SOI1010
Machine Learning II

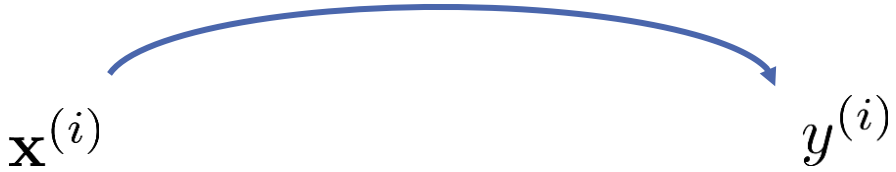
Lecture 4: Linear Regression Cont'd & Classification

Department of Data Science
Hanyang University

Recap: Machine Learning

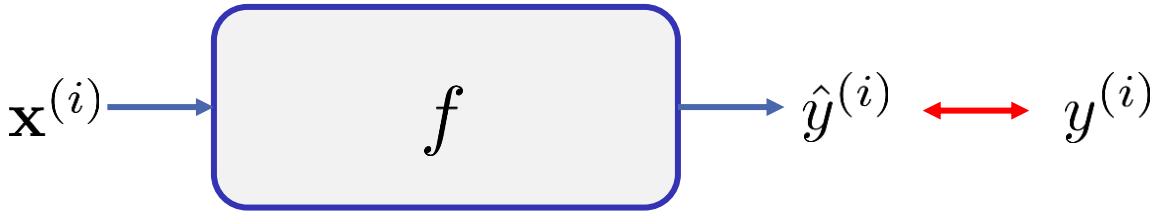
- Learn from data to achieve generalization:
  - To design algorithms that can be trained on data to make predictions for new examples/data
- Supervised Learning

Assume there is a mapping function:



Train a model f that is as close to the mapping function as possible

- To make correct predictions for new examples

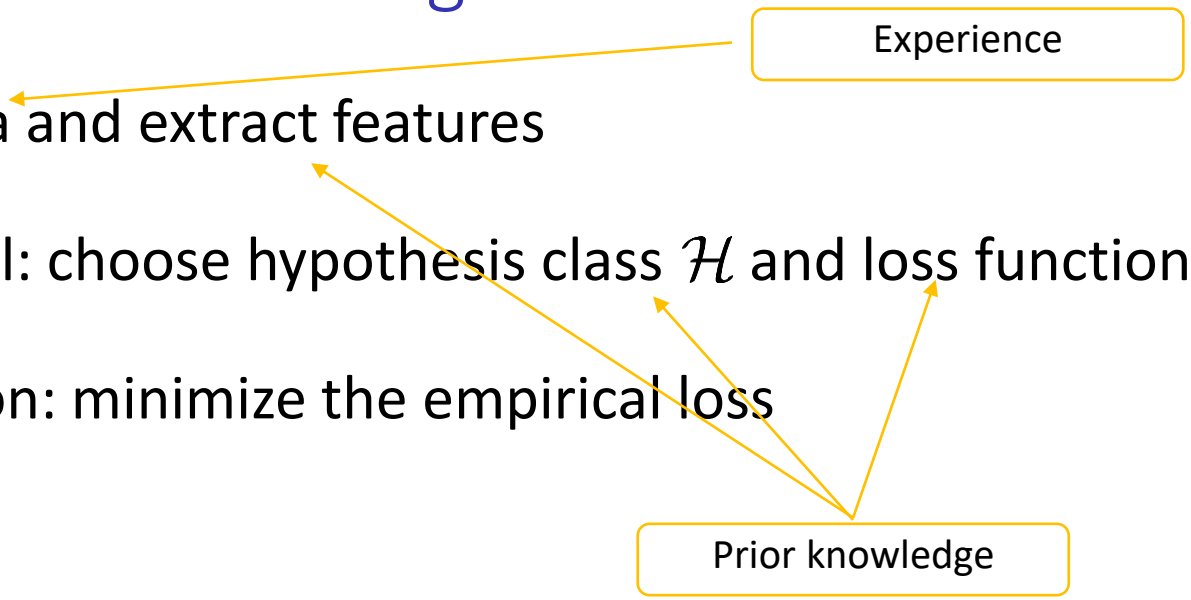


Recap: Math formulation

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Find $y = f(\boldsymbol{x}) \in \mathcal{H}$ that minimizes $\hat{\mathcal{L}}(f) = \frac{1}{n} \sum_{i=1}^{n} l(f, \boldsymbol{x}_i, y_i)$
- s.t. the expected loss is small

$$\mathcal{L}(f) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[l(f, \boldsymbol{x}, y)]$$

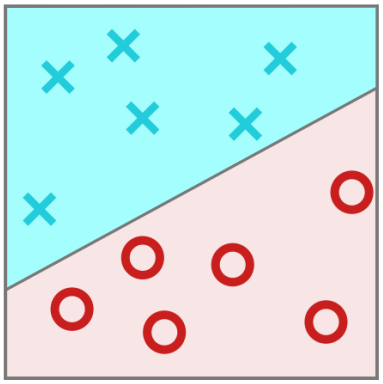
Recap: Machine Learning 1-2-3



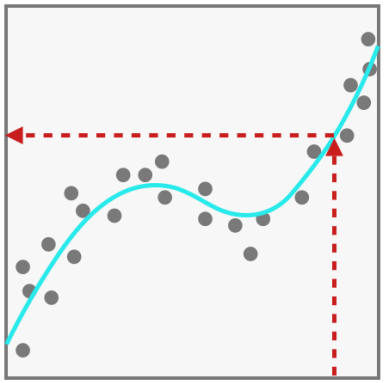
- Collect data and extract features
- Build model: choose hypothesis class $\mathcal{H}$ and loss function l
- Optimization: minimize the empirical loss

Recap: Classification vs Regression

- Classification
  - The task of predicting a category



- Regression
  - The task of predicting a continuous quantity



Recap: Linear regression math formulation

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Find $f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ that minimizes $\hat{\mathcal{L}}(f_{\boldsymbol{w}}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^T\boldsymbol{x}^{(i)} - y^{(i)})^2$

Linear model $\mathcal{H}$

l_2 loss

Why l_2 loss?

- Why not choose another loss
  - l_1 loss, hinge loss, exponential loss, etc…
- Empirical: easy to optimize
  - For linear case: $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$
- Theoretical: a way to encode prior knowledge
- Questions:
  - What kind of prior knowledge?
  - Principal way to derive loss function?

Maximum likelihood Estimation (MLE)

Maximum likelihood Estimation (MLE)

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(\boldsymbol{x}, y) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- Would like to pick $\boldsymbol{\theta}$ so that $P_{\boldsymbol{\theta}}(\boldsymbol{x}, y)$ fits the data well

Maximum likelihood Estimation (MLE)

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(\boldsymbol{x}, y) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- "fitness" of $\boldsymbol{\theta}$ to one data point $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\text{likelihood}(\boldsymbol{\theta}; \boldsymbol{x}^{(i)}, y^{(i)}) := P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)})$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(\boldsymbol{x}, y) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- "fitness" of $\boldsymbol{\theta}$ to i.i.d. data points $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$

$$\text{likelihood}(\boldsymbol{\theta}; \{\boldsymbol{x}^{(i)}, y^{(i)}\}) := P_{\boldsymbol{\theta}}(\{\boldsymbol{x}^{(i)}, y^{(i)}\}) = \prod_i P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)})$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(\boldsymbol{x}, y) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- MLE: maximize "fitness" of $\boldsymbol{\theta}$ to i.i.d. data points $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$

$$\begin{aligned}\boldsymbol{\theta}_{\mathrm{ML}} &= \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \, \mathrm{likelihood}(\boldsymbol{\theta}; \{\boldsymbol{x}^{(i)}, y^{(i)}\}) \\ &= \arg\max_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \prod_i P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)})\end{aligned}$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(\boldsymbol{x}, y) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- MLE: maximize "fitness" of $\boldsymbol{\theta}$ to i.i.d. data points $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \log[\prod_i P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)})]$$

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)})]$$

Maximum likelihood Estimation (MLE)

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(\boldsymbol{x}, y) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- MLE: negative log-likelihood loss

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)})]$$

$$l(P_{\boldsymbol{\theta}}, \boldsymbol{x}^{(i)}, y^{(i)}) = -\log(P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)}))$$

$$\hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = -\sum_i \log(P_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)}))$$

MLE: conditional log-likelihood

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- MLE: negative conditional log-likelihood loss

Only care about predicting y from $\boldsymbol{x}$; do not care about $P(\boldsymbol{x}, y)$

$$\boldsymbol{\theta}_{\text{ML}} = \text{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})]$$

$$l(P_{\boldsymbol{\theta}}, \boldsymbol{x}^{(i)}, y^{(i)}) = -\log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

$$\hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = -\sum_i \log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

MLE: conditional log-likelihood

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- MLE: negative conditional log-likelihood loss

$P(y|\boldsymbol{x})$: discriminative;
$P(\boldsymbol{x}, y)$: generative

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})]$$

$$l(P_{\boldsymbol{\theta}}, \boldsymbol{x}^{(i)}, y^{(i)}) = -\log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

$$\hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = -\sum_i \log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

MLE: conditional log-likelihood

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- MLE: negative conditional log-likelihood loss

$P(y|\boldsymbol{x})$: discriminative;
$P(\boldsymbol{x}, y)$: generative

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})]$$

$$l(P_{\boldsymbol{\theta}}, \boldsymbol{x}^{(i)}, y^{(i)}) = -\log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

$$\hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = -\sum_i \log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

MLE: conditional log-likelihood

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Let $\{P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$ be a family of distributions parameterized by $\boldsymbol{\theta}$
- Define $P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \mathcal{N}(y; f_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2)$

$$\log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})) = \frac{-1}{2\sigma^2}(f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y_i)^2 - \log(\sigma) - \frac{1}{2}\log(2\pi)$$

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})]$$

$$\hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = -\sum_i \log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\, \hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = \mathrm{argmin}_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\, \frac{1}{n}\sum_{i=1}^{n}(f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

MLE: conditional log-likelihood

- Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)}) : 1 \leq i \leq n\}$ i.i.d. from distribution $\mathcal{D}$
- Find $f_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$ that minimizes $\hat{\mathcal{L}}(f_{\boldsymbol{w}}) = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{w}^T \boldsymbol{x}^{(i)} - y^{(i)})^2$
- Define $P_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \mathcal{N}(y; f_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2)$

$$\log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})) = \frac{-1}{2\sigma^2}(f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y_i)^2 - \log(\sigma) - \frac{1}{2}\log(2\pi)$$

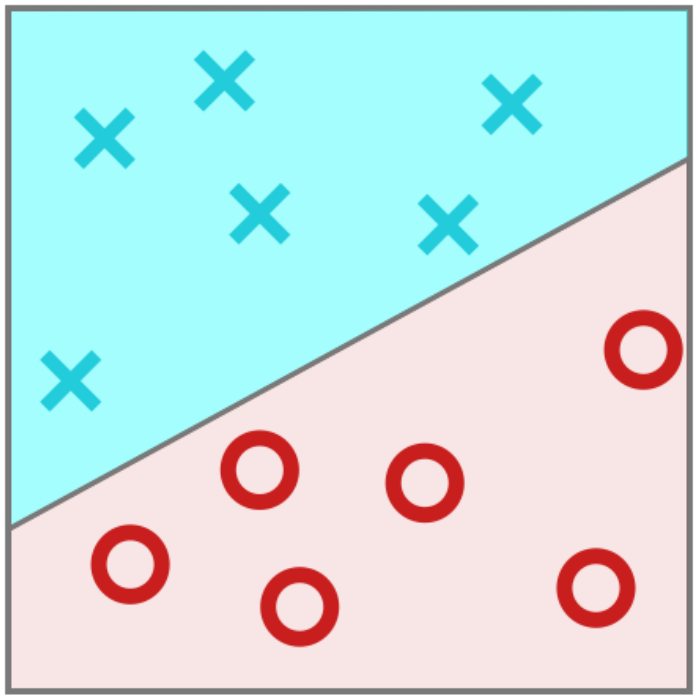
$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmax}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \sum_i \log[P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)})]$$

$$\hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = -\sum_i \log(P_{\boldsymbol{\theta}}(y^{(i)}|\boldsymbol{x}^{(i)}))$$

l_2 loss: Normal + MLE

$$\boldsymbol{\theta}_{\mathrm{ML}} = \mathrm{argmin}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \hat{\mathcal{L}}(P_{\boldsymbol{\theta}}) = \mathrm{argmin}_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \frac{1}{n} \sum_{i=1}^{n} (f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)})^2$$

Classification vs Regression



Classification

Regression

Classification

Why classification?

- Classification: a kind of summary
- Easy to interpret
- Easy for making decisions
- Example: image classification

Computer Vision

- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

- What do you see in the image?

- What is he doing?

Computer Vision

- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

- What do you see in the image?
  - Person
  - Camera
  - Tripod
  - buildings

- What is he doing?

Computer Vision

- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

- What do you see in the image?

- What is he doing?
  - Taking a picture of something
  - Or maybe taking a pose…?

Computer Vision

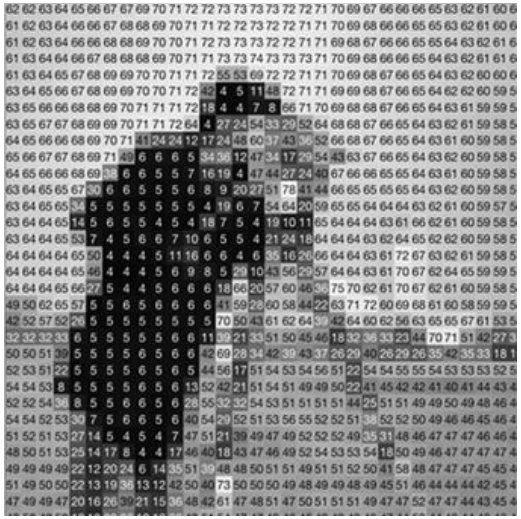
- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

- What do computers see in the image?

Computer Vision

- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

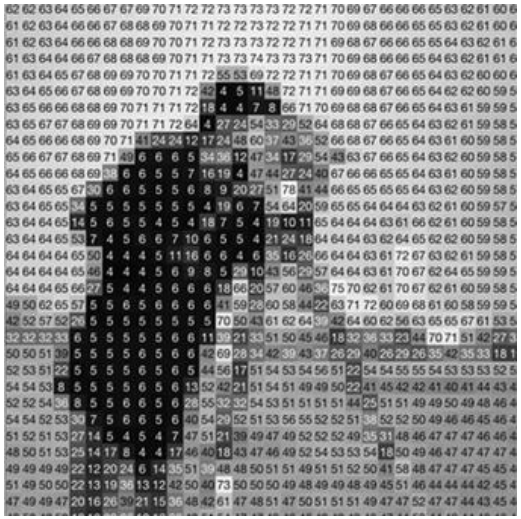
- What do computers see in the image?
  - Just numbers..!



Computer Vision

- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

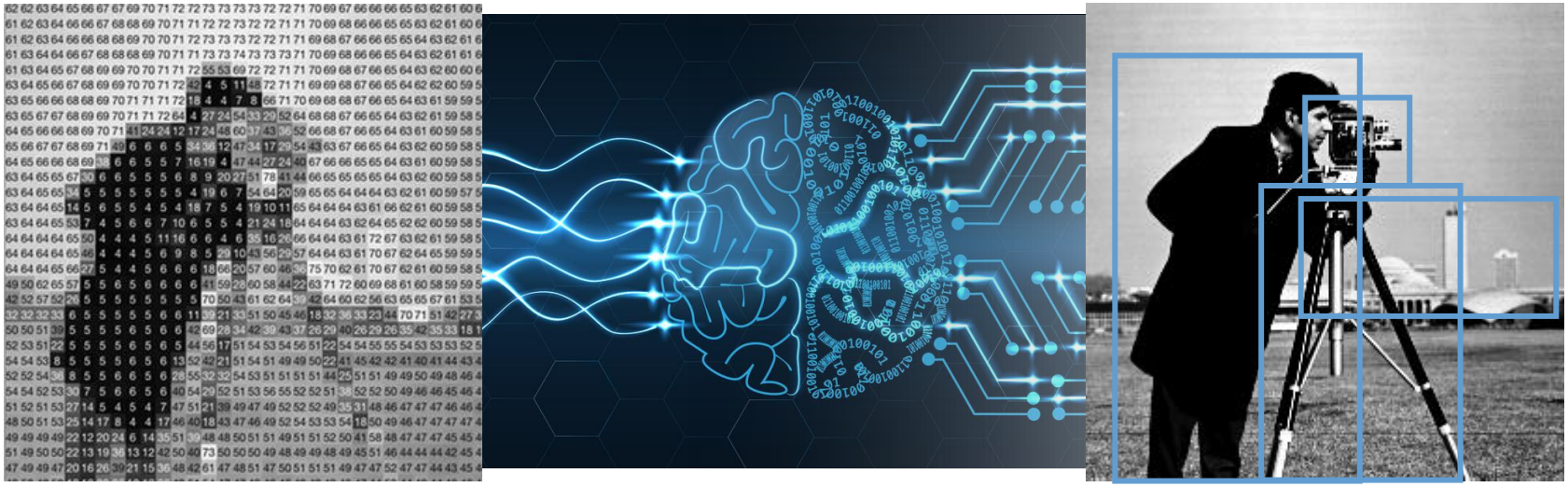
- How can we bridge the gap?



Computer Vision

- Computer Vision
  - Subfield of machine learning
  - Goal: build the computer systems that can understand images and/or videos the way humans do

- How can we bridge the gap?



Computer Vision

- To understand/predict behaviors of objects/targets
  (Action recognition/prediction)

- We need to understand their interactions with objects and surroundings
  (Scene understanding)

- We need to understand the locations of objects
  (Localization)

- We need to understand what the objects are
  (Classification)

Image Classification

- One of the most core tasks in computer vision
- Binary classification
  - Whether the image has an object of interest
- e.g.,

cat

Image Classification

There are several challenges

- Viewpoint variation

Image Classification

There are several challenges

- Background clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Image Classification

There are several challenges

- Illumination

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Image Classification

There are several challenges

- Occlusion

This image is CC0 1.0 public domain

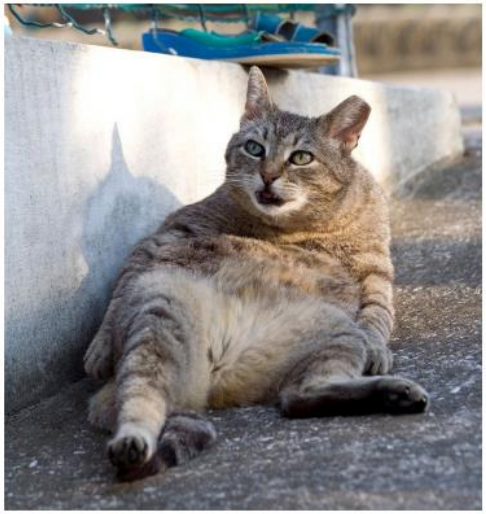
This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

Image Classification

There are several challenges

- Deformation



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0

This image by Tom Thai is licensed under CC-BY 2.0

Image Classification

There are several challenges

- Intraclass variation

This image is CC0 1.0 public domain

Image Classification

There are several challenges

- Scale variation

Image Classification

How to solve such a challenging problem?

Viewpoint variation


Illumination


Deformation


Scale Variation


Intra-class


Occlusion


Background clutter

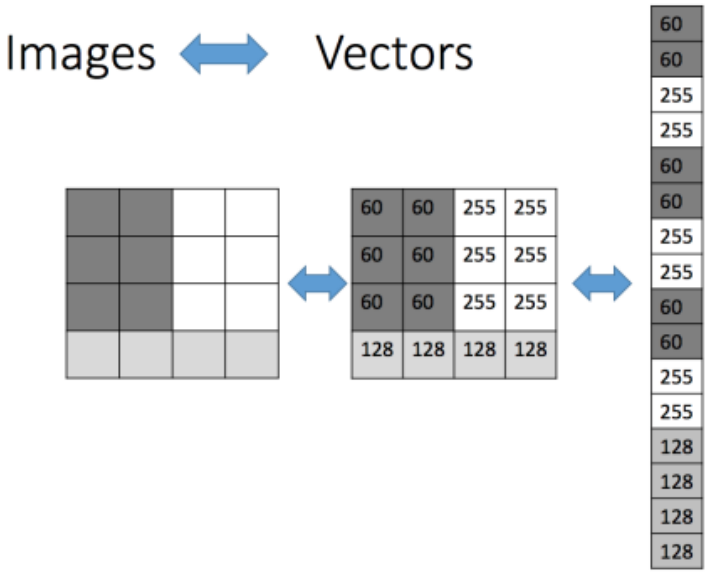
Machine Learning

Representation

- Represent the input as an input vector in $\mathbb{R}^d$
- Map the data to another space that is easy to manipulate
- Vectors are a great representation since we can do linear algebra!

Machine Learning

Representation



Can do much better if you compute a vector of meaningful features

Later, we will see that Deep Learning is good at obtaining very strong features

Machine Learning

Machine learning algorithms can be classified, depending on the assumptions w.r.t. the form of a function f

- Non-parametric
- Parametric

Machine Learning

Machine learning algorithms can be classified, depending on the assumptions w.r.t. the form of a function f

- Non-parametric
  - No assumption on the form of the function
  - Free to learn any functional form
  - Good when you have a lot of data and no prior knowledge
  - The number of parameters grows with the amount of training data
  - e.g., k-nearest neighbors, decision tree, random forest

- Parametric

Machine Learning

Machine learning algorithms can be classified, depending on the assumptions w.r.t. the form of a function f

- Non-parametric

- Parametric
  - Assumption with regards to the form of the function
  - Certain hypothesis space
  - Can simplify the learning process, but can also limit what can be learned
  - The model has a fixed number of parameters, regardless of the amount of data
  - e.g., logistic regression, LDA, neural networks

Machine Learning

Machine learning algorithms can be classified, depending on the assumptions w.r.t. the form of a function f

- Non-parametric
  - No assumption on the form of the function
  - Free to learn any functional form
  - Good when you have a lot of data and no prior knowledge
  - The number of parameters grows with the amount of training data
  - e.g., k-nearest neighbors, decision tree, random forest
- Parametric

Nearest Neighbors

Suppose we're given a novel input vector $\mathbf{x}$ we'd like to classify

Idea: Predict the label as that of the most similar training image

- "most similar": the nearest input vector to $\mathbf{x}$ in the training set
- Can formalize "nearest" in terms of Euclidean distance

$$\text{dist}(\mathbf{x}^{(a)}, \mathbf{x}^{(b)}) = \|\mathbf{x}^{(a)} - \mathbf{x}^{(b)}\|_2 = \sqrt{\sum_{j=1}^{d} (x_j^{(a)} - x_j^{(b)})^2}$$

Note: we don't need to compute the square root. Why?

Nearest Neighbors

Algorithm:

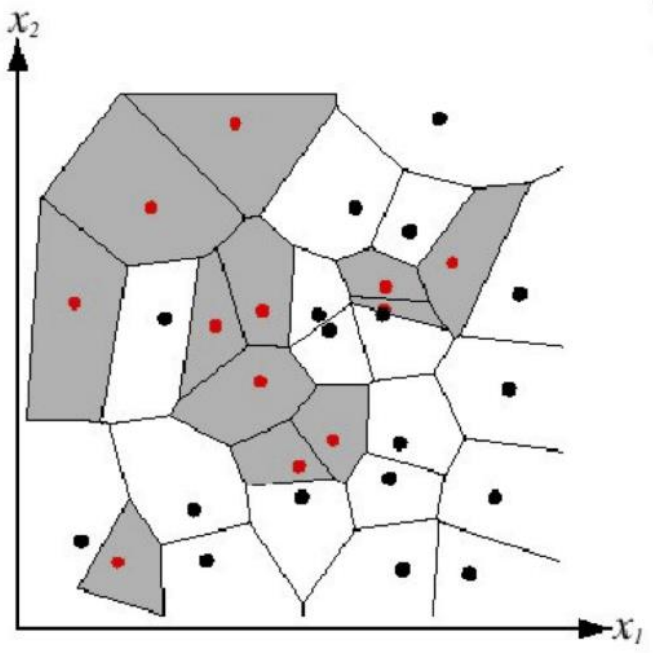
1. **Find a training example** $(\mathbf{x}^{(*)}, y^{(*)})$ **closest to** $\mathbf{x}$**:**

$$\mathbf{x}^{(*)} = \underset{\mathbf{x}^{(i)} \in \text{train. set}}{\arg\min} \ \text{dist}(\mathbf{x}^{(i)}, \mathbf{x})$$

2. **Output** $y = y^*$

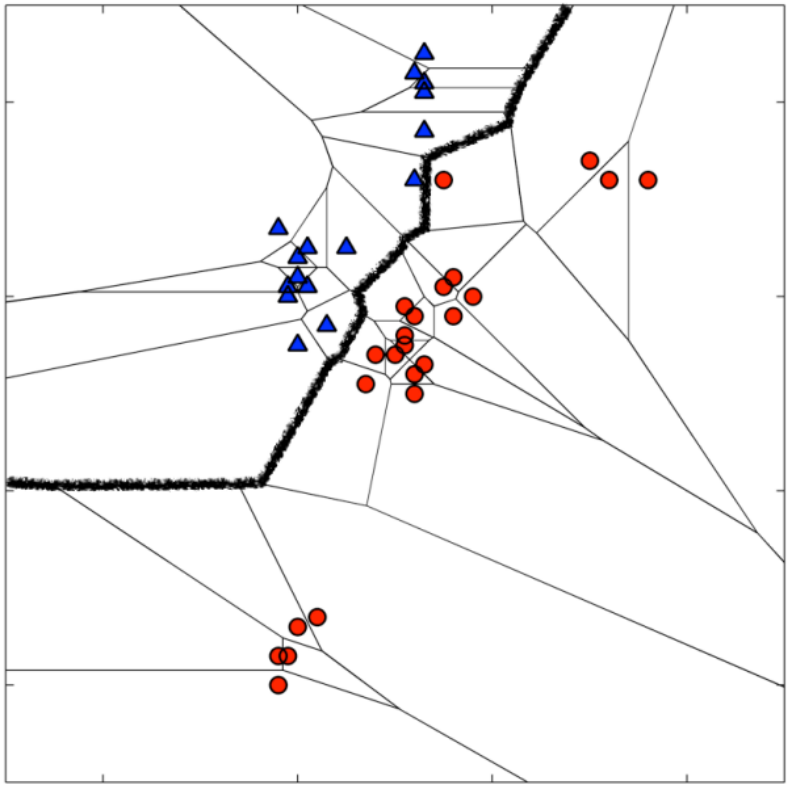
Nearest Neighbors: Decision Boundaries

We can visualize the behavior using a **Voronoi diagram**.



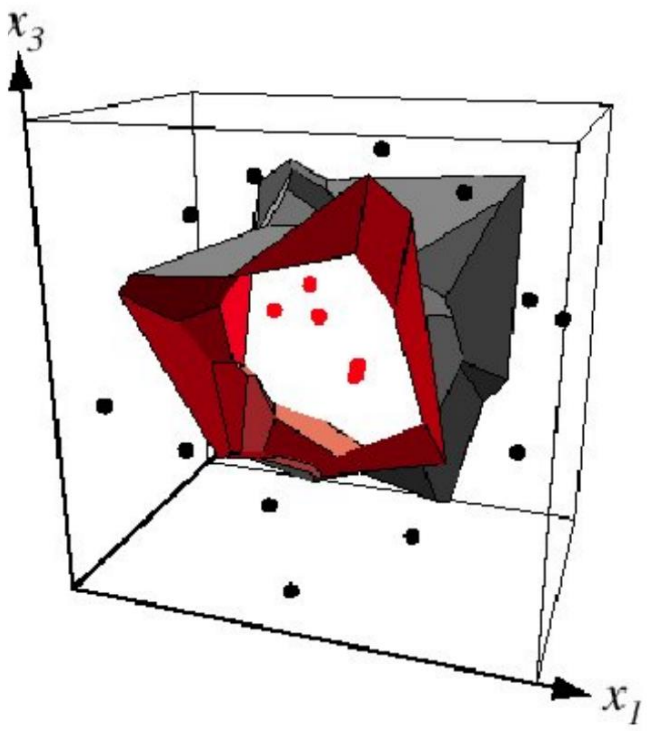
Nearest Neighbors: Decision Boundaries

Decision boundary: the boundary between regions of input space assigned to different categories (classes)



Nearest Neighbors: Decision Boundaries

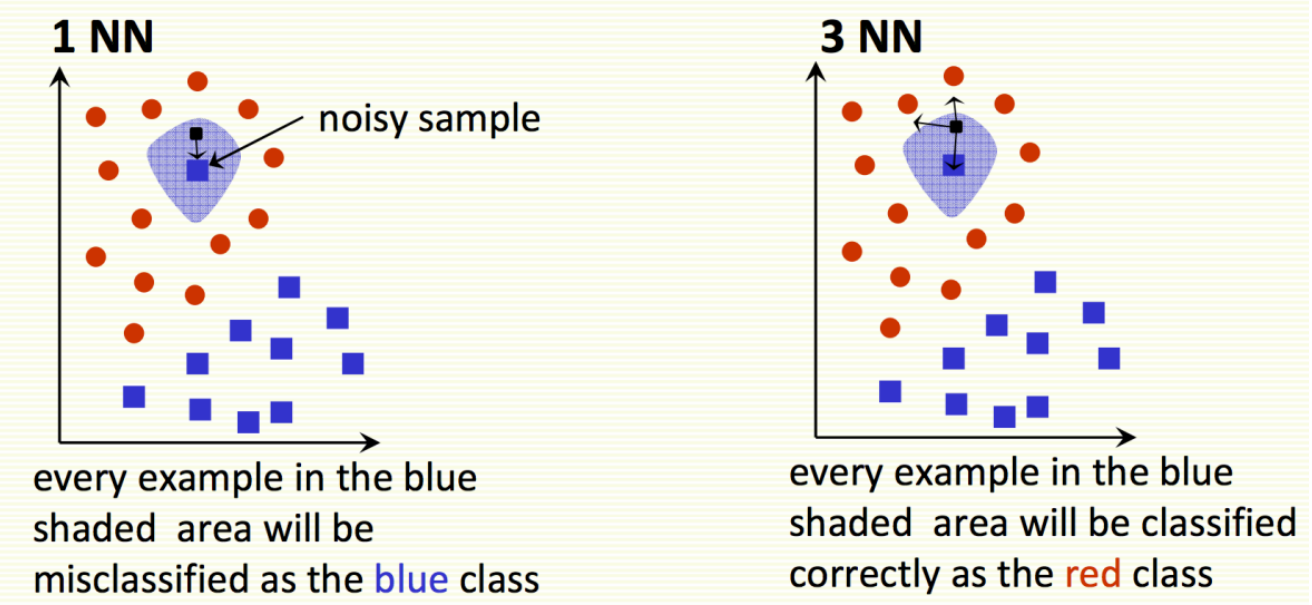
Decision boundary: the boundary between regions of input space assigned to different categories (classes)



k-Nearest Neighbors

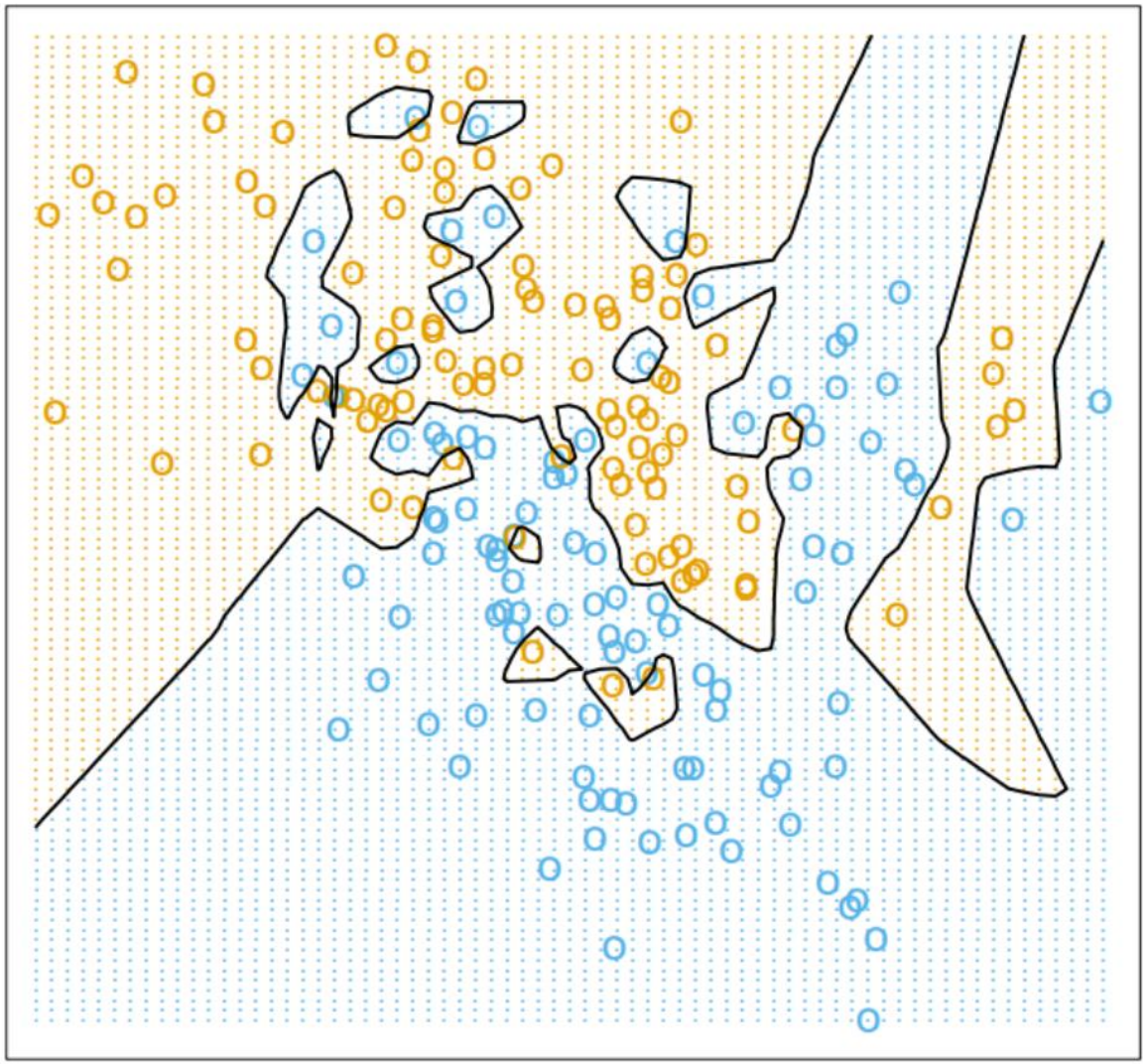
Nearest Neighbors are sensitive to **noise or mis-labeled data**

Take a majority vote from k closest points



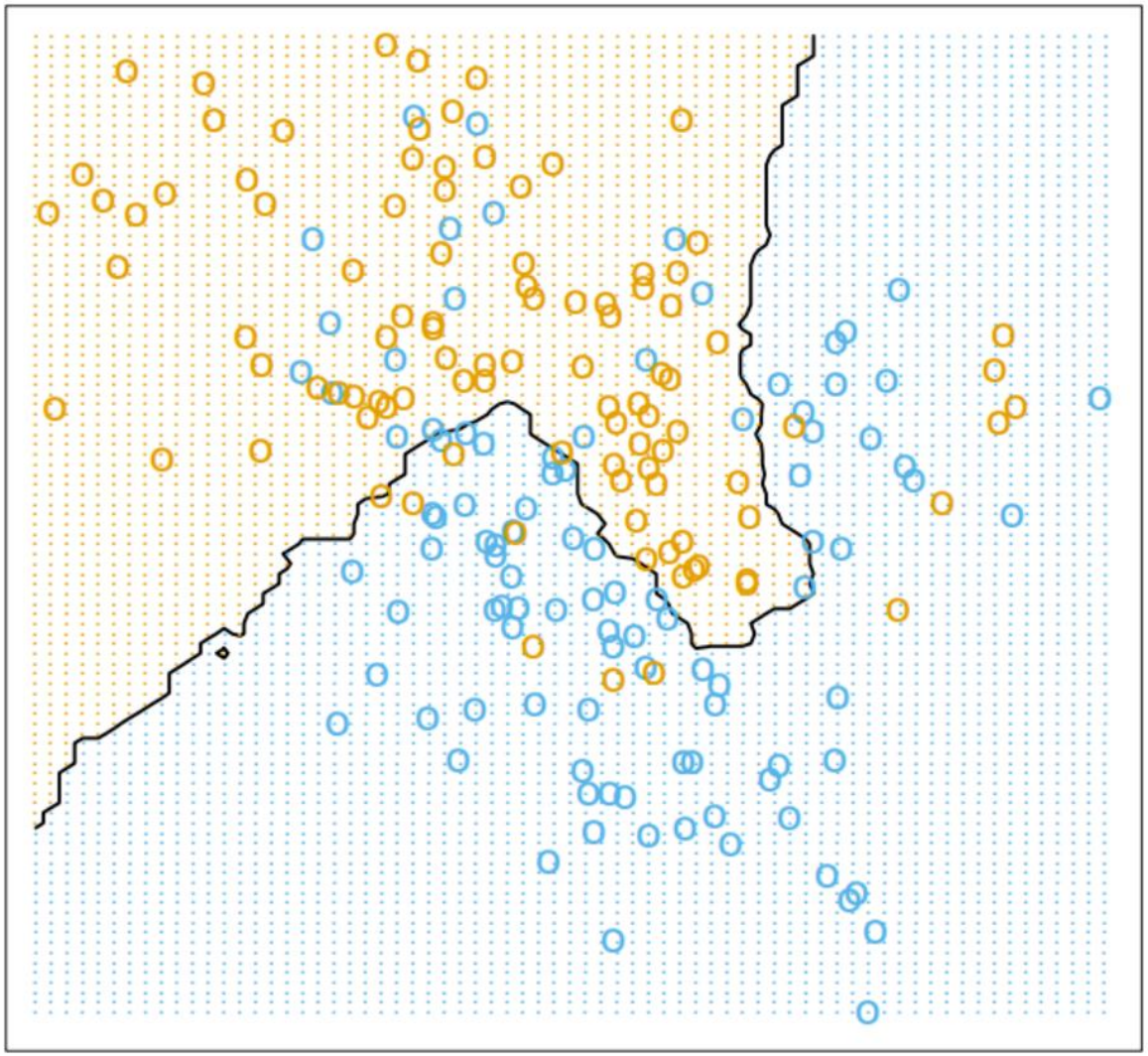
k-Nearest Neighbors

k=1



k-Nearest Neighbors

k=15



k-Nearest Neighbors

What is the best value of k? Tradeoffs in choosing k?

- Small k
  - Good at capturing fine-grained patterns
  - May **overfit,** i.e., be sensitive to random noise in the training data
- Large k
  - Makes stable predictions by averaging over examples
  - May **underfit**, i.e., fail to capture important information
- **Rule of thumb:** $k < sqrt(n)$, where n is the number of training examples

k-Nearest Neighbors : Hyperparameters

Design choices about the algorithms themselves

k is a hyperparameter.

Distance measure is a hyperparameter.

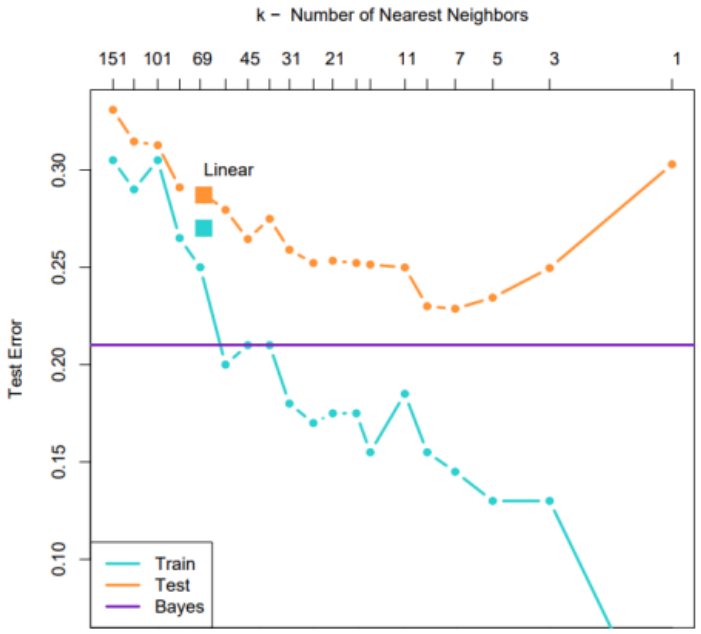
Very problem/dataset-dependent.

Must try them all out and see what works best.

k-Nearest Neighbors : Hyperparameters

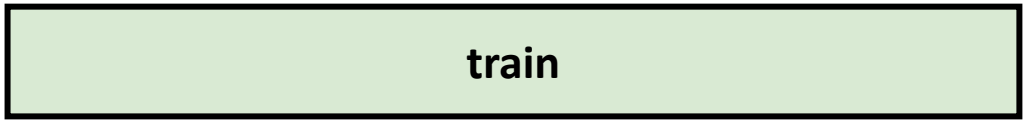
We would like our algorithm to generalize to (perform well on) data it hasn't seen before.

We want to select hyperparameters that give the lowest generalization error (error rate on new examples).



Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the **training data**



BAD: k=1 is the best option for training data

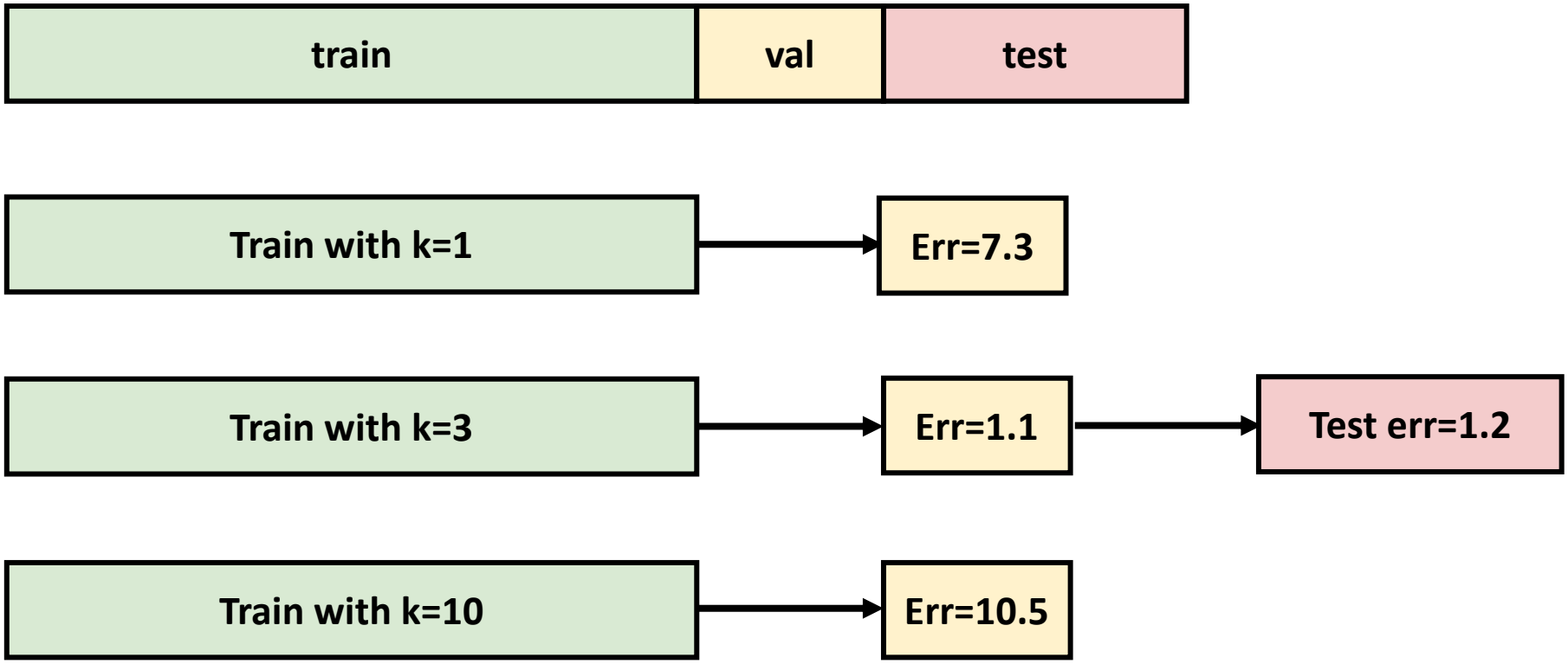
Idea #2: Choose hyperparameters that work best on **test data**

BAD: hyperparameters are overfitted to test data!
No idea how an algorithm will perform on new data

Idea #3: Split **train** data into **train & val**.
Choose hyperparameters on val and evaluate on test

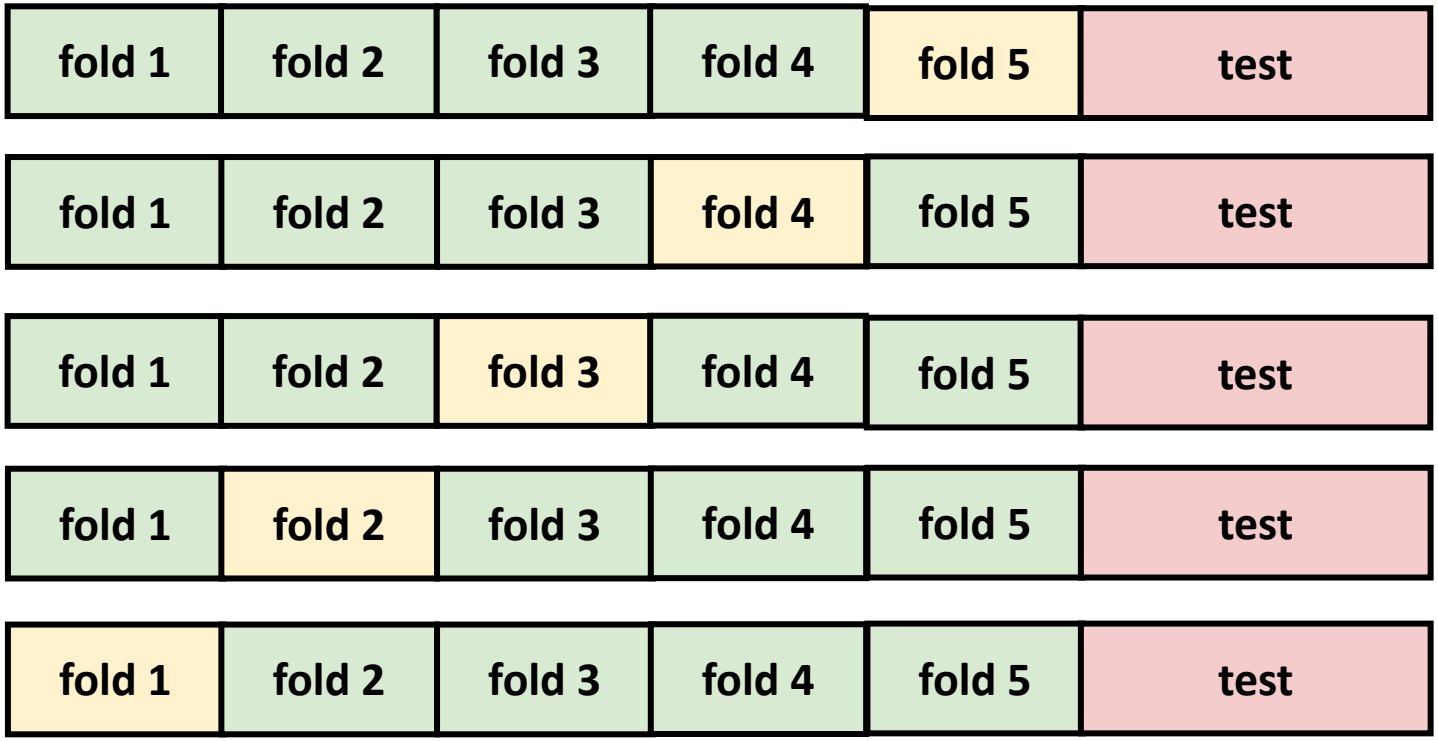
Setting Hyperparameters

Idea #3: Split **train** data into **train & val**.
Choose hyperparameters on val and evaluate on test



Setting Hyperparameters

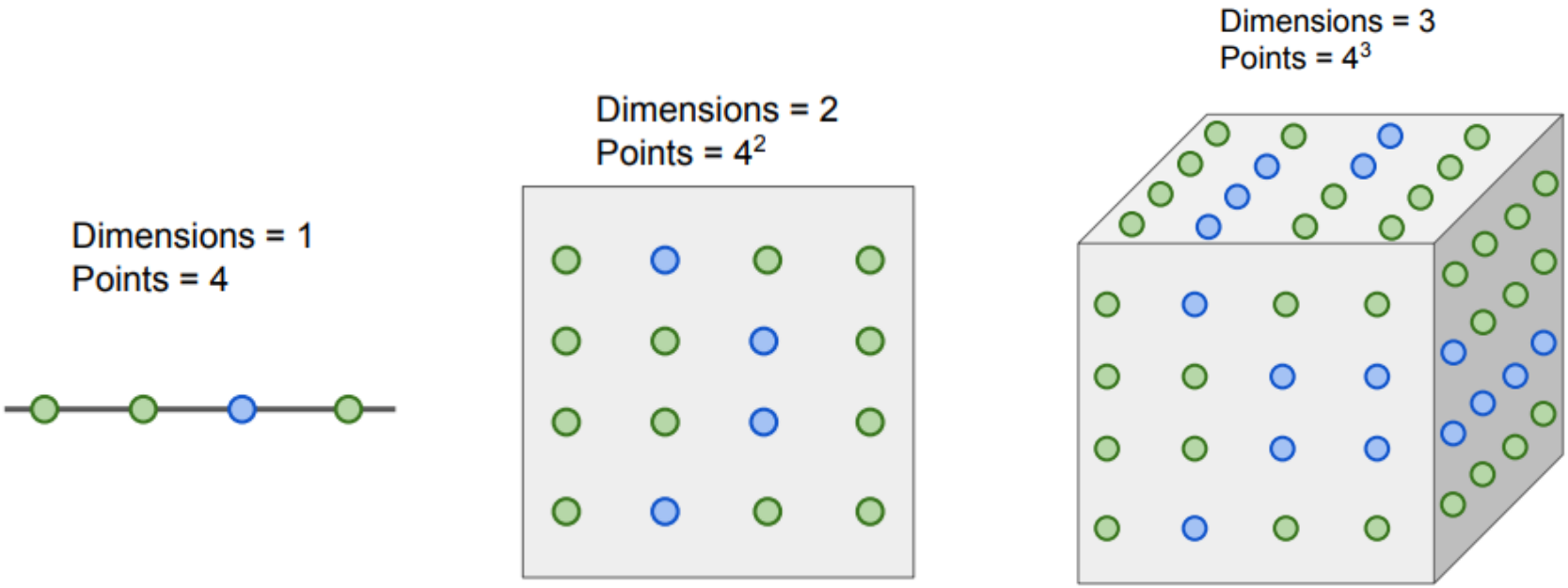
Idea #4: Cross-Validation: split data into folds, and try each fold as validation and average the results



Useful for small datasets, but not used too frequently in deep learning

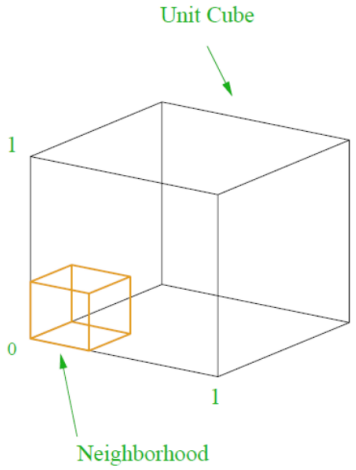
The Curse of Dimensionality

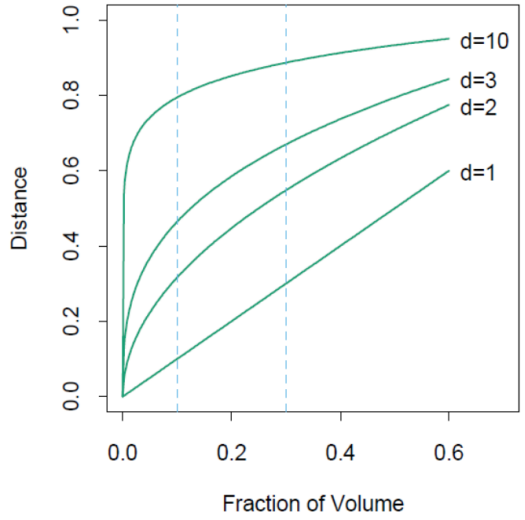
- Let's go back to data representation
- Pixels are actually never used…!
- This is due to the curse of dimensionality.
  - When the dimensionality increases, the volume of the space increases so fast that the available data become sparse



The Curse of Dimensionality

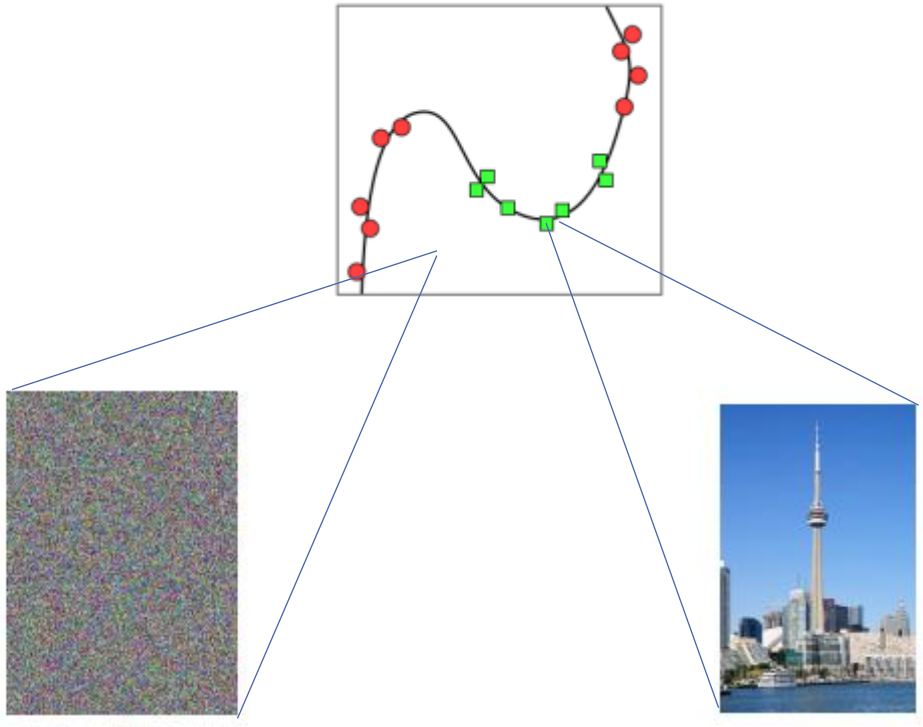
- In high dimensions, "most" points are far apart.
- If we want the nearest neighbor to be closer than ϵ, how many points do we need to guarantee it?
- The volume of a single ball radius ϵ is $\mathcal{O}(\epsilon^d)$
- The total volume of $[0,1]^d$ is 1.
- Therefore, $\mathcal{O}\left(\left(\frac{1}{\epsilon}\right)^d\right)$ balls are needed to cover the volume.





The Curse of Dimensionality

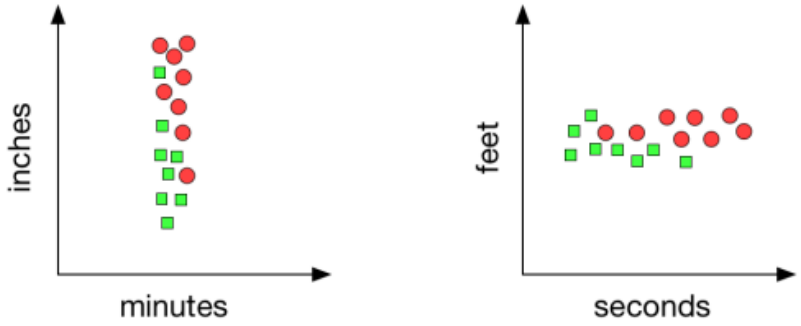
- Saving grace: some datasets (e.g., images) may have low intrinsic dimension (i.e., lie on or near a low-dimensional manifold.
- So nearest neighbors sometimes still works in high dimensions



Normalization

- Nearest Neighbors can be sensitive to the ranges of different features

- Often, the units are arbitrary:



- Simple fix: **normalize** each dimension to be zero mean and unit variance

- Caution: Depending on the problem the scale might be important!