EEP 596: Adv Intro ML || Lecture 7

Dr. Karthik Mohan

Univ. of Washington, Seattle

January 26, 2023

Logistics

A. **Conceptual Assignment 2 due sunday, Jan 29**
B. **Programming Assignment 3 due next saturday, Feb 4**
C. **Late Days - 5 late days per person for the entire quarter**

Last Time

1. Decision Trees
2. Decision Trees vs Logistic Regression

Today

1. Random Forests
2. Multi-class classification
3. Unsupervised learning
4. Clustering Methods

Lectures and Programming Assignments (Tentatively)

Week	Lecture Material	Assignment
1	Linear Regression	Housing Price Prediction
2	Classification	Spam classification (Kaggle)
3	**Classification**	Flower/Leaf classification
4	**Clustering**	MNIST digits clustering
5	Anomaly Detection	Crypto Prediction (Kaggle + P)
6	Data Visualization	Crypto Prediction (Kaggle + P)
7	Deep Learning	Visualizing 1000 images
8	Deep Learning (DL)	ECG Arrythmia Detection
9	DL in NLP	TwitterSentiment Analysis (Kaggle + P)
10	DLs in Vision	TwitterSentiment Analysis (Kaggle + P)

Decision Trees vs Logistic Regression

1. Both are interpretable in different ways

Decision Trees vs Logistic Regression

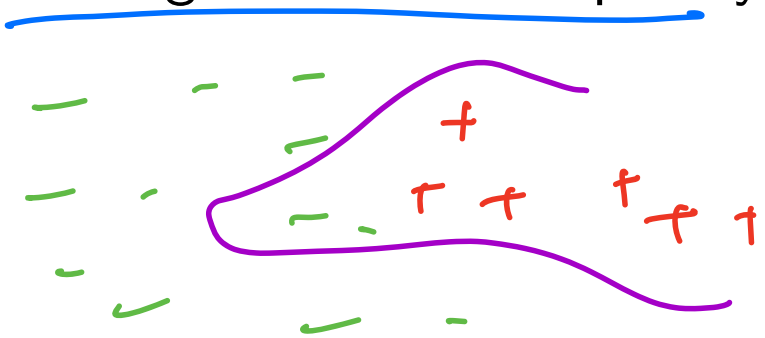
1. Both are interpretable in different ways
2. Decision trees mimick how humans make decisions and are useful in certain contexts - Like medical diagnosis or other places where number of features is not too large

Decision Trees vs Logistic Regression

1. Both are interpretable in different ways
2. Decision trees mimick how humans make decisions and are useful in certain contexts - Like medical diagnosis or other places where number of features is not too large
3. Decision Trees can easily learn non-linear decision boundaries while Logistic Regression learns linear decision boundary

Decision Trees vs Logistic Regression

1. Both are interpretable in different ways
2. Decision trees mimick how humans make decisions and are useful in certain contexts - Like medical diagnosis or other places where number of features is not too large
3. Decision Trees can easily learn non-linear decision boundaries while Logistic Regression learns linear decision boundary
4. Decision Tree has a higher model complexity as compared to Logistic Regression



Decision Trees vs Logistic Regression

1. Both are interpretable in different ways
2. Decision trees mimick how humans make decisions and are useful in certain contexts - Like medical diagnosis or other places where number of features is not too large
3. Decision Trees can easily learn non-linear decision boundaries while Logistic Regression learns linear decision boundary
4. Decision Tree has a higher model complexity as compared to Logistic Regression
5. Logistic Regerssion is less prone to over-fitting than Decision Trees with large number of features

Pitfalls of Decision Trees

1. **Overfitting**

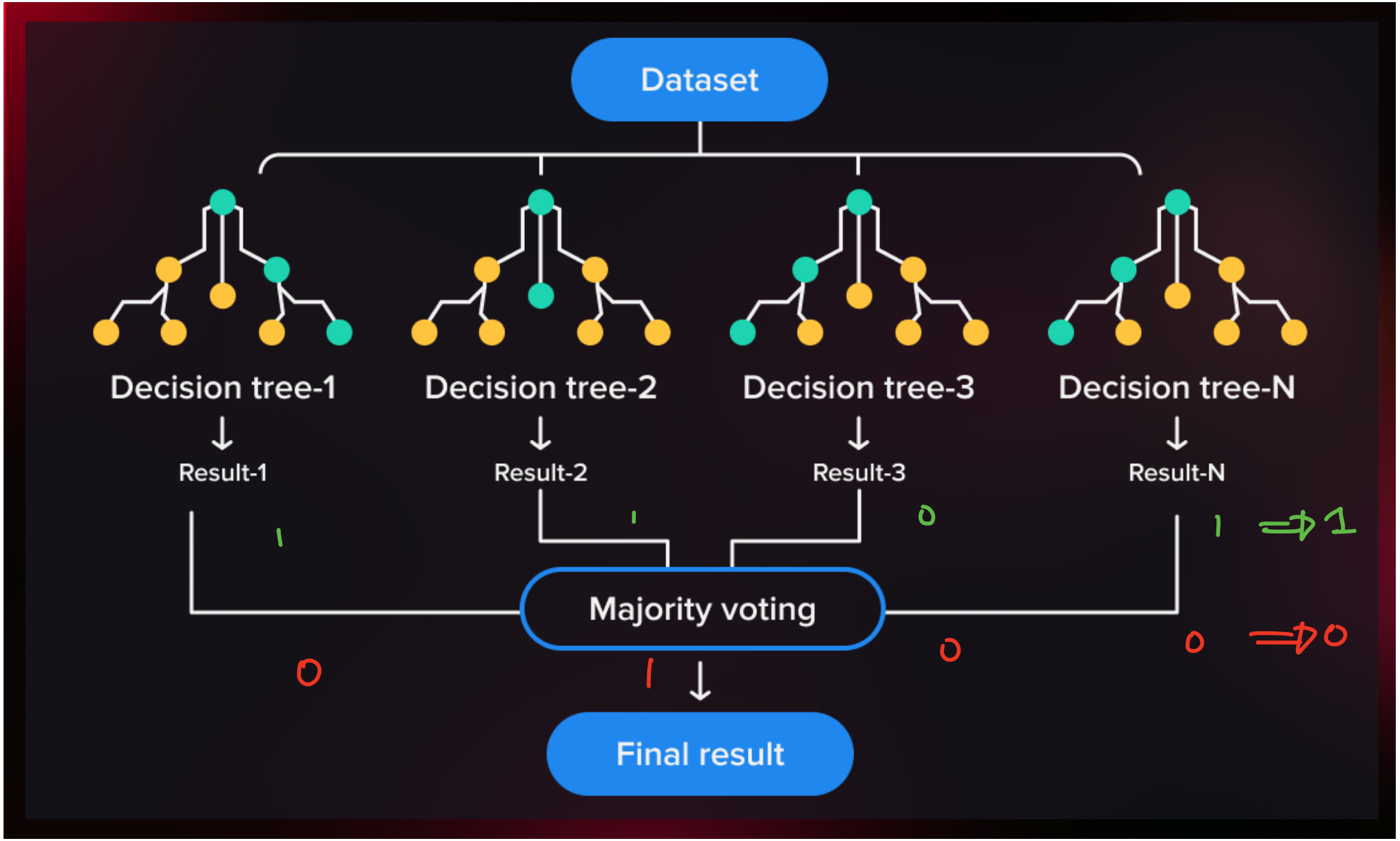
Pitfalls of Decision Trees

1. **Overfitting**
2. **Feature Engineering**

Pitfalls of Decision Trees

1. **Overfitting**
2. **Feature Engineering**
3. **Not suitable for Regression**

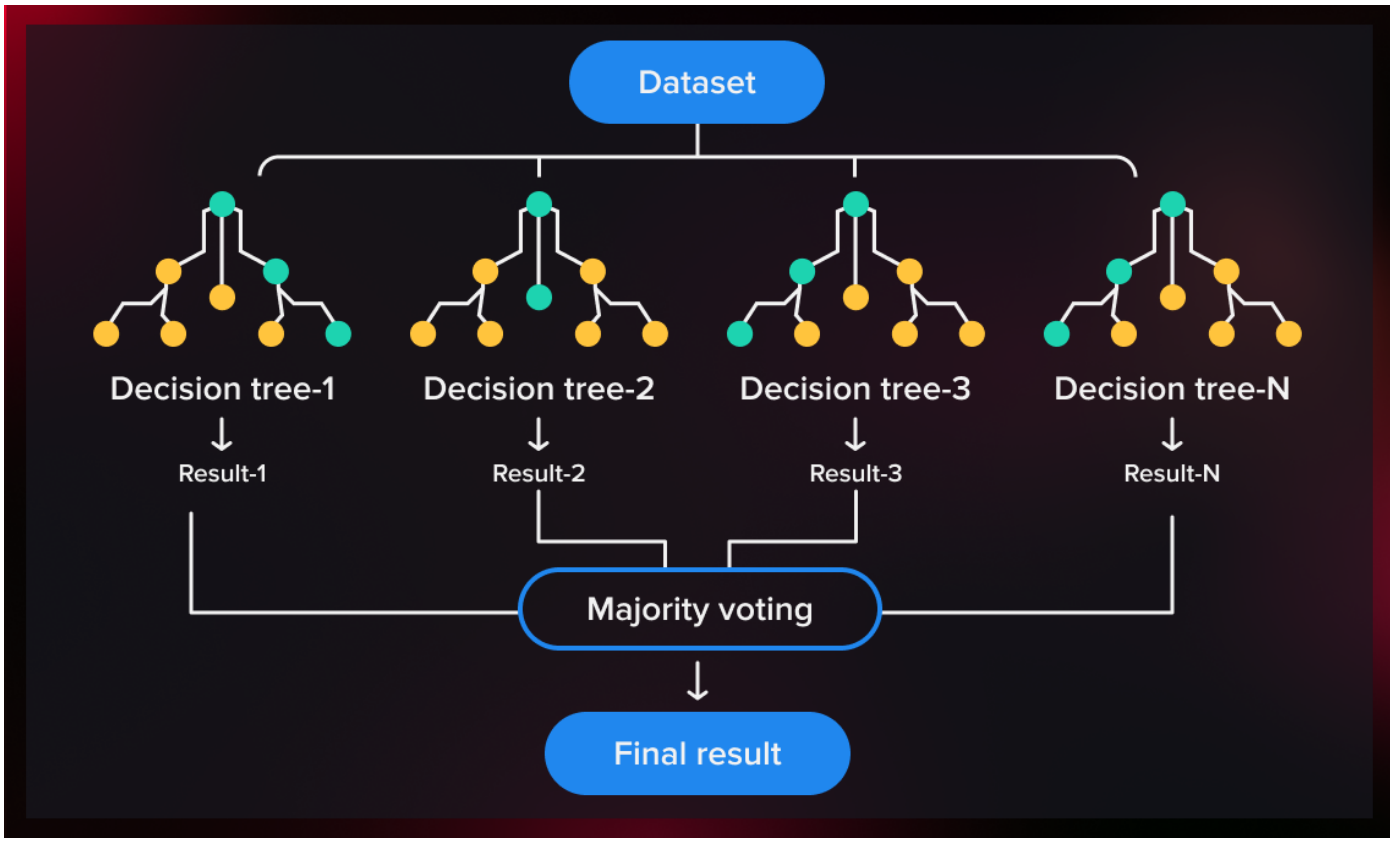
Overcoming pitfalls of Decision Trees - Random Forests



Random Forests Introduction

Random Forests

A **Random Forest** is a collection of **T Decision Trees**. Each decision tree casts a "vote" for a prediciton and the ensemble predicts the majority vote of all of its trees.



Ensemble Methods

RF is an instance of Ensemble methods for Trees

Instead of switching to a brand new type of model that is more powerful than trees, what if we instead tried to make the tree into a more powerful model.

What if we could combine many weaker models in such a way to make a more powerful model?

A **model ensemble** is a collection of (generally weak) models that are combined in such a way to create a more powerful model.

There are two common ways this is done with trees

- Random Forest (Bagging) → over-fitting issues
- AdaBoost (Boosting) → Improving Performance

Look at mistakes of a model & work on it!

Improving a classifier!

Majority Ensemble Classifier

Let's say that there are N different binary classifiers for a data set. Each classifier has a probability $\alpha > 0.5$ of predicting the right label for a data point. Let C_i be the prediction of classifier i (either 0 or 1). Consider a new classifier defined as follows: Let $Y = \sum_{i=1}^{N} C_i/N$. Let,

$$W = \begin{cases} 1 & \text{if } Y > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

So W is a new classifier, constructed out of the N classifiers. Assume that the predictions of each classifier, C_i are independent RVs. How good would the classifier W be for large N?

Ensemble Experiments!

Assume Independence of classifiers!

As N increases, what happens to the accuracy of the **Ensemble Classifier** W? We can simulate this!

Ensemble Experiments!

Example for our experiment

As an example, Consider 3 classifiers C_1, C_2, C_3 that have a probability 0.6 of matching up with the ground truth. I.e. classification accuracy on validation/test is roughly 60%!!!

Truth	C_1	C_2	C_3	Y	W	Match
1	1	0	1	0.67	1	True
1	1	1	1	1	1	True
1	0	0	1	0.33	0	False
0	0	0	0	0	0	True
1	1	1	0	0.67	1	True
1	1	1	1	1	1	True
1	1	0	1	0.67	1	True
0	0	0	1	0.33	0	True
0	0	1	1	0.67	1	False
0	1	0	0	0.33	0	True

avg C_i perf = 0.6 on ACC

Ensemble perf on ACC = ? Sur.

Ensemble Experiments

Notebook

Let's look at a Python notebook on this!

Ensemble Experiments

Notebook

Let's look at a Python notebook on this!

Independence Assumption

So **assuming** classifiers are independent and we build a majority classifier, W from $C_1, C_2, \ldots, C_K$, each of which is a **weak learner** - Then as $K \to \infty$, classification accuracy of $W \to 1$.

Ensemble Experiments

Notebook

Let's look at a Python notebook on this!

Independence Assumption

So **assuming** classifiers are independent and we build a majority classifier, W from $C_1, C_2, \ldots, C_K$, each of which is a **weak learner** - Then as $K \to \infty$, classification accuracy of $W \to 1$.

Assume classifiers are totally dependent on one another

What would be the classification accuracy of W?

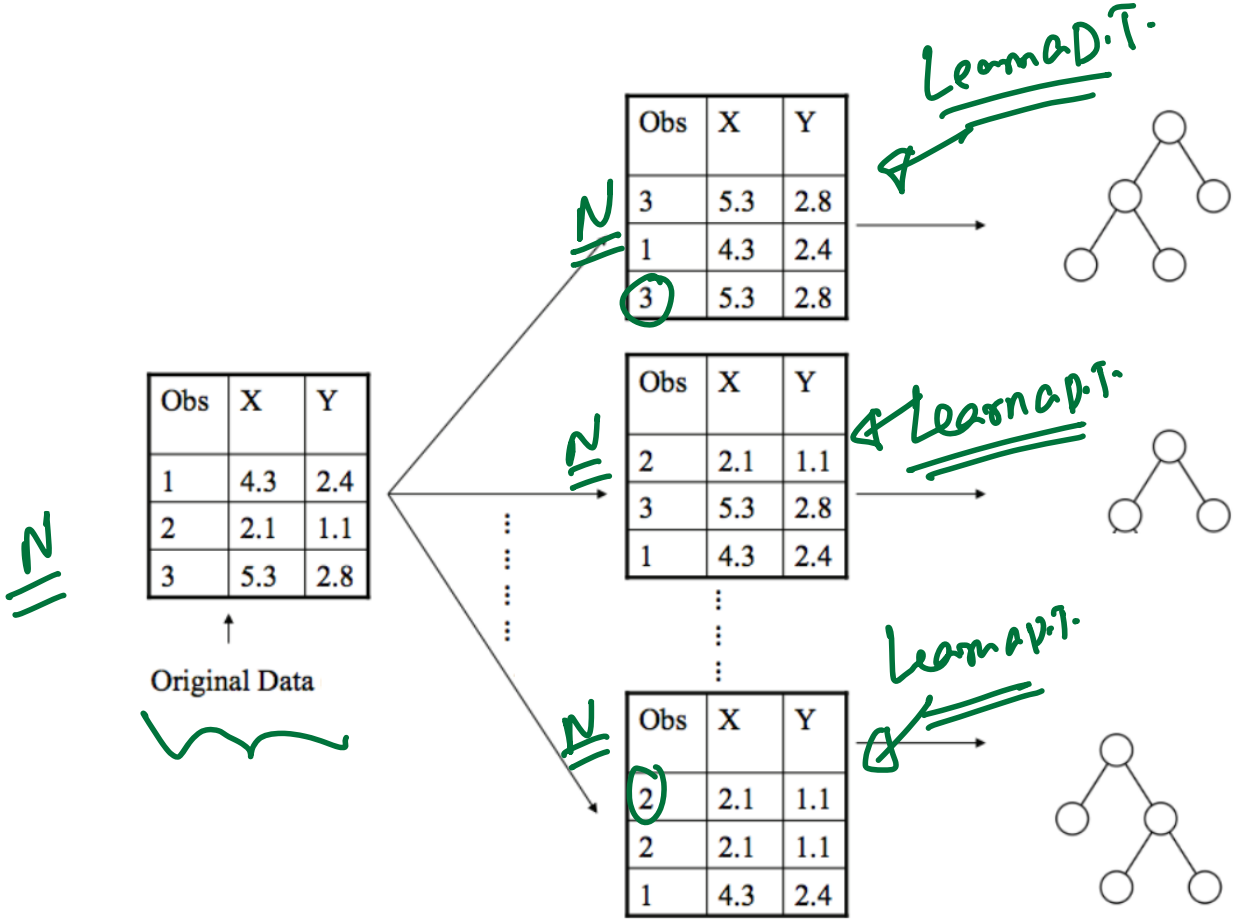
$$W = \begin{cases} 1 \text{ if } \frac{\sum C_i}{N} > 0.5 \\ 0 \text{ otherwise} \end{cases}$$

In practice:- Ensemble models are based on classifiers C_i that have some dependency between them.

Bagging for Trees

If I just have one dataset, how could I learn more than one tree?

Solve this with **bootstrapping**! Can create many similar datasets by randomly sampling *with replacement*.



Random Forest Algorithm

Training

1) Create T sets of data by sampling with replacement to build T decision trees.
2) Each data set also randomly samples features (this is key to improving performance of Random forest)
3) Train T trees in parallel and this makes up the training of Random Forest!

Can optimize "train time" & "inference time" through parallelization across the trees

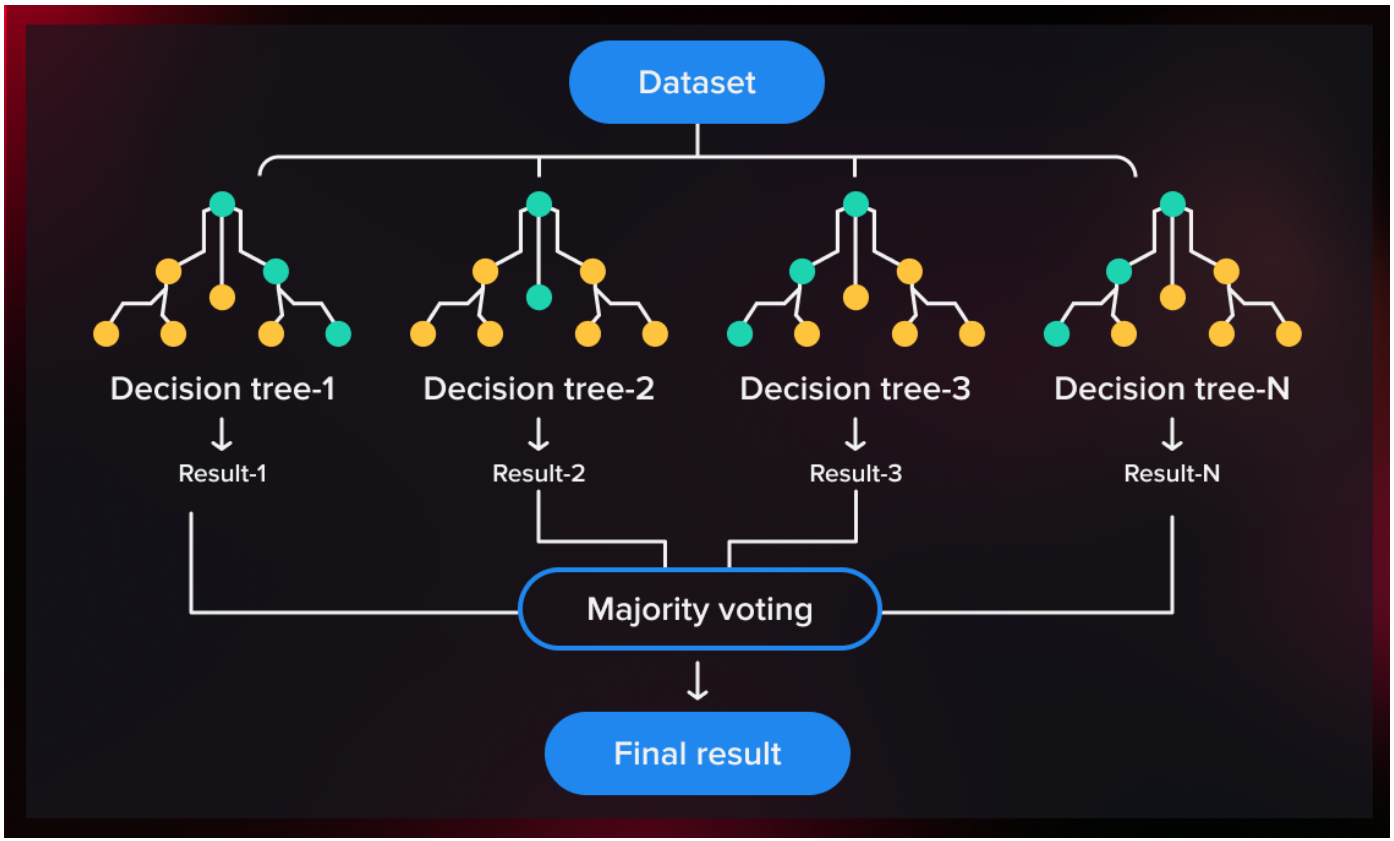
Prediction

Random Forest prediction is a majority classifier on the T trees! So predict the majority class of the T trees' predictions.

Random Forests

Random Forests

A **Random Forest** is a collection of **T Decision Trees**. Each decision tree casts a "vote" for a prediciton and the ensemble predicts the majority vote of all of its trees.



Random Forest in Msft Kinect!

Microsoft used Random Forests in their Kinect system to identify the "pose" of a person from the depth camera.

Real-Time Human Pose Recognition in Parts from Single Depth Images

Jamie Shotton Andrew Fitzgibbon Mat Cook Toby Sharp Mark Finocchio
Richard Moore Alex Kipman Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

Abstract

We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that maps the difficult pose estimation problem into a simpler per-pixel classification problem. Our large and highly varied training dataset allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. *Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes.*

The system runs at 200 frames per second on consumer hardware. Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton nearest

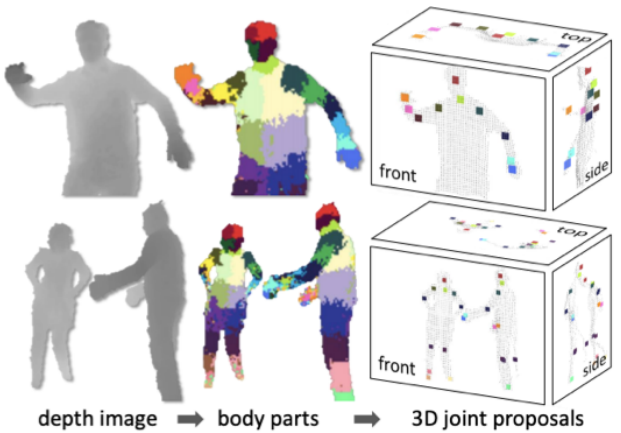


Figure 1. **Overview.** From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.

Learning and Hyper-parameters in Random Forests

ICE #1 (2 mins)

What are hyper-parameters in the random forest?

a. Number of trees in the random forest
b. The order of features to split each tree in the random forest
c. Number of tress and number of features per tree
d. Number of features per tree and weight of each feature

Decision Trees vs Random Forests

a. DT → Single Tree, RF → Collection of Trees

b. DT → overfits if tree is deep, RF → Shallow Trees are fine for performance

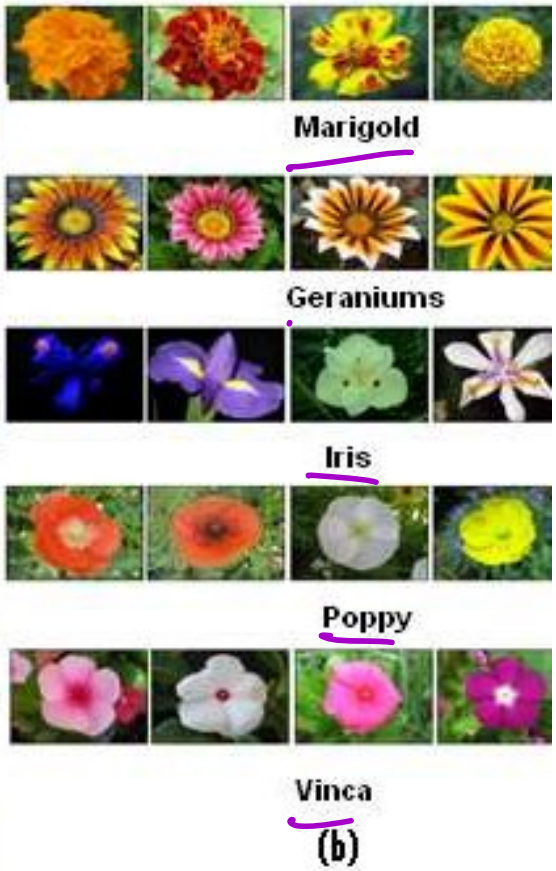
c. DT → Looks at all features but is greedy in selection, RF → Splits features & has better Coverage

d. ~~[illegible]~~

RF → 1) Takes care of overfitting of DTs

2) Can boost performance with feature sampling & ensembling

Multi-class classification



Flower classification

Classifying flowers across different species

Use image features to differentiate and classify!

Pixel values/color, convolutions

Flower classification

Classifying flowers across different species

Use image features to differentiate and classify!

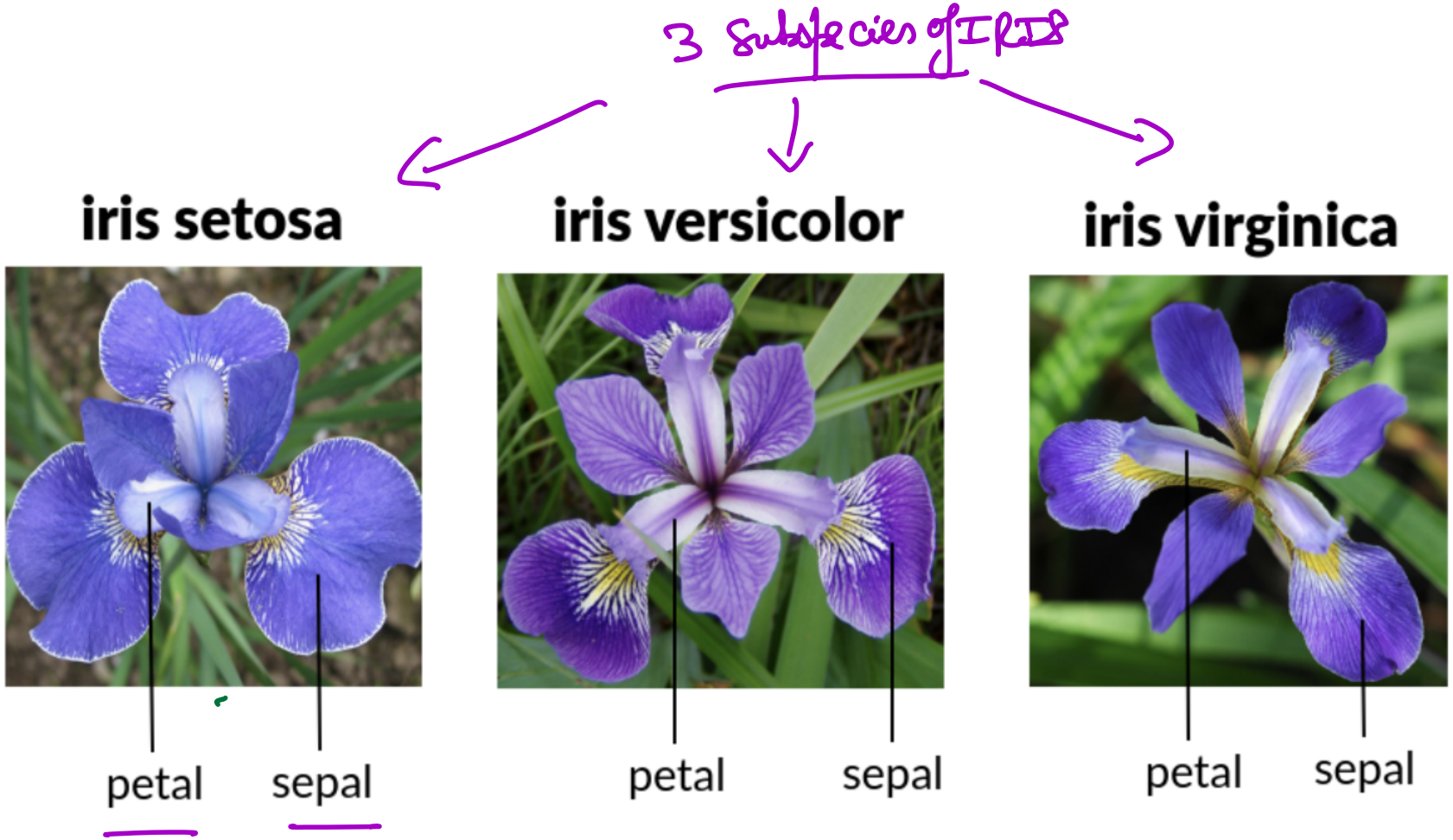
More frequent

Classify sub-types of a given flower!

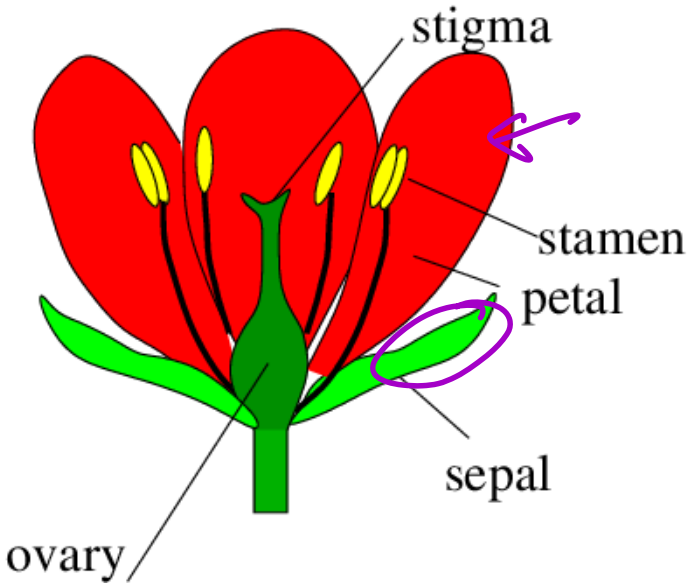
Some special features can help!

Special case

Example: IRIS flowers

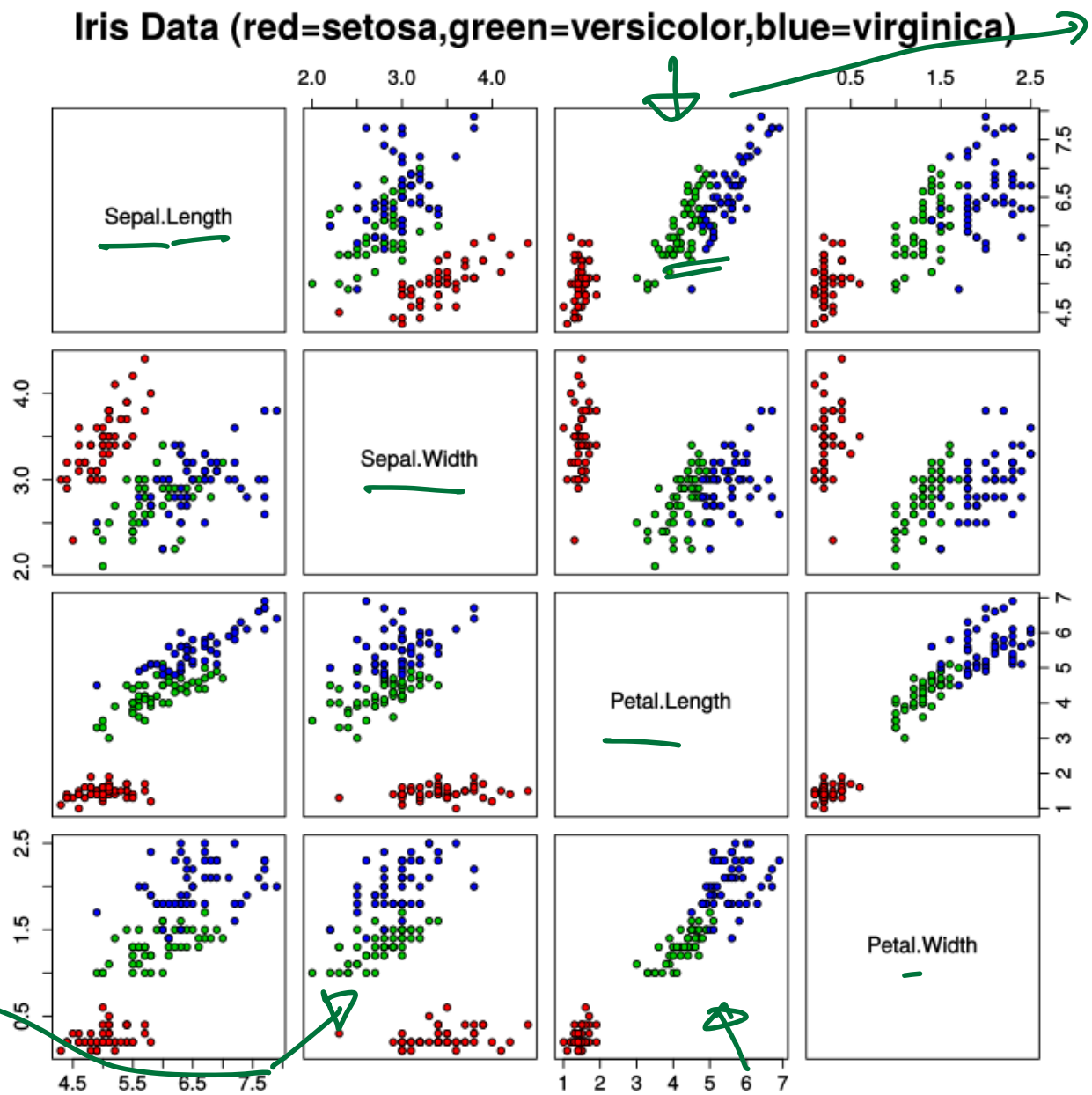


Sepal/Petal of a flower



Sepal/Petal for IRIS vs Fuschia

Pair-wise correlations in sub-types of IRIS



Multi-class classification

Binary classification

Choose between two classes. So needed a way to compute probability for one of the classes and the probability for other class follows.
$P(x^i \in \text{ Negative Class}) = 1 - P(x^i \in \text{ Positive Class}) = 1 - \hat{y}_i$)!!

Multi-class classification

Binary classification

Choose between two classes. So needed a way to compute probability for one of the classes and the probability for other class follows.
$P(x^i \in \text{ Negative Class}) = 1 - P(x^i \in \text{ Positive Class}) = 1 - \hat{y}_i$)!!

Multi-class classification

For multi-class classification - Say $K = 5$ classes, we need to generate probabilities for 5 classes such that sum of these probabilities is equal to 1!

ICE #2

Multi-class classification

Consider a 10-class flower classification problem, where we want to predict one of 10 flower classes given an image of a flower. What's the appropriate dimension of y_i and $\hat{y}_i$ so we can build a loss function over it and learn the parameters?

a. 1 & 10
b. 10 & 10
c. 10 & 1
d. 1 & 1

Multi-class classification

Recap: Sigmoid function to generate probability for binary-class

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$P(x^i \in \text{ Positive Class}) = \hat{y}_i = \sigma(\hat{w}^T x^i)$$

z - Score

$\sigma(z) = \frac{1}{1+e^{-z}}$

↳ Prob(true class)

Multi-class classification

Recap: Sigmoid function to generate probability for binary-class

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$P(x^i \in \text{ Positive Class}) = \hat{y}_i = \sigma(\hat{w}^T x^i)$

Extension to multi-class: Soft-max function

Logits: $[z_1, z_2, \ldots, z_k]$

$$\sigma(z)_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

$\hat{y} = \left[\frac{e^{z_1}}{\sum e^{z_i}}, \frac{e^{z_2}}{\sum e^{z_i}}, \ldots\right]$

Note: $\sigma(z)$ is now a vector, where $\sigma(z)_k$ gives us the probability for the *kth* class!!

Multi-class classification

Probability of kth class

Therefore,

$$\hat{y}_k^i = \sigma(z)_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

Multi-class classification

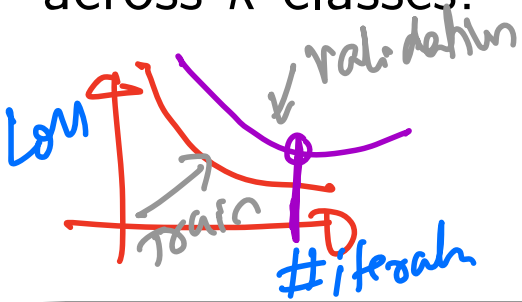
Probability of kth class

Therefore,

$$\hat{y}_k^i = \sigma(z)_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

Loss Function

Like the **binary cross-entropy loss** for binary classification, we have the **cross entropy loss** for multi-class classification. Let $y^i \in \mathcal{R}^k$ be the label for *ith* data point and $\hat{y}^i$ be the prediction for *ith* data point. Note y^i is a one-hot encoding of the true class and $\hat{y}^i$ is a probability distribution across k classes!



$$L(w; x, y) = \frac{1}{N} \sum_{i=1}^{N} \left\{ - \sum_{k=1}^{K} y_k^i \log \hat{y}_k^i \right\}$$

Avg. error across data points (N)

$y^i = [0\,0\,1\,0 \cdots 0]$

$\hat{y}^i = [0.7\,a_1\,a_2 \cdots]$

Categorical cross entropy in multi-class

Logits $\rightarrow$ Probability of a Class

Optimizing cross-entropy

Logistic Regression for multiple-classes

Naturally extends here. Still a convex function!

Optimizing cross-entropy

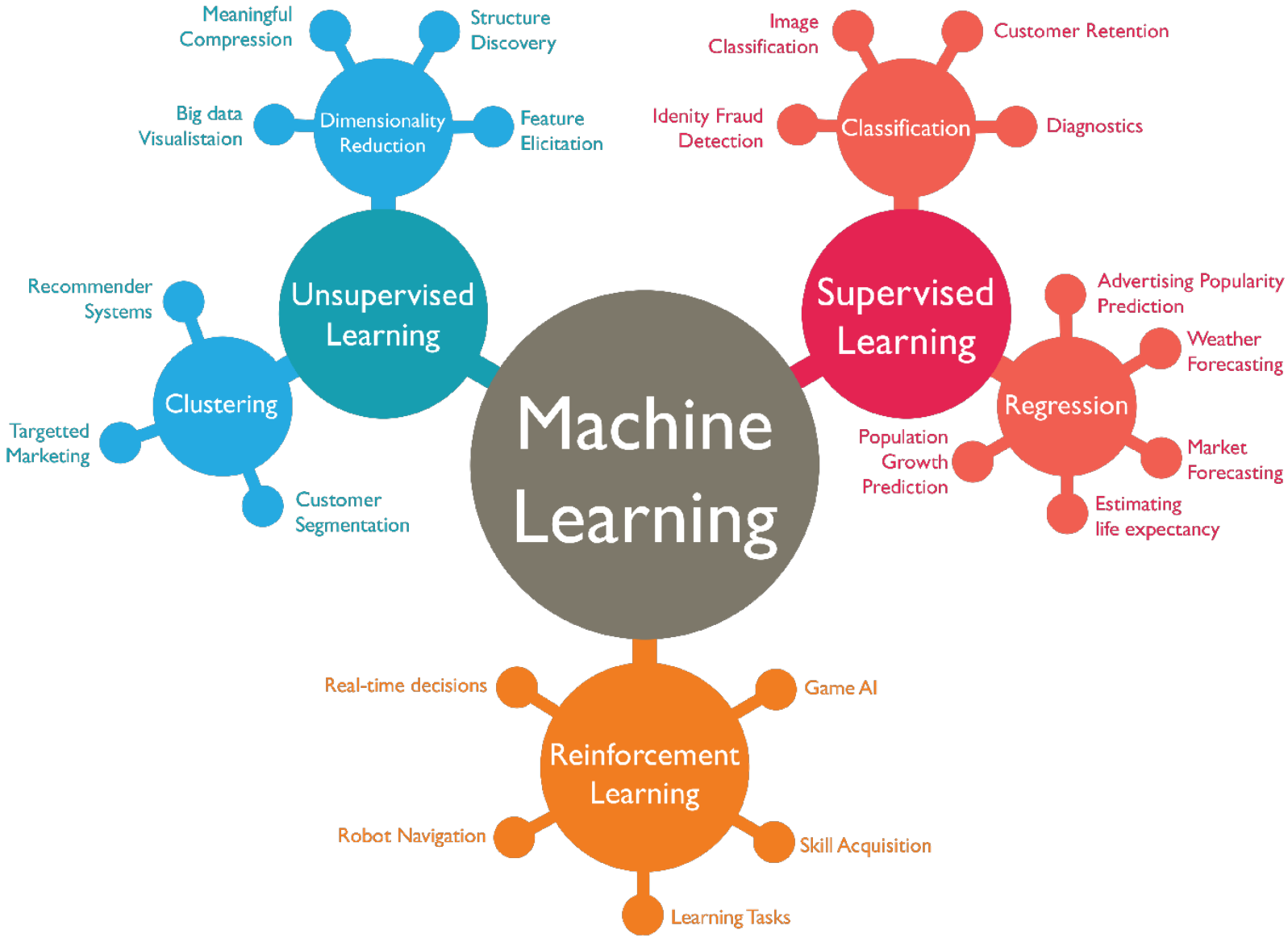
Logistic Regression for multiple-classes

Naturally extends here. Still a convex function!

Multi-class vs Multi-label classification

What's the distinction?

Un-supervised learning



The clustering problem

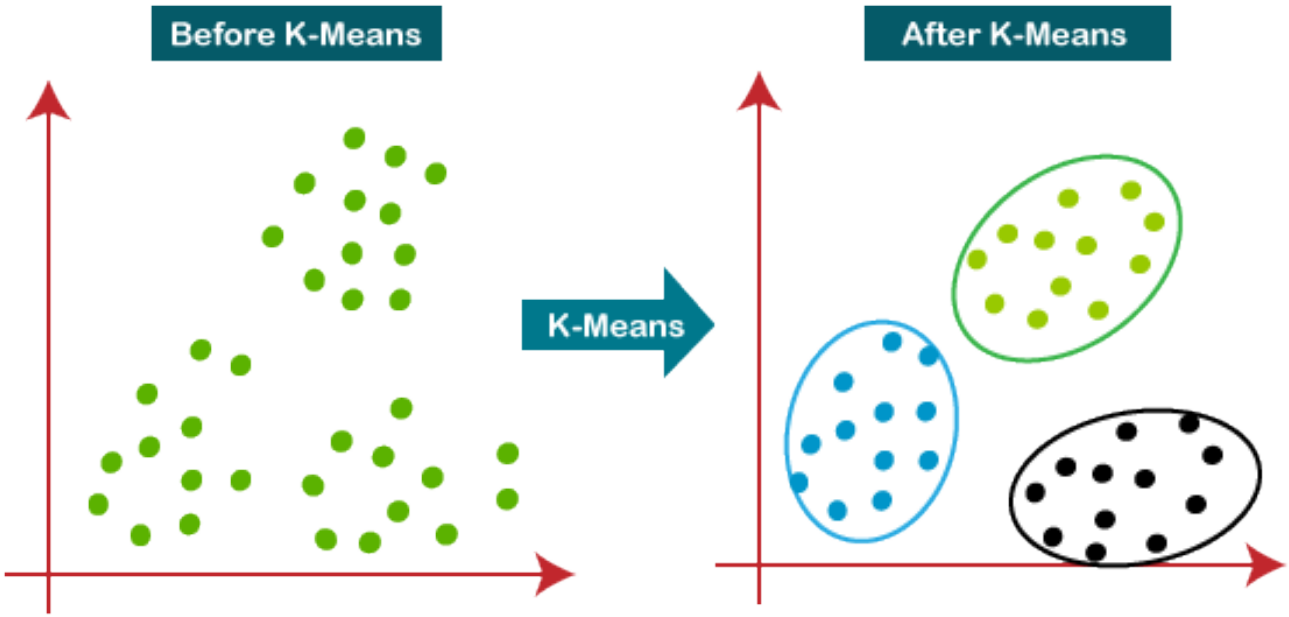
Clustering vs Classification

Difference

In the classification problem, you are given (x^i, y_i) - I.e. both the data point i and its true label y_i for training purposes. Example - a flower i and its label (flower type). Whereas in clustering problem, you are just given the data points, i.e. x^i. However, you still want to break up the data points into clusters - where each cluster has relatively similar data points.

Digits Clustering

Clustering of data points



Clustering for News

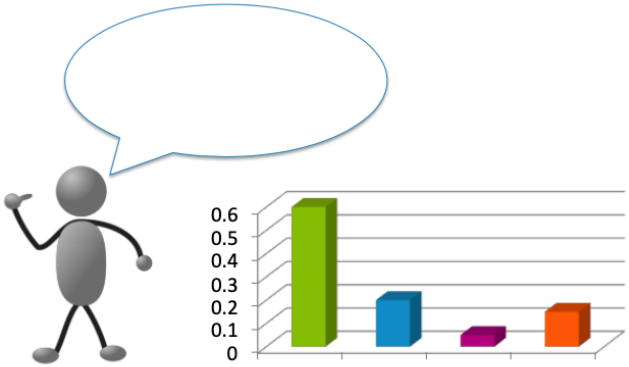
SPORTS

WORLD NEWS

Clustering for News

User preferences are important to learn, but can be challenging to do in practice.

- People have complicated preferences
- Topics aren't always clearly defined




Cluster 1

Cluster 2


Cluster 3

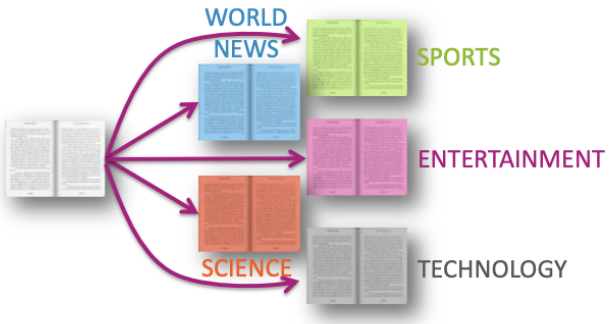

Cluster 4

Use feedback to learn user preferences over topics

Clustering for News

What if the labels are known? Given labeled training data

Can do multi-class classification methods to predict label



Clustering Basics

- In many real world contexts, there aren't clearly defined labels so we won't be able to do classification
- We will need to come up with methods that uncover structure from the (unlabeled) input data X.
- **Clustering** is an automatic process of trying to find related groups within the given dataset.

$Input: x_1, x_2, \ldots, x_n$

$Output: z_1, z_2, \ldots, z_n$

Clustering Basics

In their simplest form, a **cluster** is defined by

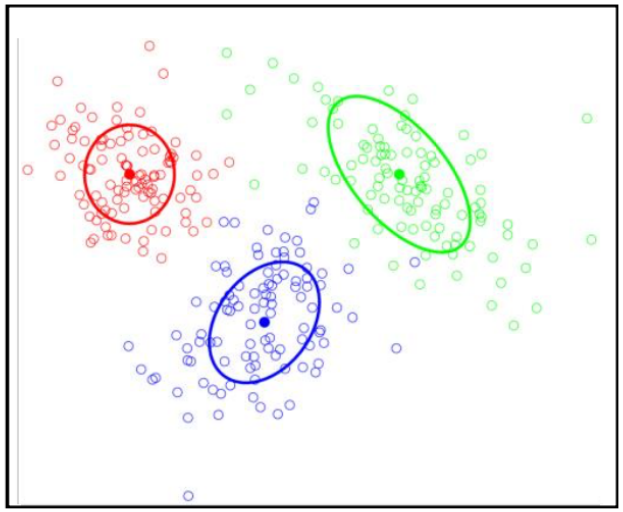
- The location of its center (**centroid**)
- Shape and size of its **spread**

Clustering is the process of finding these clusters and **assigning** each example to a particular cluster.

- x_i gets assigned $z_i \in [1, 2, \ldots, k]$
- Usually based on closest centroid

Will define some kind of score for a clustering that determines how good the assignments are

- Based on distance of assigned examples to each cluster



Distance typically used

Euclidean Distance

Distance between two points, x_1, x_1 is given by:

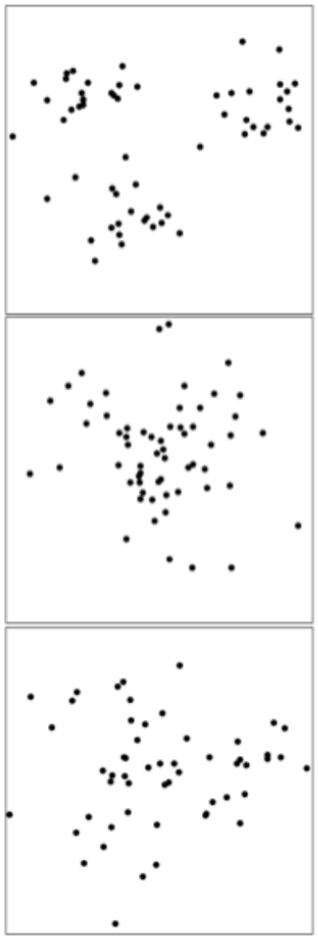
$$\|x_1 - x_2\|_2$$

Clustering on different Data sets

Clustering is easy when distance captures the clusters

Ground Truth (not visible)

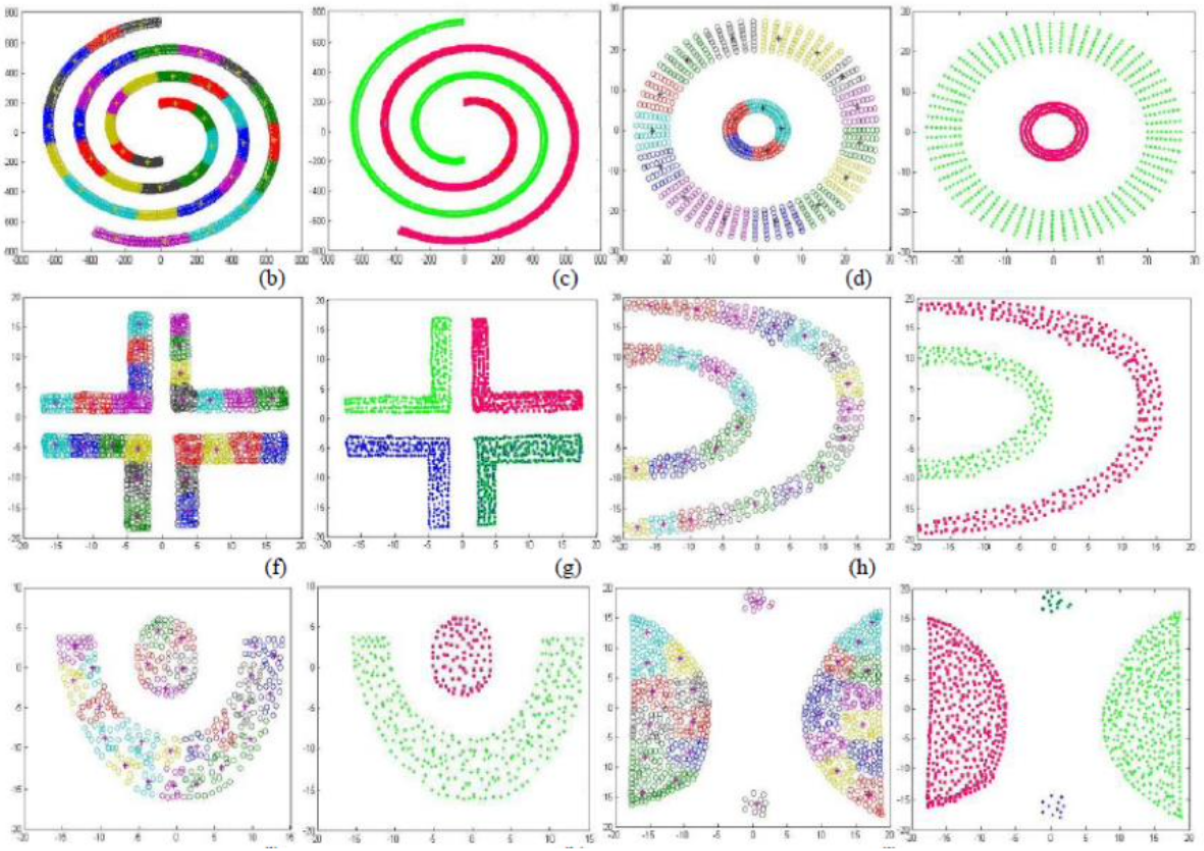
Given Data



Clustering - Hard cases

There are many clusters that are harder to learn with this setup

- Distance does not determine clusters



k-means

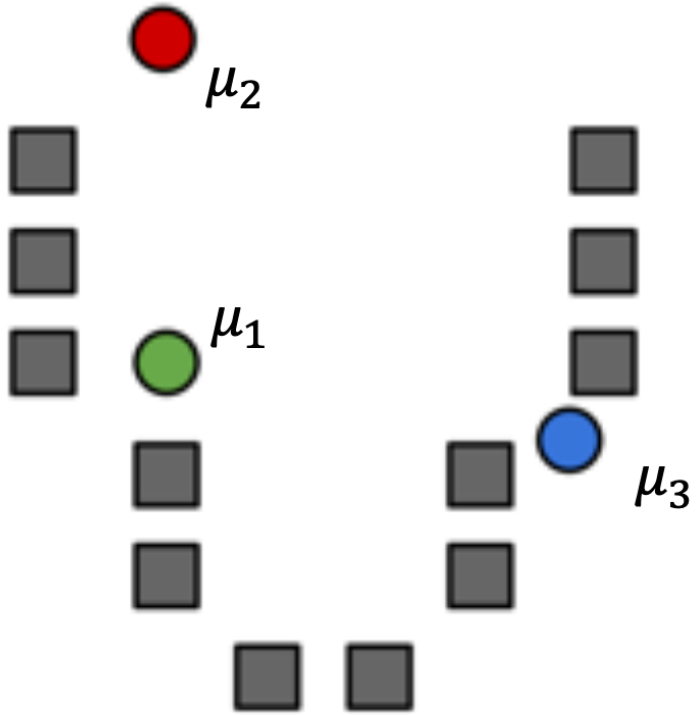
Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4: **expectation:** Assign each point to its closest centroid.
5: **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

k-means Clustering

Start by choosing the initial cluster centroids

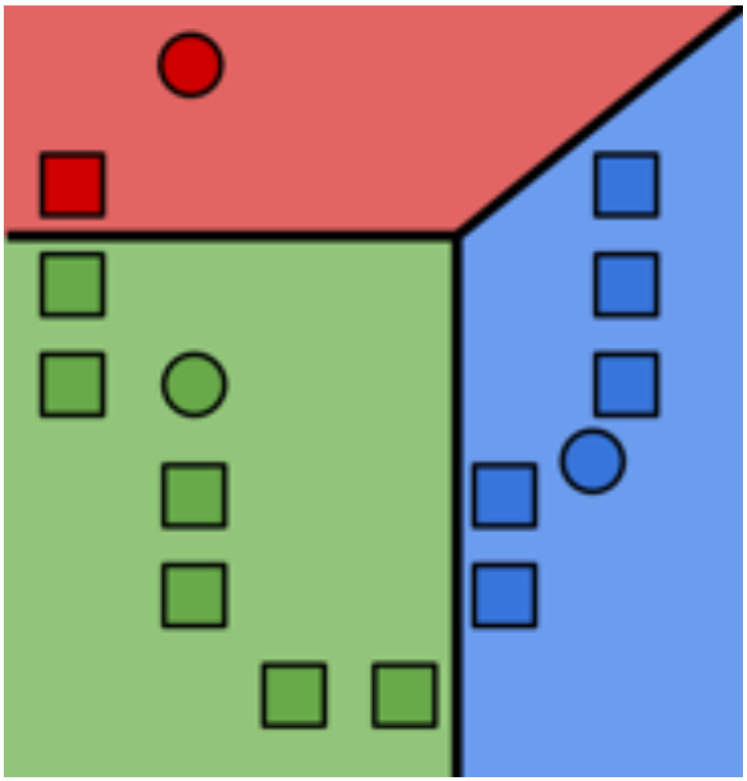
- A common default choice is to choose centroids at random
- Will see later that there are smarter ways of initializing



k-means Clustering

Assign each example to its closest cluster centroid

$$z_i \leftarrow \underset{j \in [k]}{\operatorname{argmin}} \left|\left|\mu_j - x_i\right|\right|^2$$

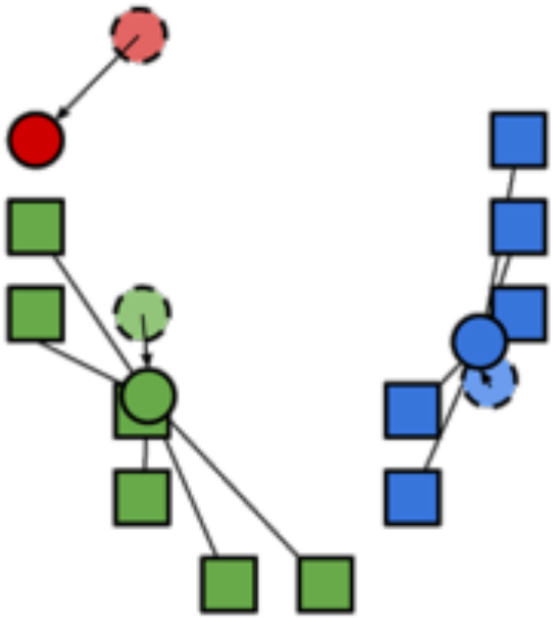


k-means Clustering

Update the centroids to be the mean of all the points assigned to that cluster.

$$\mu_j \leftarrow \frac{1}{n_j} \sum_{i:z_i=j} x_i$$

Computes center of mass for cluster!



k-means Clustering

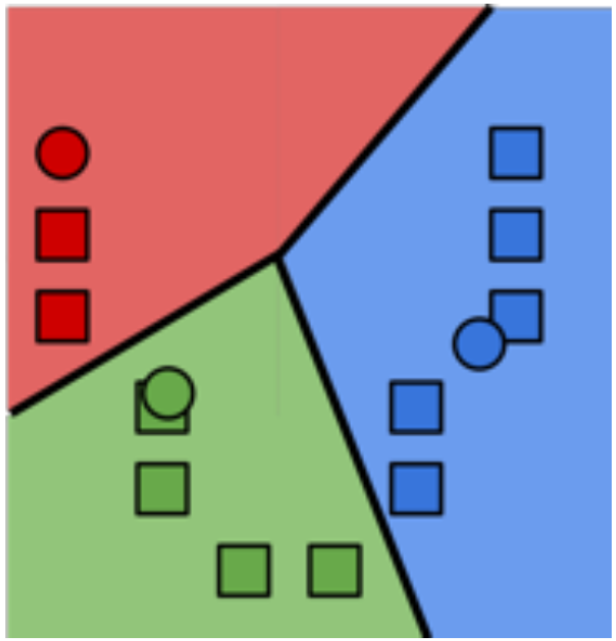
Repeat Steps 1 and 2 until convergence

Will it converge? Yes! Stop when

- Cluster assignments haven't changed
- Some number of max iterations have been passed

What will it converge to?

- Global optimum
- Local optimum
- Neither



Improving kMeans?

kMeans++

Next lecture?

Clustering in 2 dimensions - tSNE!

Data Visualization: Stochastic Neighborhood Embeddings (SNE)!

SNE

Similarity to Probabilities

Let $x_1, x_2, \ldots$ represent features of the data in their original dimensions (e.g. images).

$$p_{j|i} = \frac{e^{-\|x_i - x_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|x_i - x_k\|_2^2 / 2\sigma_i^2}}$$

Low-dimensional embedding Probabilities

Let $y_1, y_2, \ldots$ represent features of the data in lower (embedded) dimensions (e.g. 2 dimensions).

$$q_{j|i} = \frac{e^{-\|y_i - y_j\|_2^2 / 2\sigma_i^2}}{\sum_{k \neq i} e^{-\|y_i - y_k\|_2^2 / 2\sigma_i^2}}$$

Estimating low-dimensional embeddings in SNE

A similarity measure for Probabilities - KL Divergence

$$KL(p||q) = \sum_{i=1}^{d} p_i \log \frac{p_i}{q_i}$$

Estimating low-dimensional embeddings in SNE

A similarity measure for Probabilities - KL Divergence

$$KL(p||q) = \sum_{i=1}^{d} p_i \log \frac{p_i}{q_i}$$

Loss function

$$L(y_1, y_2, \ldots, y_N) = \sum_{i=1}^{N} KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

Gradient and GD

Gradient

$$\frac{\partial L}{\partial y_i} = 2 \sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Gradient and GD

Gradient

$$\frac{\partial L}{\partial y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

GD

$$y^{t+1} = y^t - \eta \frac{\partial L}{\partial y}(y^t)$$

MNIST digits data set

MNIST tSNE embeddings