Machine Learning Basics

Announcements:

1. Warmup Quiz and P(-1) and P(0) are out

2. TA office hours are posted on Canvas (location: Rhodes 503)

3. CIS Partner Finding Social (Sep 1st, 4-6pm, Upson 142)

Objective:

Get familiar with some of the common definitions, and get a big picture of supervised / unsupervised learning

Outline for Today:

1. Supervised Learning (Classification / Regression) and Unsupervised learning

2. Generalization

3. Training / validation / testing

Classification

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

Classification

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

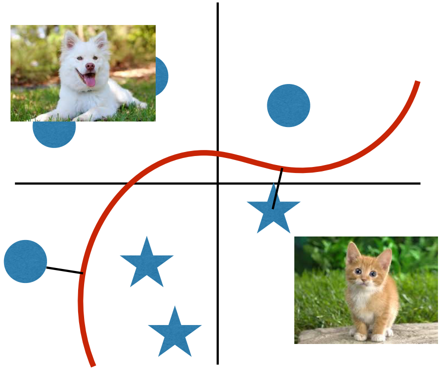
ML model

Classification

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

ML model



Mathematical formulation of the pipeline

Dataset:

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathscr{C}(\text{ e.g.,} \mathscr{C} = \{-1,1\}), (x_i, y_i) \sim \mathscr{P}$$

Mathematical formulation of the pipeline

Dataset:

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathscr{C}(\text{ e.g.,}\mathscr{C} = \{-1,1\}), (x_i, y_i) \sim \mathscr{P}$$

Hypothesis:

$$h : \mathbb{R}^d \mapsto \mathscr{C}$$

Mathematical formulation of the pipeline

Dataset:

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathscr{C}(\text{ e.g.,} \mathscr{C} = \{-1,1\}), (x_i, y_i) \sim \mathscr{P}$$

Hypothesis:

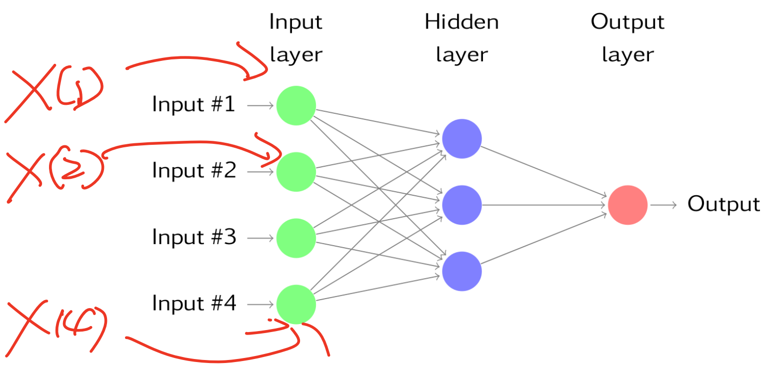
$$h : \mathbb{R}^d \mapsto \mathscr{C}$$

i.e., a neural network-based classifier that maps image to label of cat or dog

Mathematical formulation of the pipeline

Dataset:

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathscr{C}(\text{ e.g.,} \mathscr{C} = \{-1,1\}), (x_i, y_i) \sim \mathscr{P}$$



Hypothesis:

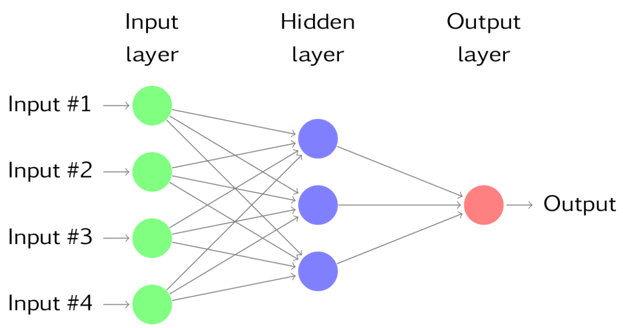
$$h : \mathbb{R}^d \mapsto \mathscr{C}$$

i.e., a neural network-based classifier that maps image to label of cat or dog

Mathematical formulation of the pipeline

Dataset:

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathcal{C}(\text{ e.g.,} \mathcal{C} = \{-1,1\}), (x_i, y_i) \sim \mathcal{P}$$



Hypothesis:

$$h : \mathbb{R}^d \mapsto \mathcal{C}$$

i.e., a neural network-based classifier that maps image to label of cat or dog

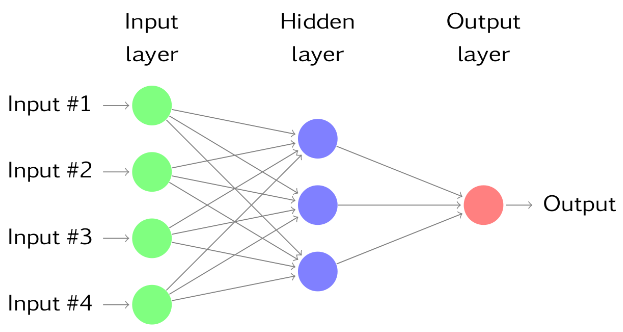
Hypothesis class

$$\mathcal{H} = \{h\}$$

Mathematical formulation of the pipeline

Dataset:

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}, x_i \in \mathbb{R}^d, y_i \in \mathscr{C}(\text{ e.g.,}\mathscr{C} = \{-1,1\}), (x_i, y_i) \sim \mathscr{P}$$



Hypothesis:

$$h : \mathbb{R}^d \mapsto \mathscr{C}$$

i.e., a neural network-based classifier that maps image to label of cat or dog

Hypothesis class

$$\mathscr{H} = \{h\}$$

i.e., a large family of NNs with different parameters

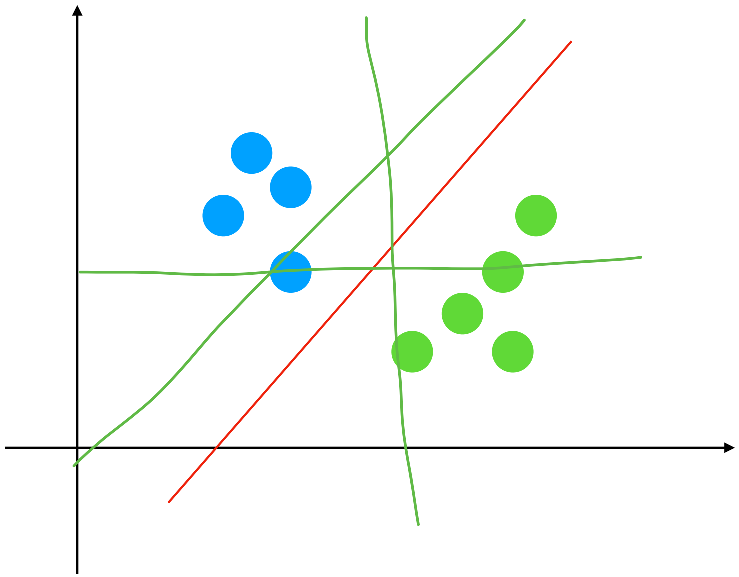
Examples of hypothesis

Inductive bias (i.e., assumptions) encoded in the hypothesis class

Examples of hypothesis

Inductive bias (i.e., assumptions) encoded in the hypothesis class

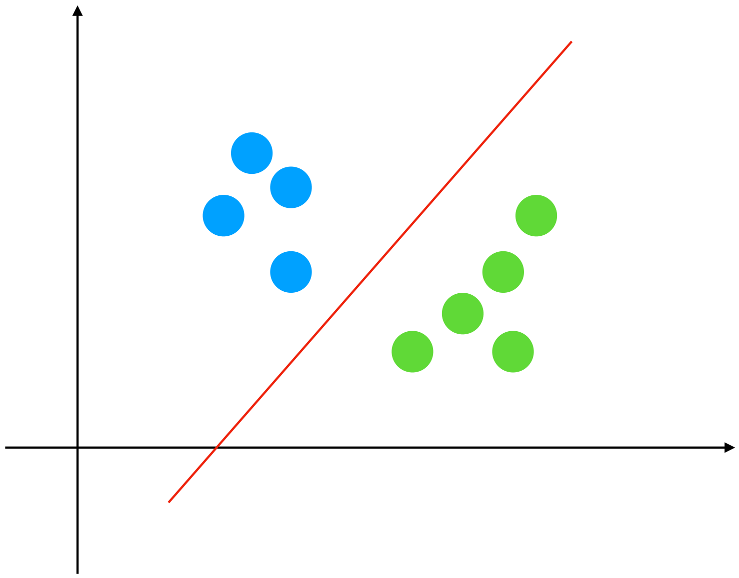
Ex: h is a linear function $h(x) = \text{sign}(w^\top x)$;
$\mathscr{H}$ contains all possible linear functions



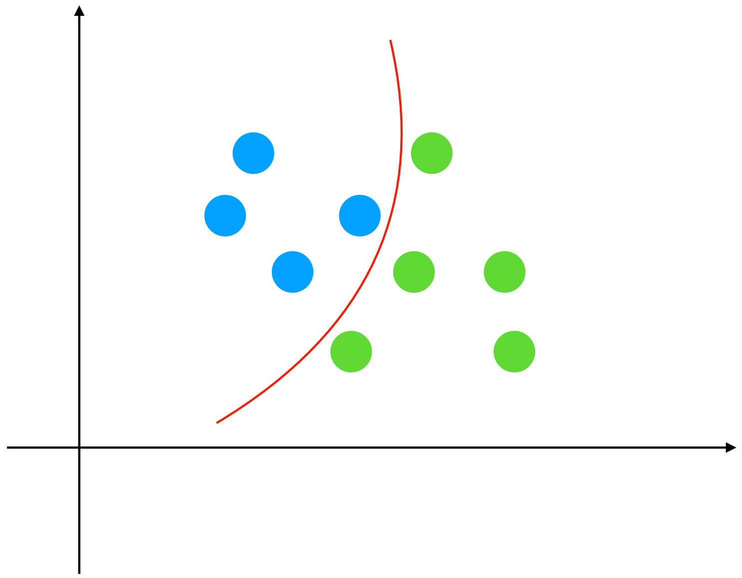
Examples of hypothesis

Inductive bias (i.e., assumptions) encoded in the hypothesis class

Ex: h is a linear function $h(x) = \mathrm{sign}(w^\top x)$;
$\mathscr{H}$ contains all possible linear functions



Ex: h is nonlinear $h(x) = \mathrm{sign}(w^\top (\mathrm{relu}(Ax)))$;
$\mathscr{H}$ contains all possible one-layer NN



Do we need to make assumptions on the data?

No free lunch theorem says that we must make such assumptions

Do we need to make assumptions on the data?

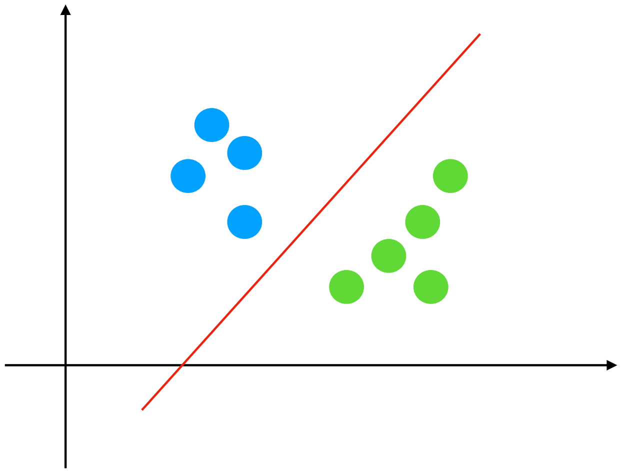
No free lunch theorem says that we must make such assumptions

Informal theorem: for any machine learning algorithm $\mathscr{A}$, there must exist a task $\mathscr{P}$ on which it will fail

Do we need to make assumptions on the data?

No free lunch theorem says that we must make such assumptions

Informal theorem: for any machine learning algorithm $\mathscr{A}$, there must exist a task $\mathscr{P}$ on which it will fail



We use prior knowledge (i.e., we believe linear function is enough) to design an ML algorithm here

The Loss Function

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

The Loss Function

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

Let's define loss function $\ell : \mathscr{H} \times \mathbb{R}^d \times \mathscr{C} \mapsto \mathbb{R}$

The Loss Function

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

Let's define loss function $\ell : \mathscr{H} \times \mathbb{R}^d \times \mathscr{C} \mapsto \mathbb{R}$

Intuitively, $\ell(h, x, y)$ tells us how bad (e.g., classification mistake) the hypothesis h is.

The Loss Function

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

Let's define loss function $\ell : \mathscr{H} \times \mathbb{R}^d \times \mathscr{C} \mapsto \mathbb{R}$

Intuitively, $\ell(h, x, y)$ tells us how bad (e.g., classification mistake) the hypothesis h is.

Examples:

Zero-one loss:

$$\ell(h, x, y) = \begin{cases} 0 & h(x) = y \\ 1 & h(x) \neq y \end{cases}$$

The Loss Function

Q: how to select the best hypothesis $\hat{h}$ from $\mathcal{H}$?

Let's define loss function $\ell : \mathcal{H} \times \mathbb{R}^d \times \mathcal{C} \mapsto \mathbb{R}$

Intuitively, $\ell(h, x, y)$ tells us how bad (e.g., classification mistake) the hypothesis h is.

Examples:

Zero-one loss:

$$\ell(h, x, y) = \begin{cases} 0 & h(x) = y \\ 1 & h(x) \neq y \end{cases}$$

Squared loss:

$$\ell(h, x, y) = (h(x) - y)^2$$

Learning/Training

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

With loss ℓ being defined, we can perform training/learning:

$$\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \ell(h, x_i, y_i)$$

Learning/Training

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

With loss ℓ being defined, we can perform training/learning:

$$\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \ell(h, x_i, y_i)$$

The hypothesis that has smallest training error

Learning/Training

Q: how to select the best hypothesis $\hat{h}$ from $\mathscr{H}$?

With loss ℓ being defined, we can perform training/learning:

$$\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{i=1}^{n} \ell(h, x_i, y_i)$$

The hypothesis that has smallest training error

e.g., total number of mistakes h makes on n training samples (training error)

Putting things together: Binary classification

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

x_1, y_1 x_2, y_2 x_3, y_3

Putting things together: Binary classification

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

x_1, y_1 x_2, y_2 x_3, y_3

ML model, e.g., neural network w/ 0-1 loss

$$\hat{h}_{nn} = \arg \min_{h_{nn} \in \mathscr{H}} \sum_{i=1}^{n} \ell_{0-1} \left(h_{nn}, x_i, y_i \right)$$

Putting things together: Binary classification

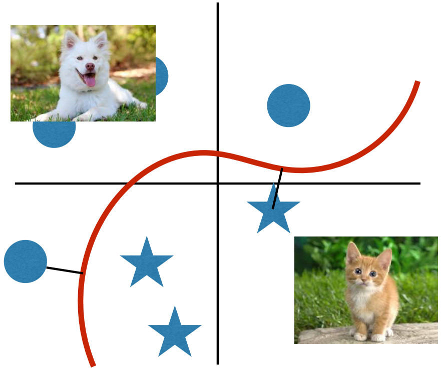
Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

x_1, y_1 x_2, y_2 x_3, y_3

ML model, e.g., neural network w/ 0-1 loss

$$\hat{h}_{nn} = \arg \min_{h_{nn} \in \mathscr{H}} \sum_{i=1}^{n} \ell_{0-1} \left(h_{nn}, x_i, y_i \right)$$



Regression

Example: learning to drive from expert

Regression

Example: learning to drive from expert

Feature x

Expert steering angle y

Regression

Example: learning to drive from expert

Feature x

Expert steering angle y

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

collected by human expert

Regression

Example: learning to drive from expert

Feature x

Expert steering angle y

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

collected by human expert

Continuous variable $(-\pi, \pi)$

Regression

Example: learning to drive from expert

Feature x

Expert steering angle y

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

collected by human expert

Continuous variable $(-\pi, \pi)$

Loss function: square loss

$$\ell(h, x, y) = (h(x) - y)^2$$

Regression

Example: learning to drive from expert

Feature x

Expert steering angle y

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

collected by human expert

Continuous variable $(-\pi, \pi)$

Loss function: square loss

$$\ell(h, x, y) = (h(x) - y)^2$$

Hypothesis class: linear functions

$$h(x) := \theta^\top x, \text{ where } \theta \in \mathbb{R}^d$$

Regression

Example: learning to drive from expert

Feature x

Expert steering angle y

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

collected by human expert

Continuous variable $(-\pi, \pi)$

Loss function: square loss

$$\ell(h, x, y) = (h(x) - y)^2$$

Hypothesis class: linear functions

$$h(x) := \theta^\top x, \text{ where } \theta \in \mathbb{R}^d$$

Training: minimizing mean squared error (MSE)

$$\arg\min_{\theta} \sum_i (\theta^\top x_i - y_i)^2$$

Application of Regression: training self-driving cars *[Pomerleau, NeurIPS '88]*

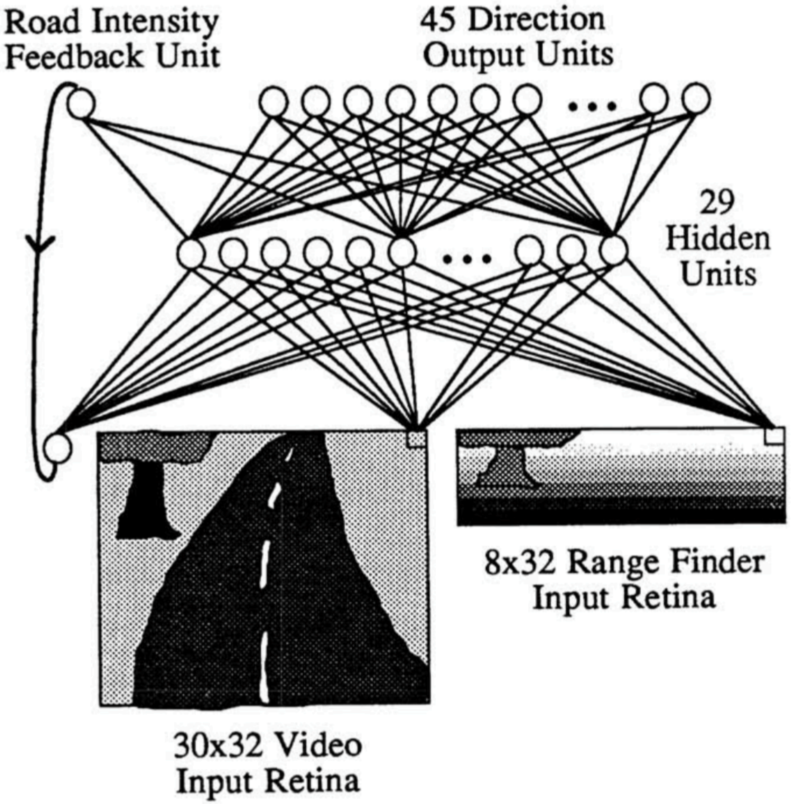
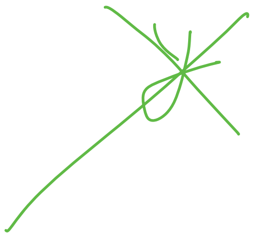


Figure 1: ALVINN Architecture

Unsupervised Learning

Dataset:

$$\mathscr{D} = \{(x_1), \ldots, (x_n)\}, x_i \in \mathbb{R}^d, x_i \sim \mathscr{P}$$

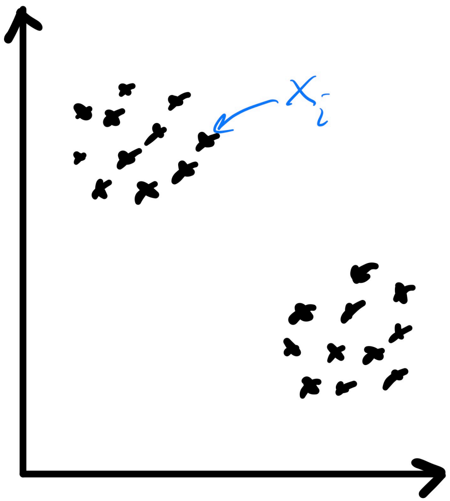


Unsupervised Learning

Dataset:

$$\mathcal{D} = \{(x_1), \ldots, (x_n)\}, x_i \in \mathbb{R}^d, x_i \sim \mathcal{P}$$

Example: Clustering

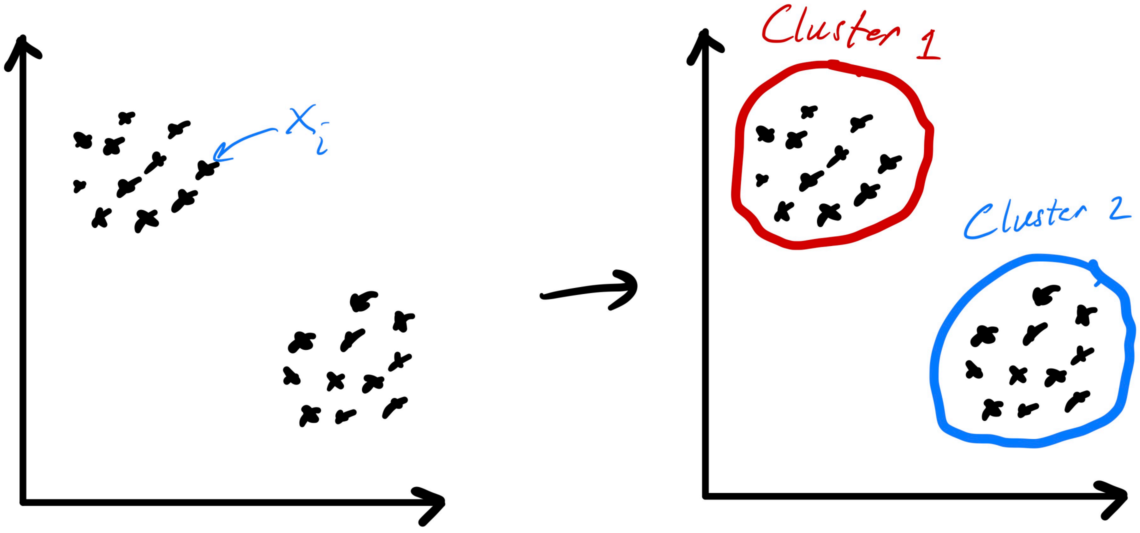


Unsupervised Learning

Dataset:

$$\mathscr{D} = \{(x_1), \ldots, (x_n)\}, x_i \in \mathbb{R}^d, x_i \sim \mathscr{P}$$

Example: Clustering

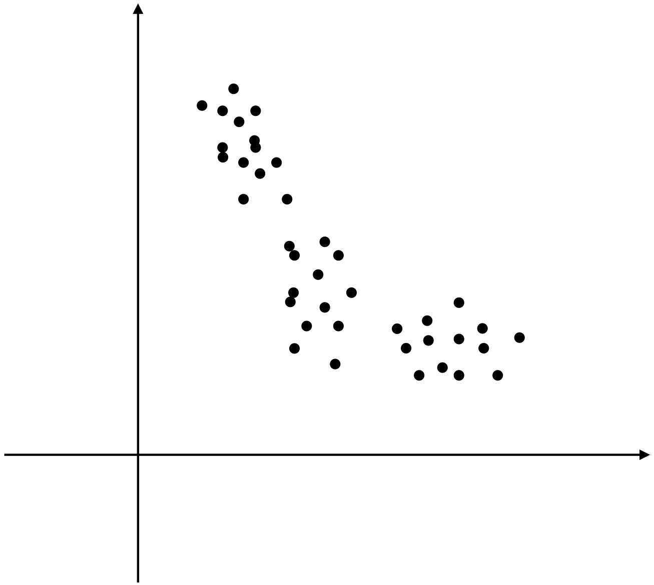


Unsupervised Learning

Dataset:

$$\mathscr{D} = \{(x_1), \ldots, (x_n)\}, x_i \in \mathbb{R}^d, x_i \sim \mathscr{P}$$

Example: distribution estimation

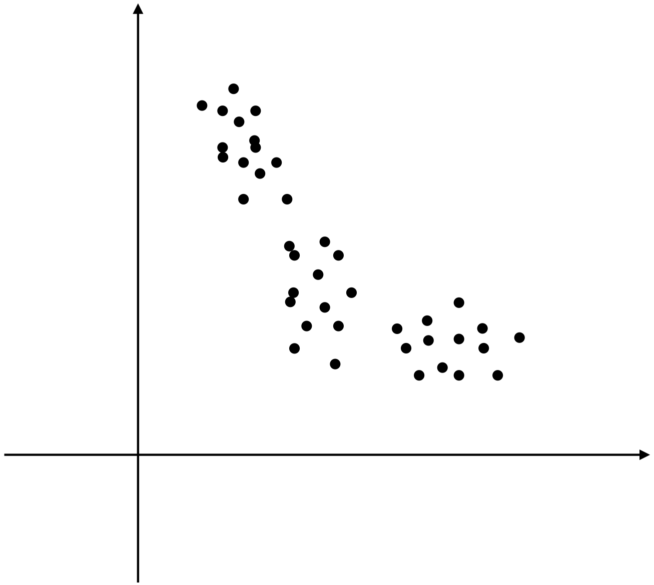


Unsupervised Learning

Dataset:

$$\mathscr{D} = \{(x_1), \ldots, (x_n)\}, x_i \in \mathbb{R}^d, x_i \sim \mathscr{P}$$

Example: distribution estimation



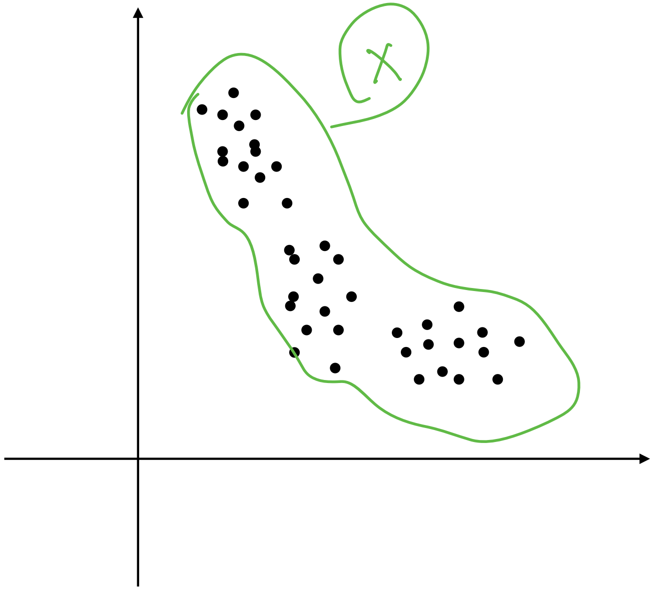
Can we construct a distribution $\hat{\mathscr{P}}$ to approximate $\mathscr{P}$?

Unsupervised Learning

Dataset:

$$\mathcal{D} = \{(x_1), \ldots, (x_n)\}, x_i \in \mathbb{R}^d, x_i \sim \mathcal{P}$$

Example: distribution estimation



Can we construct a distribution $\hat{\mathcal{P}}$ to approximate $\mathcal{P}$?

Anomaly detection / generative AI

Application of distribution estimation: face generator

Generated images:

Application of distribution estimation: face generator

Generated images:

Similar images from the dataset

Application of distribution estimation: face generator

Generated images:

Similar images from the dataset

Outline for Today:

1. Supervised Learning (Classification / Regression) and Unsupervised learning

2. Generalization

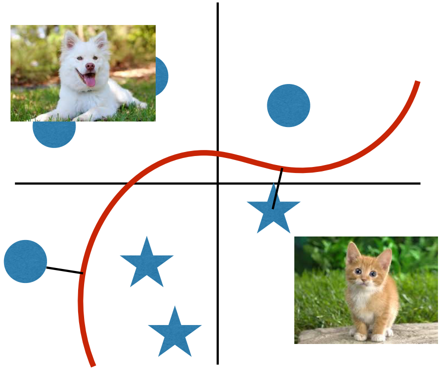
3. Training / validation / testing

Generalization

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

ML model

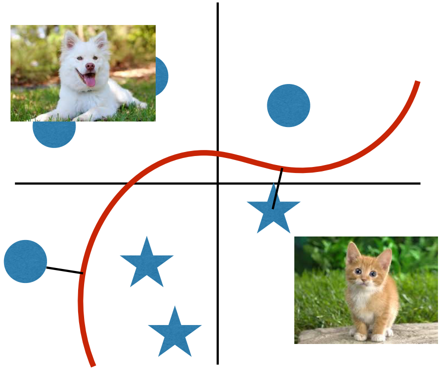


Generalization

Dataset $\mathscr{D}$

( ,cat) ( ,cat) ( ,dog)

ML model



Generalization: how well can our trained model do on <u>unseen test examples</u>?

Let's formalize this using distribution

The **Independent and identically distributed** (i.i.d) assumption:

Let’s formalize this using distribution

The **Independent and identically distributed** (i.i.d) assumption:

Training data $\mathscr{D}$ is i.i.d sampled from a distribution $\mathscr{P}$, i.e., $x_i, y_i \sim \mathscr{P}$, $\forall i \in [n]$
(i.e., all pairs are sampled from $\mathscr{P}$, and (x_i, y_i) is independent of others)

Let's formalize this using distribution

The **Independent and identically distributed** (i.i.d) assumption:

Training data $\mathscr{D}$ is i.i.d sampled from a distribution $\mathscr{P}$, i.e., $x_i, y_i \sim \mathscr{P}, \forall i \in [n]$
(i.e., all pairs are sampled from $\mathscr{P}$, and (x_i, y_i) is independent of others)

We further assume test data is also from $\mathscr{P}$, i.e., $(x, y) \sim \mathscr{P}$

Let's formalize this using distribution

The **Independent and identically distributed** (i.i.d) assumption:

Training data $\mathscr{D}$ is i.i.d sampled from a distribution $\mathscr{P}$, i.e., $x_i, y_i \sim \mathscr{P}, \forall i \in [n]$
(i.e., all pairs are sampled from $\mathscr{P}$, and (x_i, y_i) is independent of others)

We further assume test data is also from $\mathscr{P}$, i.e., $(x, y) \sim \mathscr{P}$

Generalization error: $\mathbb{E}_{x,y\sim\mathscr{P}}\left[\ell(\hat{h}, x, y)\right]$

Let's formalize this using distribution

The **Independent and identically distributed** (i.i.d) assumption:

Training data $\mathscr{D}$ is i.i.d sampled from a distribution $\mathscr{P}$, i.e., $x_i, y_i \sim \mathscr{P}$, $\forall i \in [n]$
(i.e., all pairs are sampled from $\mathscr{P}$, and (x_i, y_i) is independent of others)

We further assume test data is also from $\mathscr{P}$, i.e., $(x, y) \sim \mathscr{P}$

Generalization error: $\mathbb{E}_{x,y\sim\mathscr{P}}\left[\ell(\hat{h}, x, y)\right]$

e.g., expected classification error of $\hat{h}$

Overfitting

Overfitting: we have a small training error but large generalization error

Overfitting

Overfitting: we have a small training error but large generalization error

Example

Hypothesis $\tilde{h}$ that **memorizes** the whole training set

$$\tilde{h}(x) = \begin{cases} y_i & \exists (x_i, y_i) \in \mathscr{D} \text{ w/ } x_i = x \\ -1 & \text{else} \end{cases}$$

Overfitting

Overfitting: we have a small training error but large generalization error

Example

Hypothesis $\tilde{h}$ that **memorizes** the whole training set

$$\tilde{h}(x) = \begin{cases} y_i & \exists (x_i, y_i) \in \mathscr{D} \text{ w/ } x_i = x \\ -1 & \text{else} \end{cases}$$

What is the training error? Is this a good classifier?

Traing Err = $\sum_{i=1}^{n} \mathbb{1}(\tilde{h}(x_i) \neq y_i)$

Overfitting

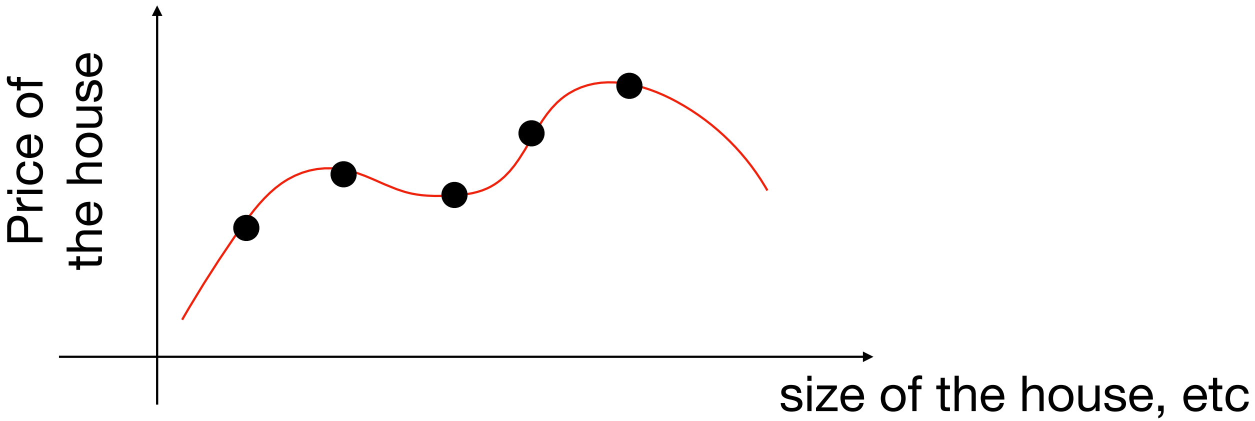
Overfitting: we have a small training error but large generalization/test error

Example

Overfitting

Overfitting: we have a small training error but large generalization/test error

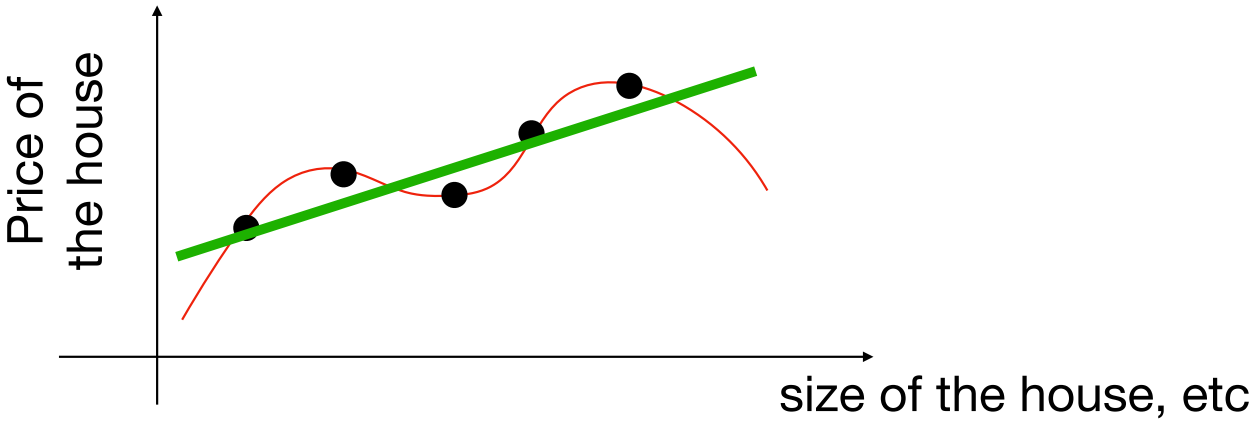
Example



Overfitting

Overfitting: we have a small training error but large generalization/test error

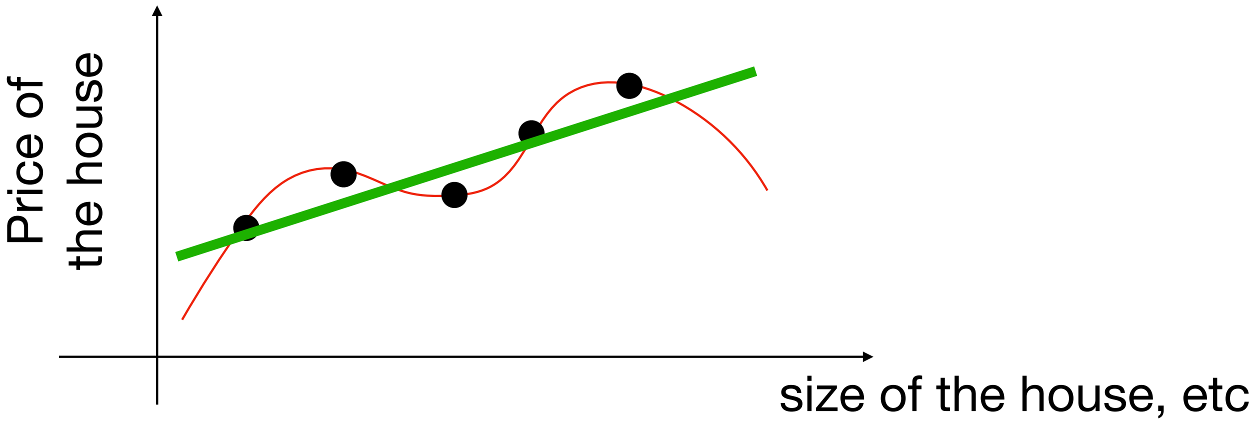
Example



Overfitting

Overfitting: we have a small training error but large generalization/test error

Example



Training error = 0 (e.g., we probably overfit to noises), but could do terribly on test examples

Overfitting

How to tell that our models overfit?

Outline for Today:

1. Supervised Learning (Classification / Regression) and Unsupervised learning

2. Generalization

3. Training / validation / testing

Training, validation, and testing

Given a training dataset $\mathscr{D}$, we can split it into three sets:

$\mathscr{D}_{TR}$: training set

$\mathscr{D}_{VA}$: validation set

$\mathscr{D}_{TE}$: test set

Training, validation, and testing

Given a training dataset $\mathscr{D}$, we can split it into three sets:

$\mathscr{D}_{TR}$: training set

$\mathscr{D}_{VA}$: validation set

$\mathscr{D}_{TE}$: test set

Before training/learning, we often **randomly** split it with size proportional to 80% / 10% / 10%

Selecting models using validation set

We can use validation set to select models, i.e., select hypothesis class, tune parameters, etc

Selecting models using validation set

We can use validation set to select models, i.e., select hypothesis class, tune parameters, etc

Small avg error on $\mathscr{D}_{TR}$ but larger avg error on $\mathscr{D}_{VA}$ indicates overfitting

Selecting models using validation set

We can use validation set to select models, i.e., select hypothesis class, tune parameters, etc

Small avg error on $\mathscr{D}_{TR}$ but larger avg error on $\mathscr{D}_{VA}$ indicates overfitting

$$\Downarrow$$

Revise model on $\mathscr{D}_{TR}$ (e.g., add regularization, change neural network structures, etc)

Selecting models using validation set

We can use validation set to select models, i.e., select hypothesis class, tune parameters, etc

Small avg error on $\mathscr{D}_{TR}$ but larger avg error on $\mathscr{D}_{VA}$ indicates overfitting

$\Downarrow \Uparrow$

Revise model on $\mathscr{D}_{TR}$ (e.g., add regularization, change neural network structures, etc)

Do not use test set to train/select models

We should not touch test set during training!

Do not use test set to train/select models

We should not touch test set during training!

This makes sure that the test set $\mathscr{D}_{TE}$ is independent of our model $\hat{h}$

Do not use test set to train/select models

We should not touch test set during training!

This makes sure that the test set $\mathcal{D}_{TE}$ is independent of our model $\hat{h}$

Such independence implies that:

$$\frac{1}{|\mathcal{D}_{TE}|} \sum_{x,y \in \mathcal{D}_{TE}} \ell(\hat{h}, x, y) \approx \mathbb{E}_{x,y \sim \mathcal{P}}[\ell(\hat{h}, x, y)]$$

(Due to law of large numbers)

Other ways to split the data?

Can we split data based on features, or labels?

Summary

1. Given a task and a dataset
   $\mathcal{D} = \{x_i, y_i\}, x_i, y_i \sim \mathcal{P}$

Summary

1. Given a task and a dataset $\mathscr{D} = \{x_i, y_i\}, x_i, y_i \sim \mathscr{P}$

2. Design hypothesis class $\mathscr{H}$ and loss function ℓ (encodes inductive bias)

Summary

1. Given a task and a dataset
$\mathcal{D} = \{x_i, y_i\}, x_i, y_i \sim \mathcal{P}$

2. Design hypothesis class $\mathcal{H}$ and loss function ℓ (encodes inductive bias)

3. Train: $\hat{h} = \arg\min_{h \in \mathcal{H}} \sum_{(x, y \in \mathcal{D})} \ell(h, x, y)$

Summary

1. Given a task and a dataset
$\mathscr{D} = \{x_i, y_i\}, x_i, y_i \sim \mathscr{P}$

2. Design hypothesis class $\mathscr{H}$ and loss function ℓ (encodes inductive bias)

3. Train: $\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{(x, y \in \mathscr{D})} \ell(h, x, y)$

Often repeated many times using validation $\mathscr{D}_{VA}$

Summary

1. Given a task and a dataset $\mathscr{D} = \{x_i, y_i\}, x_i, y_i \sim \mathscr{P}$

2. Design hypothesis class $\mathscr{H}$ and loss function ℓ (encodes inductive bias)

3. Train: $\hat{h} = \arg\min_{h \in \mathscr{H}} \sum_{(x,y \in \mathscr{D})} \ell(h, x, y)$

4. Output: $\hat{h}$ that has small generalization error $\mathbb{E}_{x,y\sim\mathscr{P}}[\ell(\hat{h}, x, y)]$

Often repeated many times using validation $\mathscr{D}_{VA}$