Dimensionality reduction & clustering

Lecture 16

Changho Suh

October 5, 2021

Recap: DTs

A decision-based model of the tree structure.

Training algorithm: **CART**

Hyperparameters:

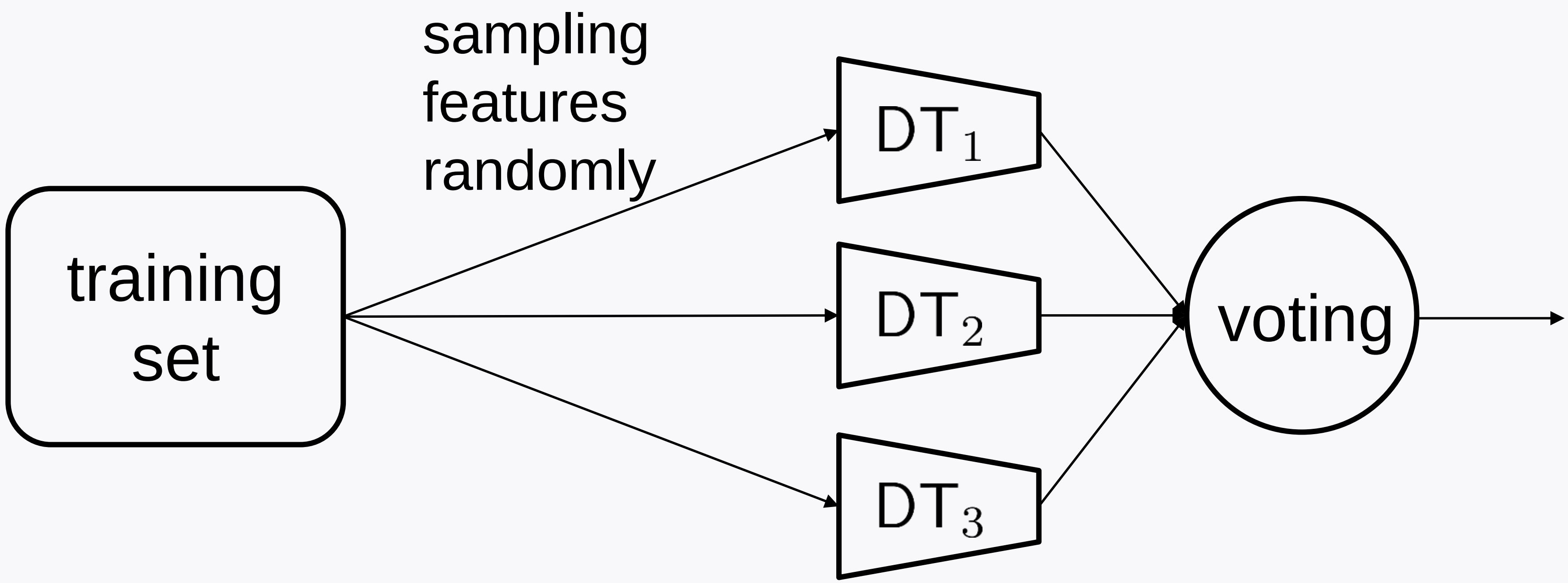
“max_depth” “min_samples_split”

“max_leaf_nodes” “min_samples_leaf”

Challenge: Sensitive to small variations in training data

Recap: RFs

An ensemble of DTs, each trained on the random subspace method



Hyperparameters: **"max_features"** **"n_estimators"**

A measure for *interpretation*: **Feature importance**

Recap: Coding for DTs and RFs

```
from    sklearn.tree    import DecisionTreeClassifier
from    sklearn.tree    import plot_tree

tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X_train,y_train)
plot_tree(tree_clf)
```

```
from    sklearn.ensemble      import RandomForestClassifier

rnd_clf = RandomForestClassifier(n_estimators=500,max_leaf_nodes=16)
rnd_clf.fit(X_train,y_train)
feature_importances = rnd_clf.feature_importances_
```

Recap: Coding for hyperparameter search

```
from    sklearn.model_selection       import GridSearchCV
forest_clf=RandomForestClassifier(max_depth=2)
param_grid={'n_estimators':[3,10,100,500],'max_features':[1,2,3,4]}
grid_search=GridSearchCV(forest_clf,param_grid,cv=5,scoring='accuracy')
grid_search.fit(X_train,y_train)

From    sklearn.model_selection       import RandomizedSearchCV
param_distributions={'n_estimators':range(1,500),'max_features':range(1,5)}
randomized_search=RandomizedSearchCV(forest_clf,param_distributions,
                                     cv=5,n_iter=50,scoring='accuracy')
randomized_search.fit(X_train,y_train)

randomized_search.best_params_
randomized_search.best_estimator_
randomized_search.best_estimator_.feature_importances_
```

Question

So far: Learned about **DNNs, CNNs, RNNs** & **RFs.**

What if still unsatisfactory performances?

This may be due to:

1. $n \gg m \longleftarrow$ # of examples and/or
   (↖ data dimension)

2. data distribution is pretty wide.
   i.e., data characteristics are quite distinct across examples.

Techiques for addressing such scenarios

Scenario 1: $n \gg m$

⇨ dimensionality reduction

Scenario 2: data distribution is pretty wide.

⇨ clustering

Outline of today's lecture

Will study dimensionality reduction & clustering:

1. Explore the most popular dimensiona reduction technique: Principal Component Analysis (**PCA**)

2. Investigate another prominent technique:

   t-distributed Stochastic Neighbor Embedding (**t-SNE**)

3. Study clustering methods.

Focus of Lecture 16

Will study dimensionality reduction & clustering:

1. Explore the most popular dimensiona reduction technique: Principal Component Analysis (**PCA**)

2. Investigate another prominent technique:

   t-distributed Stochastic Neighbor Embedding (**t-SNE**)

3. Study clustering methods.

Dimensionality reduction

Reducing # of features in data by obtaining a set of principal components

Three major roles:

1. Improve generalization performance
2. Speed up training
3. Data visualization

Principal Component Analysis (PCA)

The most popular **dimensionality reduction** technique!

Karl Pearson 1901

PCA in words

A technique that does the following:

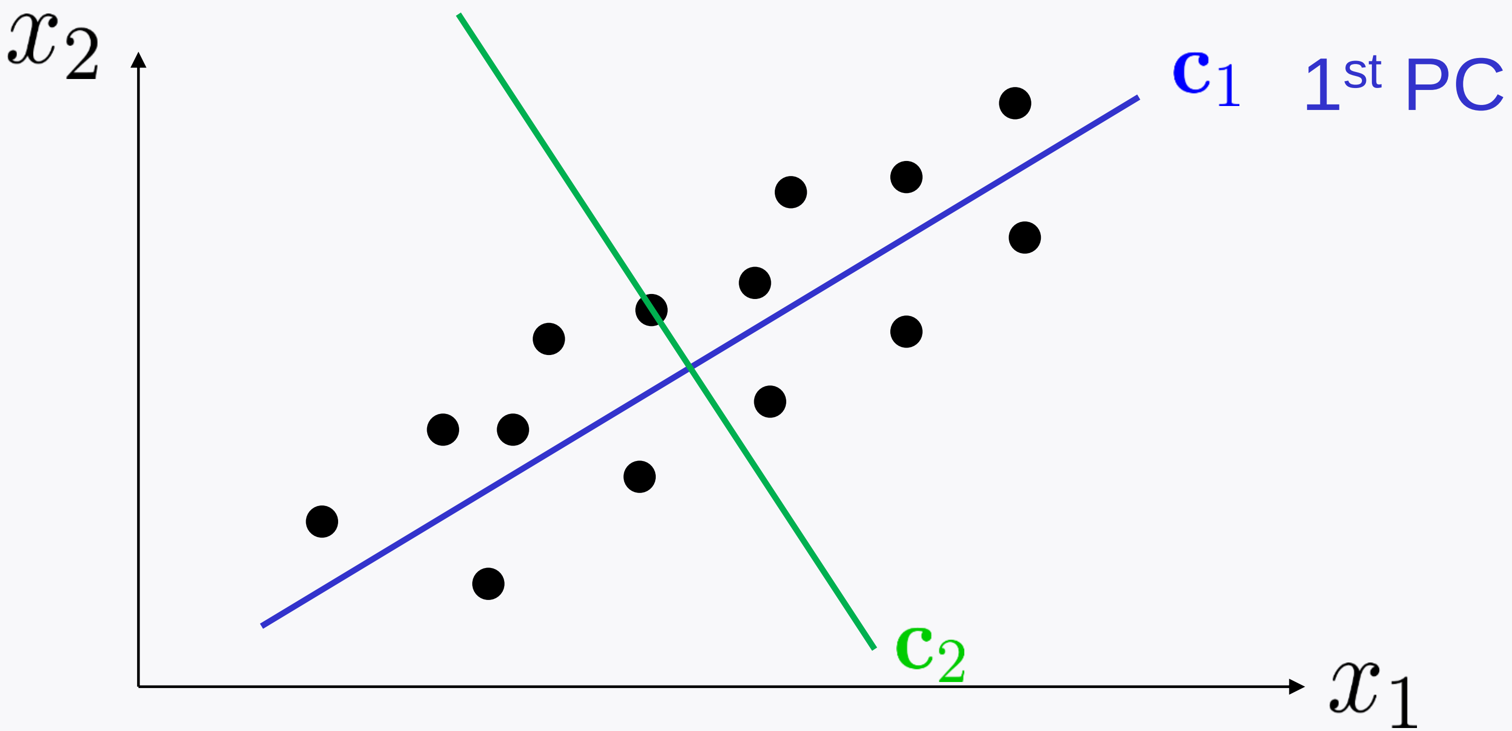
1. Identify a vector along which data points are most *explained*. 1st principal component (PC)

2nd PC

2. Find another vector along which data points are most explained *subject to orthogonality to the 1st one*

3. Repeat until reaching desired # of vectors.

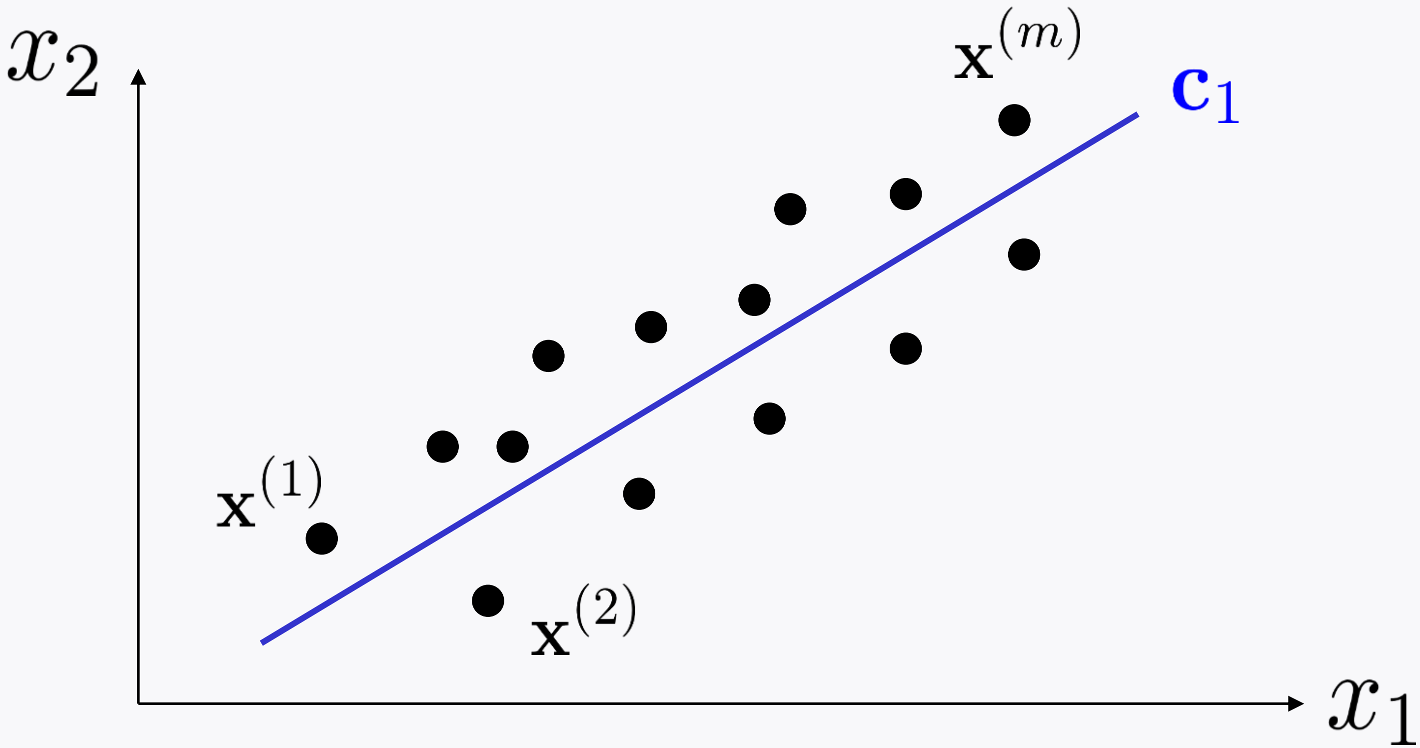
PCA in picture



Find a vector along which data points are most *spread*.

Similarly find another vector yet orthogonal to 1st one.

How to find c_1?



$$\mathbf{c}_1 := \arg\max_{\|\mathbf{v}\|=1} \sum_{i=1}^{m} (\mathbf{x}^{(i)T}\mathbf{v})^2$$

How to find $\mathbf{c}_1$?

$$\mathbf{c}_1 := \arg\max_{\|\mathbf{v}\|=1} \underbrace{\sum_{i=1}^{m} (\mathbf{x}^{(i)T}\mathbf{v})^2}_{\left\| \begin{bmatrix} \mathbf{x}^{(1)T}\mathbf{v} \\ \vdots \\ \mathbf{x}^{(m)T}\mathbf{v} \end{bmatrix} \right\|^2}$$

How to find $\mathbf{c}_1$?

$$\mathbf{c}_1 := \arg\max_{\|\mathbf{v}\|=1} \underbrace{\sum_{i=1}^{m} (\mathbf{x}^{(i)T}\mathbf{v})^2}$$

$$\left\| \underbrace{\begin{bmatrix} \mathbf{x}^{(1)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix}}_{=: \mathbf{X}} \mathbf{v} \right\|^2$$

training data matrix

How to find $\mathbf{c}_1$?

$$\mathbf{c}_1 := \arg\max_{\|\mathbf{v}\|=1} \underbrace{\sum_{i=1}^{m} (\mathbf{x}^{(i)T}\mathbf{v})^2}_{\mathbf{v}^T \underbrace{\mathbf{X}^T\mathbf{X}}_{\text{symmetric matrix}} \mathbf{v}}$$

Eigenvalue decomposition: $\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$

$$\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$$

$$\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$$

How to find $\mathbf{c}_1$?

$$\mathbf{c}_1 := \arg\max_{\|\mathbf{v}\|=1} \underbrace{\sum_{i=1}^{m} (\mathbf{x}^{(i)T}\mathbf{v})^2}$$

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$

$$\mathbf{\Lambda} = \mathsf{diag}(\lambda_1, \ldots, \lambda_n)$$

$$\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$$

$$\mathbf{v}^T\mathbf{X}^T\mathbf{X}\mathbf{v} = \sum_{i=1}^{n} \lambda_i (\mathbf{v}^T\mathbf{v}_i)^2$$

$$\lambda_1 \geq \cdots \geq \lambda_n \qquad \rightarrow \mathbf{v}^* = \mathbf{v}_1$$

How to find $\mathbf{c}_1$?

$$\mathbf{c}_1 := \arg\max_{\|\mathbf{v}\|=1} \sum_{i=1}^{m} (\mathbf{x}^{(i)T}\mathbf{v})^2 \qquad \rightarrow \mathbf{c}_1 = \mathbf{v}_1$$

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T \qquad \mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$$

$\mathbf{c}_1 = \mathbf{v}_1$ (1st eigenvector of $\mathbf{X}^T\mathbf{X}$)

(1st **singular** vector of $\mathbf{X}$)

How to find PCs?

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}^{(1)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \in \mathbf{R}^{m \times n}$$ training data matrix

$$\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \mathbf{v}_n]$$

$$\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$$

"d" number of PCs

$$\mathbf{c}_1 = \mathbf{v}_1$$
$$\mathbf{c}_2 = \mathbf{v}_2$$
$$\mathbf{c}_d = \mathbf{v}_d$$

How to choose *d*?

Choose *d* so as to explain a sufficiently large portion of data (e.g., 95%)

A measure that captures the proportion of data reflected in a principal component:

EVR (explained variance ratio) $\in [0, 1]$

Choose *d* such that: $$\sum_{i=1}^{d} \text{EVR}(\mathbf{c}_i) \approx 0.95$$

PCA output?

$$\mathbf{X} := \begin{bmatrix} \mathbf{x}^{(1)T} \\ \vdots \\ \mathbf{x}^{(m)T} \end{bmatrix} \in \mathbf{R}^{m \times n}$$ training data matrix

$$\mathbf{V}_d = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \mathbf{v}_d]$$ principal components matrix

$$\mathbf{x}_{\text{proj}}^{(1)T} = [\mathbf{x}^{(1)T}\mathbf{v}_1, \ldots, \mathbf{x}^{(1)T}\mathbf{v}_d] = \mathbf{x}^{(1)T}\mathbf{V}_d$$

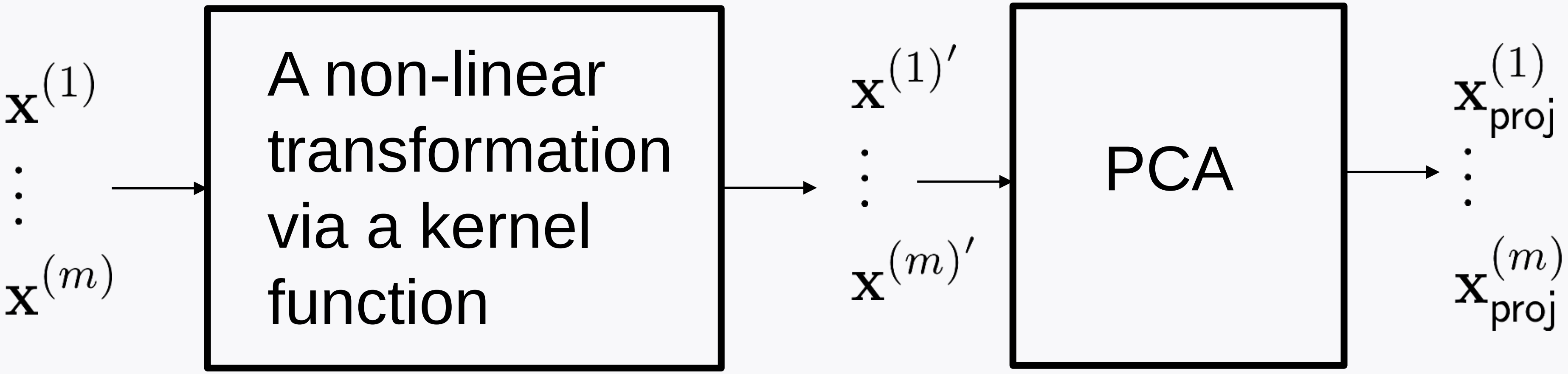
$$\mathbf{X}_{\text{proj}} = \begin{bmatrix} \mathbf{x}_{\text{proj}}^{(1)T} \\ \vdots \\ \mathbf{x}_{\text{proj}}^{(m)T} \end{bmatrix} = \mathbf{X}\mathbf{V}_d$$

Another technique?

Note: PCA is a linear technique.

There is a non-linear version of PCA:

Kernel PCA



Look ahead

Will study another non-linear technique:

t-distributed Stochastic Neighbor Embedding (**t-SNE**)