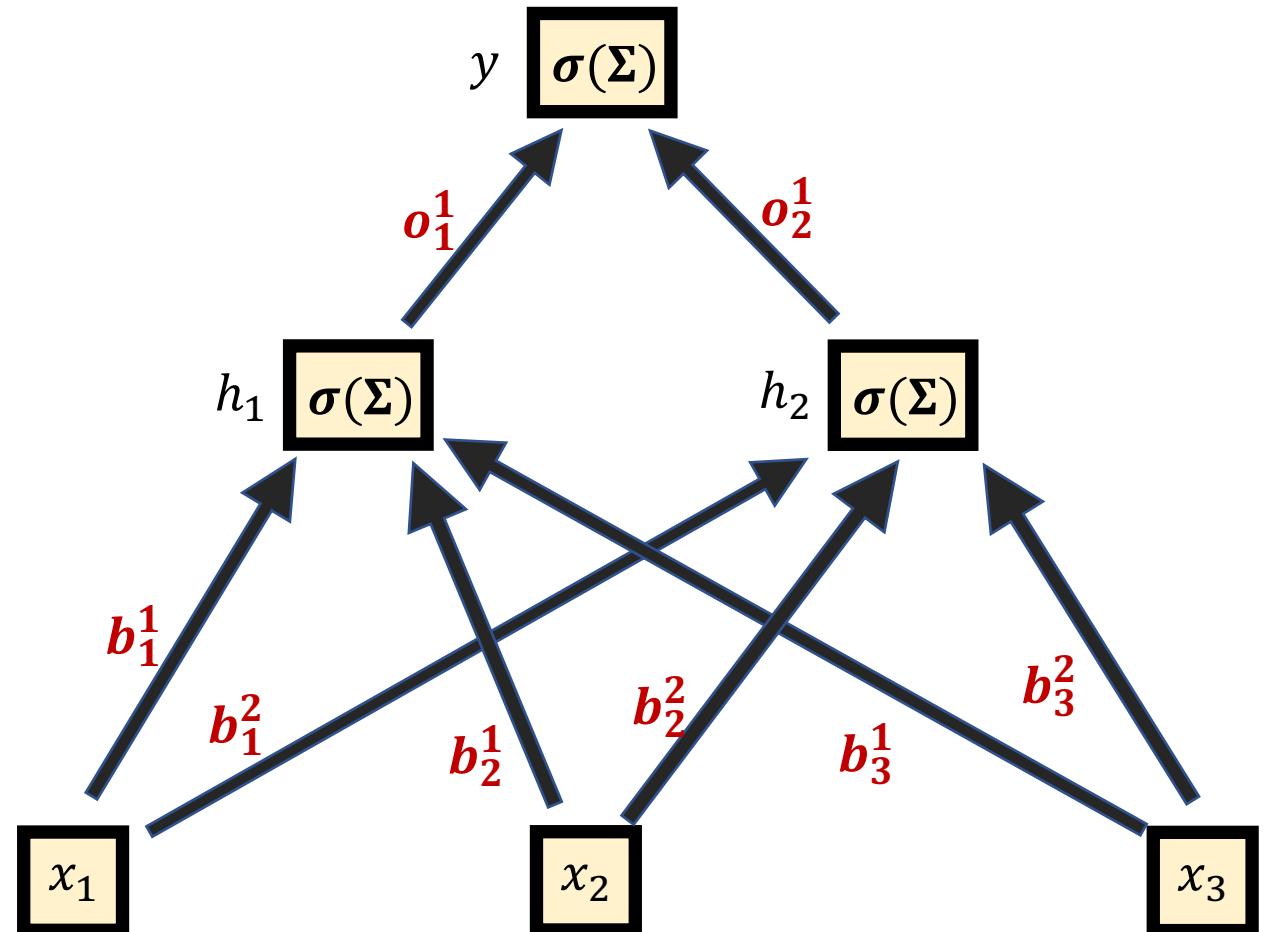


Graphically
(NN format)

Feed-Forward Neural Network

Training:

Initialize θ with random values



Graphically
(NN format)

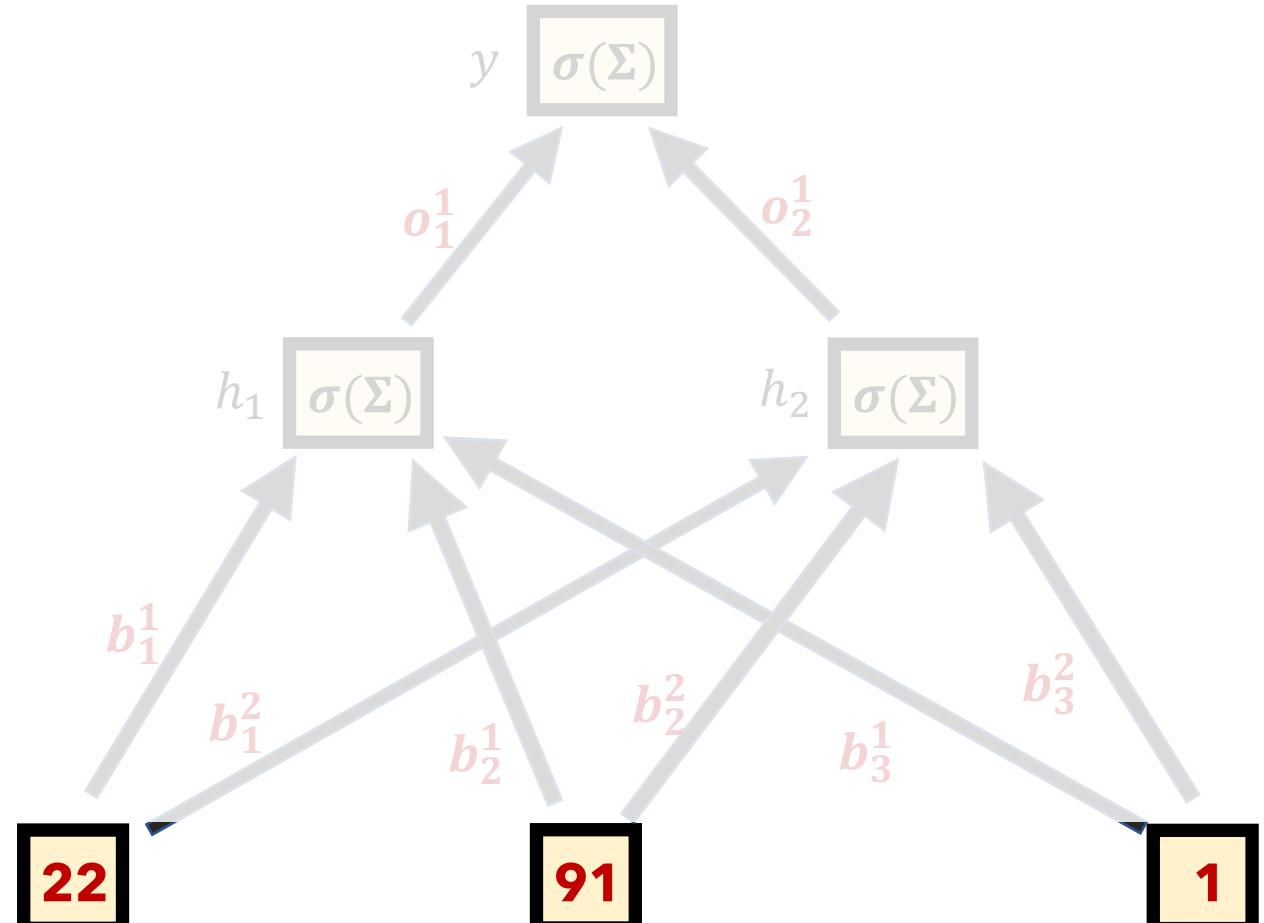
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network



Graphically
(NN format)

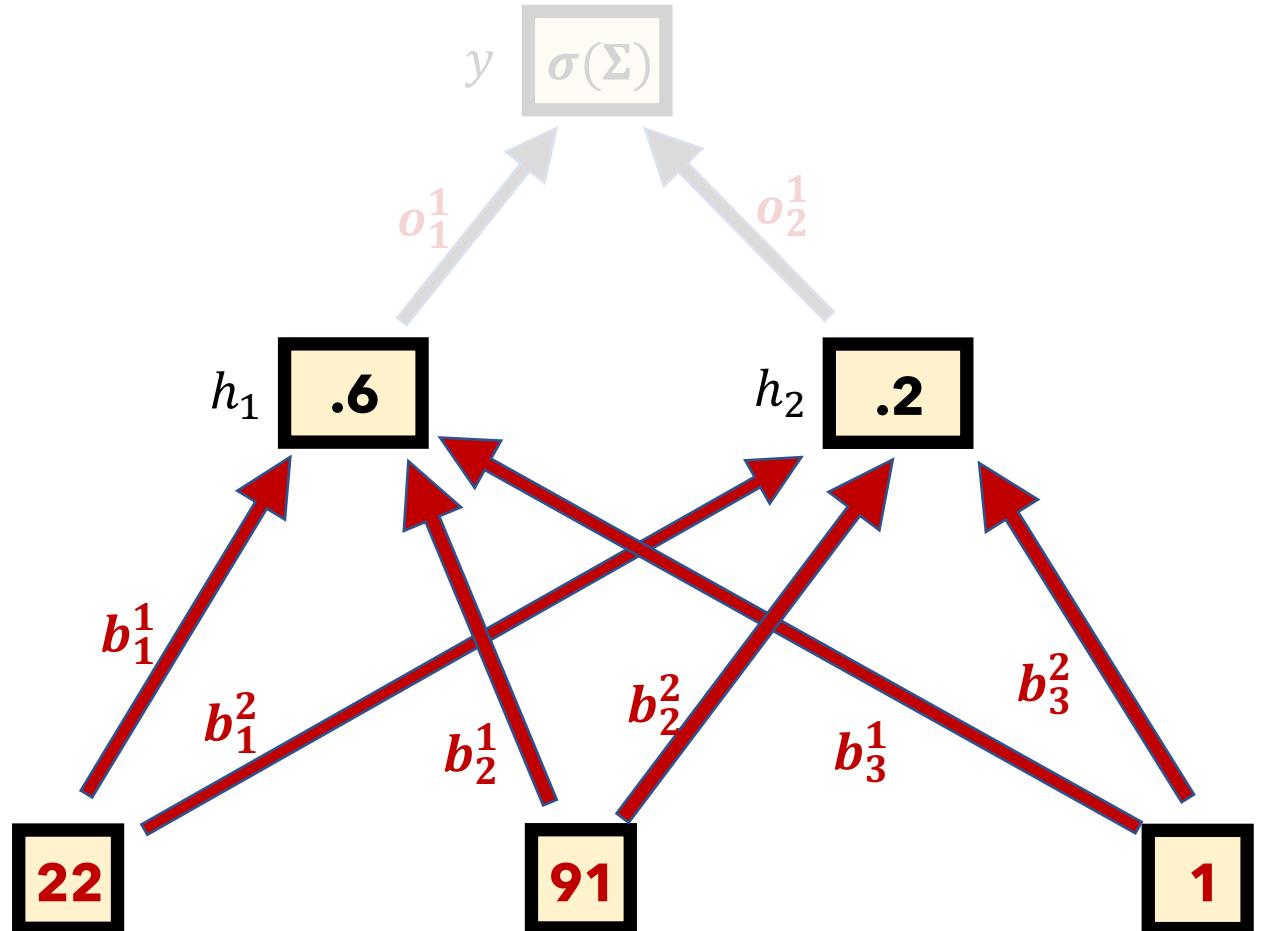
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network



Graphically
(NN format)

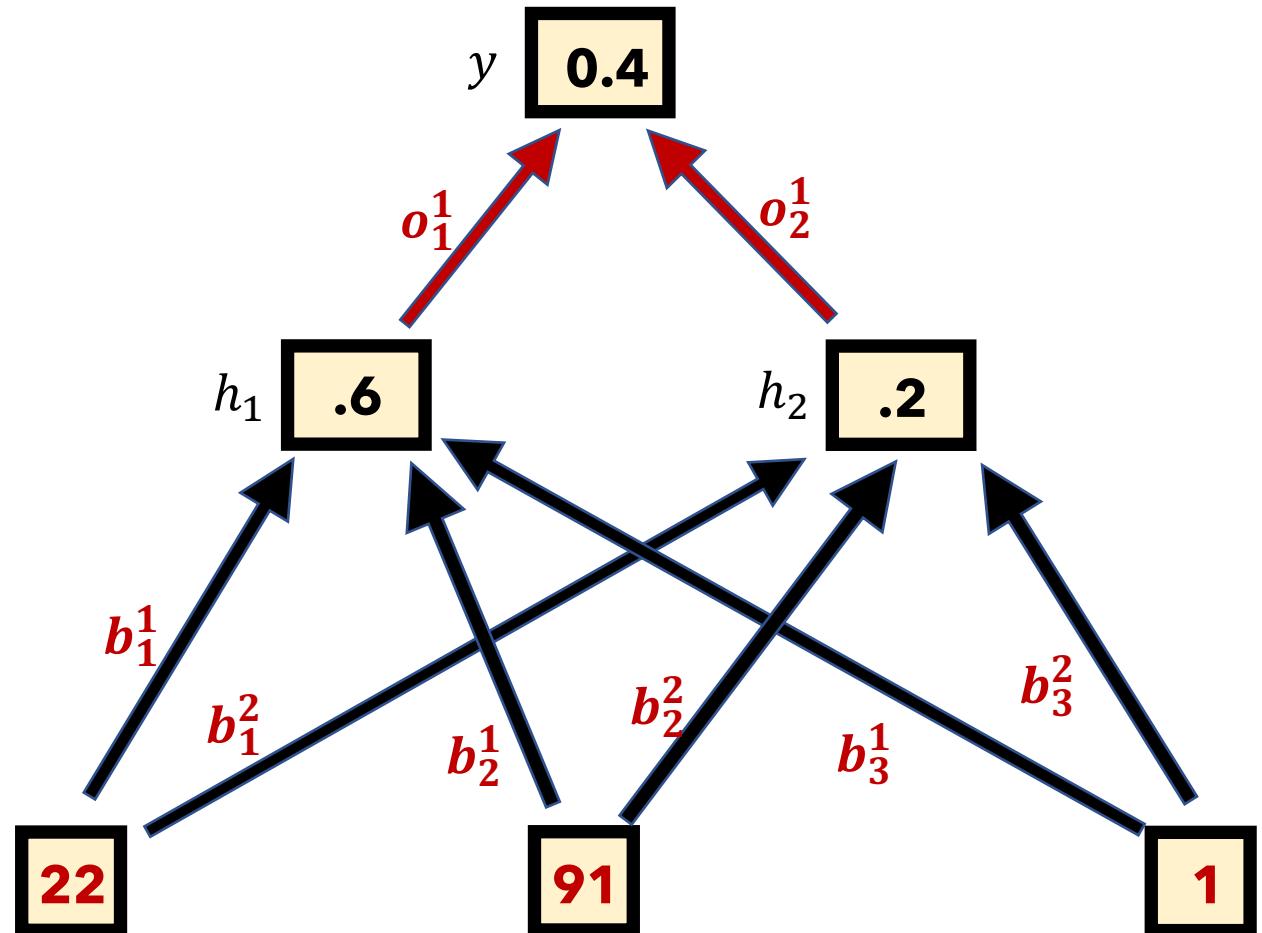
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network



Graphically
(NN format)

Feed-Forward Neural Network

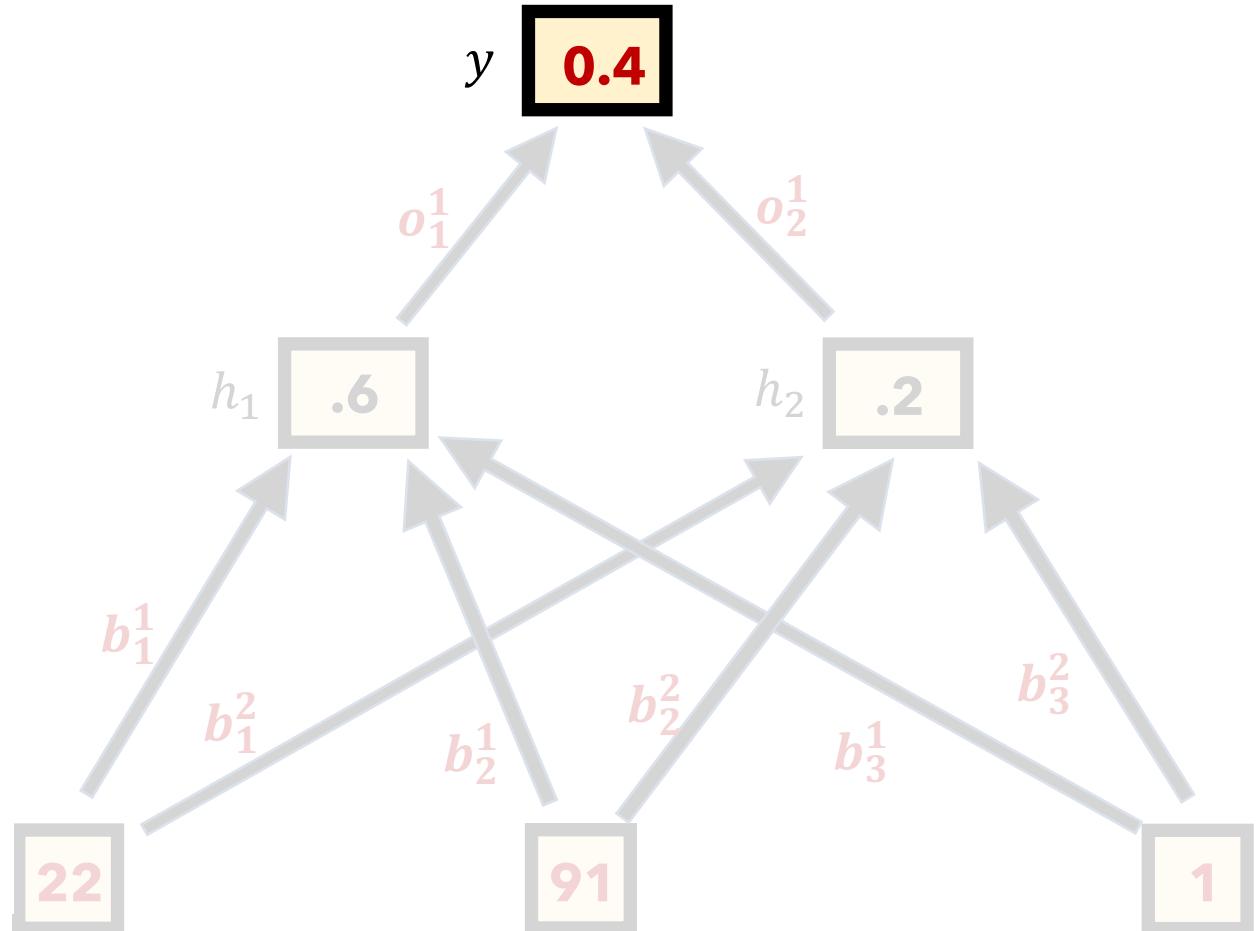
Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss

$$J(\theta) = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$



Graphically
(NN format)

Feed-Forward Neural Network

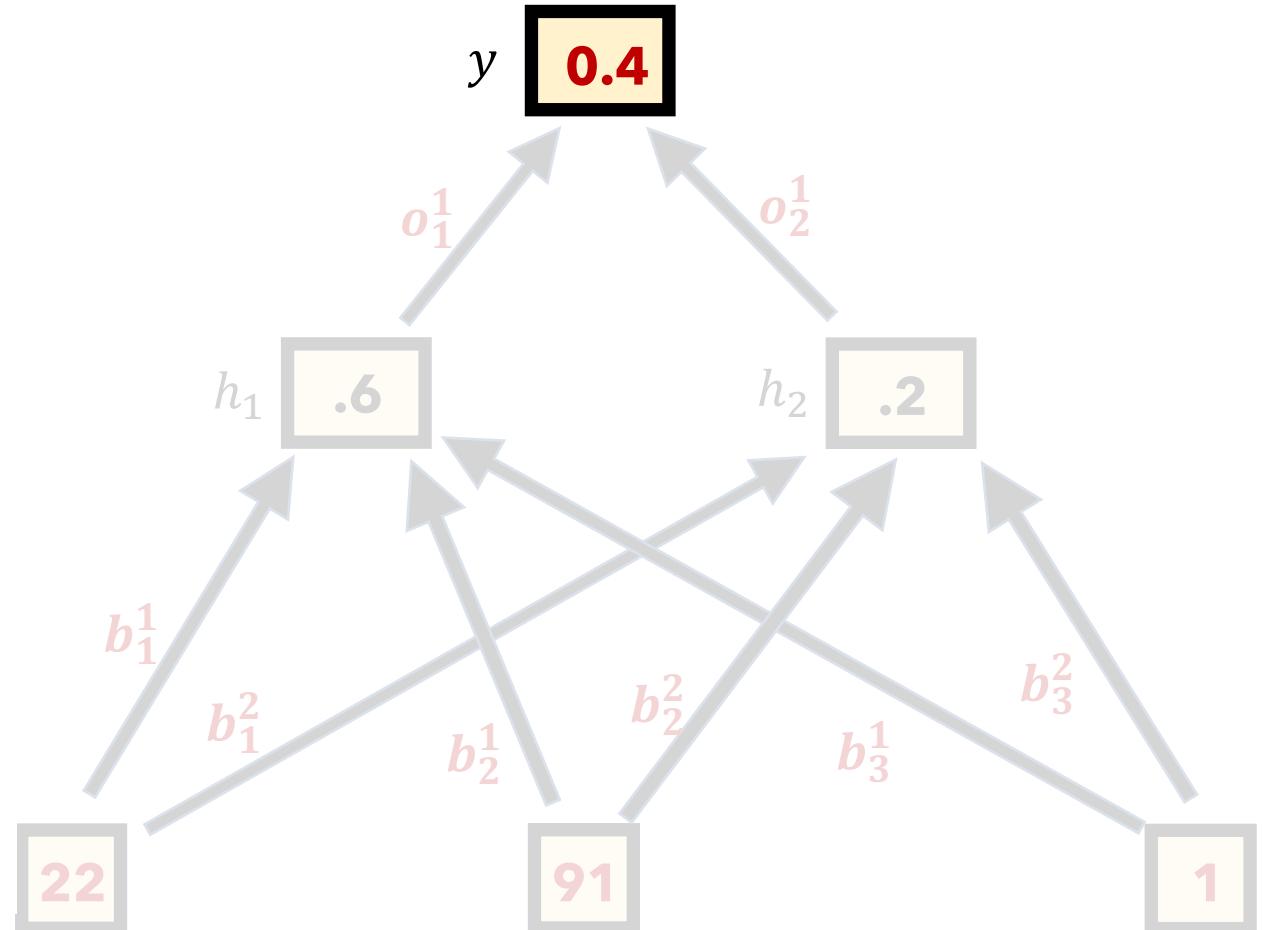
Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss

$$J(\theta) = -[0 + (1 - 0) \log(1 - 0.4)]$$



Graphically
(NN format)

Feed-Forward Neural Network

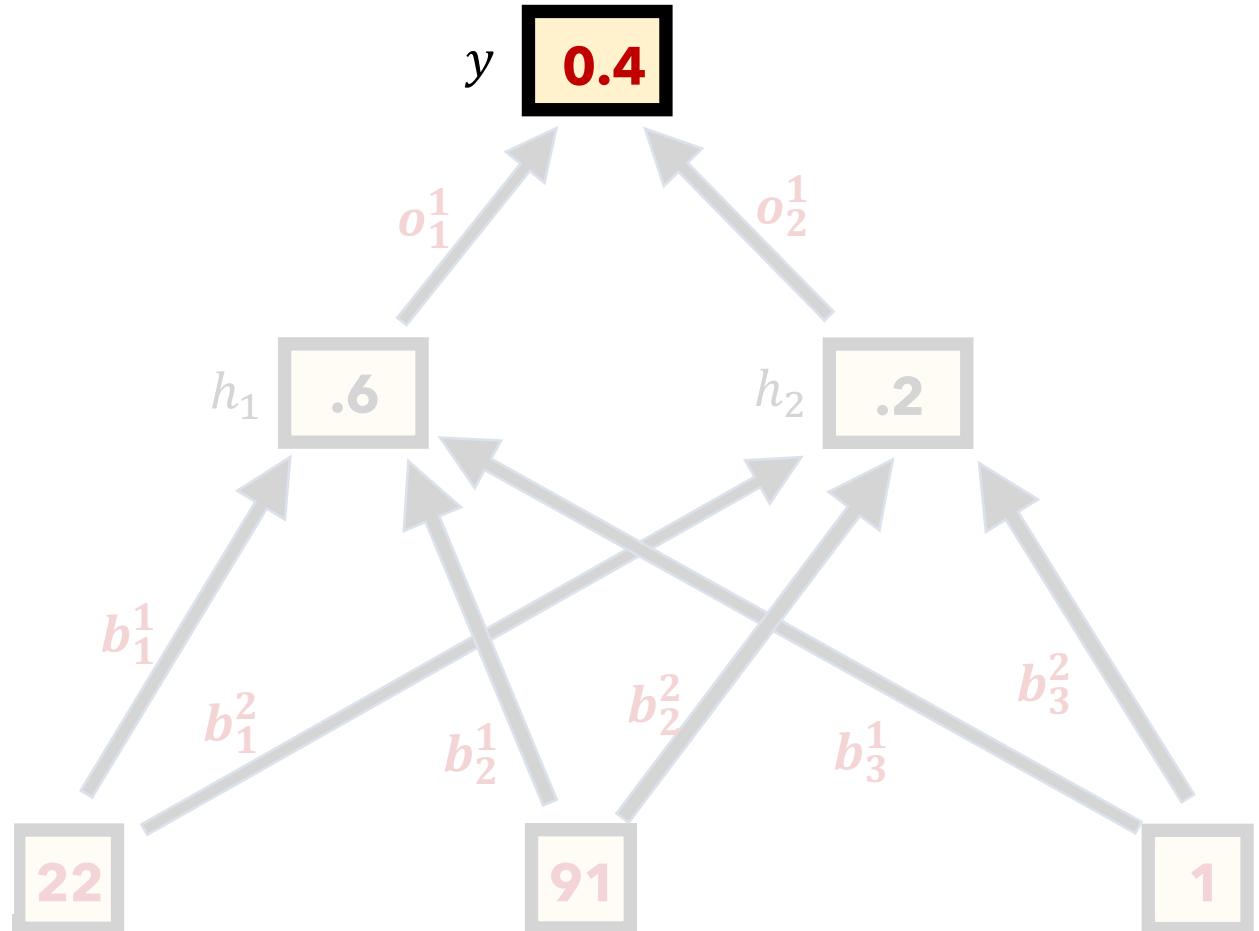
Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss

$$J(\theta) = -[0 + (1 - 0) \log(0.6)]$$



Graphically
(NN format)

Feed-Forward Neural Network

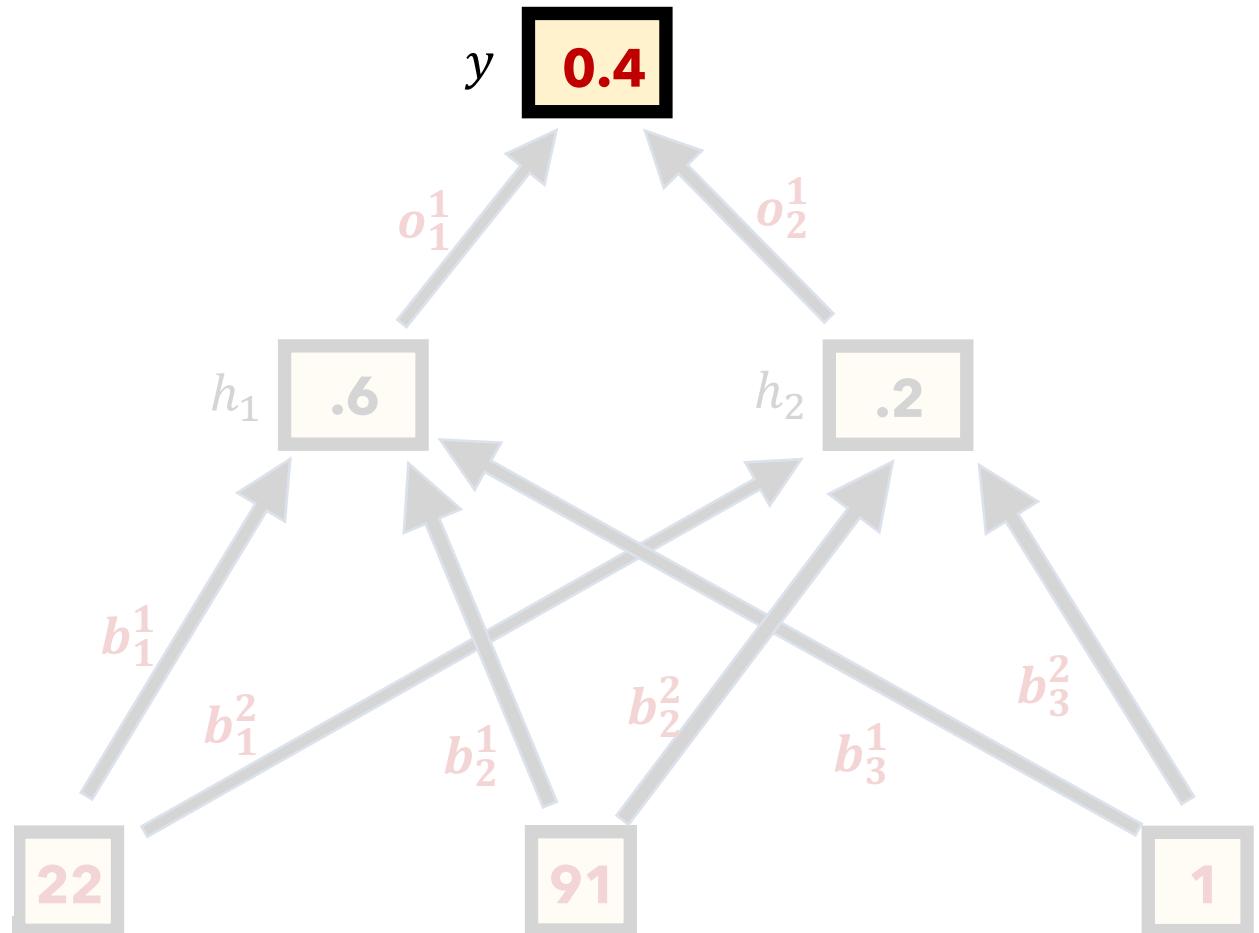
Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss

$$J(\theta) = -\log(0.6)$$



Graphically
(NN format)

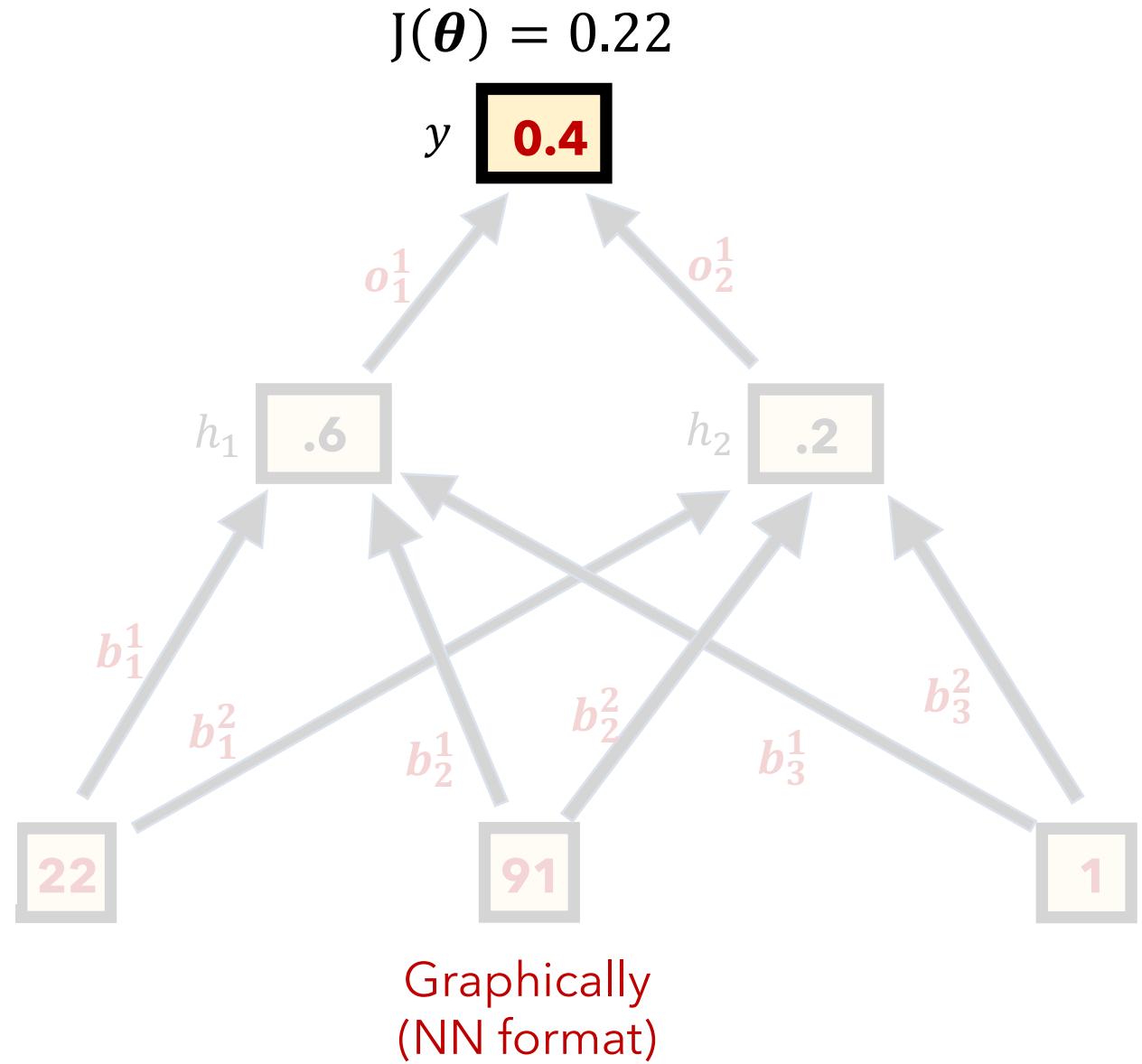
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss



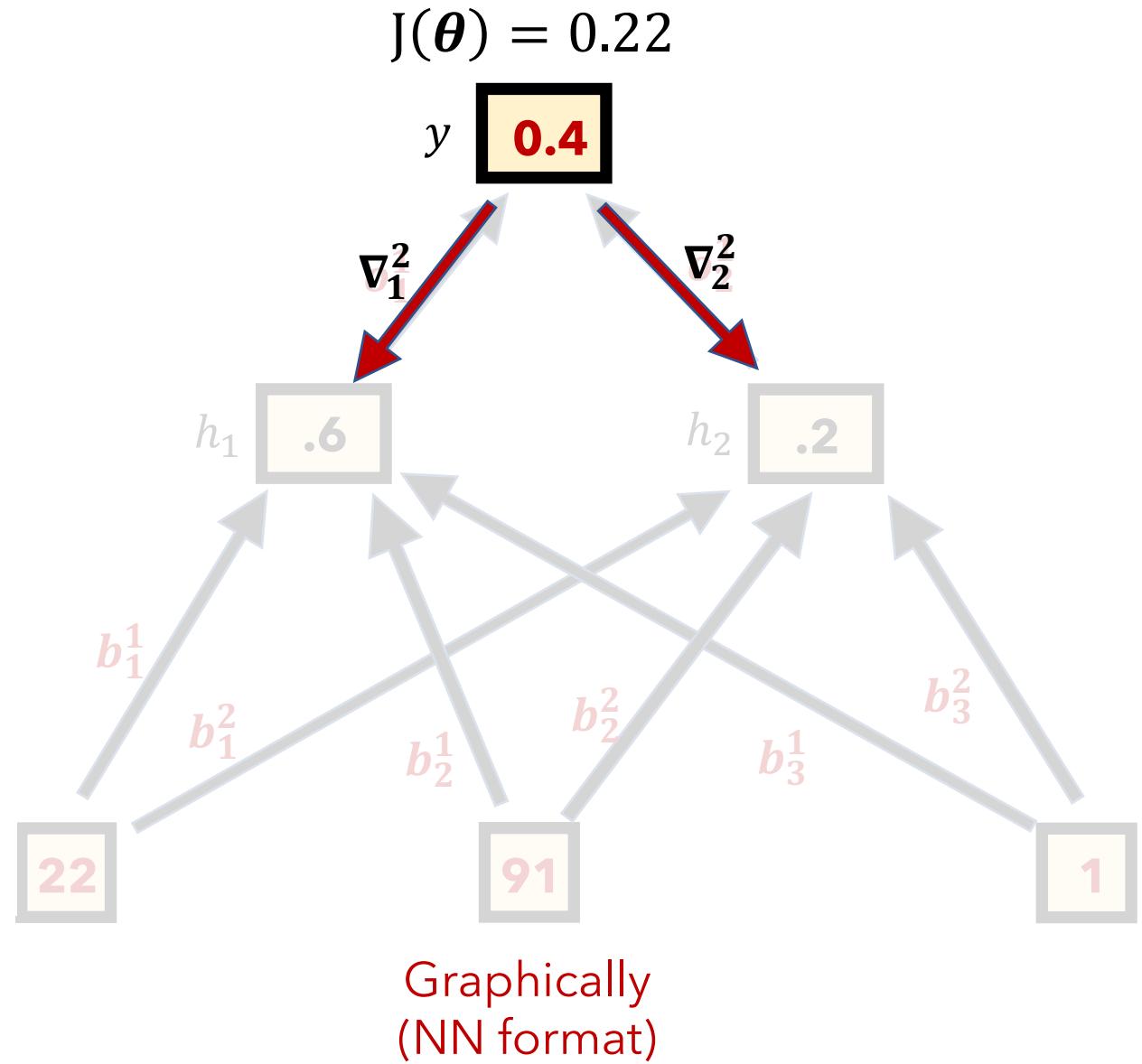
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss
4. Calculate gradients via backpropagation



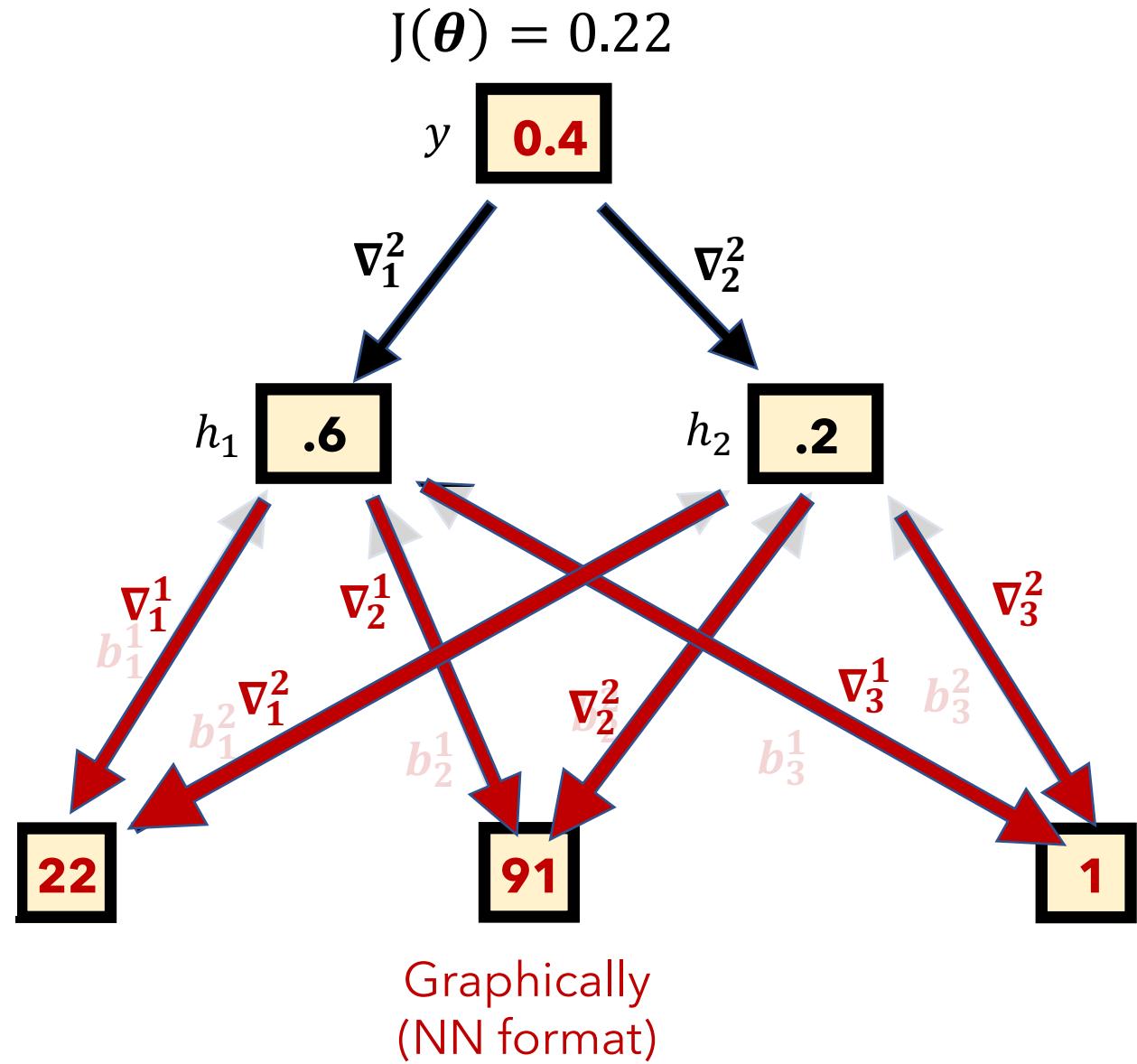
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss
4. Calculate gradients via backpropagation



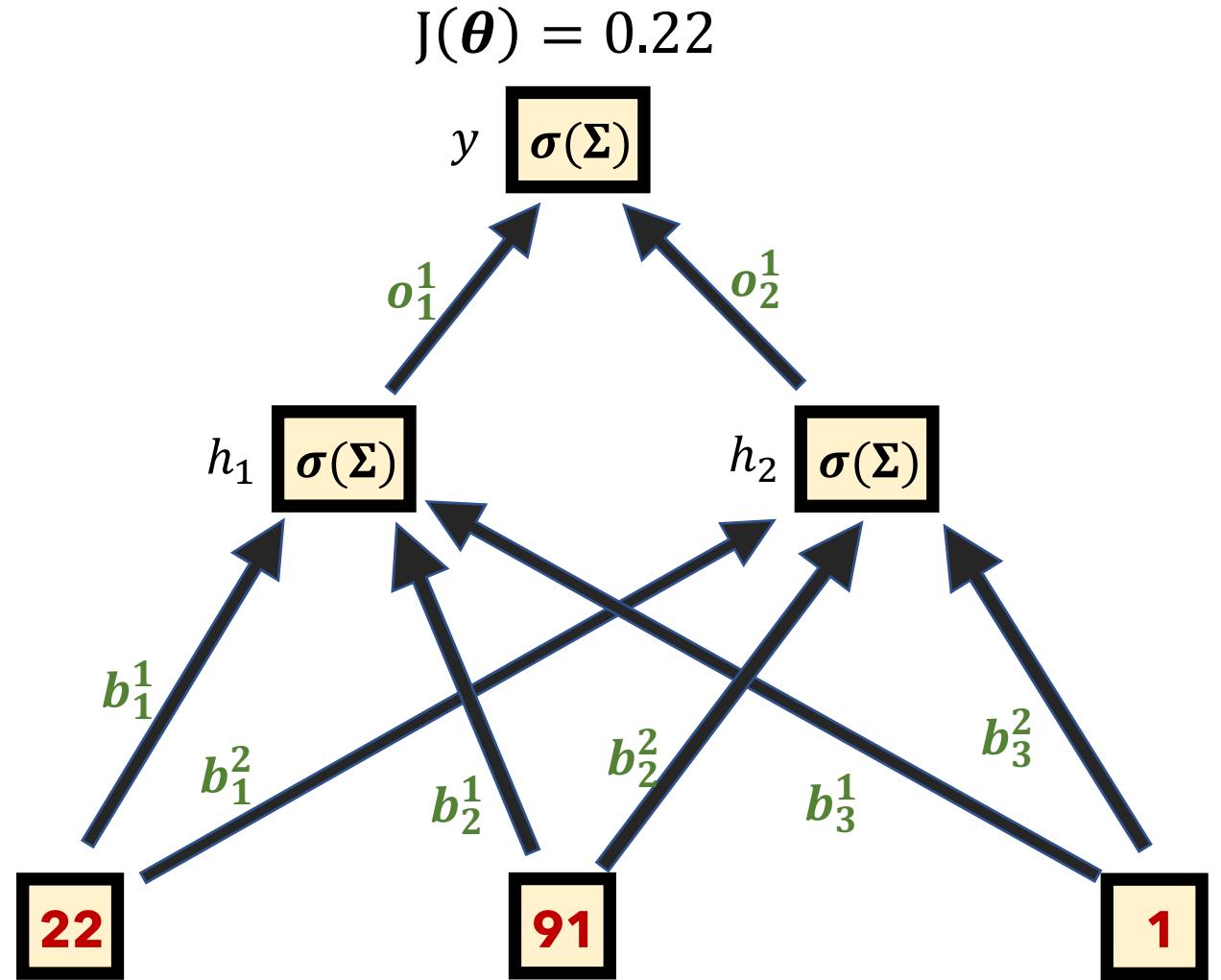
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss
4. Calculate gradients via **backpropagation**
5. Update the weights (aka θ) via **gradient descent**



Graphically
(NN format)

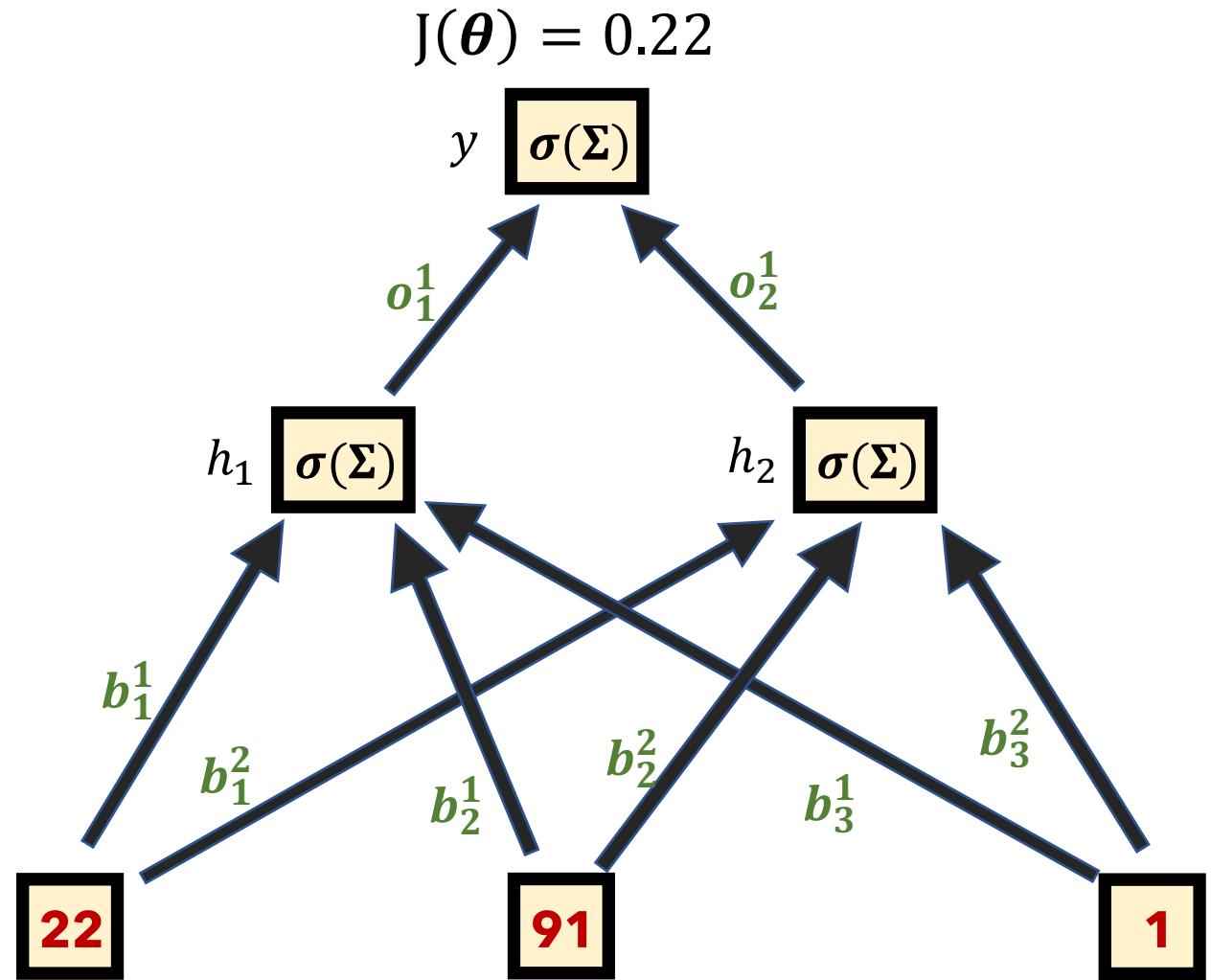
Feed-Forward Neural Network

Training:

Initialize θ with random values

Repeat until convergence:

1. Provide input x_i to the network
2. Propagate the values through the network
3. Calculate the cost/loss
4. Calculate gradients via backpropagation
5. Update the weights (aka θ) via gradient descent



Graphically
(NN format)

Feed-Forward Neural Network

PROS

- Fits many linear or **non-linear** activation functions f_i to combinations of input X
- Can model highly complex behavior
- When designed well, can provide state-of-the-art results on most tasks
- Incredible resources, libraries, and support

CONS

- Sensitive to architecture choices and hyperparameters
- Tricky to debug
- Can be computationally expensive
- Poor interpretability

	Supervised vs Unsupervised	Regression vs Classification	Parametric vs Non-Parametric	Generative vs Discriminative
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	Supervised	Classification	Parametric	Discriminative
k-NN	Supervised	either	Non-Parametric	Discriminative
	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
PCA	Unsupervised	neither	Non-Parametric	neither
Clustering	Unsupervised	neither	Non-Parametric	Generative
GAMs	Supervised	either	Parametric	Discriminative
Feed-Forward Net	Supervised	either	Parametric	Discriminative

IMPORTANT

When **training** any supervised model, be careful of **overfitting** your model

A good model should generalize well to unseen (i.e., testing) data

Consider adding **regularization** term $R(\theta)$ to your cost function

Imposes a penalty based on your parameter values θ

L1 regularization: $R(\theta) = \sum_{i=1}^n |\theta_i|$ Prefers many small-weight values

L2 regularization: $R(\theta) = \sum_{i=1}^n \theta_i^2$ Prefers sparse weights (many 0's)

Additionally, you can add dropout to Neural Networks

IMPORTANT

When **training** any supervised model, wisely use your training data

A good model should generalize well to unseen (i.e., testing) data

- a. Shuffle your training data and optionally bootstrap samples
- b. Perform cross-validation

MLE vs MAP

So far, whenever we've discussed training a model, we've assumed our data was i.i.d. and we framed the problem as maximizing the similarity of the predictions and the gold truth by adjusting the parameters θ

e.g.

Q1

When training our model, how do we measure its m predictions \hat{y} ?

A1

Cost function $J(\theta) = \frac{1}{2} \sum_{i=1}^m (\hat{y} - y)^2$

"Least Squares"

MLE vs MAP

We were performing the **maximum likelihood estimate**

Def:

maximum likelihood estimate (MLE) asserts that we should choose θ so as to maximize the probability of the observed data (i.e., our \hat{y} should become as close to y as possible)

MLE vs MAP

In other words, we were searching for $\hat{\theta}_{MLE}$

Say we have the likelihood function $P(D|\theta)$

$$\hat{\theta}_{MLE} = \operatorname{argmax}_\theta P(D|\theta)$$

MLE vs MAP

MAP stands for maximum a posteriori and is interested
in calculating $P(\theta|D)$

If we have knowledge about the prior distribution $P(\theta)$,
we can calculate:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(X)} = \propto P(D|\theta)P(\theta)$$

MLE vs MAP

MAP stands for maximum a posteriori and is interested
in calculating $P(\theta|D)$

If we have knowledge about the prior distribution $P(\theta)$,
we can calculate:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(X)} = \propto P(D|\theta)P(\theta)$$

$$\hat{\theta}_{MAP} = \operatorname{argmax}_\theta P(D|\theta)P(\theta)$$

MLE vs MAP

MAP stands for maximum a posteriori and is interested
in calculating $P(\theta|D)$

NOTE: If the prior $P(\theta)$ is uniform (i.e., not Gaussian or any other distribution), then $\hat{\theta}_{MAP} = \hat{\theta}_{MLE}$

Thus, MLE is a special case of MAP!

$$\hat{\theta}_{MAP} = \operatorname{argmax}_{\theta} P(D|\theta)P(\theta)$$

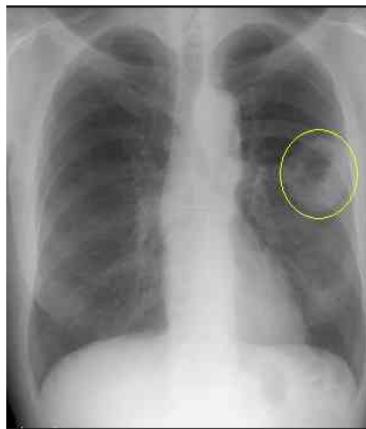
CS109B: What's next

We have learned a lot so far, with the assumption
that our data is “flat” (each feature/column is
independent of the others)

But what if our data is different?

CS109B: What's next

Scenario: imagine having picture data, whereby each pixel is a feature. Obviously, pixels near one another in 2D space (both vertically and horizontally) are highly correlated.



Detecting lung cancer

A flattened vector wouldn't work well.

CS109B: What's next

Solution: **CNNs**

CS109B: What's next

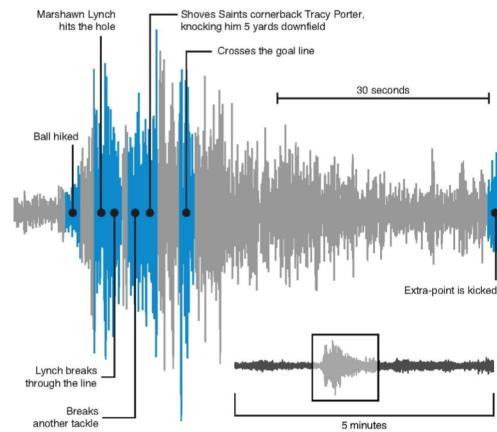
We have learned a lot so far, with the assumption that our data is **i.i.d.** (each row is independent from one another)

But what if our data is different?

CS109B: What's next

Scenario: imagine having data that is sequential in nature (e.g., natural text, speech, video frames, time series data)

"Today, I went to the _____"



UNDERSTANDING
LANGUAGE

PREDICTING
EARTHQUAKES

CS109B: What's next

Solution: **RNNs / LSTMs**

CS109B: What's next

We have learned that PCA can transform our data while maintaining variance. However, it's unsupervised. Can we **learn** a better representation of our data?

Perhaps, we can learn how the data was "generated"?

Solution: **Autoencoders**

CS109B: What's next

Can we generate realistic, synthetic data, and do so in such a realistic way that it **increases the performance** of our classifiers?

Solution: **GANs (not GAMs)**

DeepFake is an example that uses GANs

CS109B: What's next

Instead of making just 1 prediction per preset input, sometimes we may want to get real-time feedback as to what our prediction's effects were. For example, navigating through an environment or game (Mars or Chess Board)

We need to represent the updated environment, possible actions to take, risks of those actions, etc.

Solution: **Reinforcement Learning**

Questions?