CS109B Recap

What have we learned, and where are we now?

Harvard IACS

CS109B

Pavlos Protopapas, Mark Glickman, and Chris Tanner

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

Your Data 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

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

Your Data 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

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

Your Data 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

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

Your Data 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

- 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**)

Your Data 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

- 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: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$

X

Age	Play	Rainy
22	N	Y
29	Y	N
31	N	N
23	Y	N
37	N	Y
41	Y	N
29	Y	Y
21	N	N
30	Y	N

Y

Temp
91
89
56
71
72
83
97
64
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: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$
- Want a model f that is:
  - Supervised

X

Age	Play	Rainy
22	N	Y
29	Y	N
31	N	N
23	Y	N
37	N	Y
41	Y	N
29	Y	Y
21	N	N
30	Y	N

Y

Temp
91
89
56
71
72
83
97
64
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: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$
- Want a model f that is:
  - Supervised

X

Age	Play	Rainy
22	N	Y
29	Y	N
31	N	N
23	Y	N
37	N	Y
41	Y	N
29	Y	Y
21	N	N
30	Y	N

Y

Temp
91
89
56
71
72
83
97
64
68

Def:

<u>Supervised</u> 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		Y
	Play	**Rainy**	**Temp**
	N	Y	91
	Y	N	89
	N	N	56
	Y	N	71
	N	Y	72
	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: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$
- Want a model f that is:
  - Supervised
  - Predicts real numbers (regression model)

X

Age	Play	Rainy
22	N	Y
29	Y	N
31	N	N
23	Y	N
37	N	Y
41	Y	N
29	Y	Y
21	N	N
30	Y	N

Y

Temp
91
89
56
71
72
83
97
64
68

- Given some data such that each

Def:

<u>Regression</u> models are **supervised** models, whereby **Y** are *continuous* values.

- Predicts real numbers (regression model)

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

- Given some data such that each

Def:

<u>Regression</u> models are **supervised** models, whereby **Y** are *continuous* values.

<u>Classification</u> models are **supervised** models, whereby **Y** are *categorical* values.

- Predicts real numbers (regression model)

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

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 $\boldsymbol{\theta}$ are

2. Our **optimization** algorithm?

   Determines how we update our parameters $\boldsymbol{\theta}$ so that our model better fits our training data (e.g., closed-form equations; gradient descent)

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 $\boldsymbol{\theta}$ are

2. Our **optimization** algorithm?

   Determines how we update our parameters $\boldsymbol{\theta}$ so that our model better fits our training data (e.g., closed-form equations; gradient descent)

When **testing** your model's predictions, be mindful of your metric selection:

3. Our **evaluation metric**

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

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.

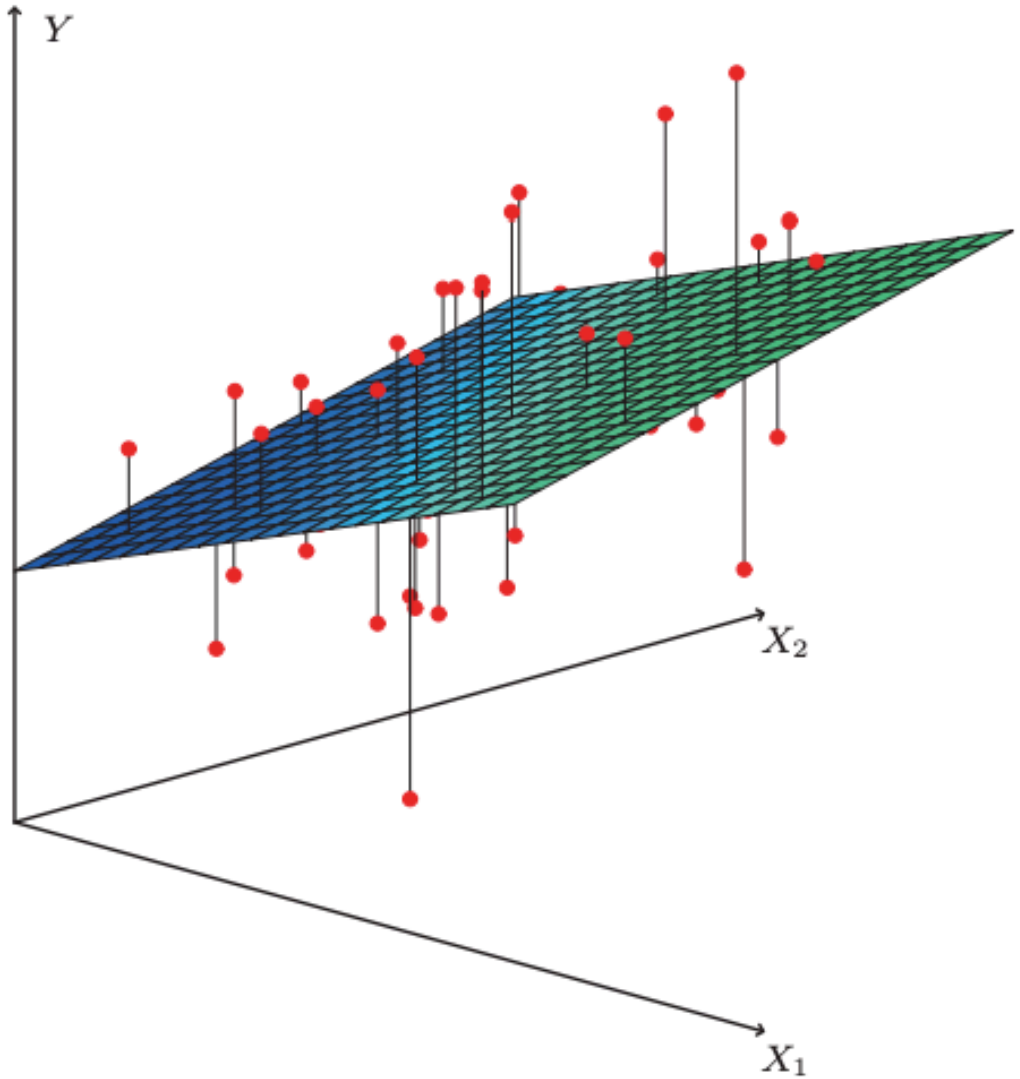


Photo from "*An Introduction to Statistical Learning*" (James, et al. 2017)

Linear Regression

PROS

- **Simple** and **fast** approach to model linear relationships
- Interpretable results via $\boldsymbol{\theta}$ ($\boldsymbol{\beta}$ 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: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$

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: $\boldsymbol{Y} = f(\boldsymbol{X}) + \varepsilon$
- Want a model that is:
  - Supervised
  - Predicts categories/classes (classification model)
- **Q**: What model could we use?

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

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**)

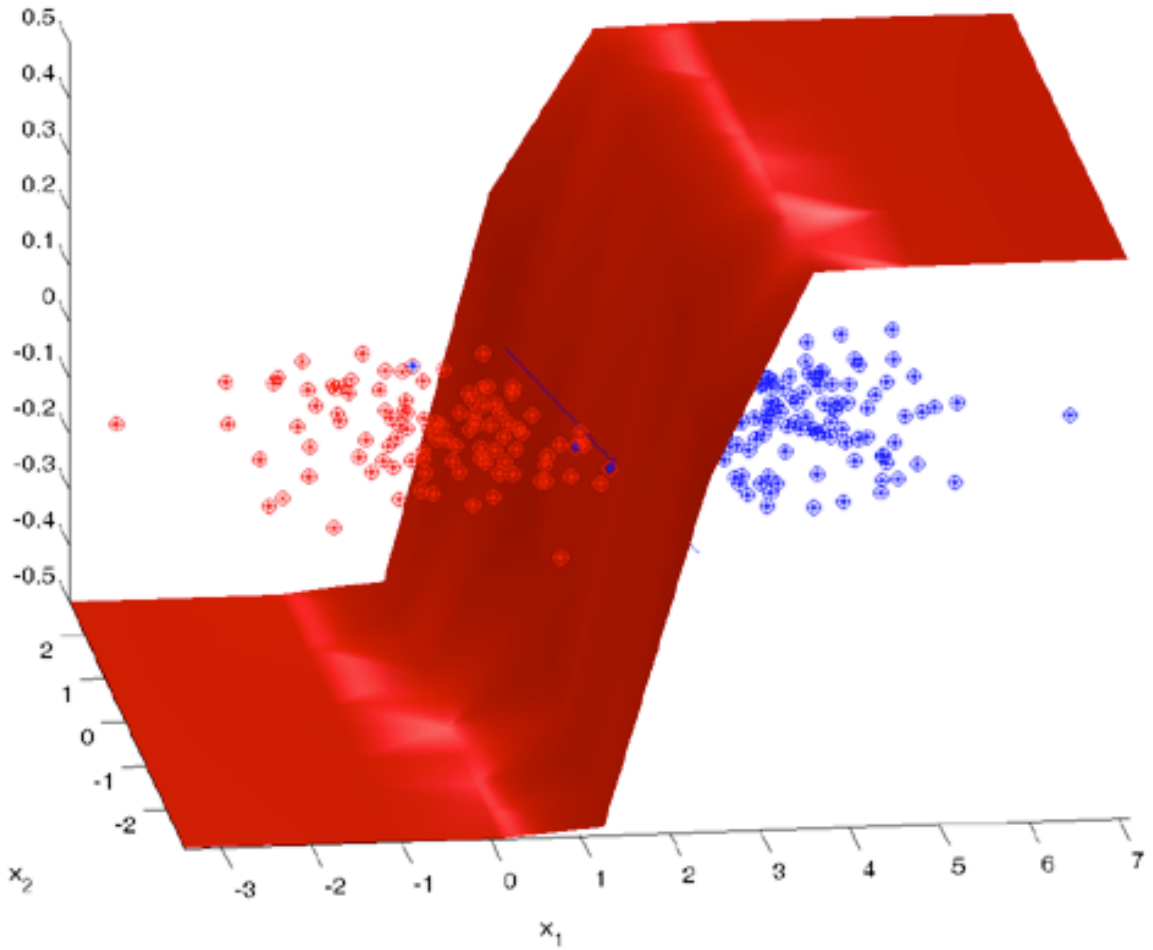


Photo from http://strijov.com/sources/demoDataGen.php (Dr. Vadim Strijov)

Parametric Models

- So far, we've assumed our data **X** and **Y** can be represented by an underlying model $\boldsymbol{f}$ (i.e., $\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}) + \varepsilon$) that has a particular form (e.g., a linear relationship, hence our using a linear model)

- We fit the model $\boldsymbol{f}$ by estimating its parameters $\boldsymbol{\theta}$

- Parametric models make the above assumptions. Namely, that there exists an underlying model $\boldsymbol{f}$ 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.

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*.

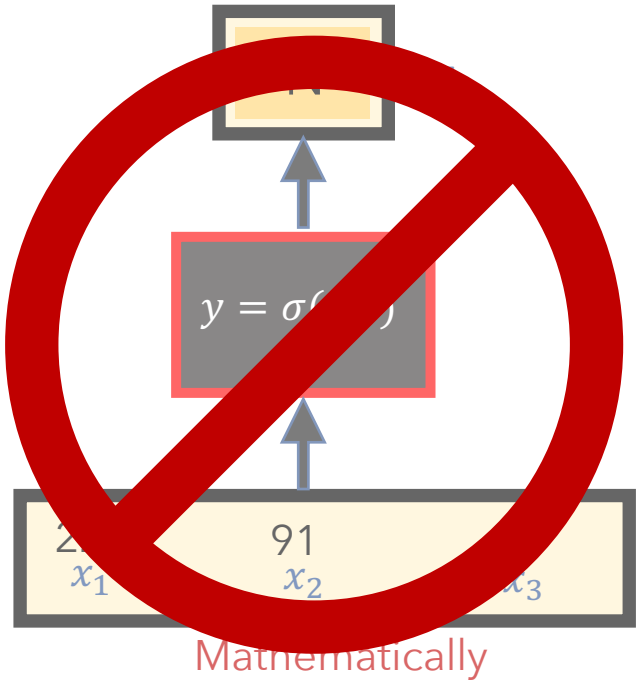
- Returning to our data, let's again predict if a person will **Play**

- If we do not want to assume any particular form 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

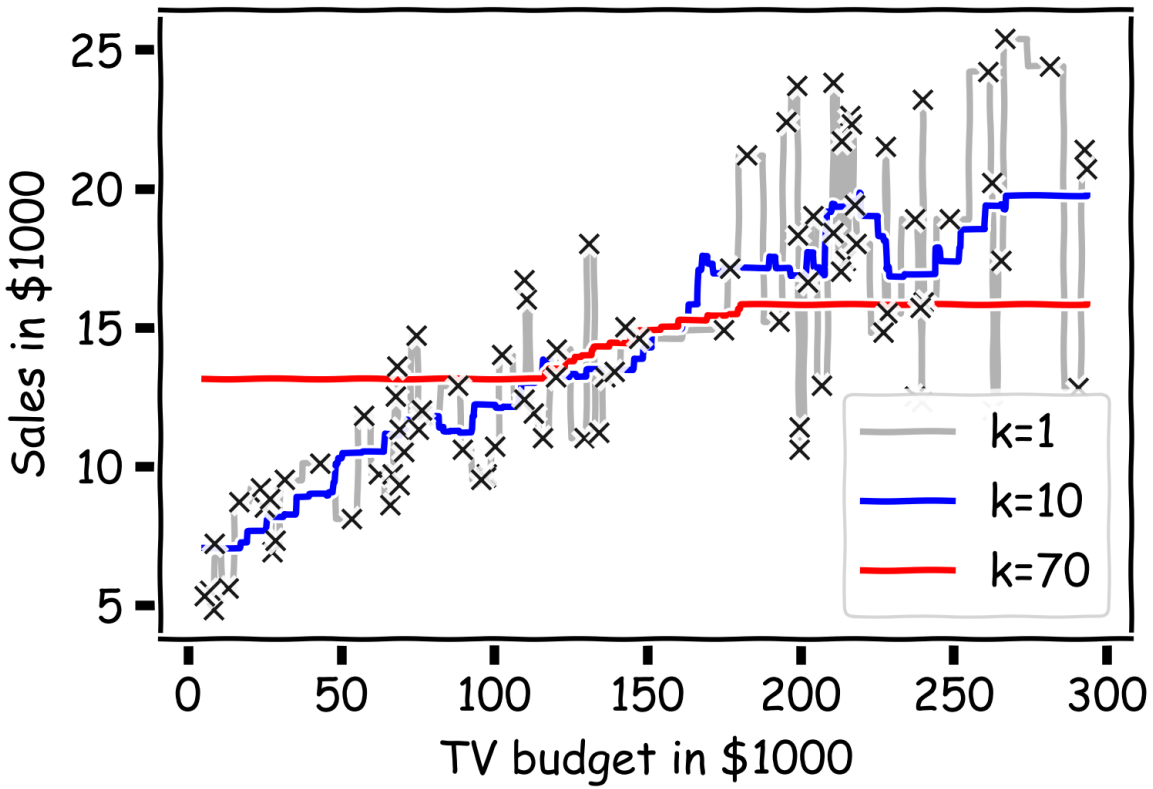
Refresher:



- k-NN doesn't train a model
- One merely specifies a $\boldsymbol{k}$ value
- At test time, a new piece of data $\boldsymbol{a}$:
  - must be compared to all other training data $\boldsymbol{b}$, to determine its <u>k-nearest neighbors</u>, per some distance metric $\boldsymbol{d}(\boldsymbol{a}, \boldsymbol{b})$
  - 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 $\boldsymbol{X}$ or the form of $\boldsymbol{f}(\boldsymbol{X})$
- k-NN is a **non-parametric model**



k-NN

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 any particular form about how **X** and **Y** relate, believing that <u>no single equation</u> 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

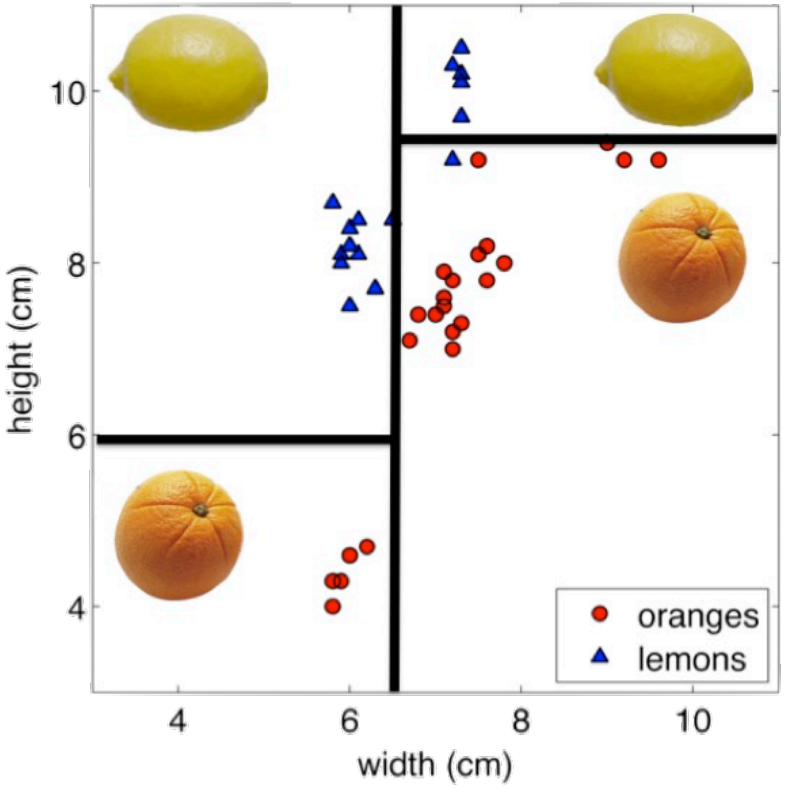
Refresher:

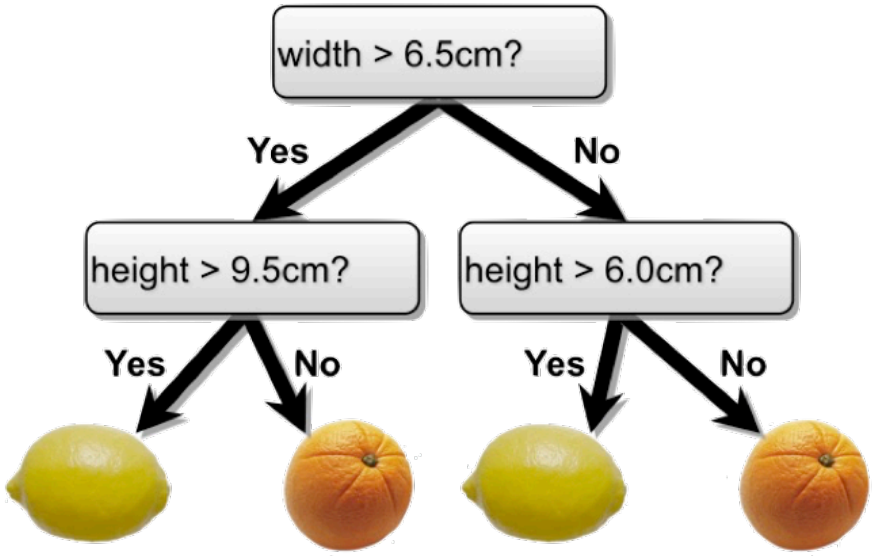
- A **Decision Tree** iteratively determines how to split our data by the best feature value so as to minimize the entropy (uncertainty) of our resulting sets.

- 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 <u>a single feature</u>. Typically, at each iteration, we split once along one dimension (one predictor).





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

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

Your Data 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

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 that 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

- 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
- i.e., **unsupervised** learning

Your Data 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

Refresher:

- There are many approaches to clustering

  (e.g., k-Means, hierarchical, DBScan)

- Regardless of the approach, we need to specify a distance metric

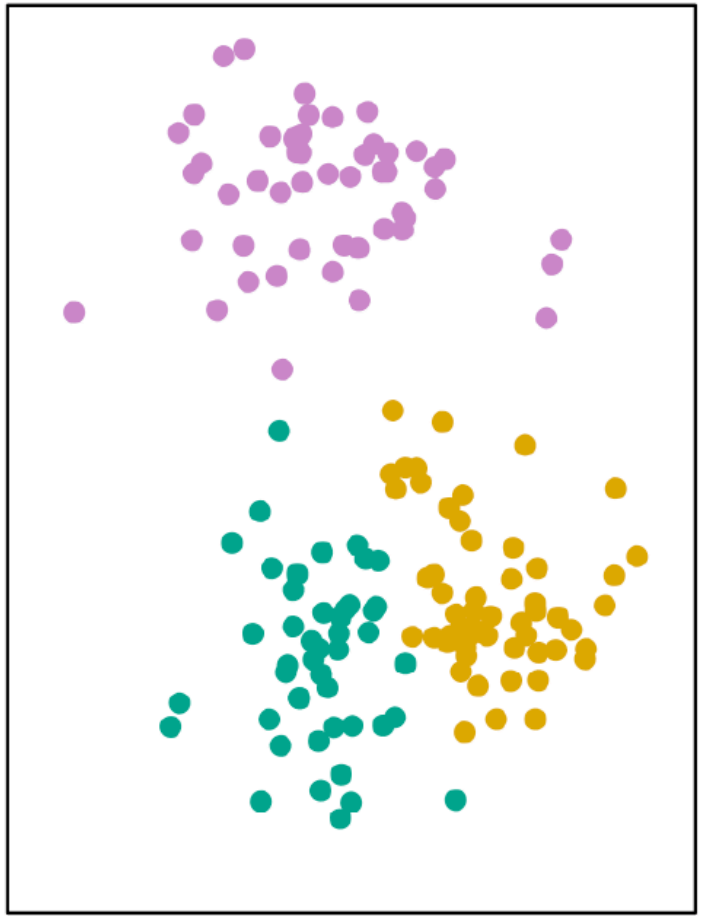
  (e.g., Euclidean, Manhattan)

- **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)

k-Means example:

- Although we are not explicitly using any column Y, one could imagine that the 3 resulting cluster labels are our 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



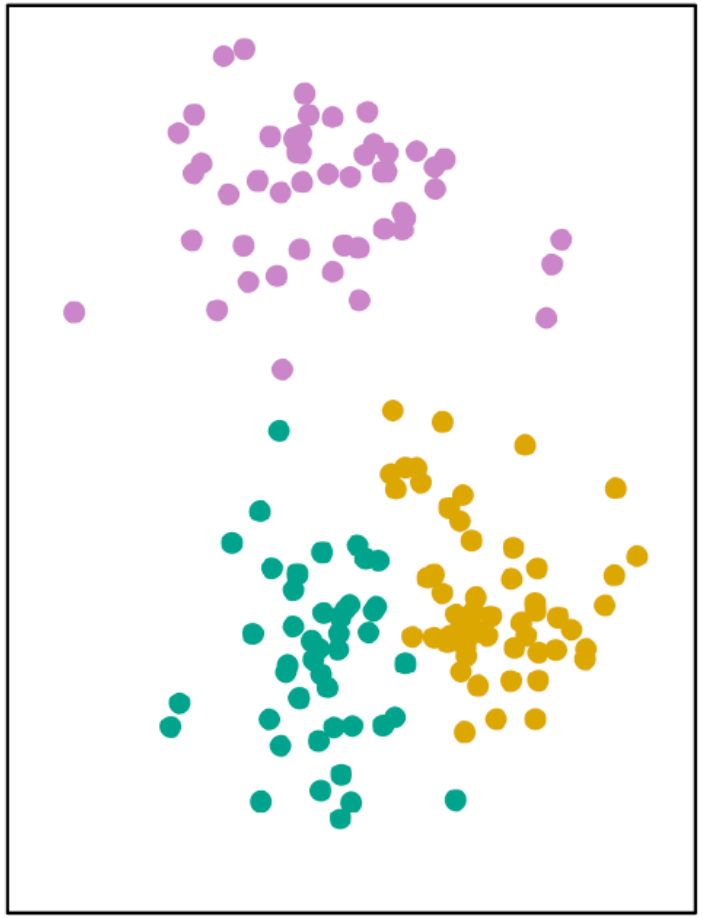
Visual Representation

k-Means example:

- Although we are not explicitly using any column Y, one could imagine that the 3 resulting cluster labels are our 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

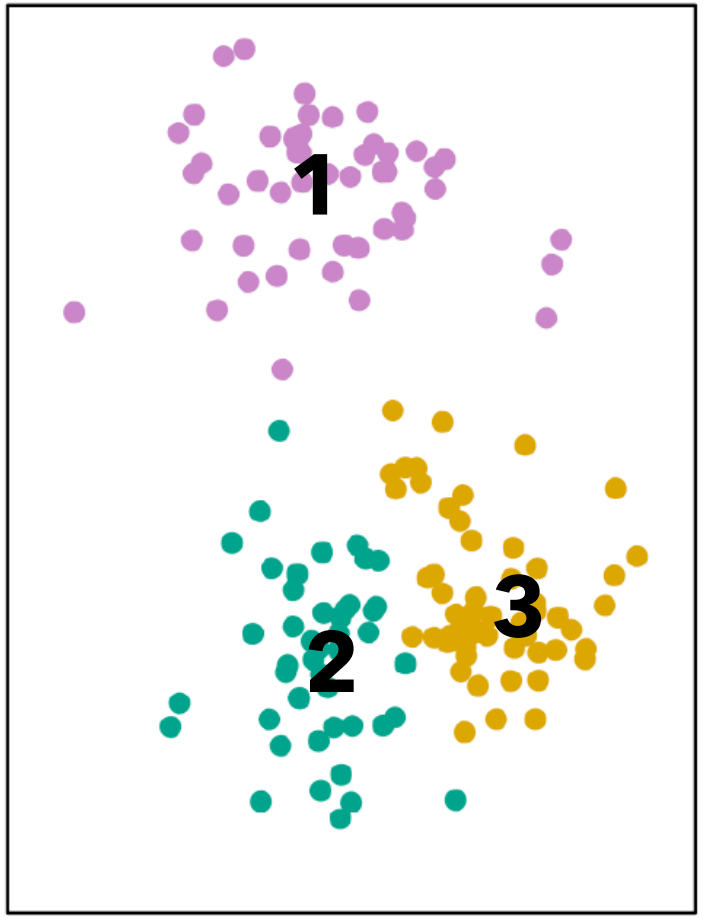
- Yet, one could imagine a narrative whereby our data points were **generated** by these 3 classes.



Visual Representation

k-Means example:

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

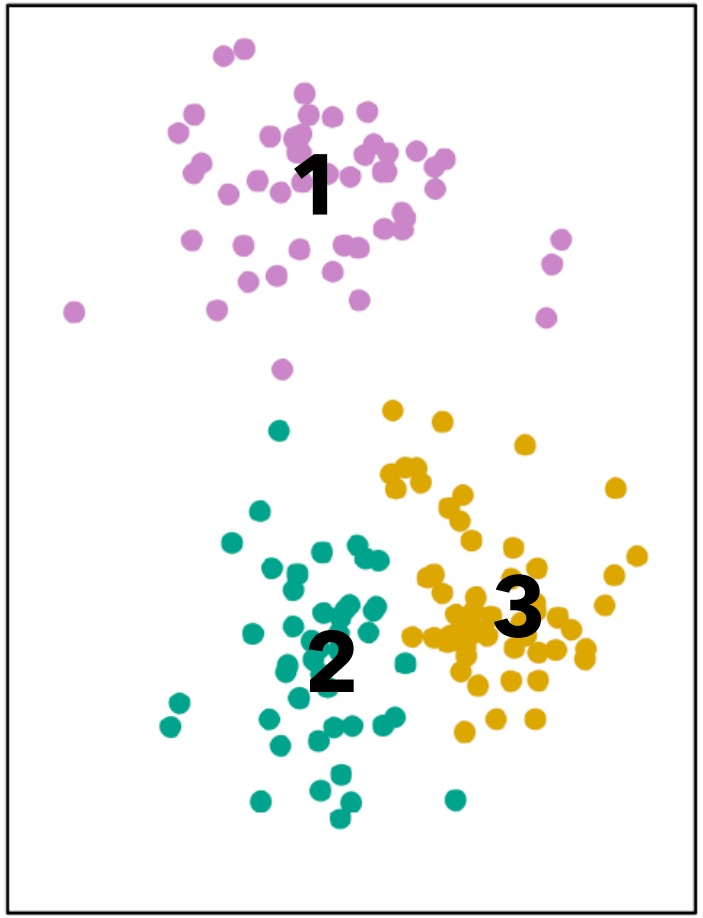


Visual Representation

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 $\boldsymbol{X}$ exists and has the following class labels $\boldsymbol{Y}$. This is a **generative** model.

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



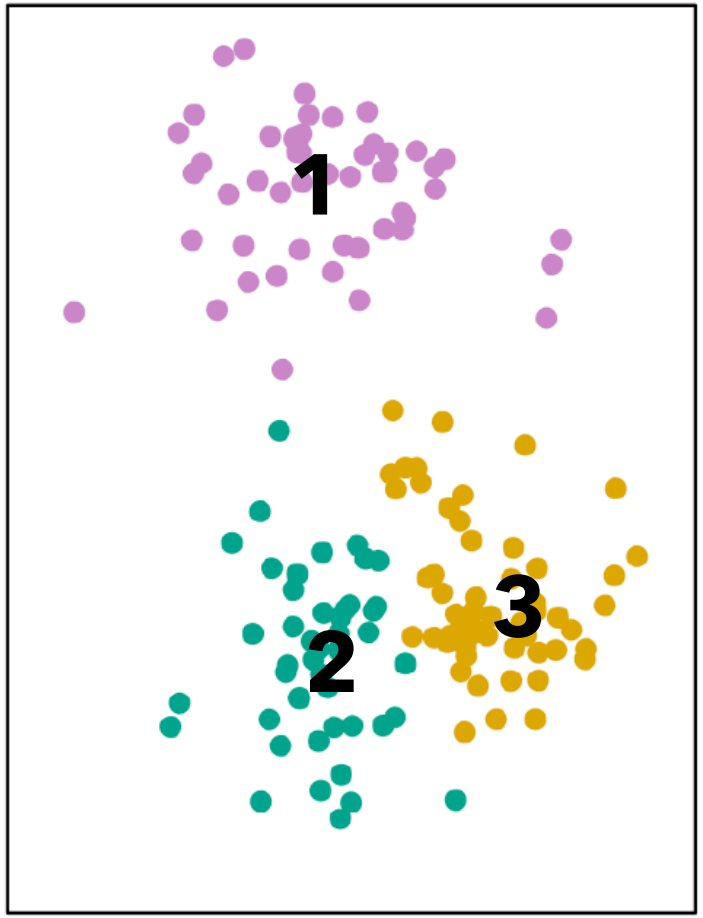
Visual Representation

k-Means example:

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

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

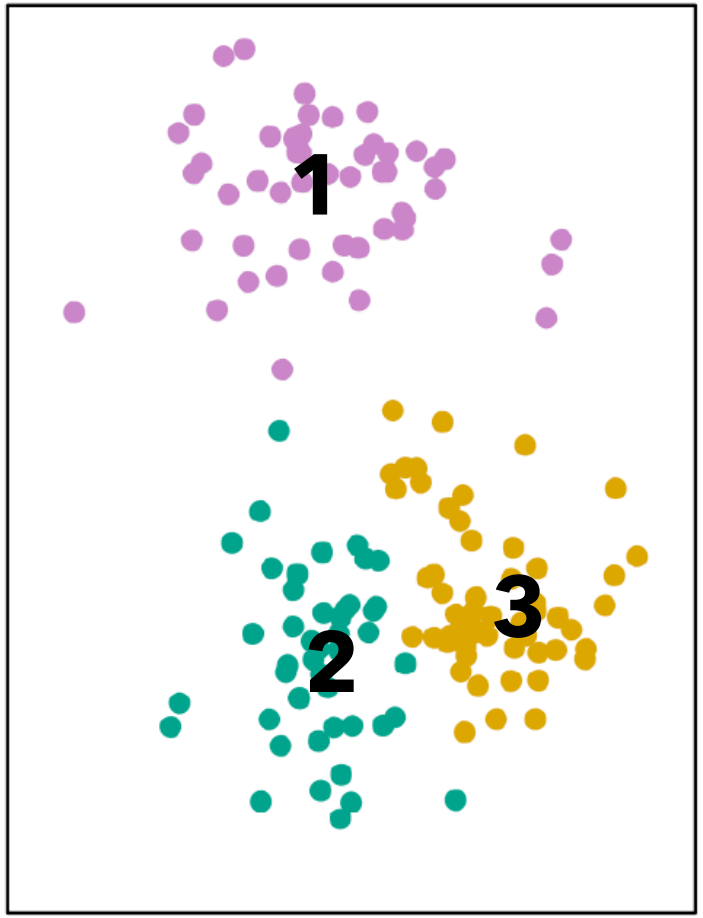
- We can estimate the joint probability of $P(Y, X)$



Visual Representation

k-Means example:

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



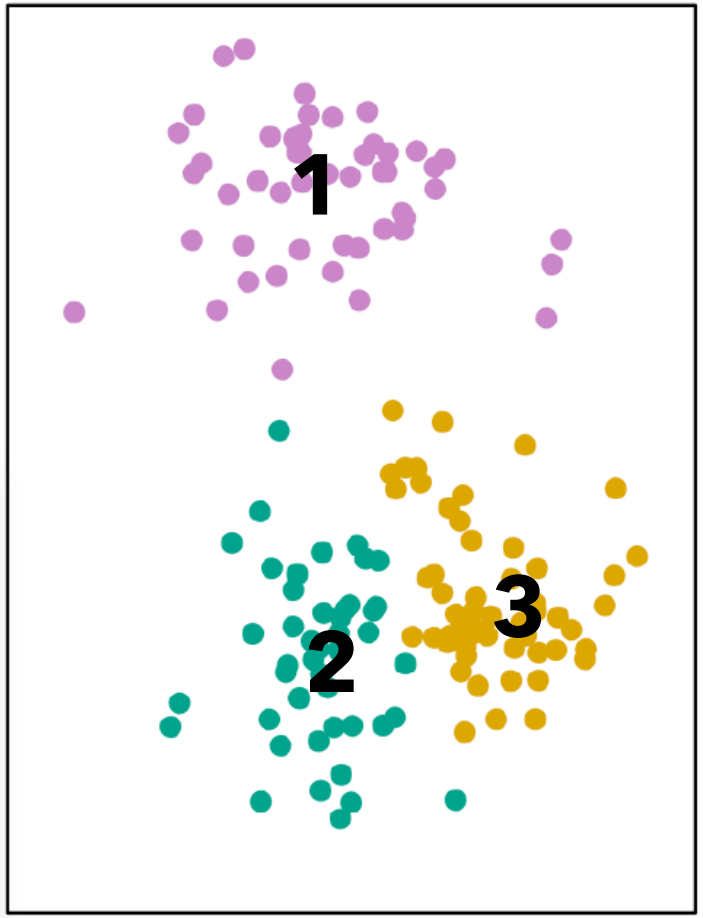
Visual Representation

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 $P(\boldsymbol{Y}|\boldsymbol{X})$:

Supervised models are given some data $\boldsymbol{X}$ and want to calculate the probability of $\boldsymbol{Y}$.

They learn to **discriminate** between different values of possible $\boldsymbol{Y}$'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
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	Sup		ric	Discriminative
Decision Tree	Sup		ric	Discriminative
PCA	Unsu		ic	neither
Clustering	Unsu		ric	Generative

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

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

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

	Supervised vs **Unsupervised**	**Regression** vs **Classification**	**Parametric** vs **Non-Parametric**	**Generative** vs **Discriminative**
Linear Regression	Supervised	Regression	Parametric	Discriminative
Logistic Regression	S		c	**Discriminative**
k-NN	S		etric	Discriminative
Decision Tree	S		etric	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			metric	**Discriminative**
Decision Tree			metric	Discriminative
PCA			metric	neither
Clustering	Unsupervised	neither	Non-Parametric	Generative

Given training X, learns to discriminate between possible Y values (quantitative or 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	Supervised	either	Non-Parametric	Discriminative
Decision Tree			metric	**Discriminative**
PCA			metric	neither
Clustering			metric	Generative

Given training X, learns decision boundaries so as to discriminate between possible Y values (quantitative or 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	Supervised	either	Non-Parametric	Discriminative
Decision Tree	Supervised	either	Non-Parametric	Discriminative
PCA				neither
Clustering				Generative

PCA is a **process**, not a model, so it doesn't make sense to consider it as a Discriminate or Generative model

	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	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

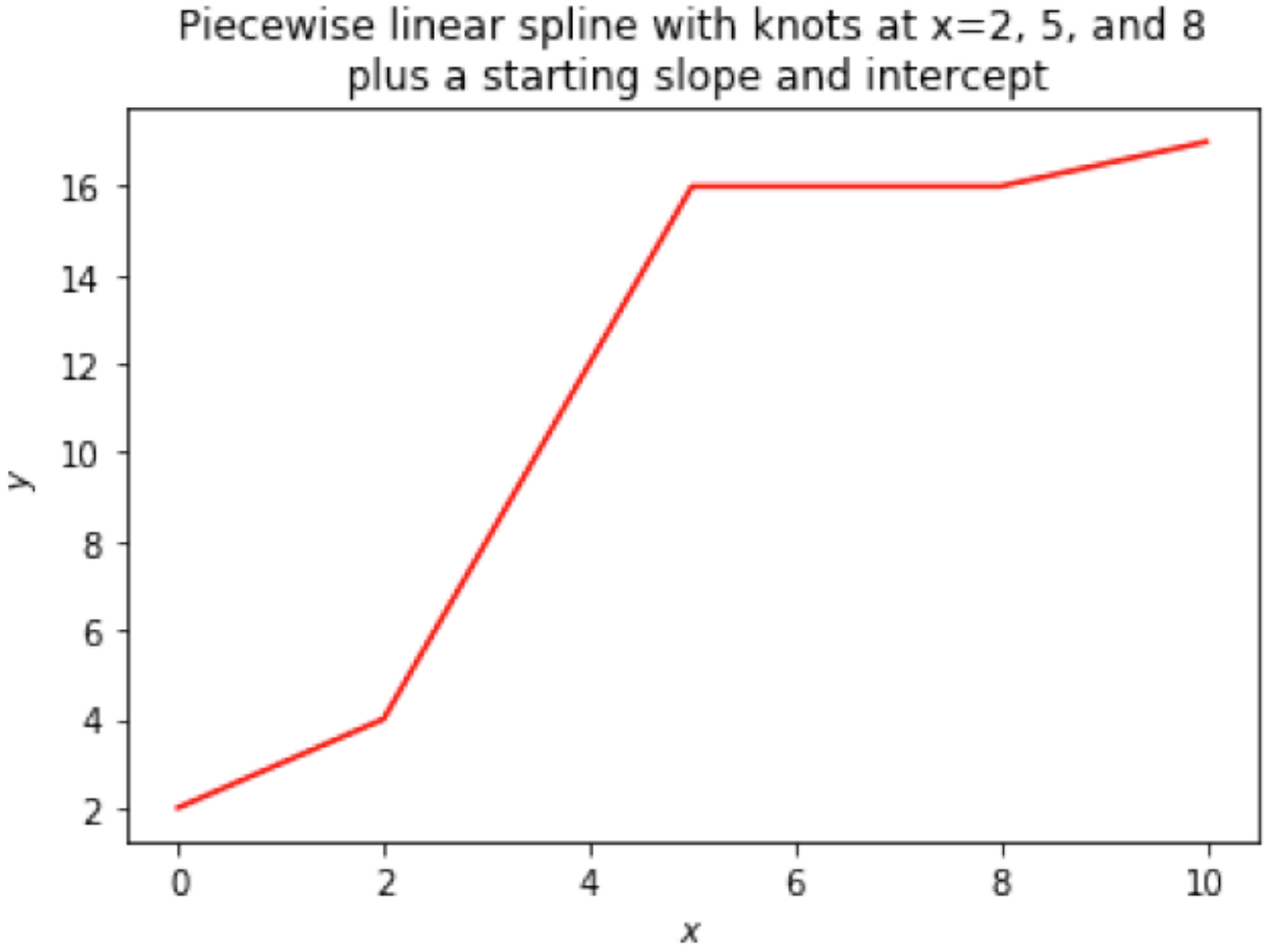
Age	Play	Rainy
22	N	Y
29	Y	N
31	N	N
23	Y	N
37	N	Y
41	Y	N
29	Y	Y
21	N	N
30	Y	N

Y

Temp
91
89
56
71
72
83
97
64
68

Refresher:

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

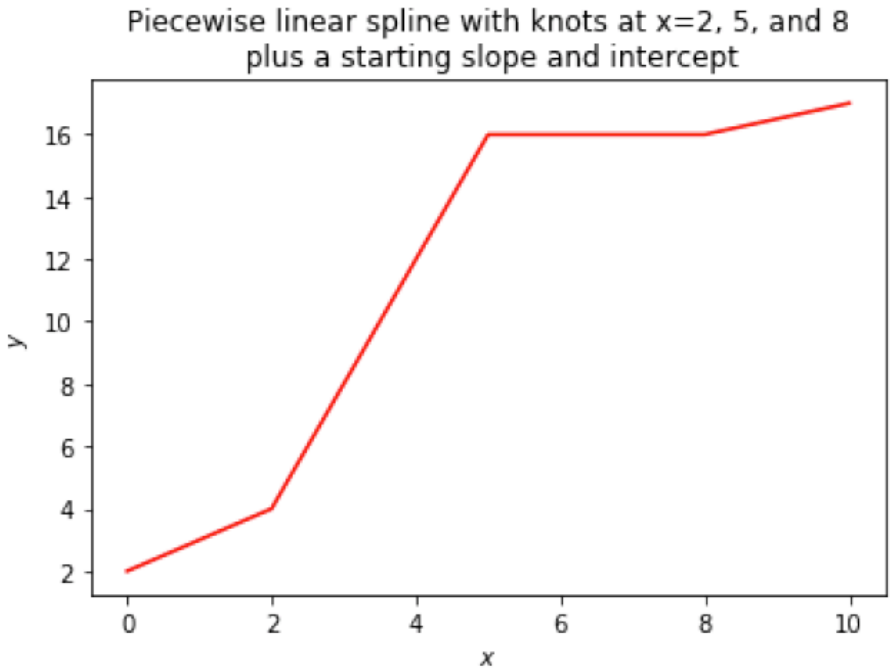


Refresher:

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

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

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



Generalized Additive Models (GAMs)

Y

$f(\boldsymbol{X})$

X

High-level

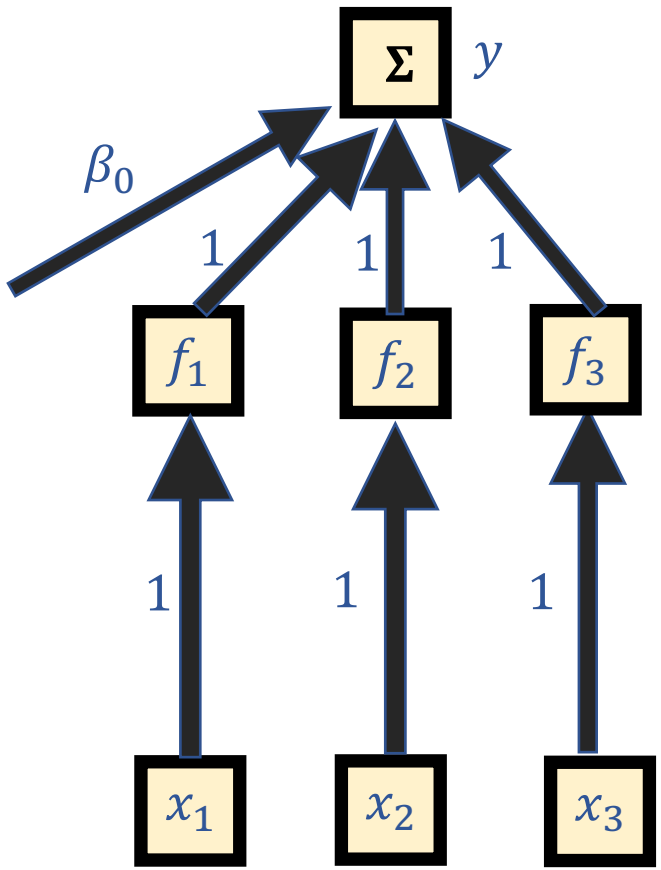
91

y

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

22	N	Y
x_1	x_2	x_3

Mathematically



Graphically
(NN format)

Generalized Additive Models (GAMs)

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

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

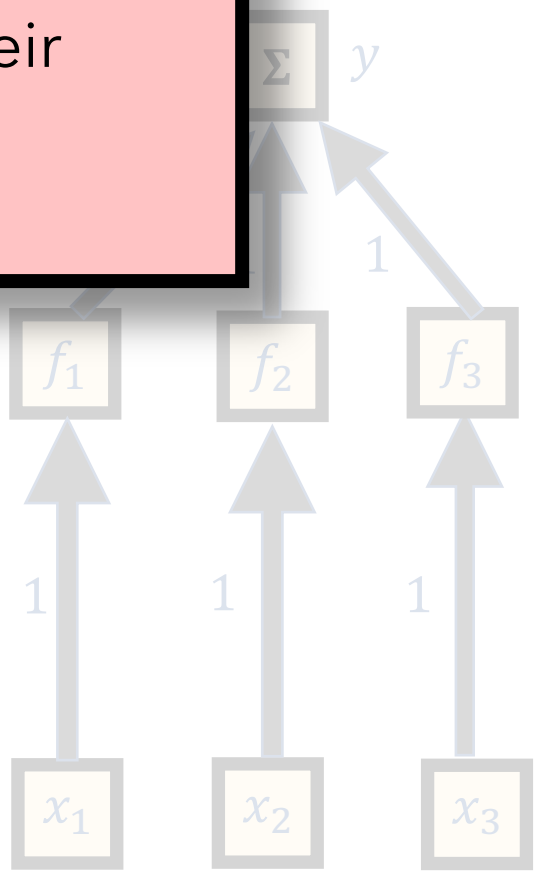
$f(X)$

X

High-level

22	N	Y
x_1	x_2	x_3

Mathematically



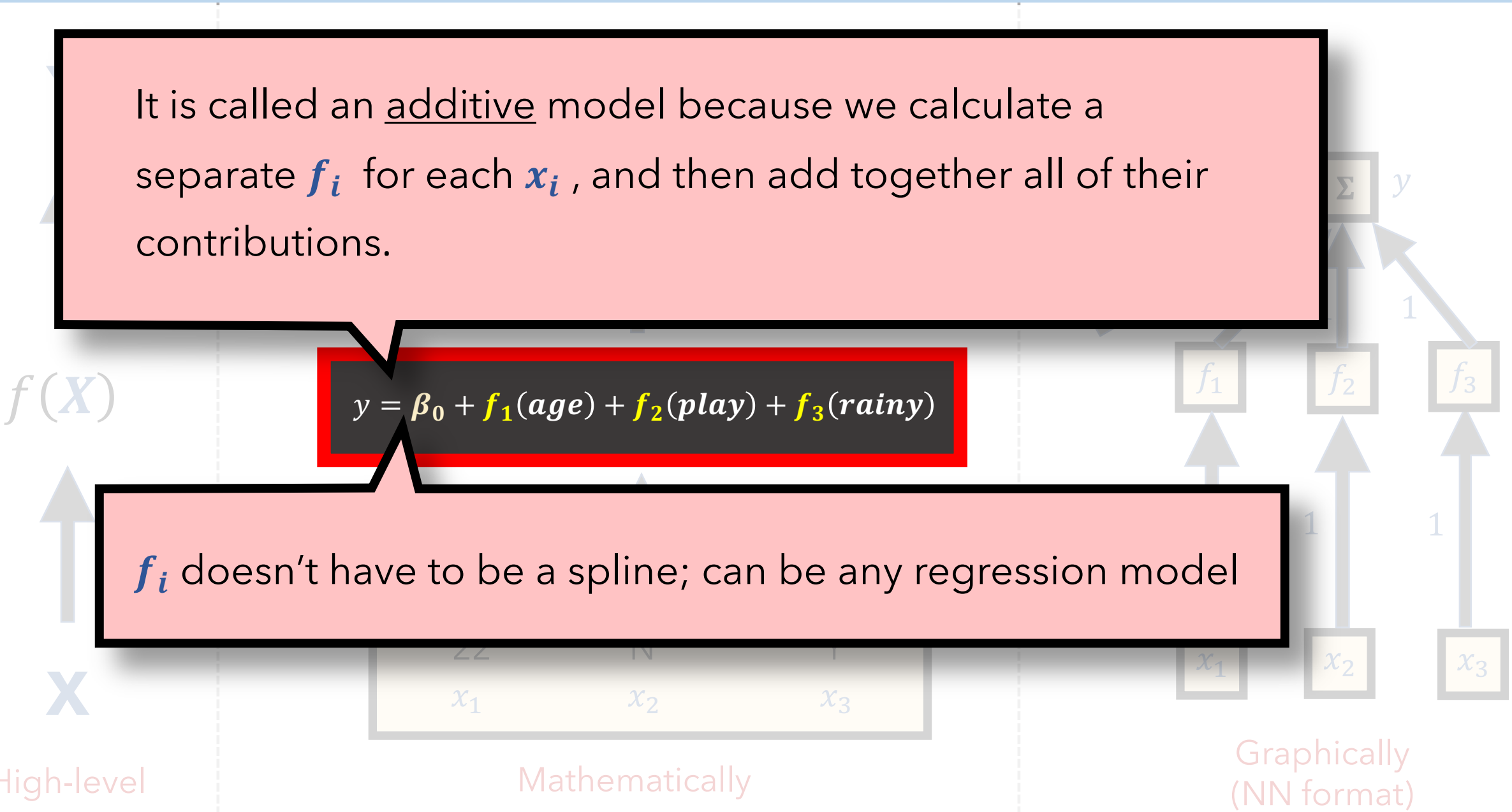
Graphically
(NN format)

Generalized Additive Models (GAMs)

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

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

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



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
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 <u>don't care about interpretability</u>

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

Feed-Forward Neural Network

Y

$f(\boldsymbol{X})$

X

High-level

N

y

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

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

22	N	Y
x_1	x_2	x_3

Mathematically

Feed-Forward Neural Network

Y

$f(\boldsymbol{X})$

X

High-level

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

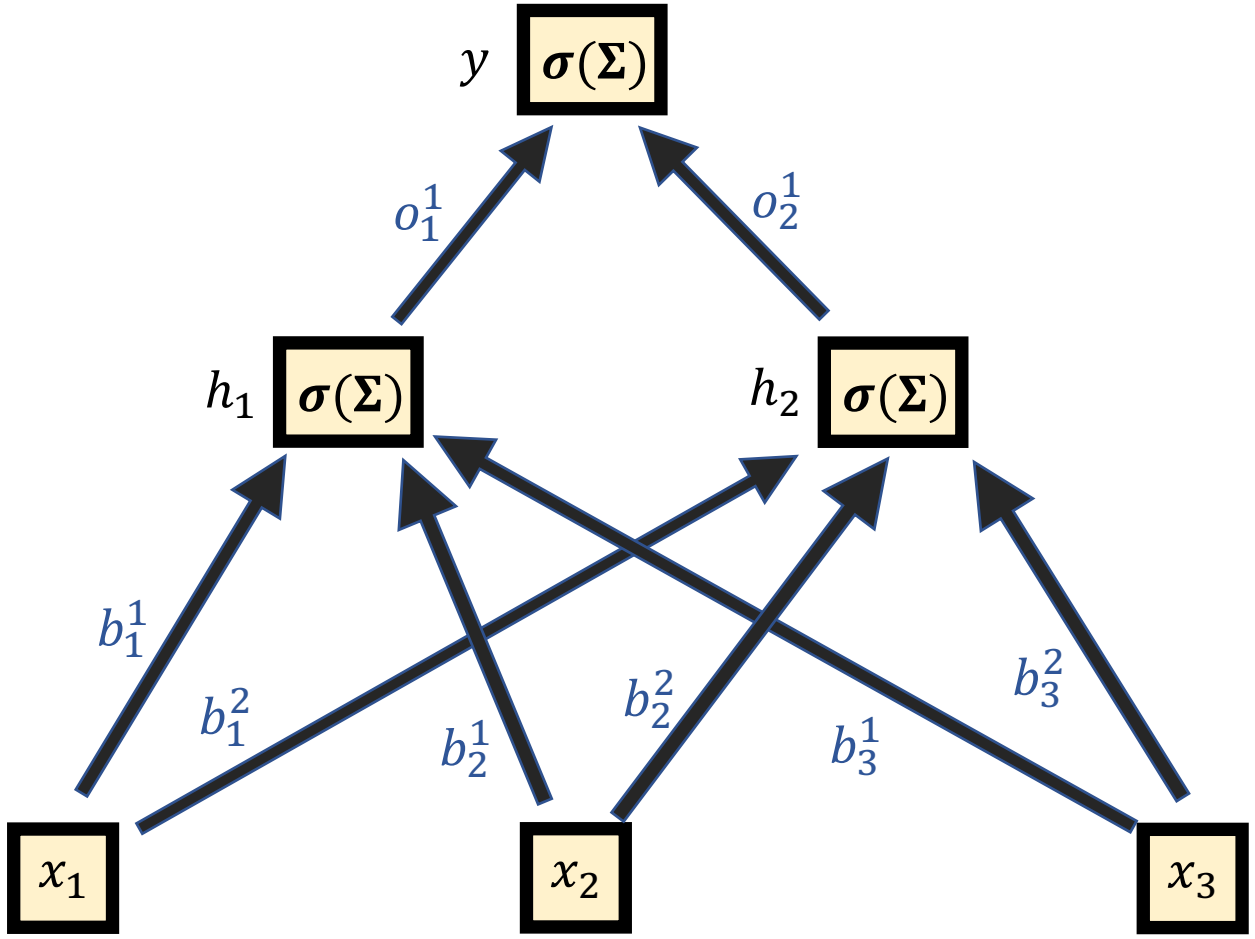
$$h_i = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3)}} = \sigma(\beta^1 X)$$

NOTE: a Neural Network can be viewed as a function $f(\boldsymbol{X})$, 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)

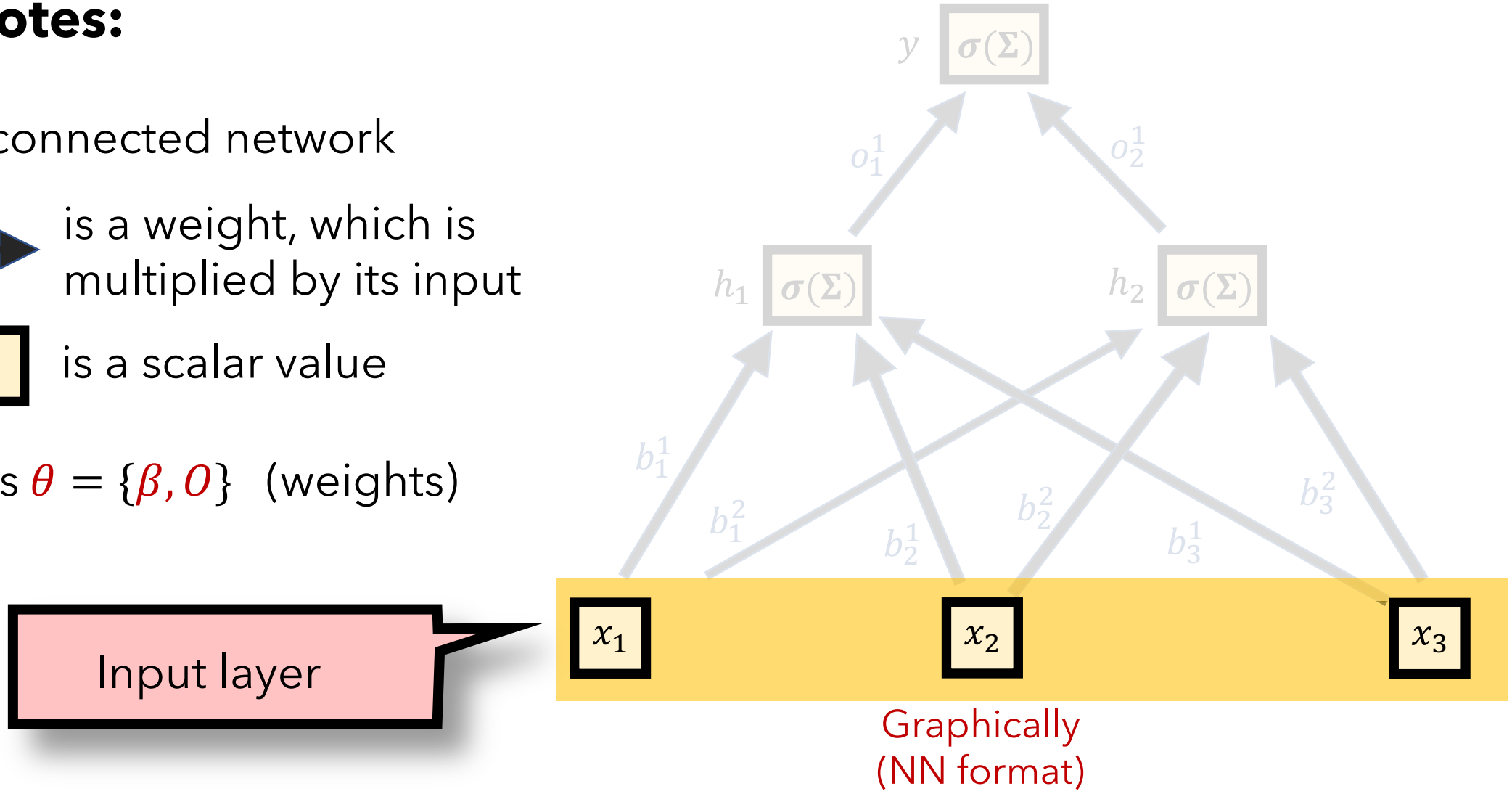


Graphically
(NN format)

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)

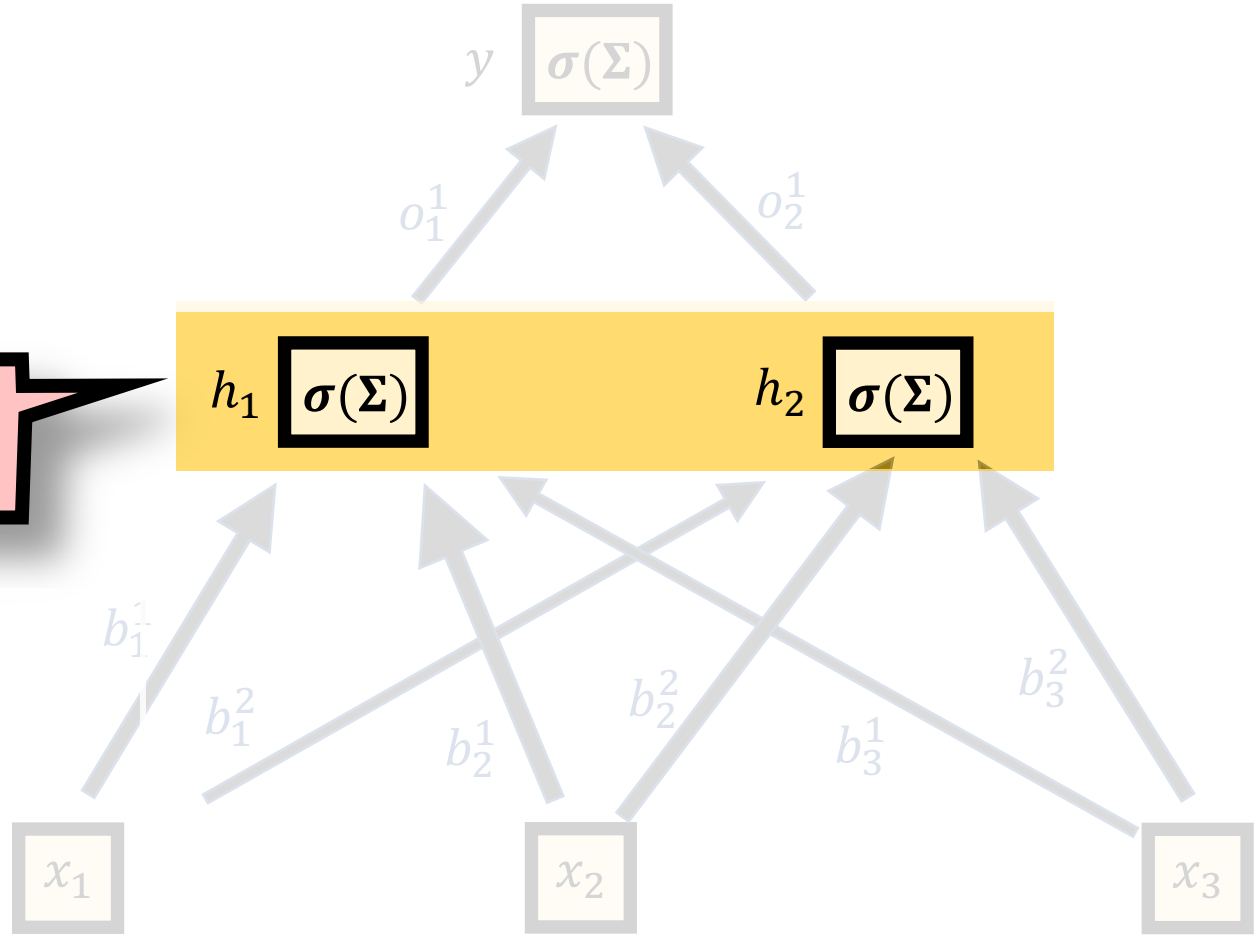


General Notes:

- It's a fully connected network
- Every → is a weight, which is mult
- Every ▭ is a s

?

- Parameters $\theta = \{\beta, O\}$ (weights)



Graphically
(NN format)

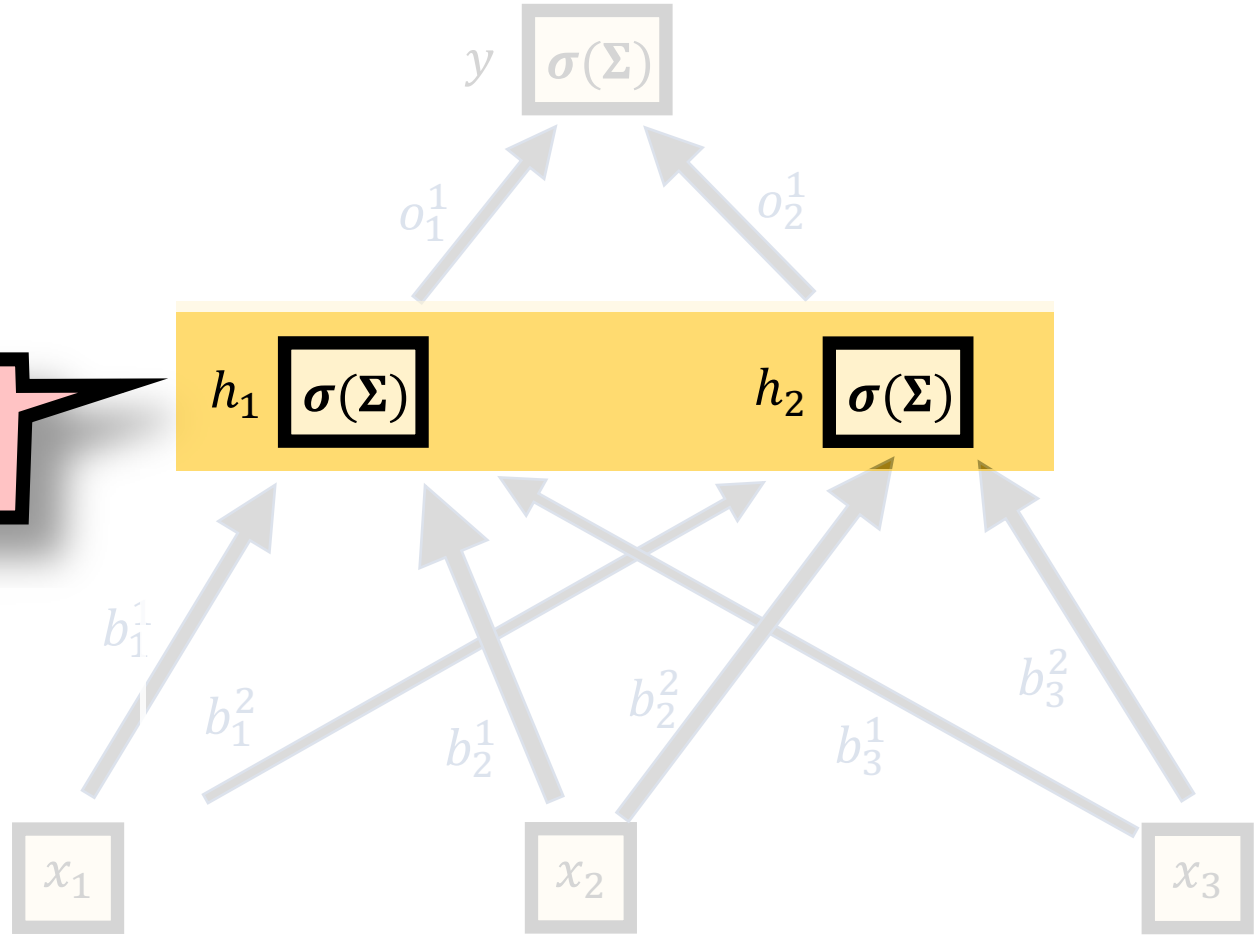
Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Every → is a weight, which is mult
- Every ▭ is a s

Hidden layer

- Parameters $\theta = \{\beta, O\}$ (weights)

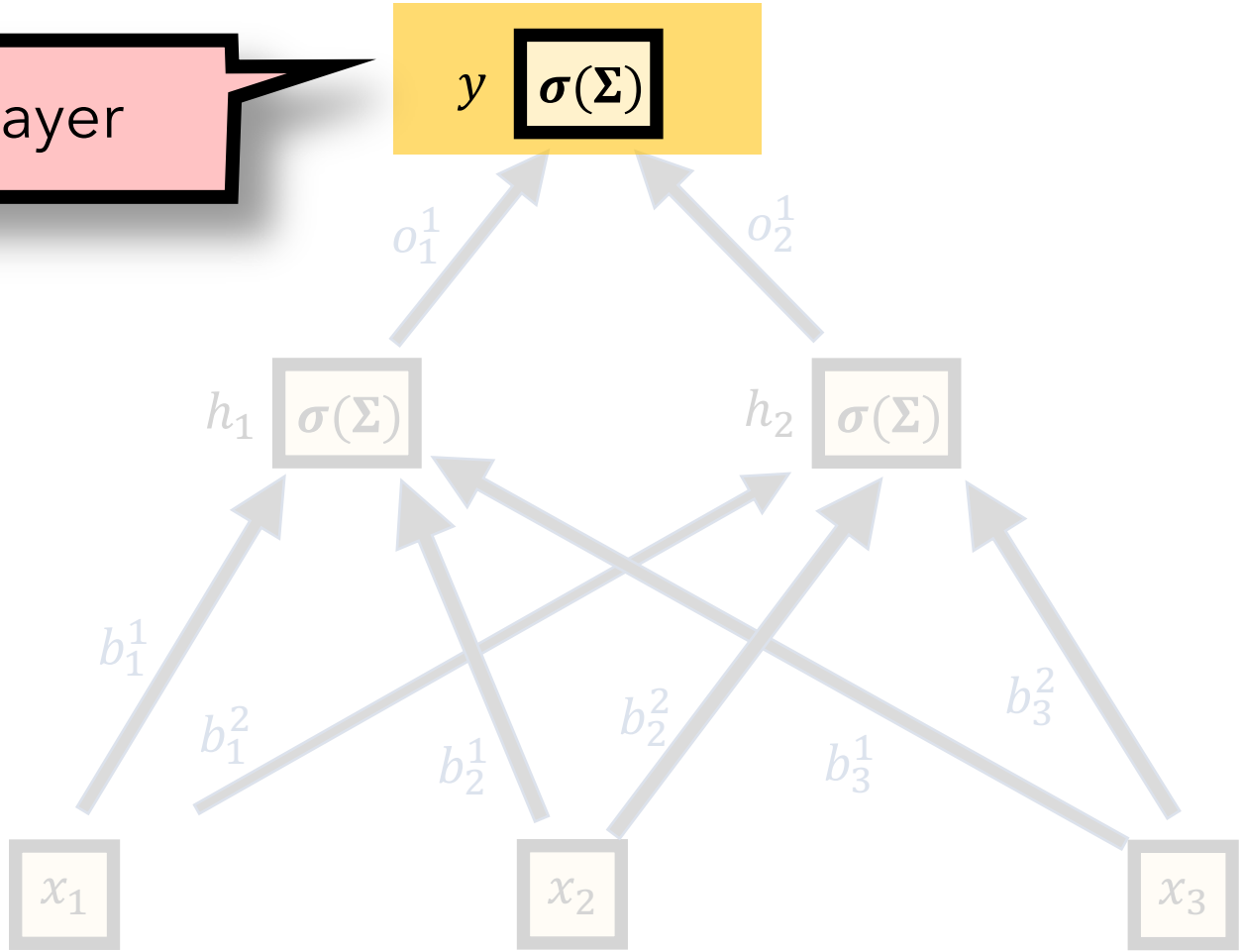


Graphically
(NN format)

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)

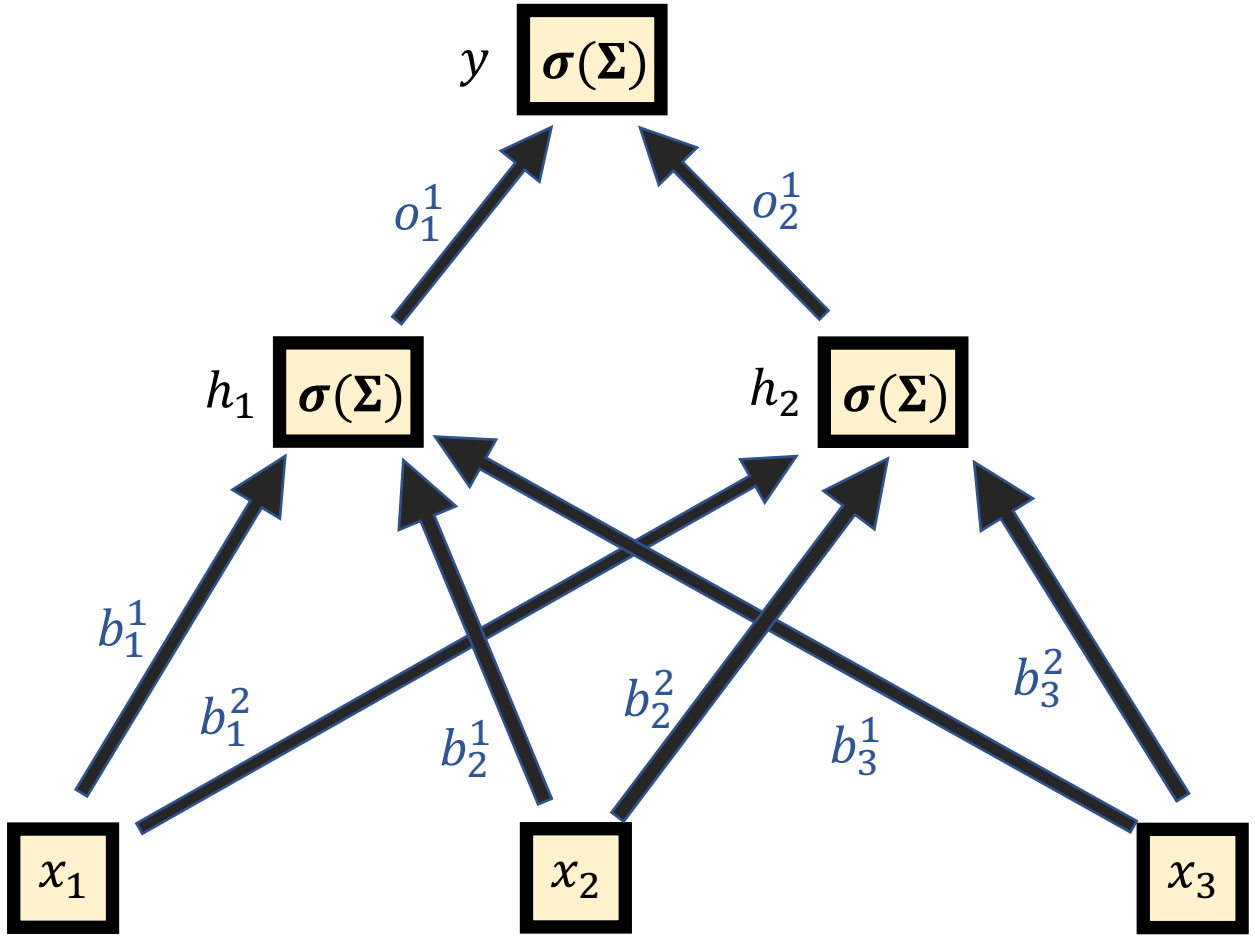


Graphically
(NN format)

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)

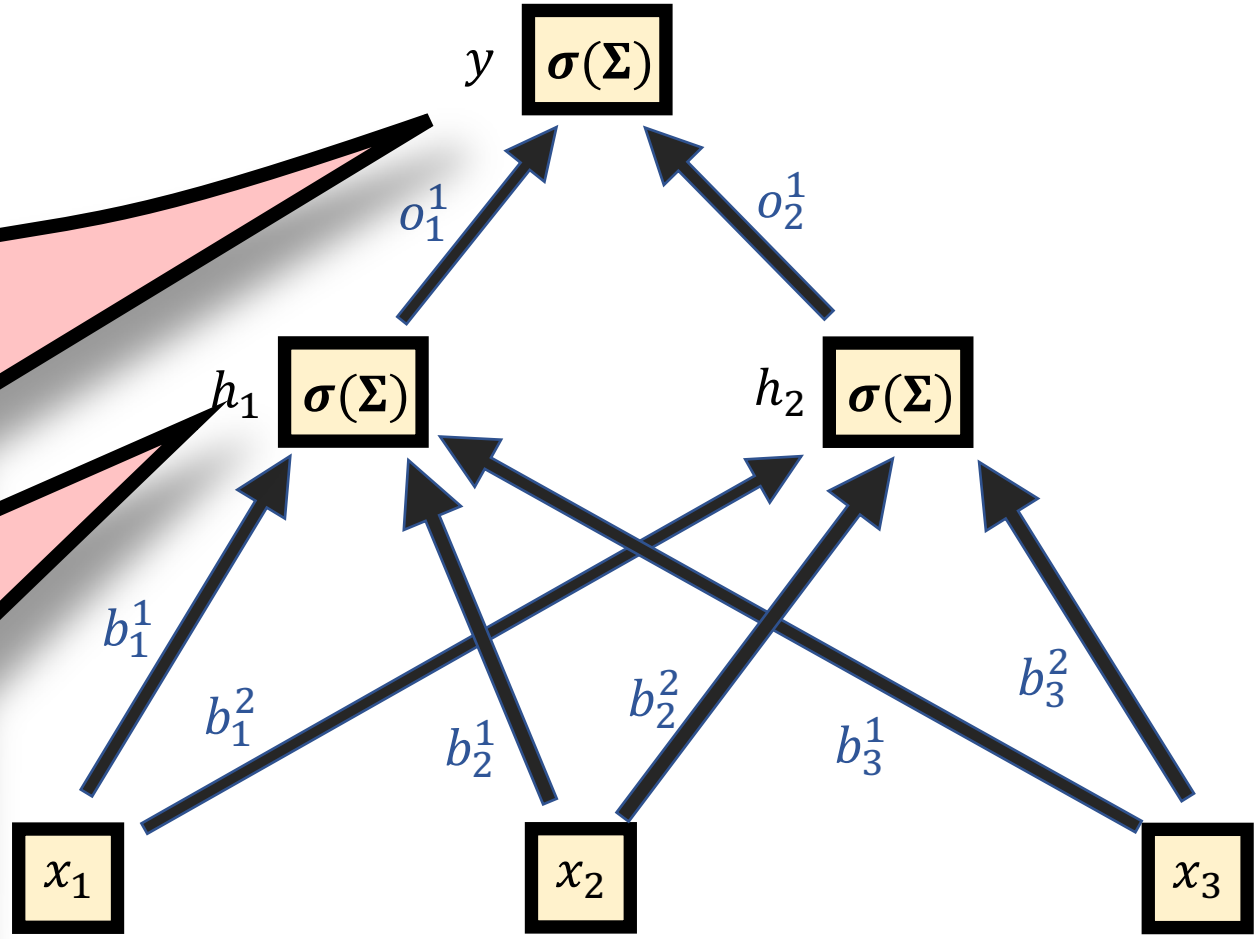


Graphically
(NN format)

General Notes:

- It's a fully connected network
- Ev
- Ev
- Pa

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



Graphically
(NN format)

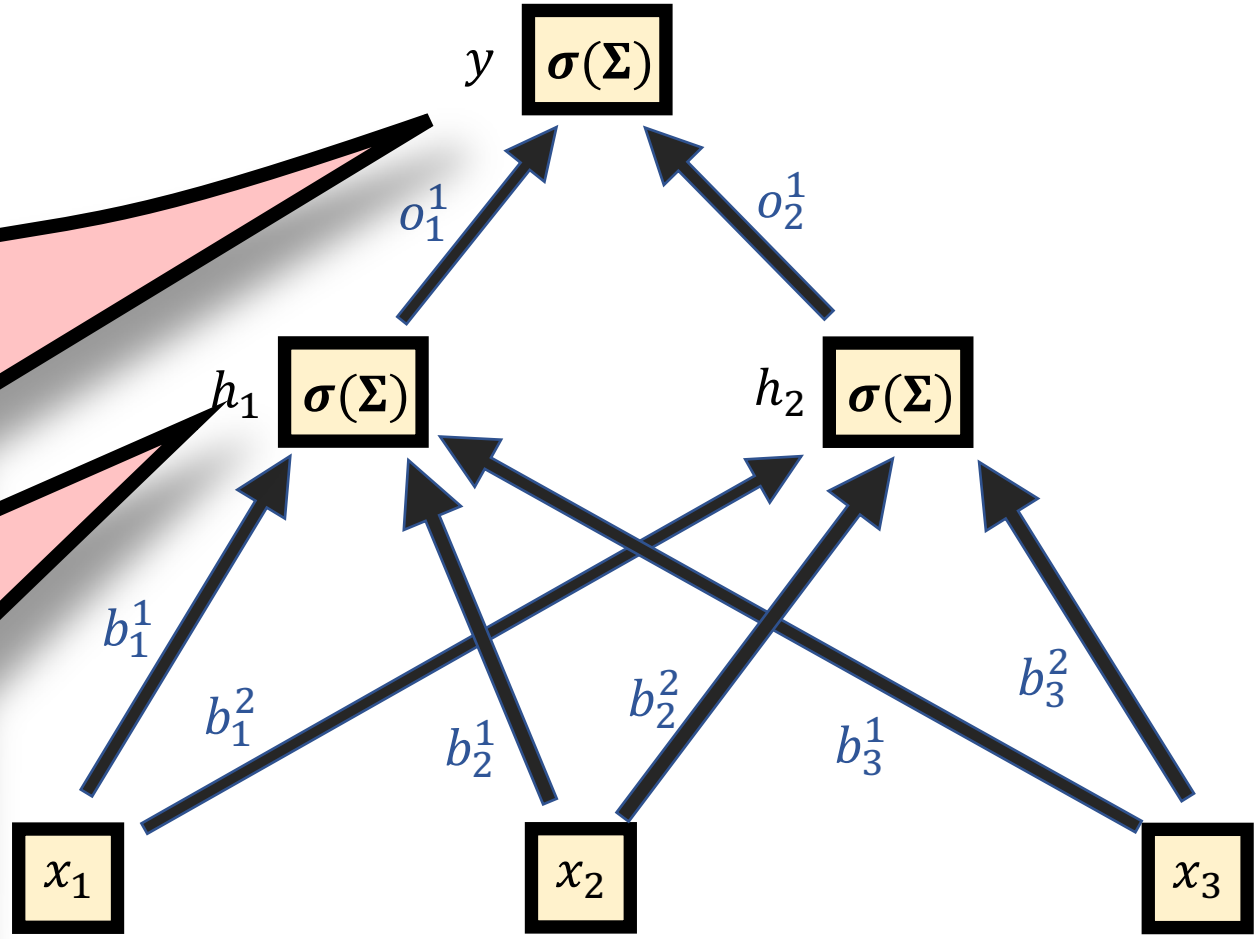
Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Ev
- Ev
- Pa

Every , except for the input layer's, is called an **activation function.**

The sigmoid function $\boldsymbol{\sigma}$ is a common choice and is equivalent to performing **logistic regression** on its given inputs.



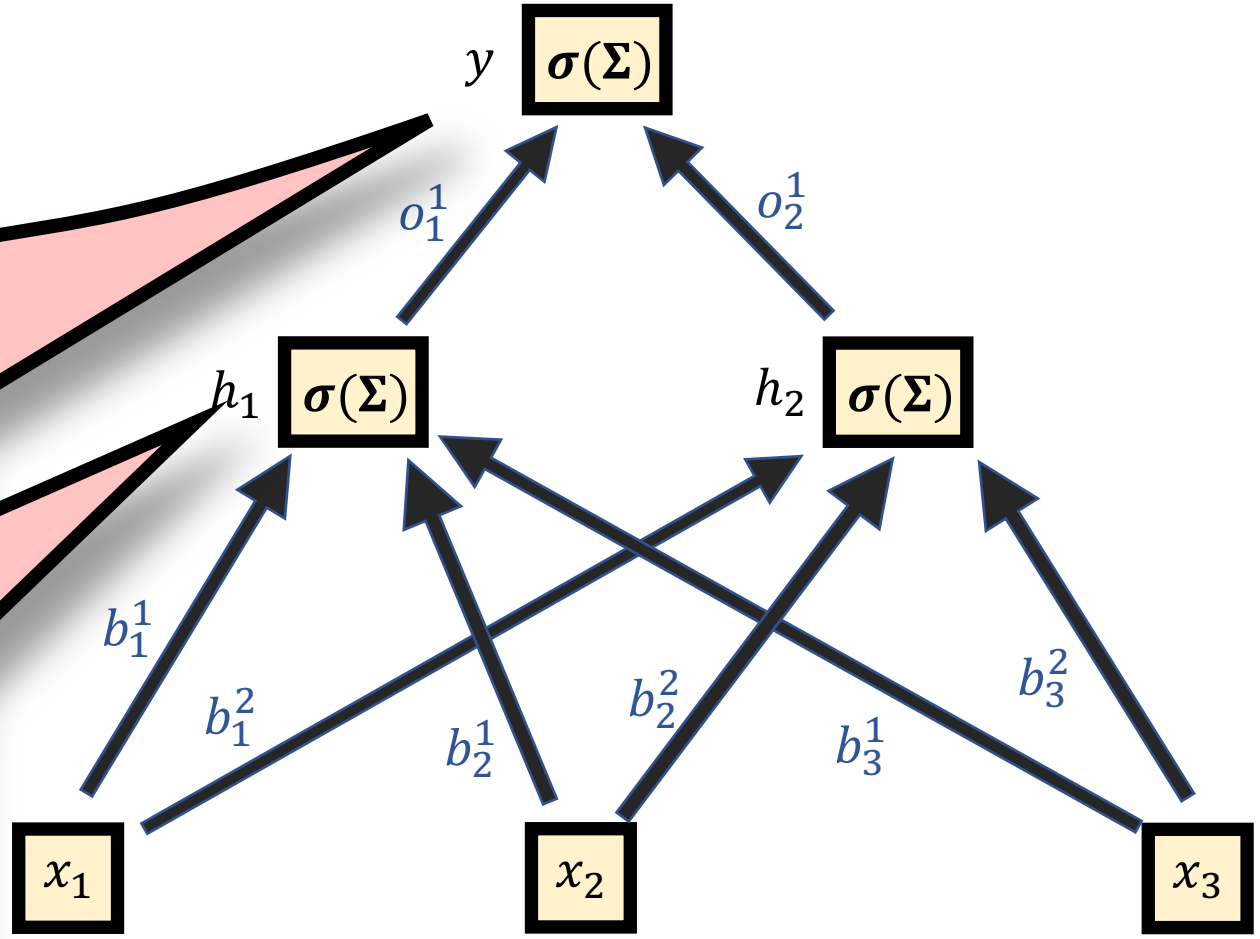
Graphically
(NN format)

Feed-Forward Neural Network

General Notes:

- It's a fully connected network
- Ev
- Ev
- Pa

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

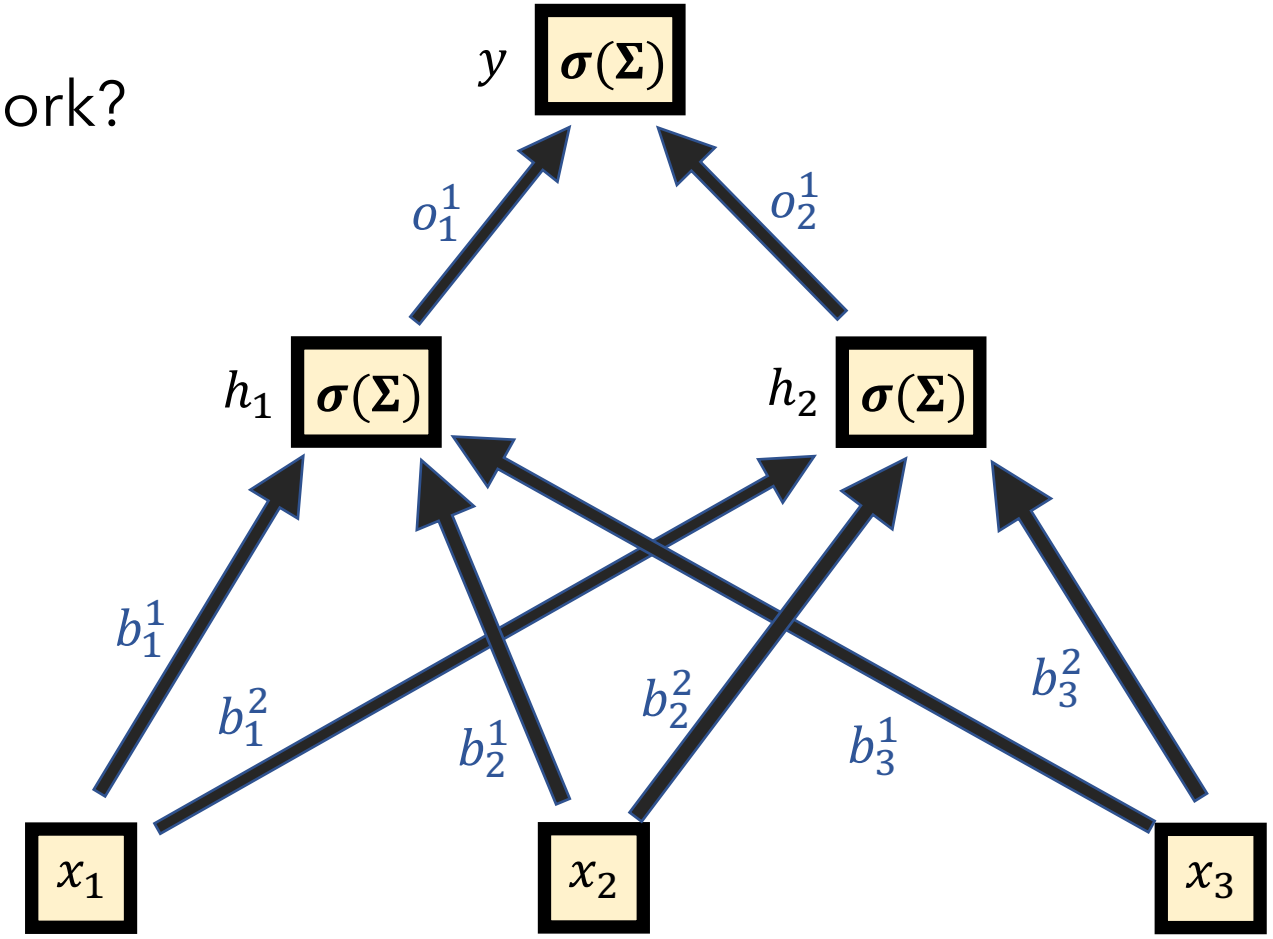


Graphically
(NN format)

Feed-Forward Neural Network

Training:

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



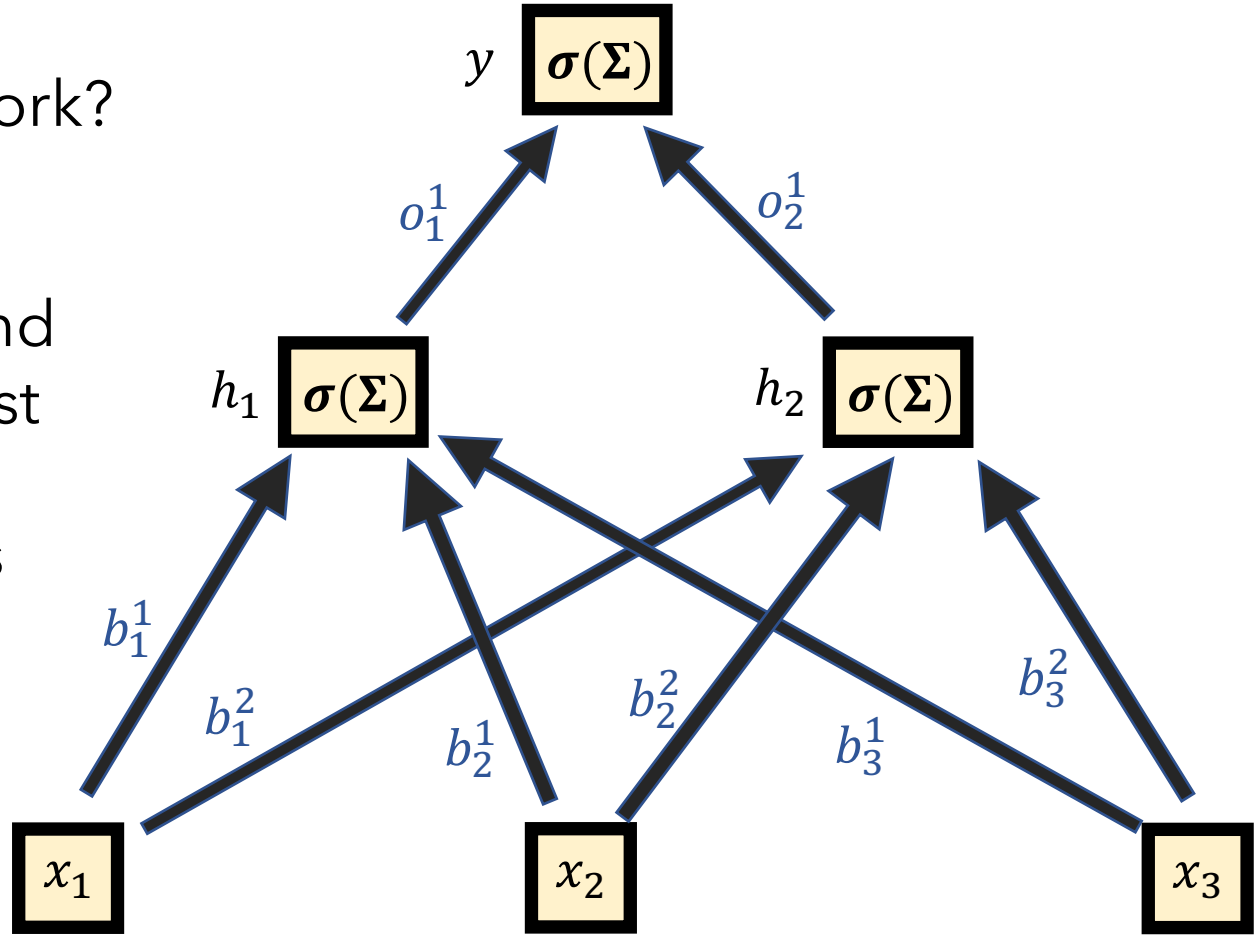
Graphically
(NN format)

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)

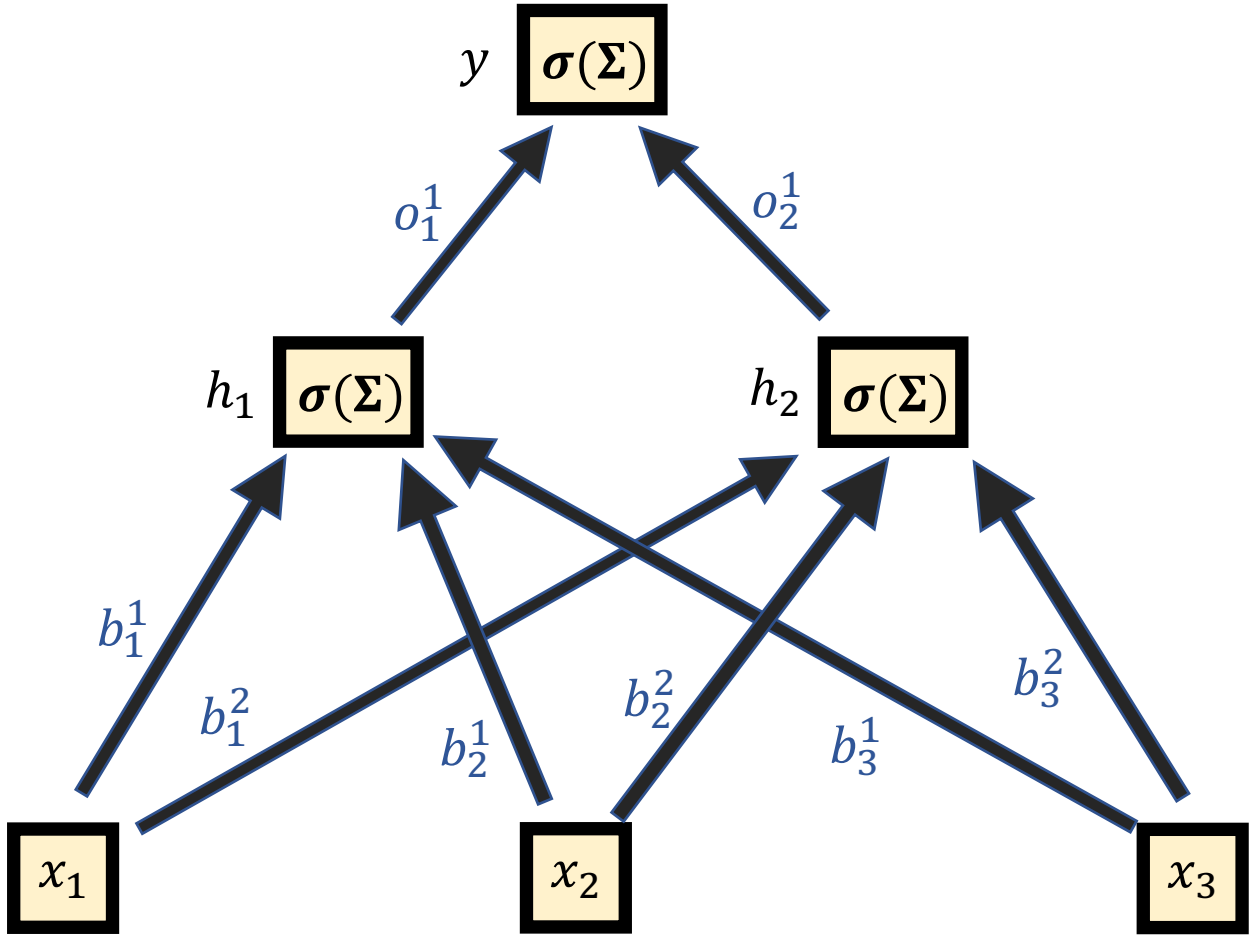
Feed-Forward Neural Network

Training:

Cost function

$$J(\boldsymbol{\theta}) = \underbrace{-[y \log \hat{y} + (1-y)\log(1-\hat{y})]}_{\text{"Cross-Entropy" aka "Log loss"}}$$

Update the $\boldsymbol{\theta}$ via
gradient descent



Graphically
(NN format)

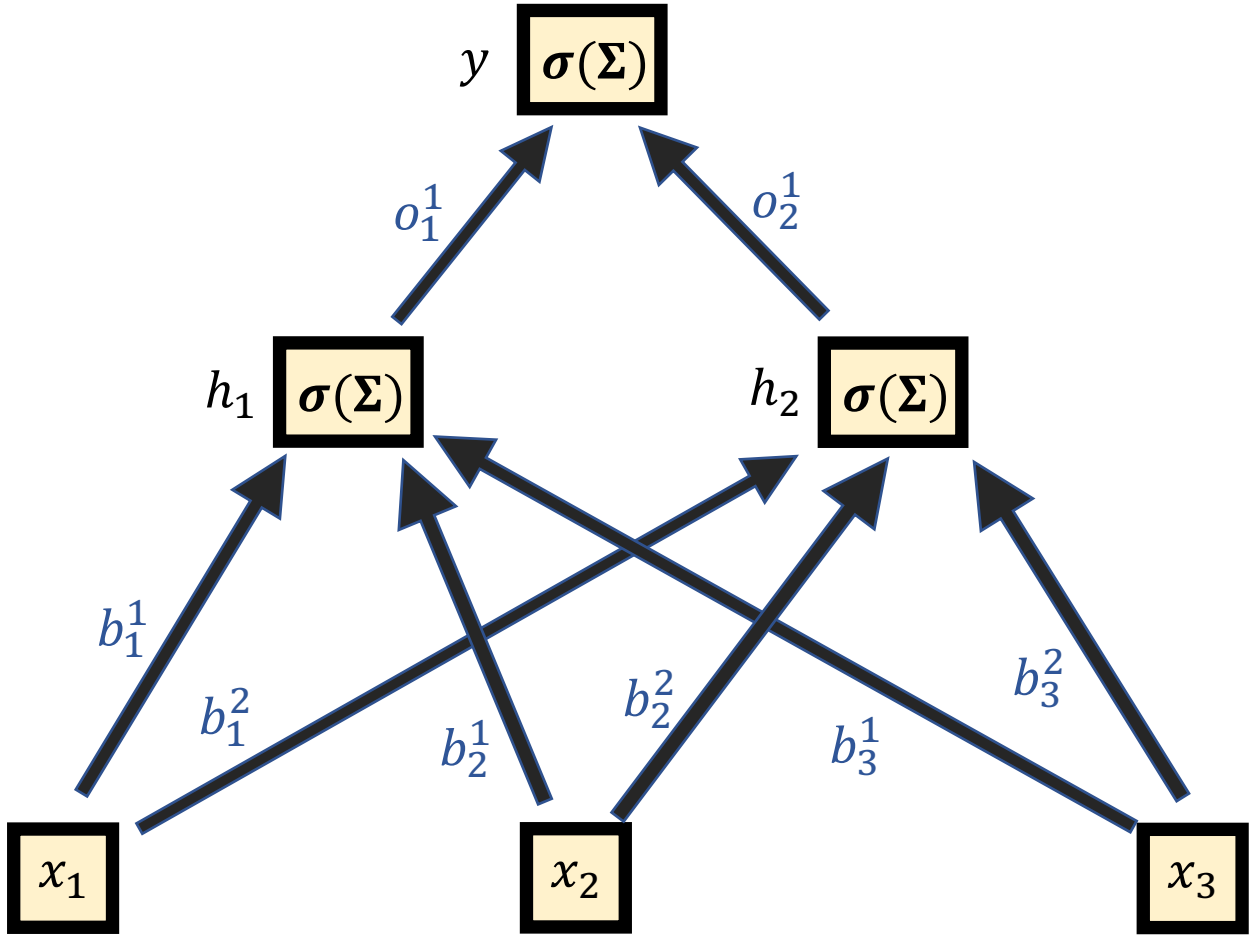
Feed-Forward Neural Network

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

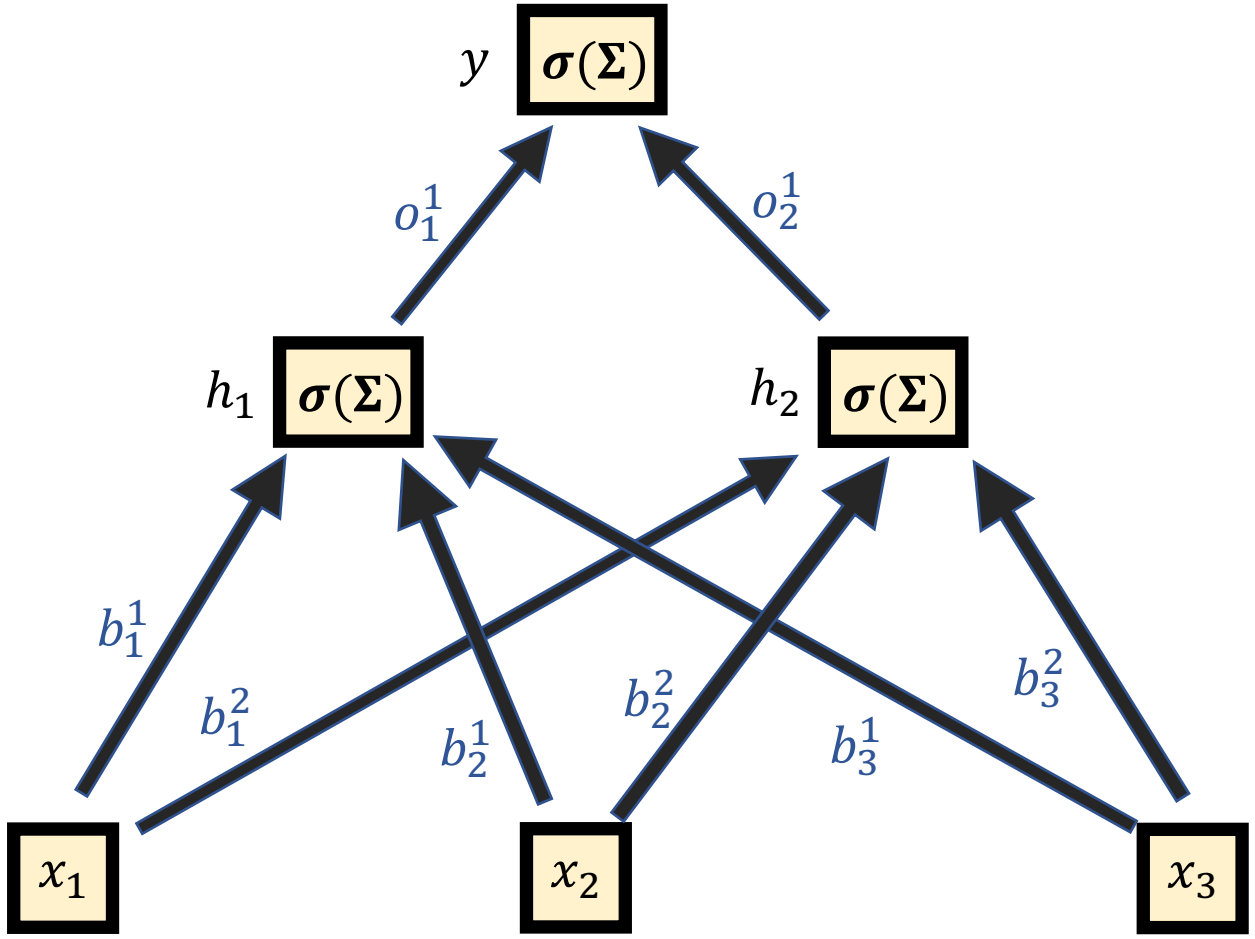
1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network
3. Calculate the cost/loss
4. Calculate gradients via backpropagation
5. Update the weights (aka $\boldsymbol{\theta}$) via gradient descent



Graphically
(NN format)

Training:

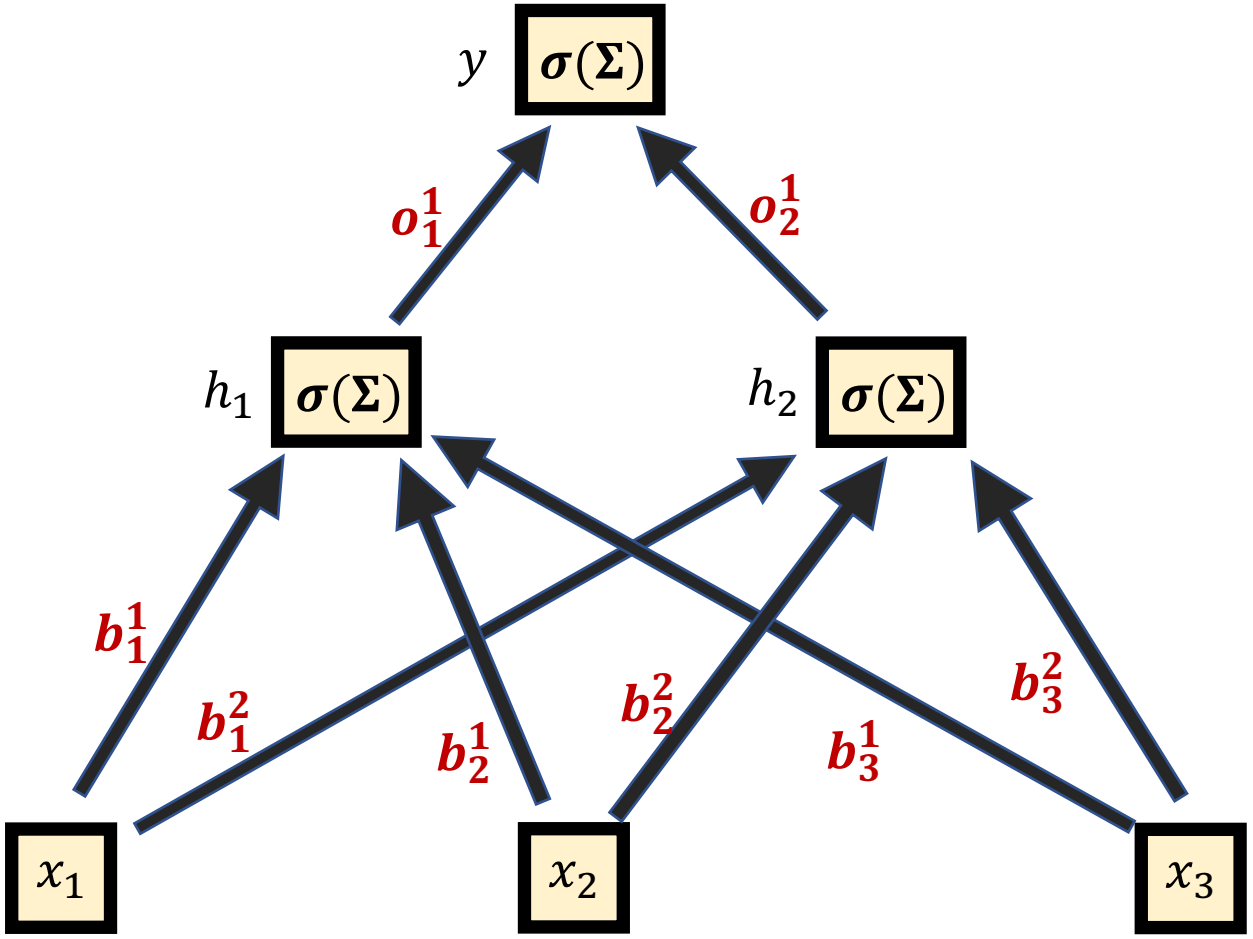
Initialize $\boldsymbol{\theta}$ with random values



Graphically
(NN format)

Training:

Initialize $\boldsymbol{\theta}$ with random values



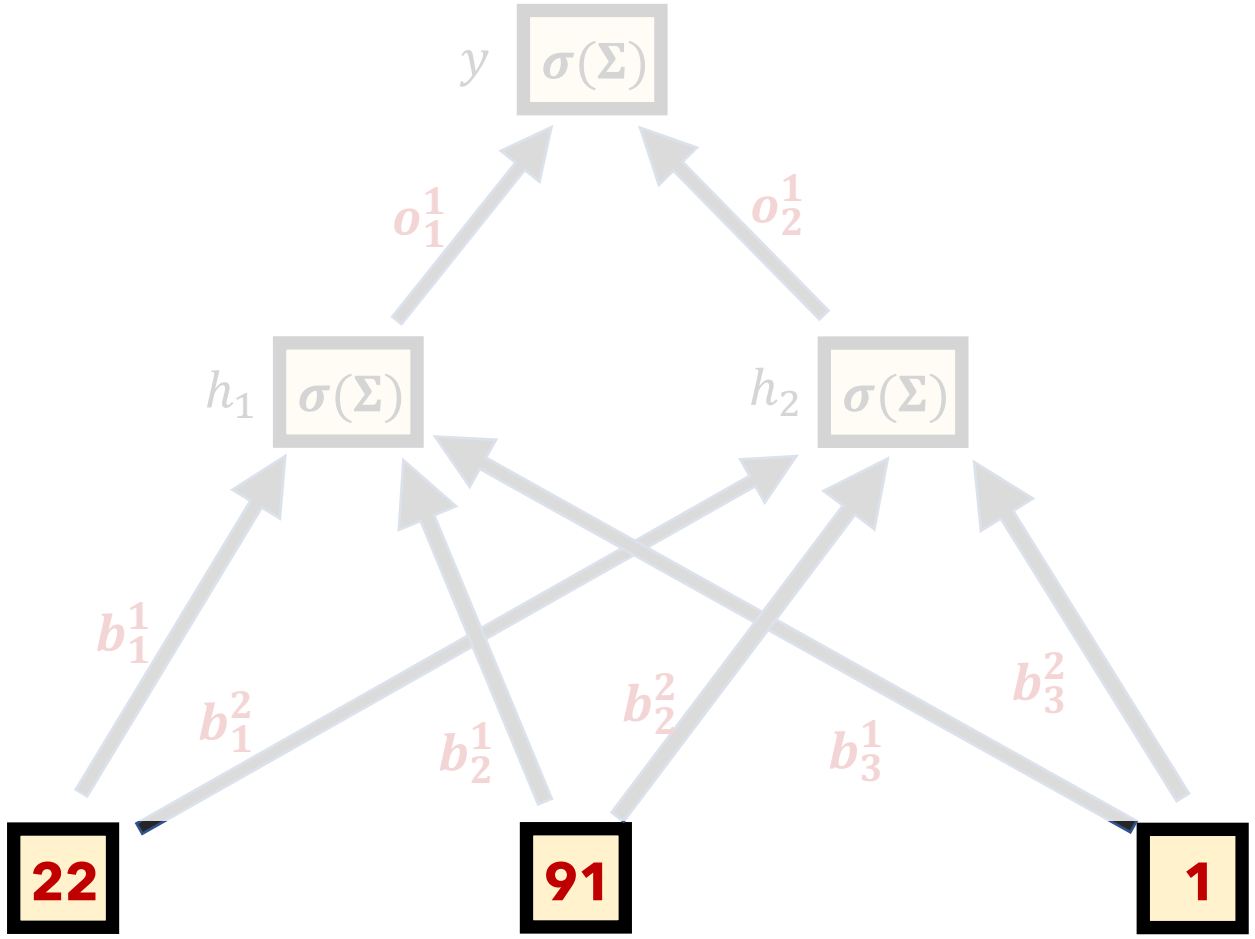
Graphically
(NN format)

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network



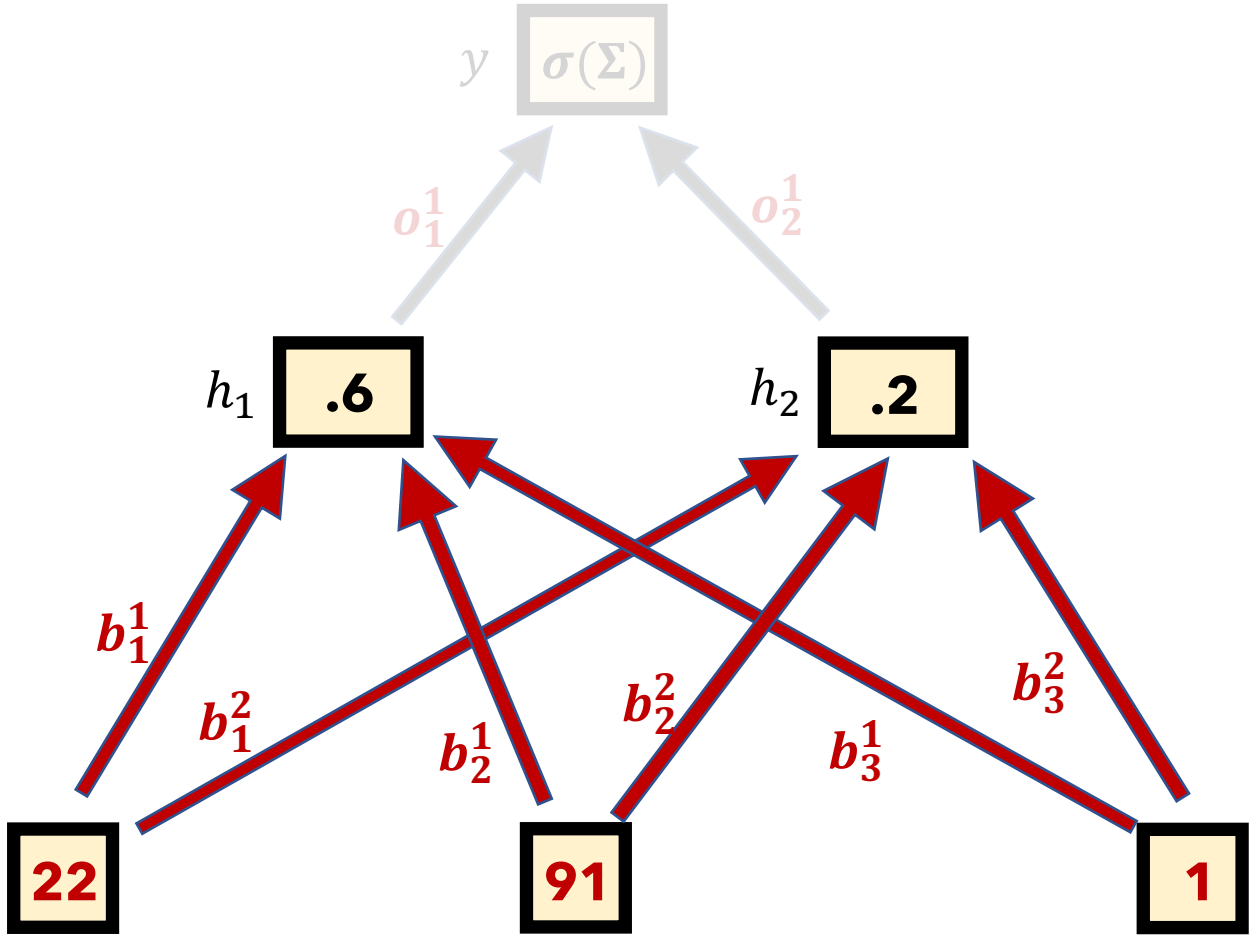
Graphically
(NN format)

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network



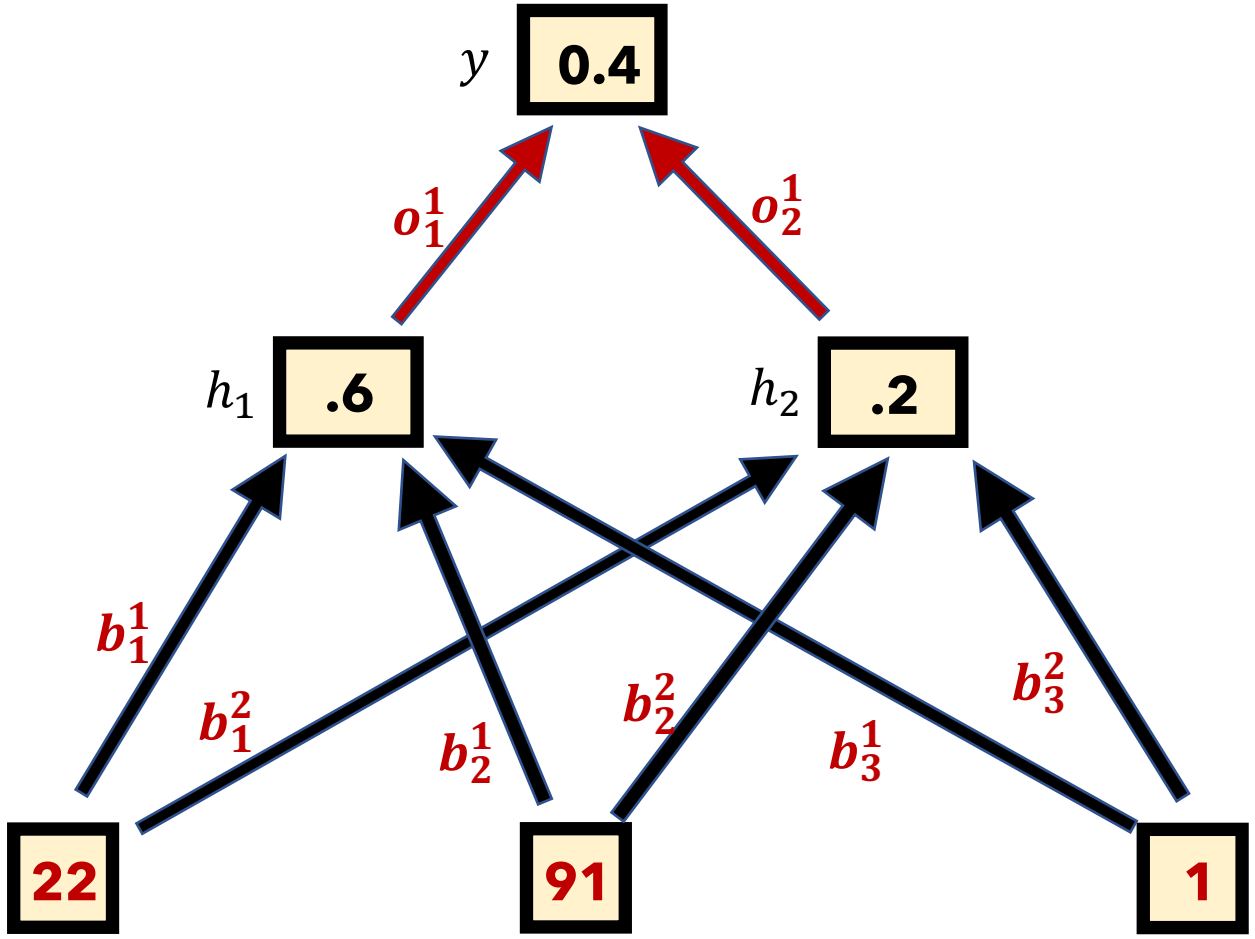
Graphically
(NN format)

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network



Graphically
(NN format)

Feed-Forward Neural Network

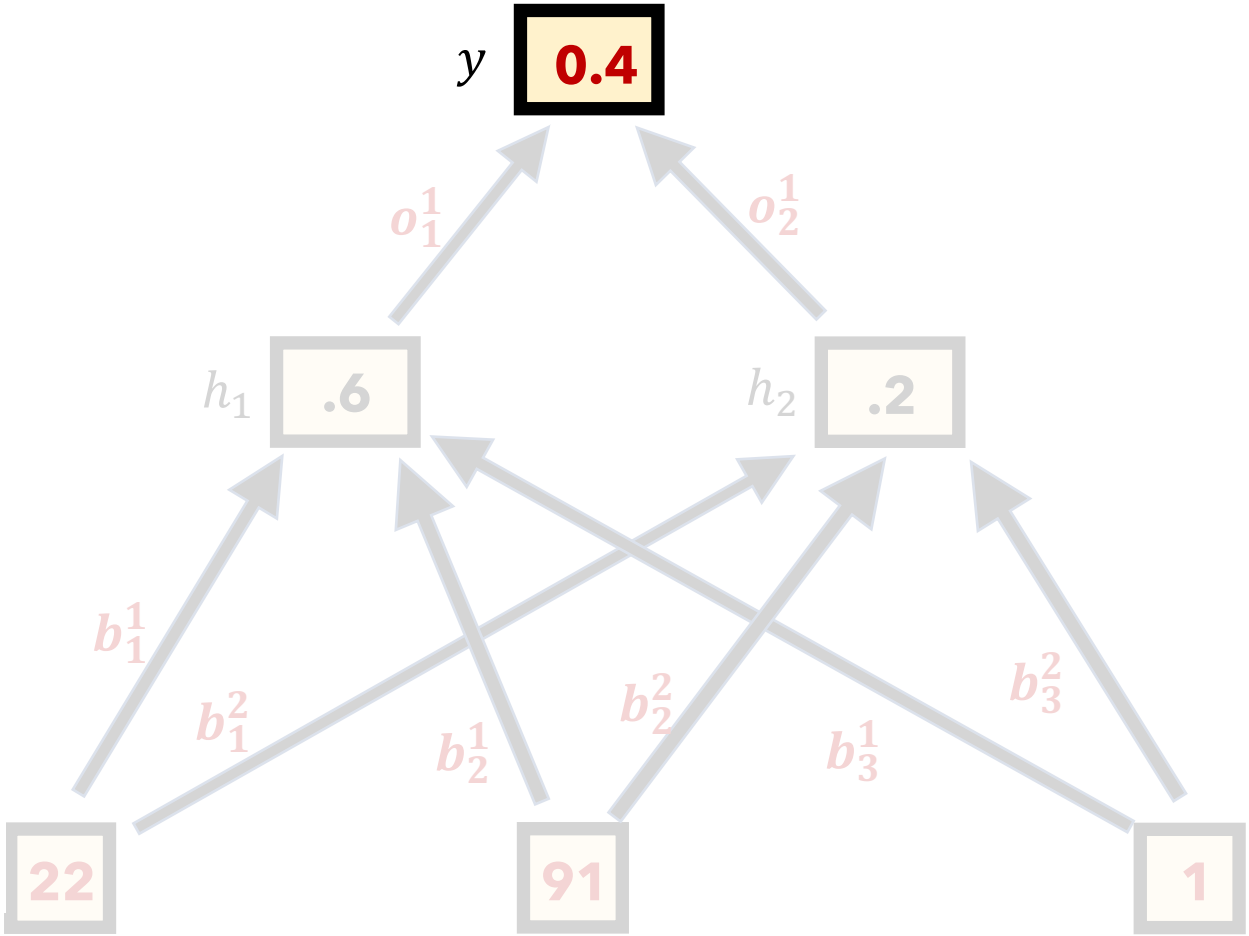
Training:

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

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network
3. Calculate the cost/loss



Graphically
(NN format)

Feed-Forward Neural Network

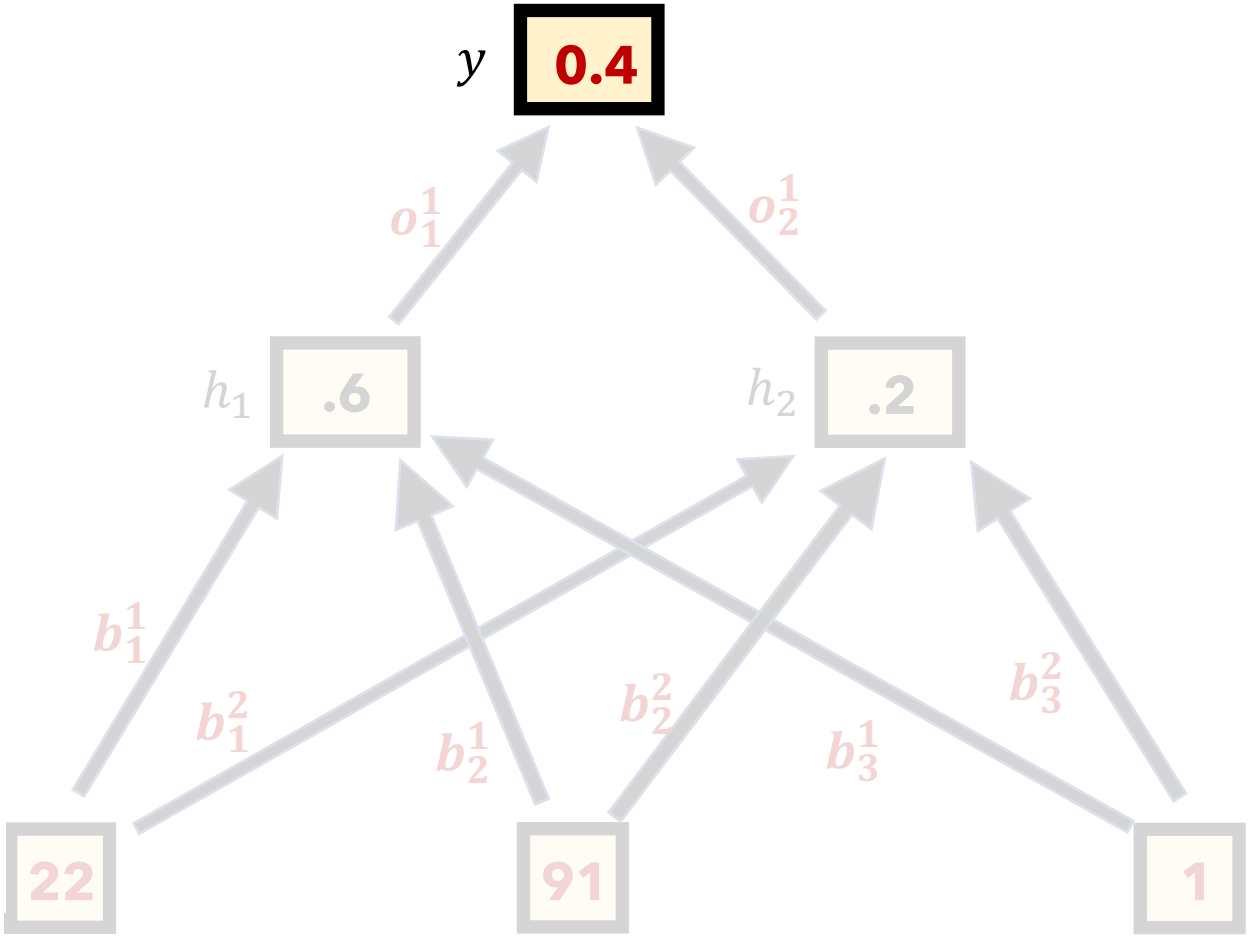
Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network
3. Calculate the cost/loss

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



Graphically
(NN format)

Feed-Forward Neural Network

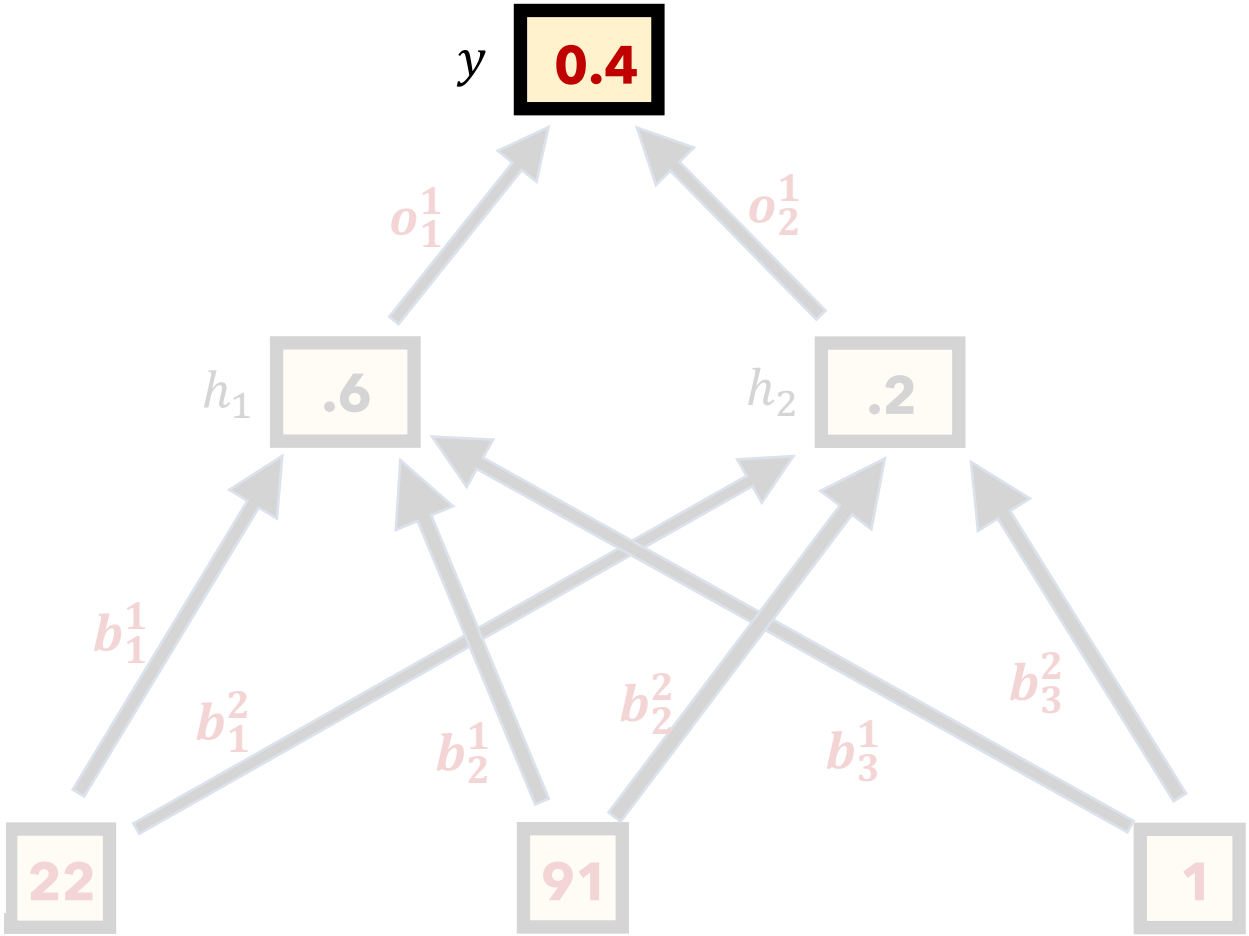
Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network
3. Calculate the cost/loss

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



Graphically
(NN format)

Feed-Forward Neural Network

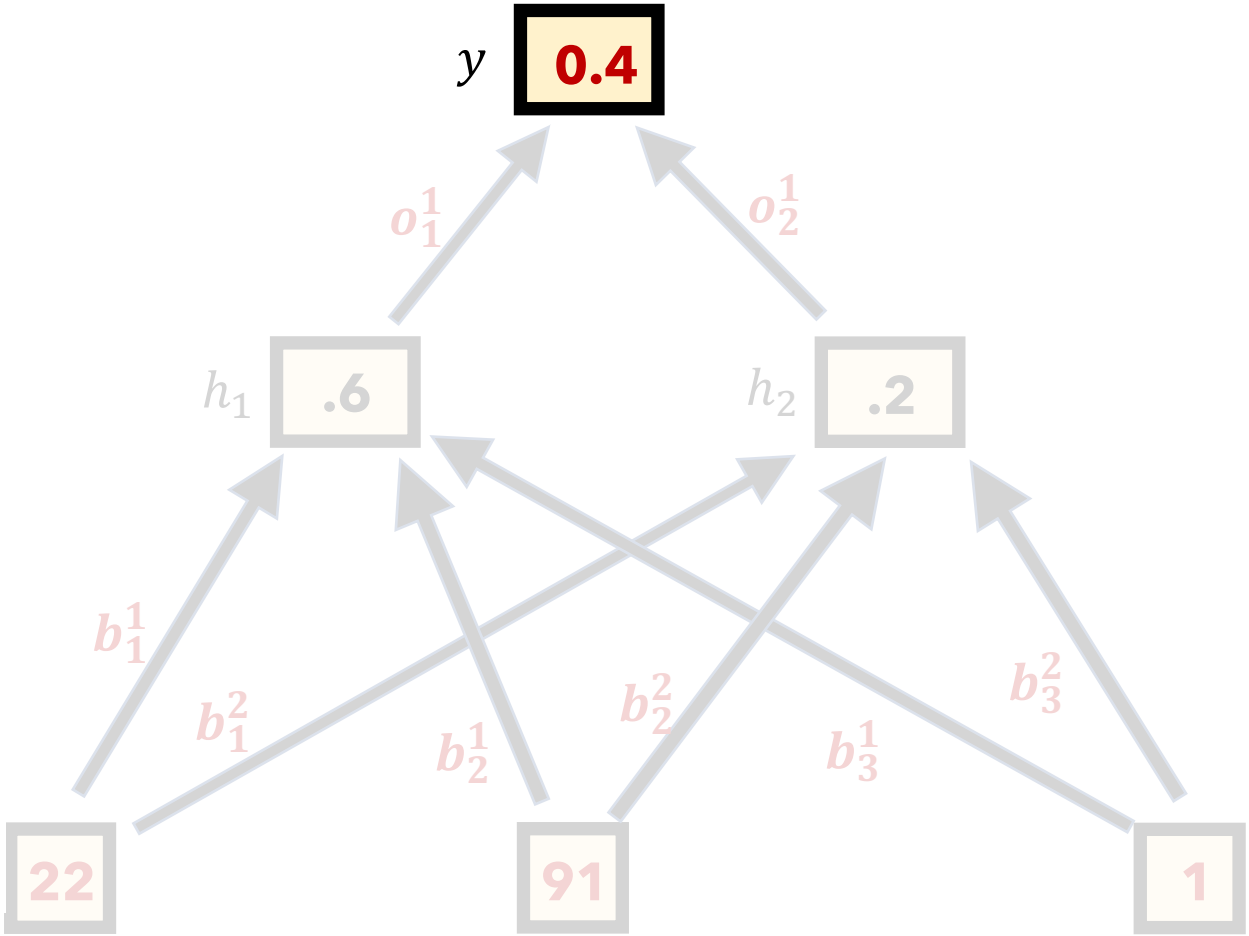
Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network
3. Calculate the cost/loss

$$J(\boldsymbol{\theta}) = -\log(0.6)$$



Graphically
(NN format)

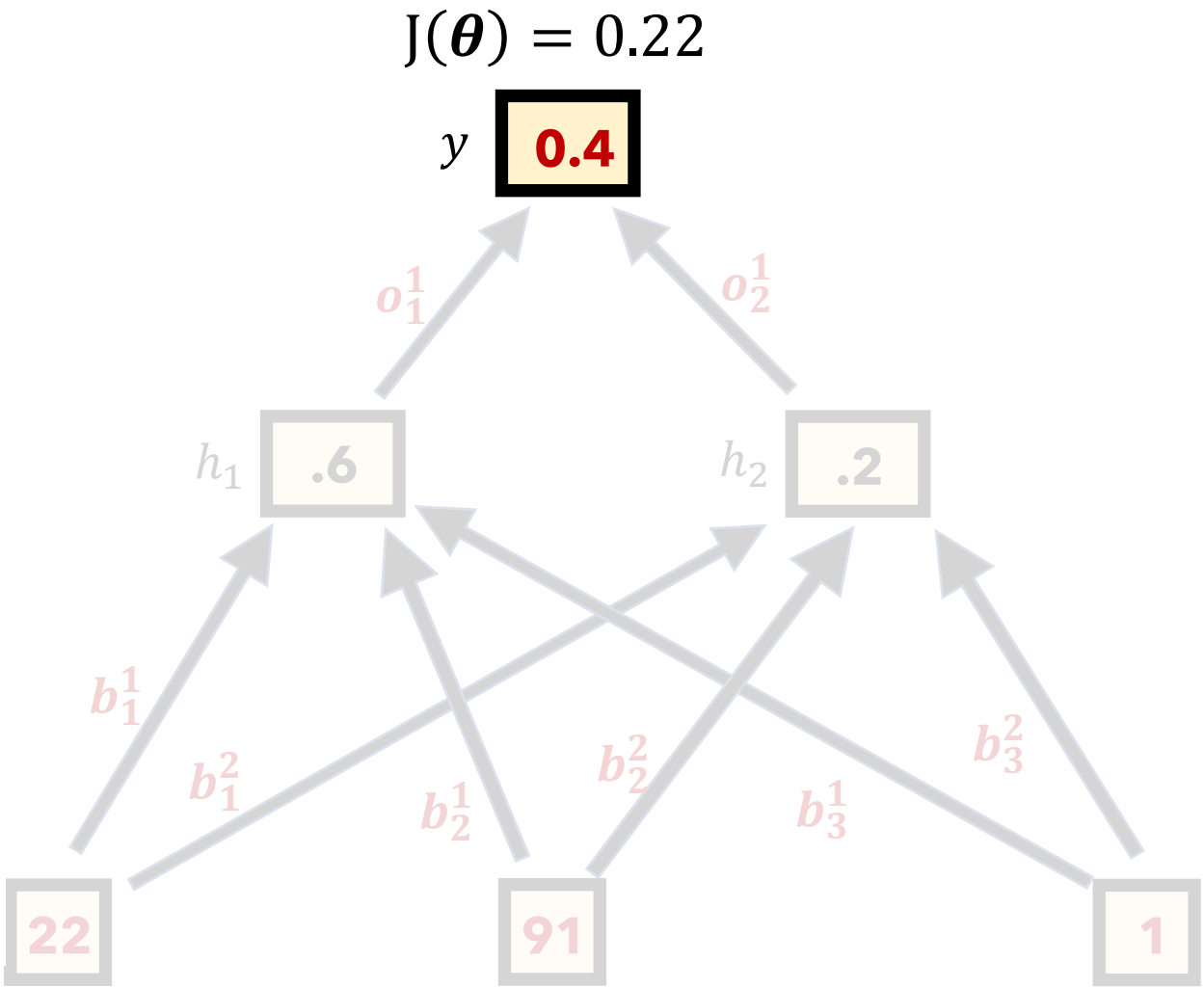
Feed-Forward Neural Network

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ to the network
2. Propagate the values through the network
3. Calculate the cost/loss



Graphically
(NN format)

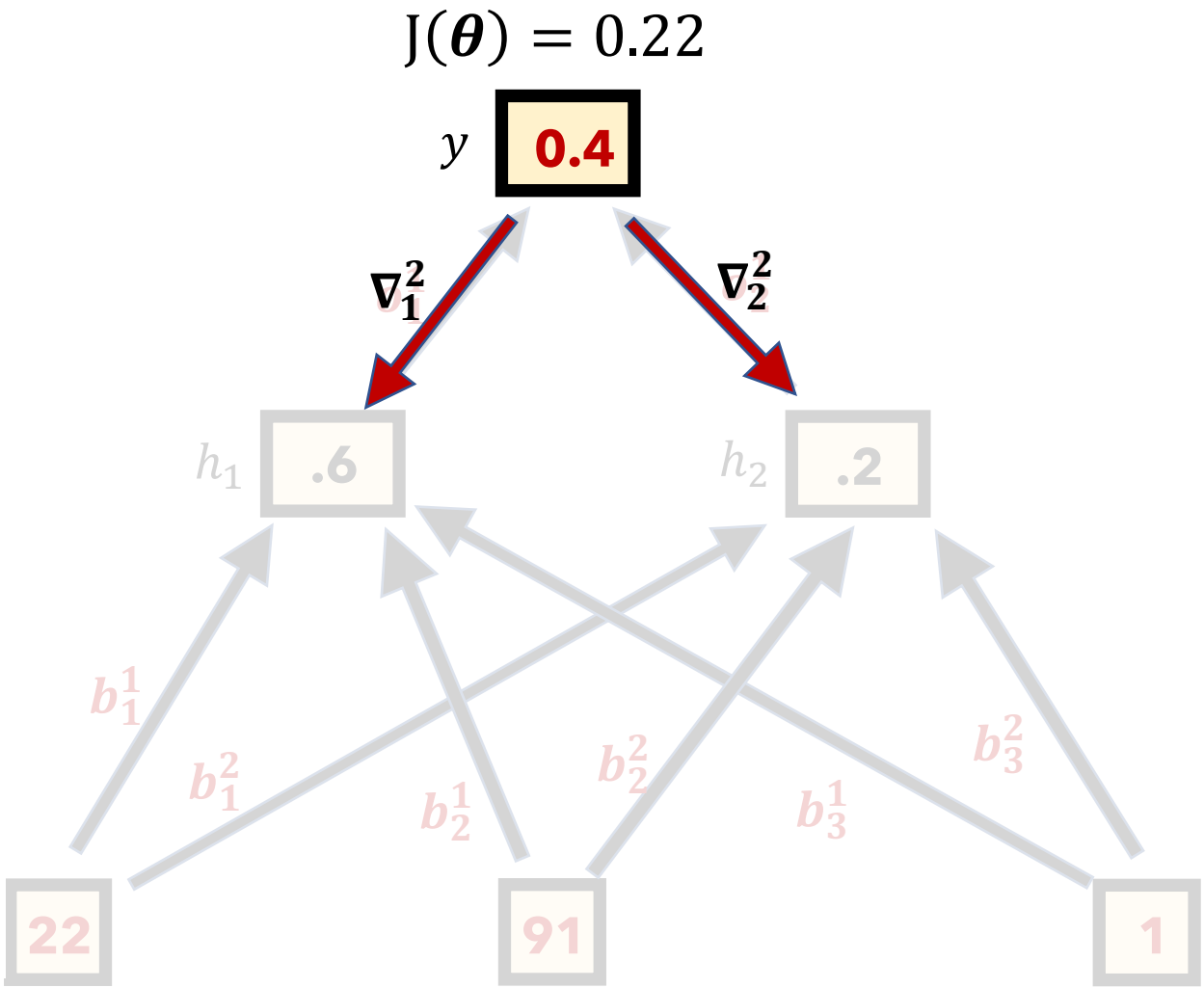
Feed-Forward Neural Network

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

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



Graphically
(NN format)

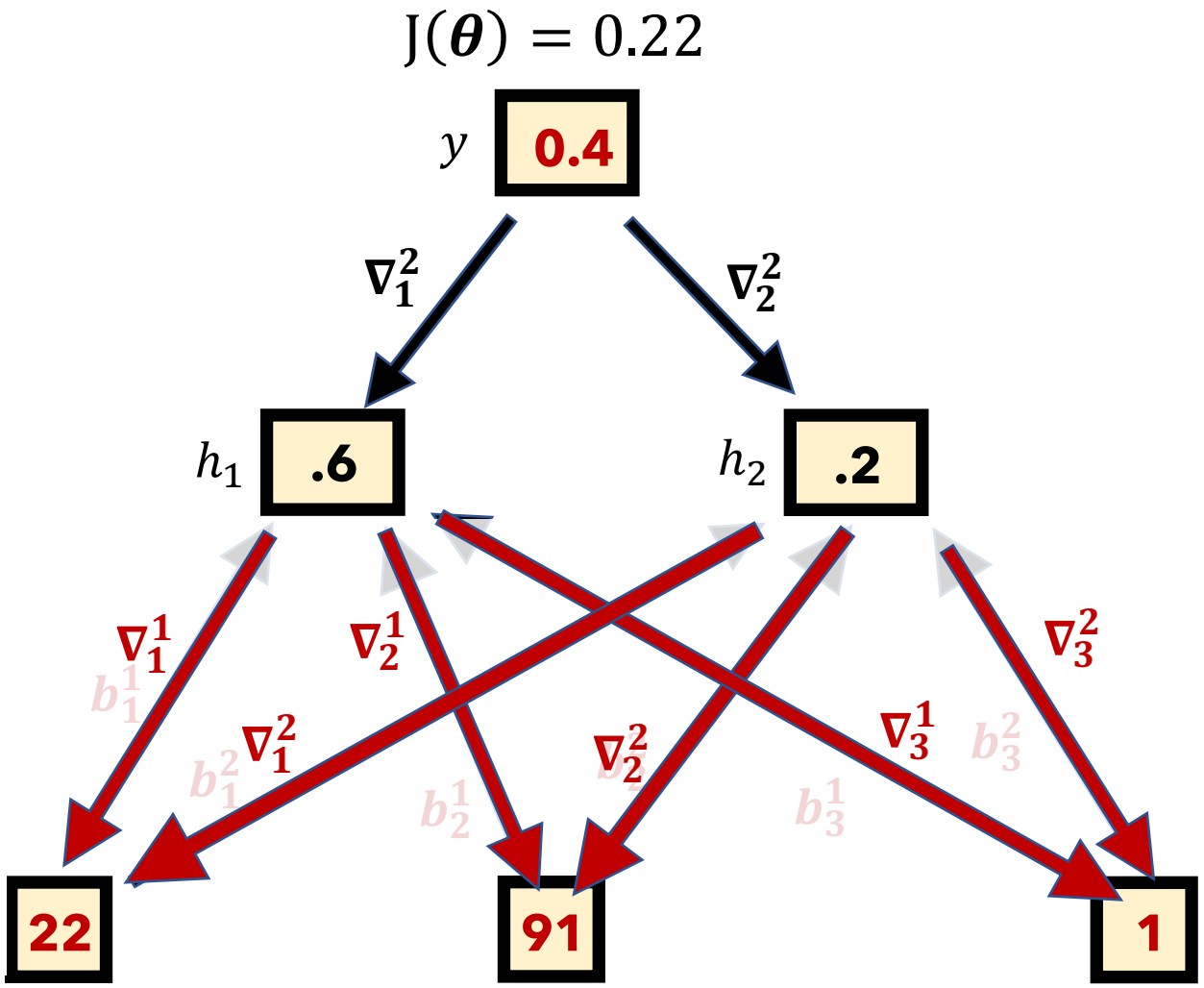
Feed-Forward Neural Network

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

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



Graphically
(NN format)

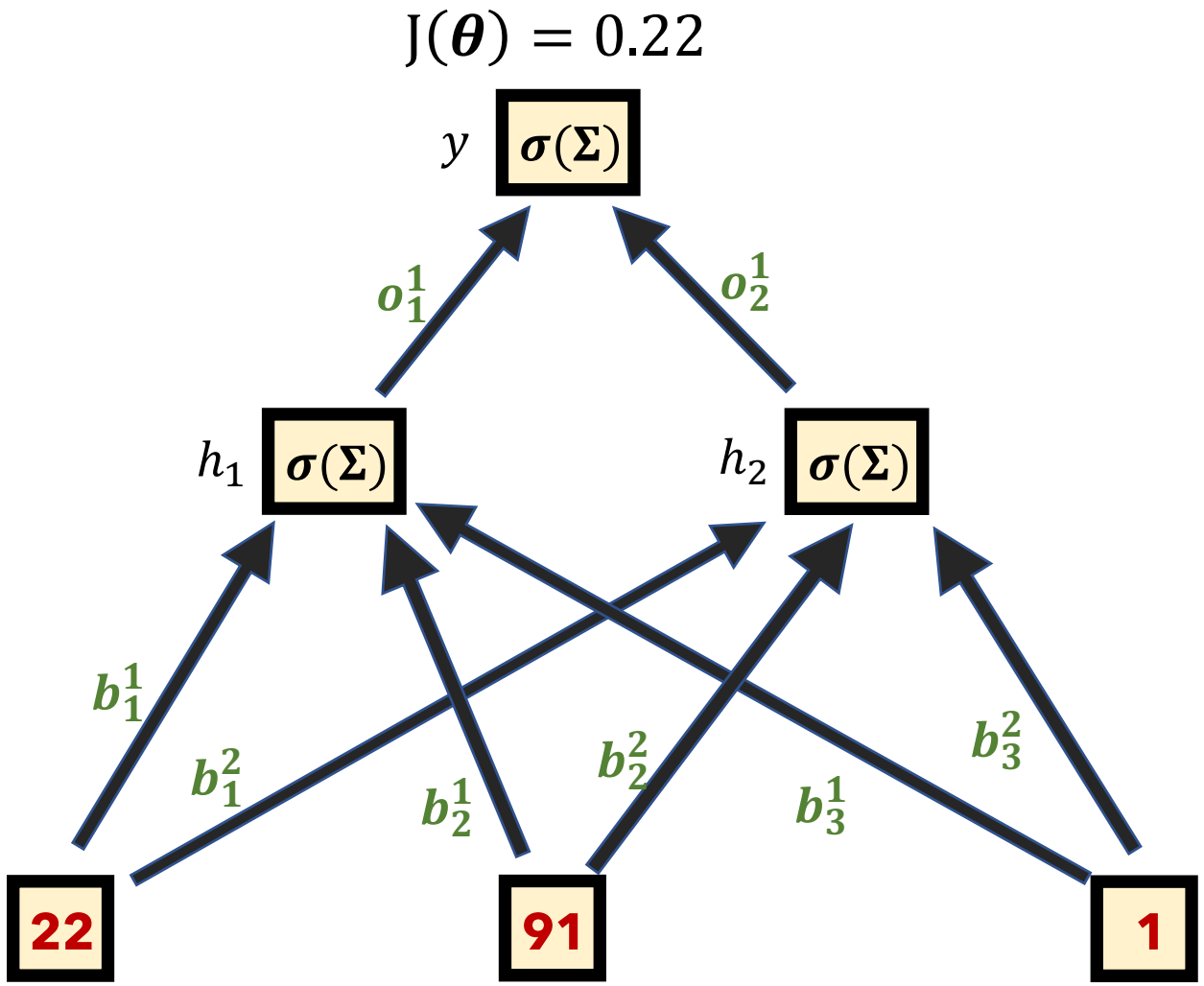
Feed-Forward Neural Network

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ 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 $\boldsymbol{\theta}$) via gradient descent



Graphically
(NN format)

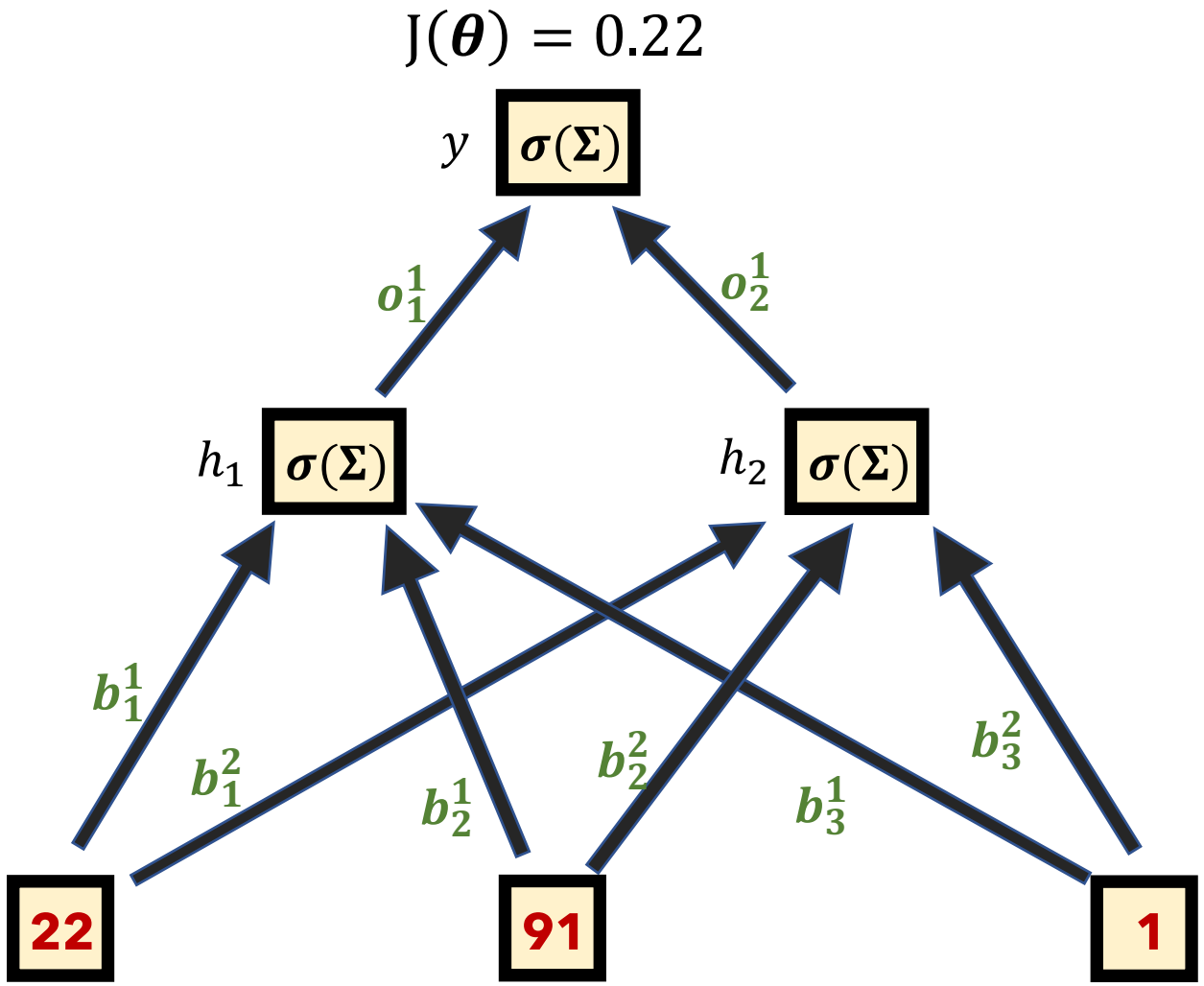
Feed-Forward Neural Network

Training:

Initialize $\boldsymbol{\theta}$ with random values

Repeat until convergence:

1. Provide input $\boldsymbol{x_i}$ 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 $\boldsymbol{\theta}$) via gradient descent



Graphically
(NN format)

Feed-Forward Neural Network (and all other neural nets)

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
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 $\boldsymbol{\theta}$

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 $\boldsymbol{\theta}$

e.g.

Q1 When training our model, how do we measure its m predictions $\hat{\boldsymbol{y}}$?

A1 Cost function $$\mathrm{J}(\boldsymbol{\theta}) = \underbrace{\frac{1}{2}\sum_{i=1}^{m}(\hat{y} - y)^2}_{\text{"Least Squares"}}$$

MLE vs MAP

We were performing the **maximum likelihood estimate**

Def:

maximum likelihood estimate (MLE) asserts that we should choose $\boldsymbol{\theta}$ so as to maximize the likelihood 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 $P(D|\theta)$

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

MLE vs MAP

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

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

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

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

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!

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

CLOSING NOTE

The model ≠ the data ≠ the data's representation

Many models can work well with data from different domains. The model restrictions concern your assumptions about the data, and what you're trying to do with the data. The representation of the data can permit/limit the model's ability to fit/learn the data.

Now let's turn to the world of **language data** and useful **models** for it