Introduction to Machine Learning

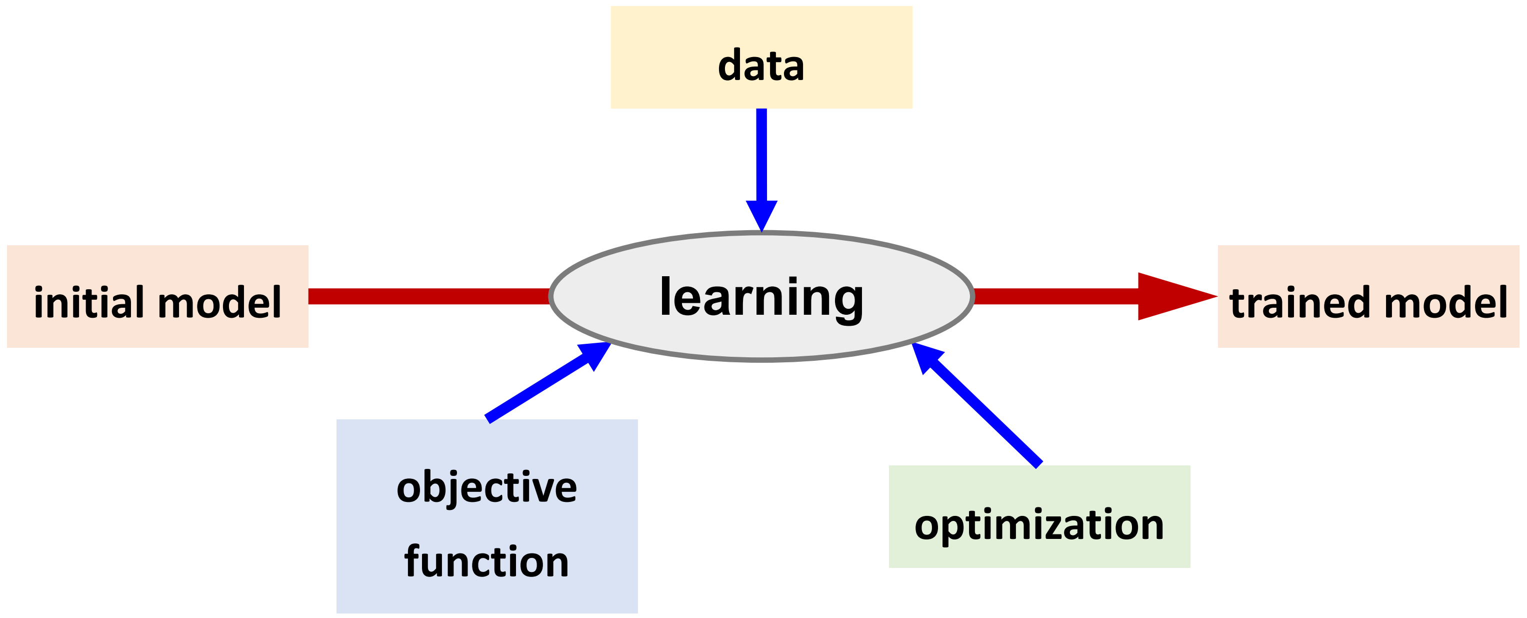
ECE30007 Intro to AI Project

Outline

- definition
- workflow and components
- categories
- applications
- prerequisite

what is machine learning?

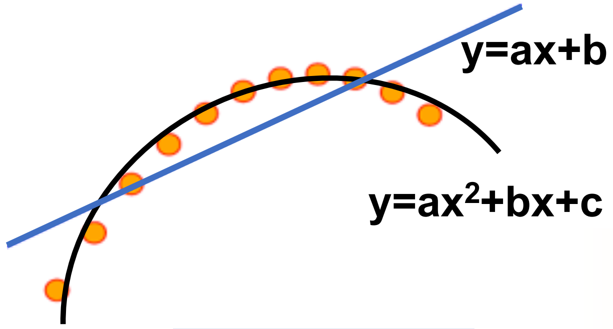
- definition of “learning” (Mitchell 1997)
  - a computer program is said to ***learn*** from experience *E*
    with respect to some class of tasks *T* and performance measure *P*,
    if its performance at the tasks improves with the experiences



machine learning (ML)

- manually designed rules are not enough to solve complex problems.

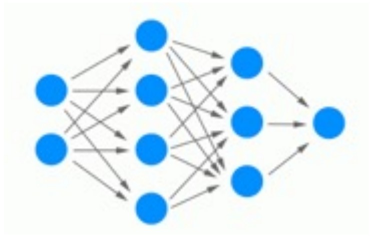
simple model
(e.g., linear regression)

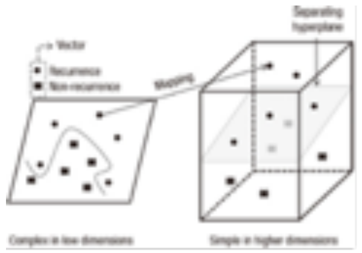


$y=f\,(x,w)$

complexity data & model

complex model

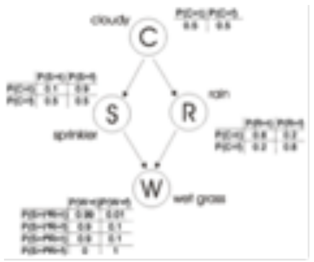




neural networks **support vector machine**

$y=f\,(x,w)$

decision tree **Bayesian networks**

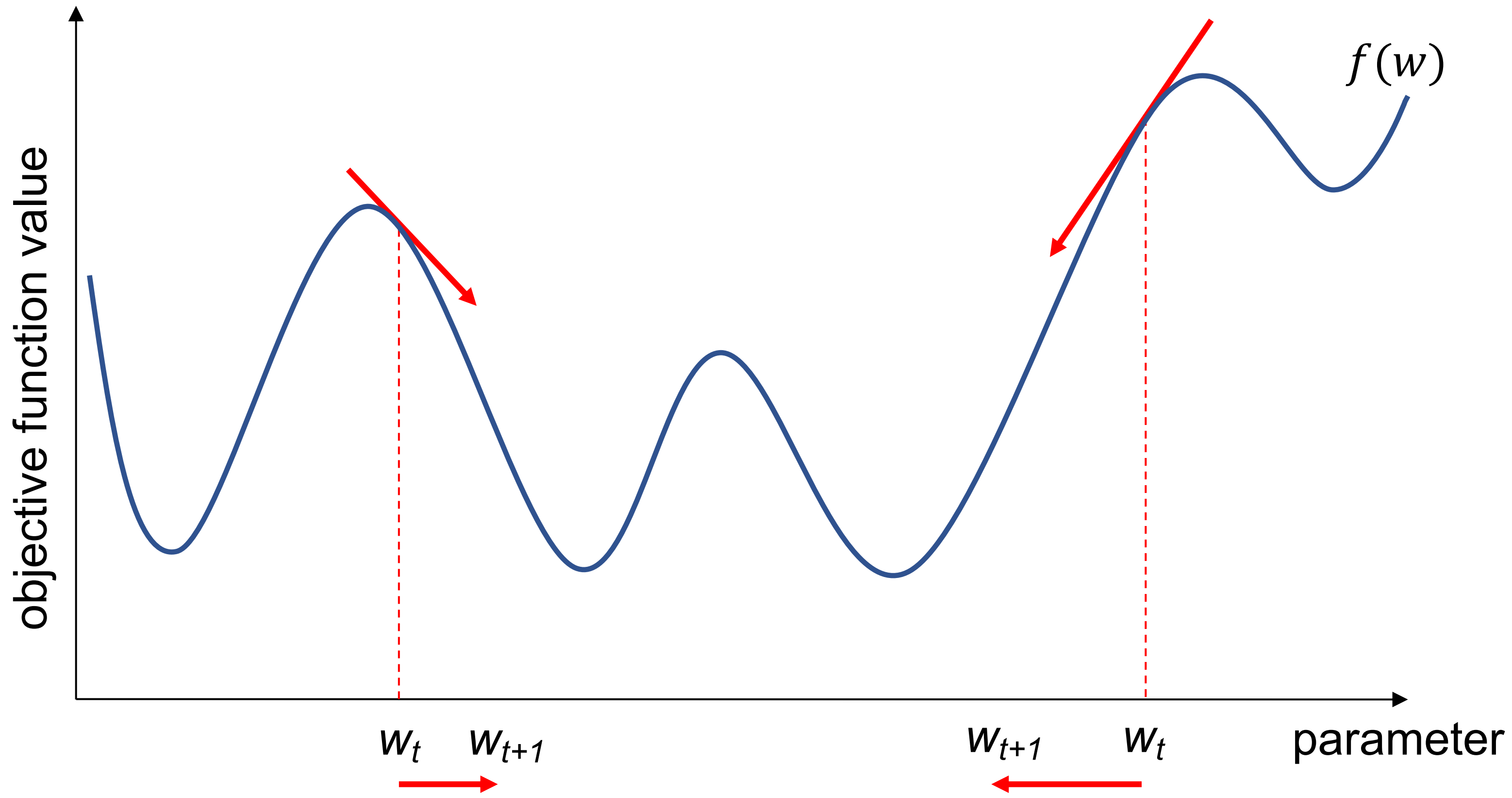


for supervised learning

- **learning:** given data (x,y), estimating w
- **recognition:** given x, calculating f(x,w) to know y

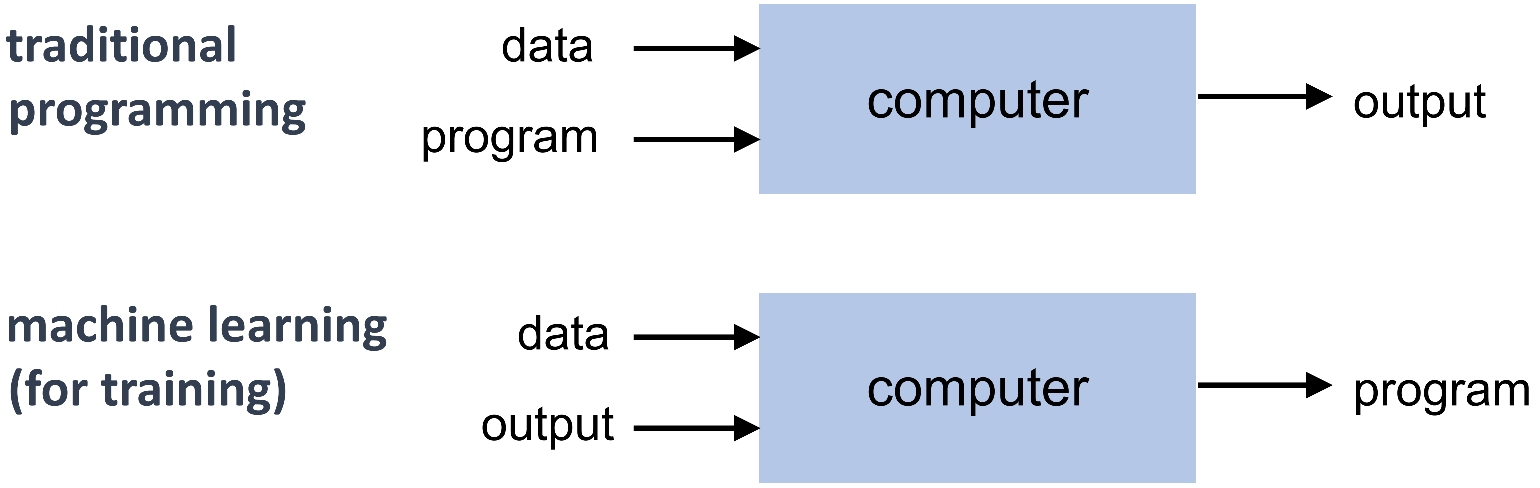
gradient for optimization

- gradient says the direction to optimize the function



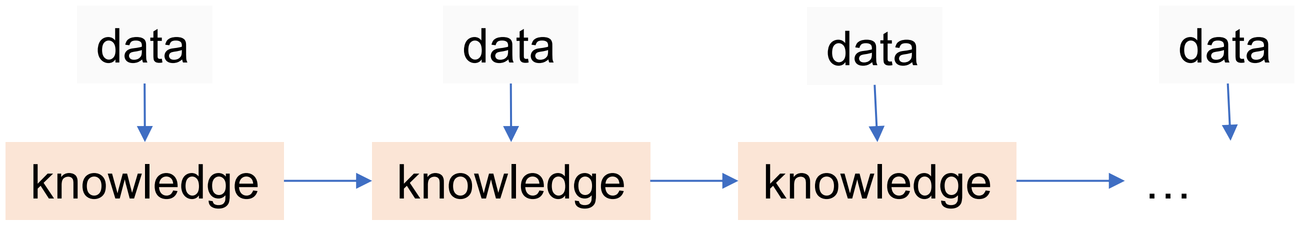
traditional programming vs. machine learning

- machine learning generates "program" by training

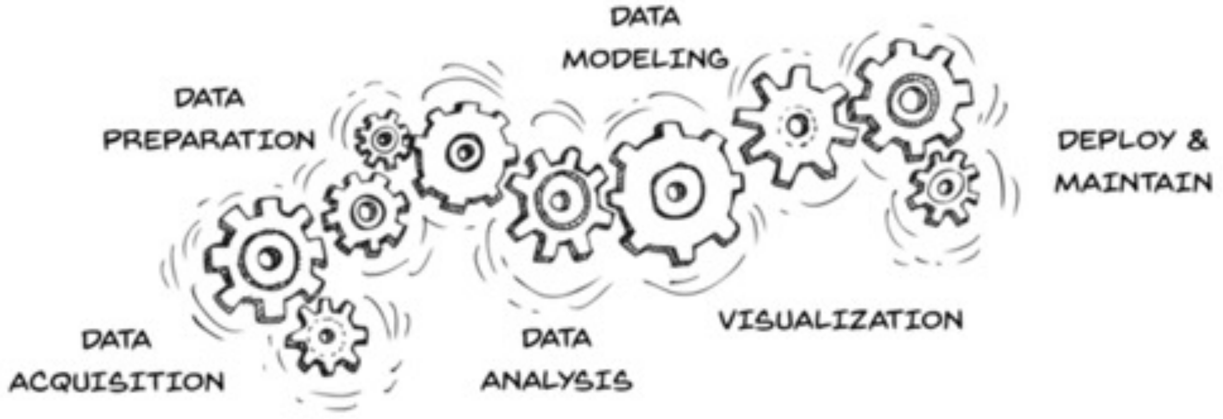


(from P. Domingos's slides)

- source of knowledge is data



ML workflow



from Charmgil Hong's slide

- ***acquisition*** - data is gathered/collected from various sources
  - sensors, activity trackers (apps), social media platforms
  - experiments, surveys, meta-data analysis
  - manual collection from non-digital sources
- ***preparation*** - data is cleaned, preprocessed, and eventually becomes a dataset
  - removing errors, mistakes, duplicates, and inconsistencies in data
  - data curation or annotation
  - data integration - combining data from different sources

ML workflow (continued)

- ***analysis*** - data is evaluated to run and customize reports (to better understand data)
  - various queries and data mergers are applied to tell a better and more informed story than when you look at each source independently
- ***modeling*** - data is patternized and generalized as models
  - models explain the general patterns that frequently observed in data
  - models are often used to make predictions or inferences
- ***visualization*** - data is visualized to provide intuitive overview
- ***deployment and maintenance*** - the outcomes of the work are applied to the field/domain to make productive effects

from Charmgil Hong's slide

Components of ML

- if someone is working on ML, he is working on the followings.

data →

- features
- label
- sequential
- format
- training
- validation
- testing

models →

- SVM
- neural networks
- naïve Bayes
- Bayesian network
- logistic regression
- random forests
- K-means
- etc

objectives →

- cross-entropy
- RMSE
- likelihood
- a posterior
- WER in ASR
- BLEU in MT
- etc

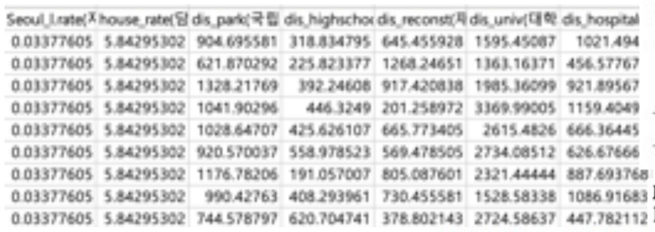
optimizations

- gradient descent
- Newton method
- linear programming
- convex optimization
- etc

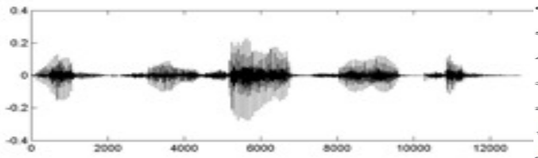
- selection depends on
  - application scenarios (classification, dimension reduction, etc)

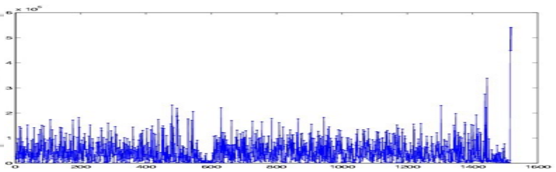
Data

- a set of values of qualitative or quantitative variables
  - measured from nature, user behavior, industrial process, and so on
  - in many different types

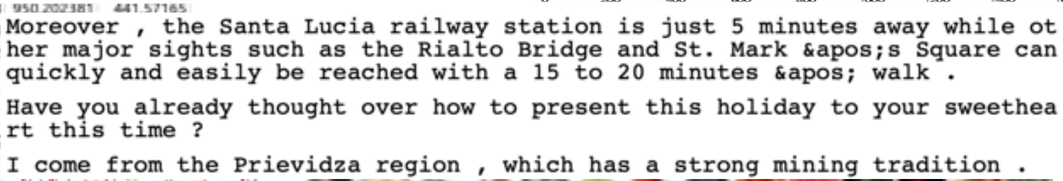


Seoul_l.rate(지	house_rate(담	dis_park(국립	dis_highschoo	dis_reconst(재	dis_univ(대학	dis_hospital
0.03377605	5.84295302	904.695581	318.834795	645.455928	1595.45087	1021.494
0.03377605	5.84295302	621.870292	225.823377	1268.24651	1363.16371	456.57767
0.03377605	5.84295302	1328.21769	392.24608	917.420838	1985.36099	921.89567
0.03377605	5.84295302	1041.90296	446.3249	201.258972	3369.99005	1159.4049
0.03377605	5.84295302	1028.64707	425.626107	665.773405	2615.4826	666.36445
0.03377605	5.84295302	920.570037	558.978523	569.478505	2734.08512	626.67666
0.03377605	5.84295302	1176.78206	191.057007	805.087601	2321.44444	887.693768
0.03377605	5.84295302	990.42763	408.293961	730.455581	1528.58338	1086.91683
0.03377605	5.84295302	744.578797	620.704741	378.802143	2724.58637	447.782112





950.202381 441.57165

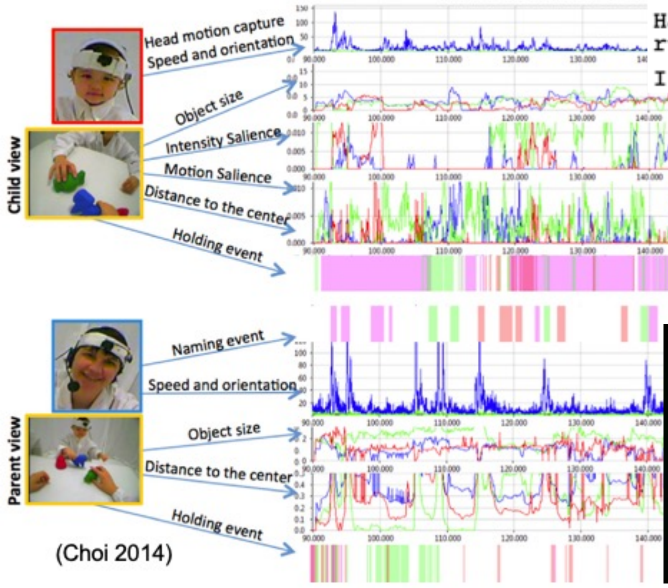


Moreover , the Santa Lucia railway station is just 5 minutes away while other major sights such as the Rialto Bridge and St. Mark &apos;s Square can quickly and easily be reached with a 15 to 20 minutes &apos; walk .

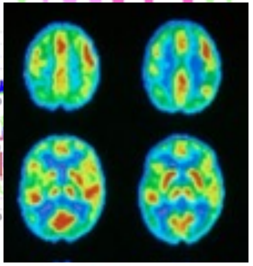
Have you already thought over how to present this holiday to your sweetheart this time ?

I come from the Prievidza region , which has a strong mining tradition .

Variables from a trial of one



(Choi 2014)



How do data look?

- structured / unstructured
  - structured data: ex) review rate about restaurants
    - a matrix (example, dimension) or
    - higher order tensor (example, dimension, time)
  - unstructured data: ex) review comments about restaurants

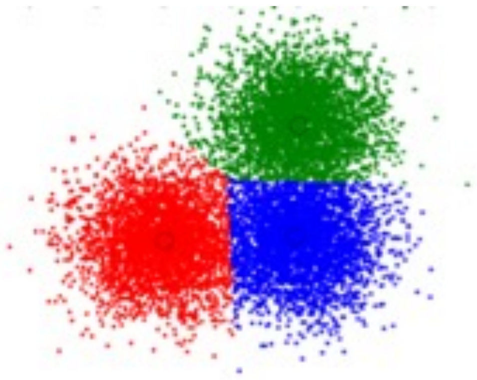
- usually, it is messy
  - data cleansing and preparation is crucial and time-consuming process
  - it is crucial in ML to prepare a clean dataset.
  - quality and quantity both matter

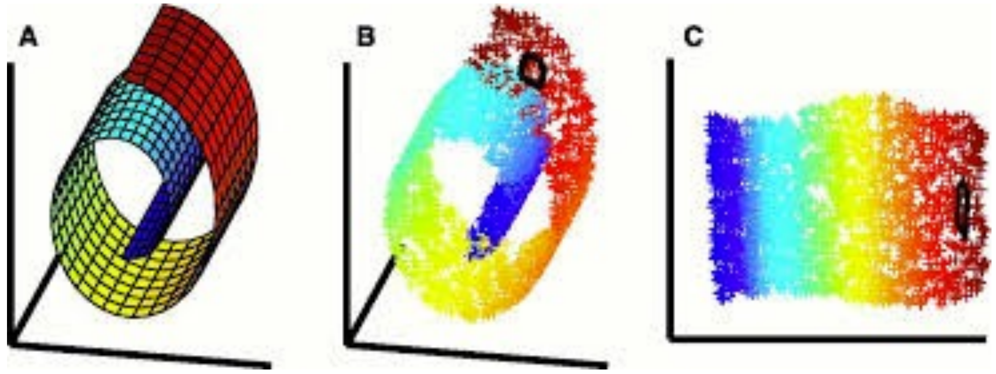
Categories in machine learning

- unsupervised learning
  - e.g., clustering, dimension reduction
- supervised learning
  - e.g., speech recognition, face recognition
- semi-supervised learning
  - e.g., cancer detection
- reinforcement learning
  - e.g., AlphaGo, self-driving car

Unsupervised learning

- e.g., clustering, dimension reduction

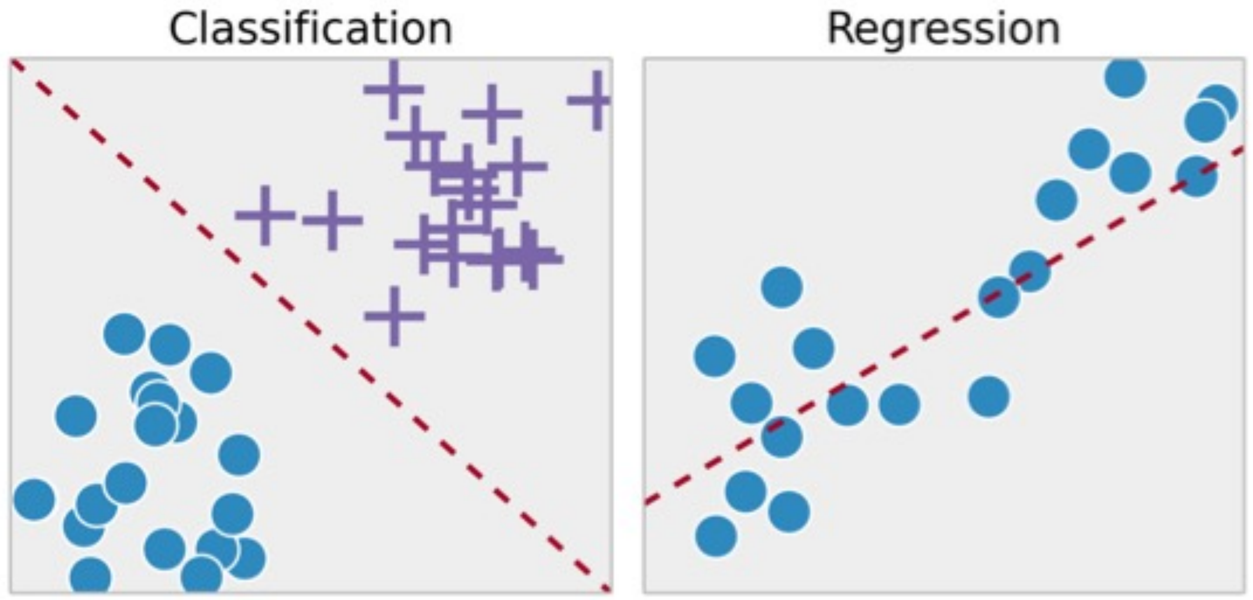




- density estimation
- pretraining

Supervised learning

- e.g., speech/face recognition, house price estimation

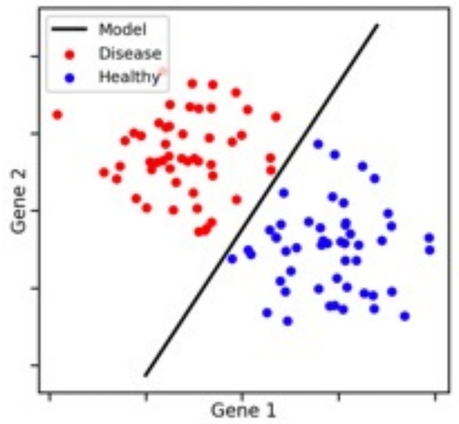


interested in the boundary of classes interested in the relationship of input and output

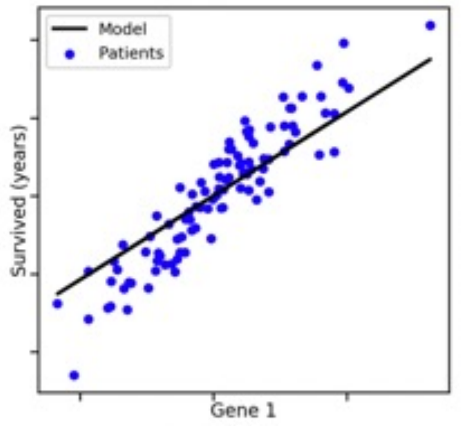
Classification and regression

- classification and regression are both supervised learning

- classification:
  - predicting a discrete label of input
  - usually evaluated by accuracy or so
  - interested in the boundary of classes



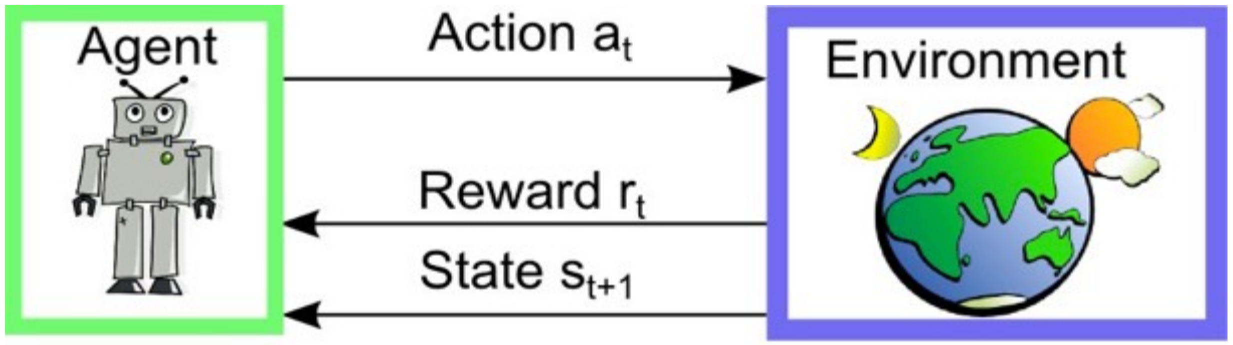
- regression:
  - predicting the quantity of output.
  - usually evaluated by root mean square error (RMSE)
  - interested in the relationship of input and output



from the web

Reinforcement learning

- e.g., AlphaGo, self-driving

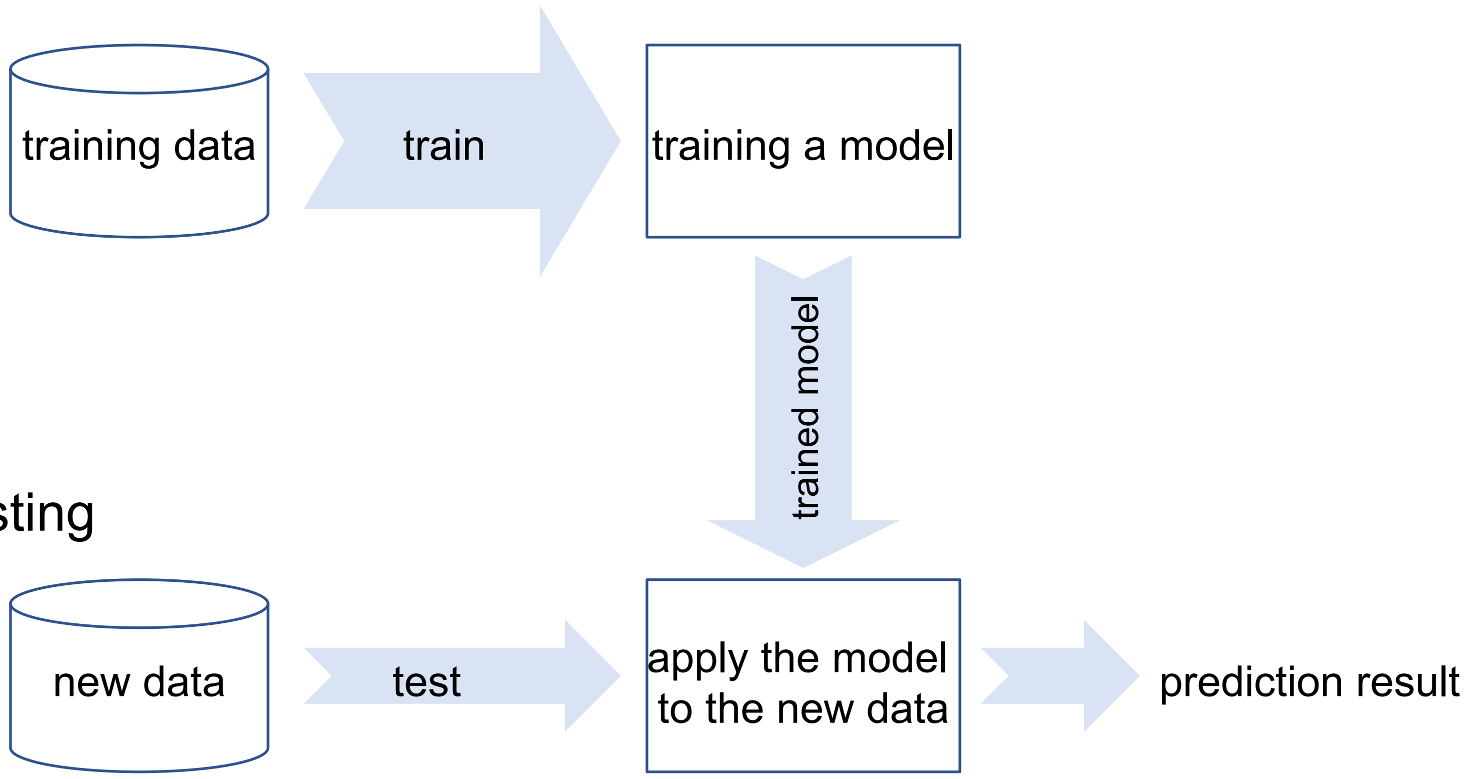


Reinforcement Learning Setup

- credit assignment problem (due to the delayed reward)
- trade-off between exploration and exploitation

workflow for supervised learning

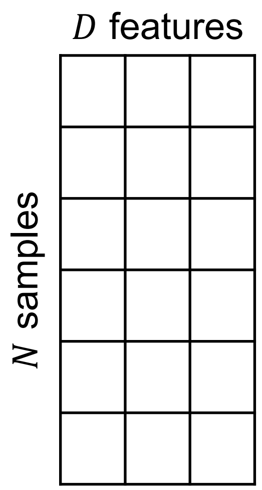
- training
- testing



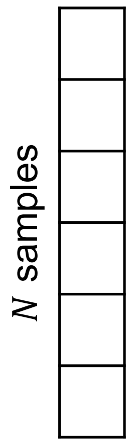
Data for supervised learning

- data is represented as a matrix
  - a row: an observation or a data instance
  - a column: one feature or attribute
  - $X = x_1, x_2, \dots, x_N$: N samples
    $x_i = (x_i^1, x_i^2, \dots, x_i^D)$: ith input sample with D attributes or features
  - $Y = y_1, y_2, \dots, y_N$: N outputs (or classes or labels)

$X \in R^{N \times D}$

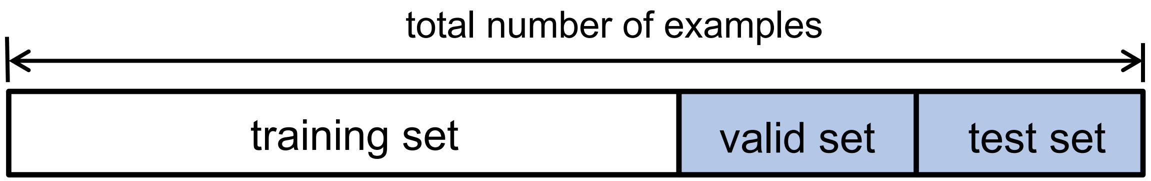


$Y \in R^N$



training, validation and testing

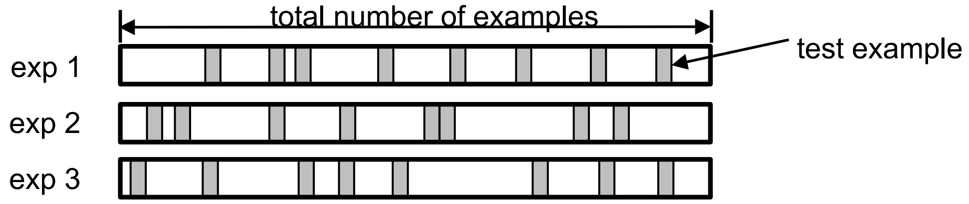
- training the model with training data (X_{tr}, Y_{tr})
  - learning the parameters by optimizing an objective function
- validation with validation data (X_{val}, Y_{val})
  - to evaluate the model, or to avoid overfitting
  - when there is no validation data,
    split the data into training data and validation data.
- testing with (X_{test}, Y_{test})
  - predicting the output of new data using the parameters learned
  - test data should not be used in training at all



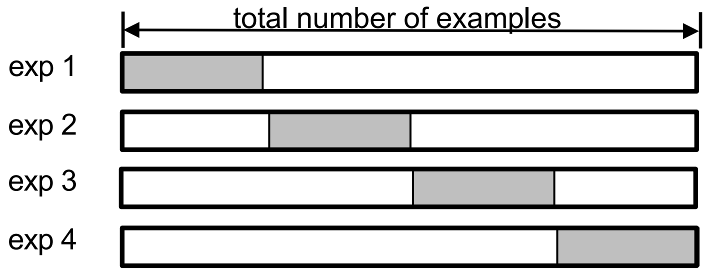
cross-validation

- a resampling procedure to evaluate ML models on a limited data.
  - split the data into training (including validation) and testing
  - evaluate the model
  - repeat the above steps

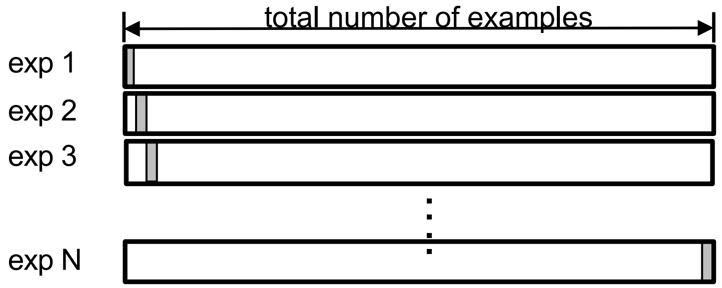
- split method 1: random subsampling



- method 2: K-fold cross-validation

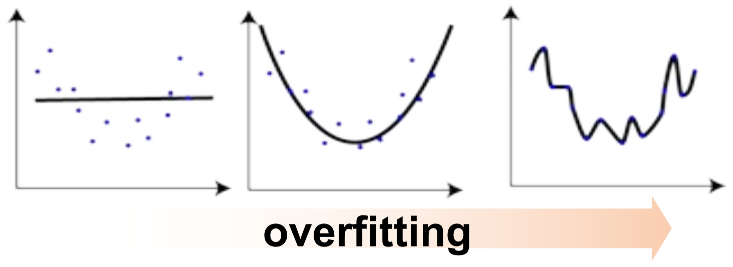


- method 3: leave-one-out cross validation



model complexity and overfitting

- overfitting: a model is too closely fit to a limited set of training data.



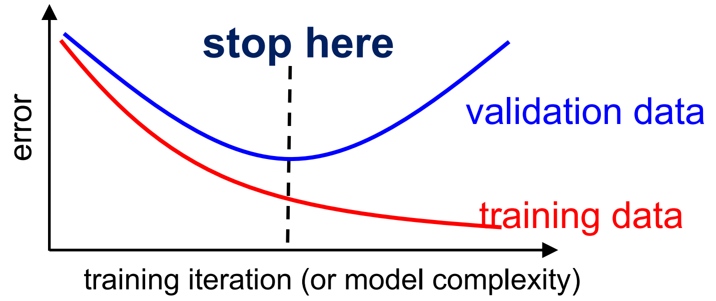
Occam's razor

"Entities should not be multiplied unnecessarily"

- William of Ockham

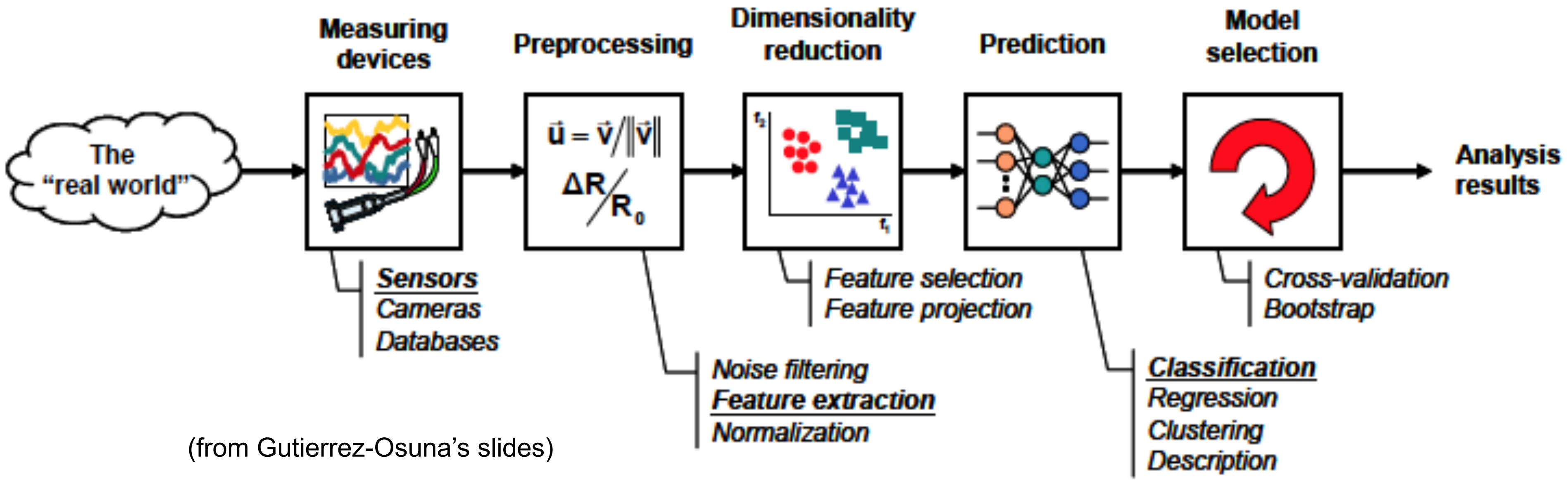
- avoid overfitting
  - spreading out the probability mass from the training samples
    - to the assumed manifold that is smooth.
  - discovering underlying abstractions/explanatory factors.

- practical approaches for overfitting
  - more data samples
  - simpler model
  - regularization methods
  - early stopping



pattern recognition

- **supervised learning**
- "The assignment of a physical object or event to one of several pre-specified categories" –*Duda and Hart*
- "A problem of estimating density functions in a high dimensional space and dividing the space into the regions of categories or classes" – *Fukunaga*



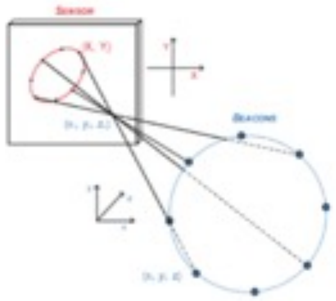
(from Gutierrez-Osuna's slides)

pattern recognition

face detection/recognition

speech recognition

beacon recognition

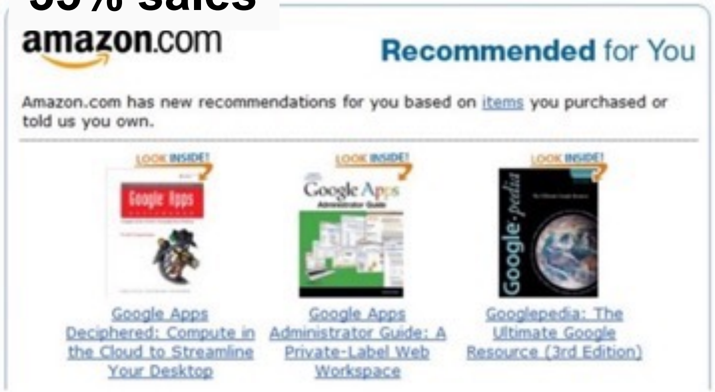


[Katake & Choi 2010]

text categorization
sentiment analysis
etc

recommendation systems

35% sales


amazon.com

Recommended for You

Amazon.com has new recommendations for you based on items you purchased or told us you own.

Google Apps Deciphered: Compute in the Cloud to Streamline Your Desktop

Google Apps Administrator Guide: A Private-Label Web Workspace

Googlepedia: The Ultimate Google Resource (3rd Edition)

2/3 of the movie watch

Congratulations! Movies we think You will ♥

Add movies to your Queue, or Rate ones you've seen for even better suggestions.

38% more clicks

Google News

Netflix dataset
- # of users: 500k
- # of items: 17k
- the total # of possible ratings: 500k x 17k = 8.5B
- the total # of actual ratings: 10M
- the portion of non-zero entries: **0.11%**

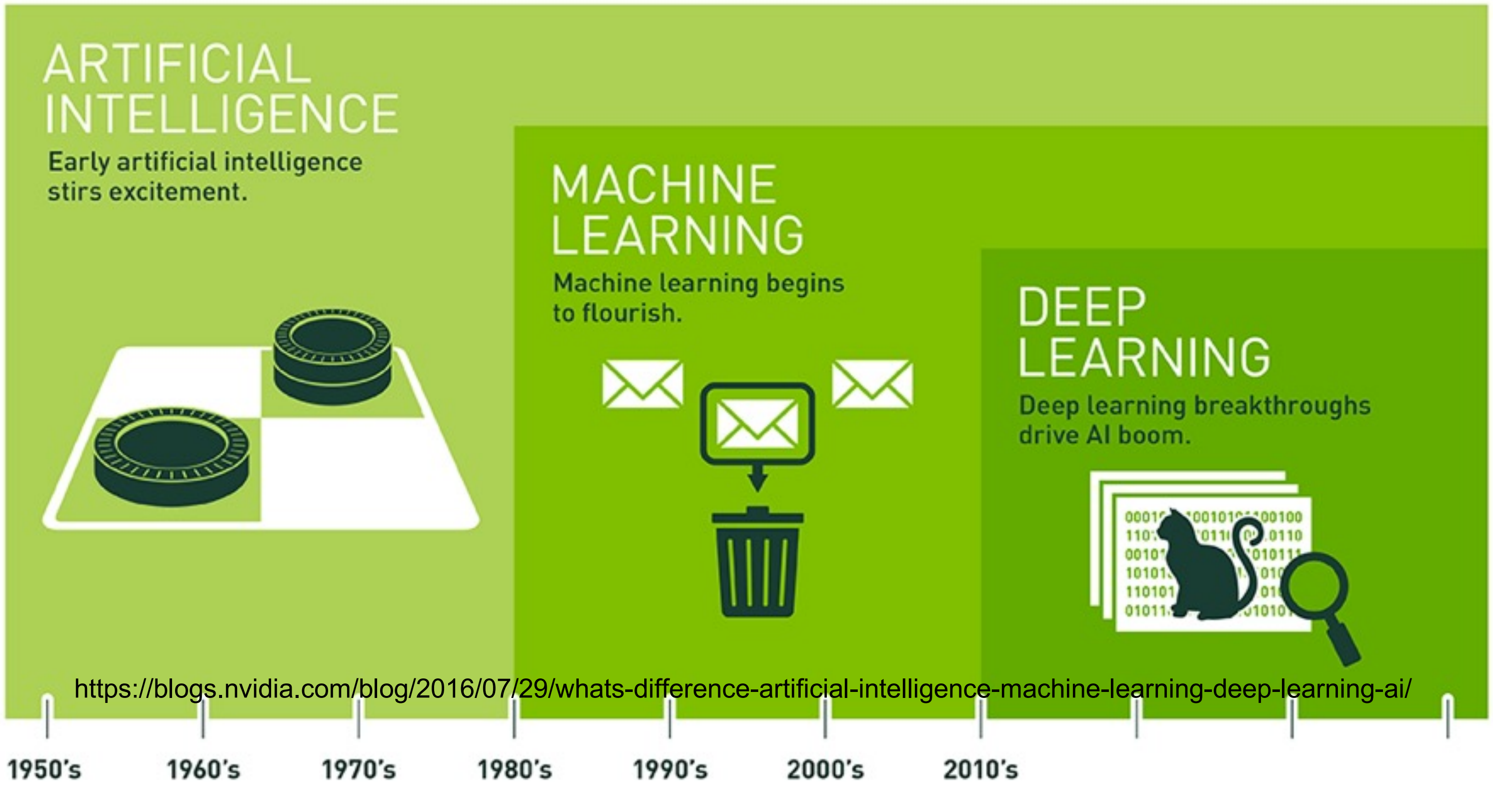
more applications

- Web search
- Speech recognition
- Handwriting recognition
- Machine translation
- Information extraction
- Document summarization
- Question answering
- Spelling correction
- Image recognition
- 3D scene reconstruction
- Human activity recognition
- Autonomous driving
- Music information retrieval
- Automatic composition
- Social network analysis
- Product recommendation
- Advertisement placement
- Smart-grid energy optimization
- Household robotics
- Robotic surgery
- Robot exploration
- Spam filtering
- Fraud detection
- Fault diagnostics
- AI for video games
- Financial trading
- Dynamic pricing
- Protein folding
- Medical diagnosis
- Medical imaging

(from Yu's slides)

recent advances in AI are attributed to deep learning

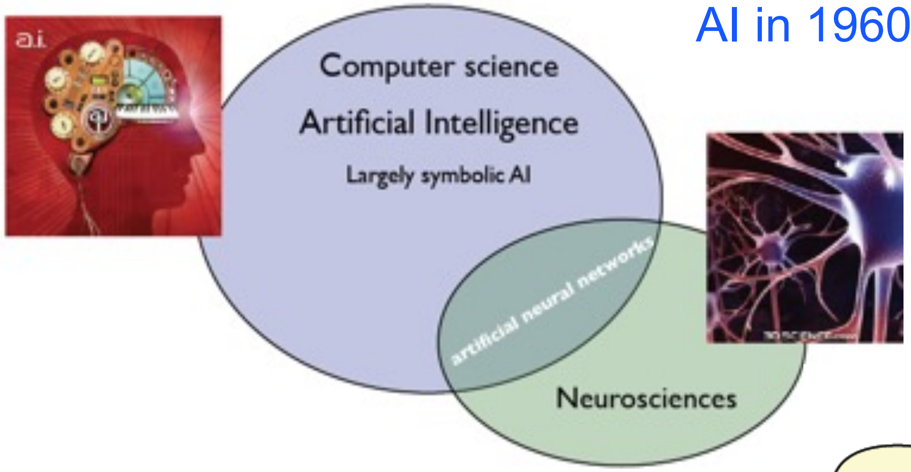
- why deep learning (DL)?
  - to understand complex problems, our model should be powerful enough.
  - DL is expressive and generalizing well (distributed representation)


ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

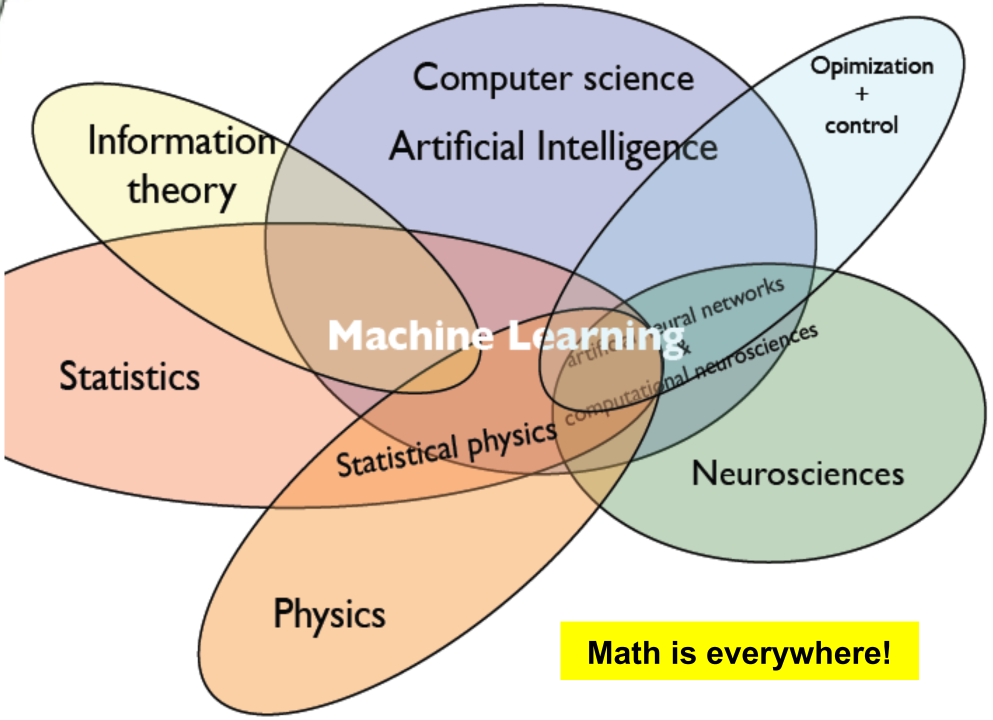
https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

interdisciplinary

AI in 1960s



recently

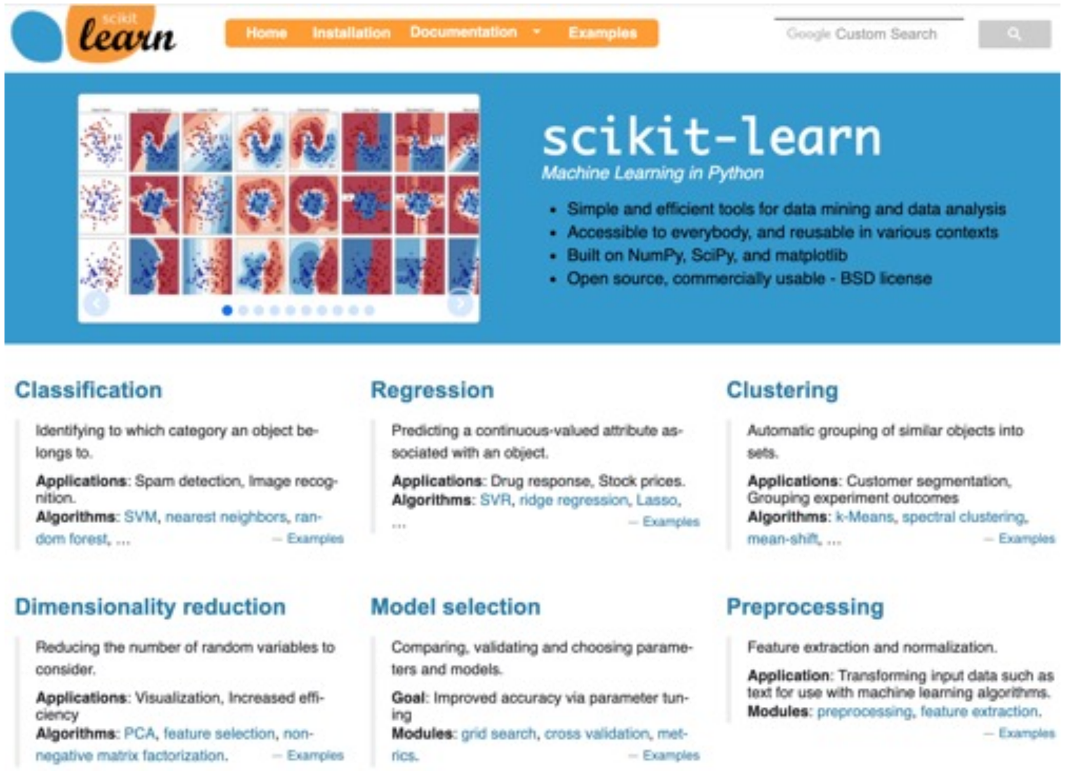


(from Vincent's slides)

Math is everywhere!

Scikit-Learn (sklearn)

- well-established ML algorithms in Python
- open source and commercially usable with BSD license
- built on NumPy, SciPy and matplotlib
- well documented with examples
- https://scikit-learn.org/



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ... — Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ... — Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ... — Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization. — Examples

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: grid search, cross validation, metrics. — Examples

Preprocessing

Feature extraction and normalization.

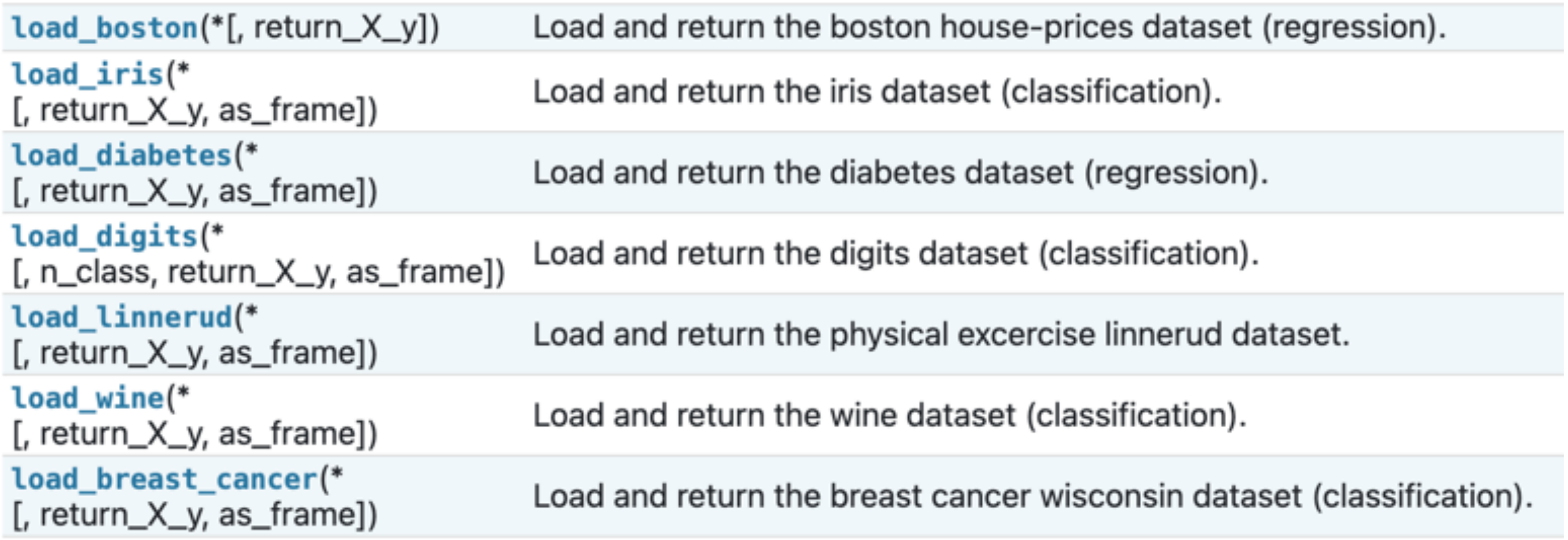
Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction. — Examples

key classes

- key components are implemented as classes.
- https://scikit-learn.org/stable/modules/classes.html
    - datasets: sklearn.datasets
    - models: sklearn.tree, sklearn.svm, etc
    - evaluation metrics: sklearn.metrics
    - experiment: sklearn.model_selection

key class: datasets

- scikit-learn comes with a few small standard datasets
  - do not require to download any file from some external website.
- They can be loaded using the following functions:



load_boston(*[, return_X_y])	Load and return the boston house-prices dataset (regression).
load_iris(* [, return_X_y, as_frame])	Load and return the iris dataset (classification).
load_diabetes(* [, return_X_y, as_frame])	Load and return the diabetes dataset (regression).
load_digits(* [, n_class, return_X_y, as_frame])	Load and return the digits dataset (classification).
load_linnerud(* [, return_X_y, as_frame])	Load and return the physical excercise linnerud dataset.
load_wine(* [, return_X_y, as_frame])	Load and return the wine dataset (classification).
load_breast_cancer(* [, return_X_y, as_frame])	Load and return the breast cancer wisconsin dataset (classification).

from https://scikit-learn.org/stable/datasets/toy_dataset.html

key class: datasets

- example: load_iris

Iris Versicolor　Iris Setosa　Iris Virginica

```
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)
print(X.shape, y.shape)
```

```
(150, 4) (150,)
```

```
X[:5]
```

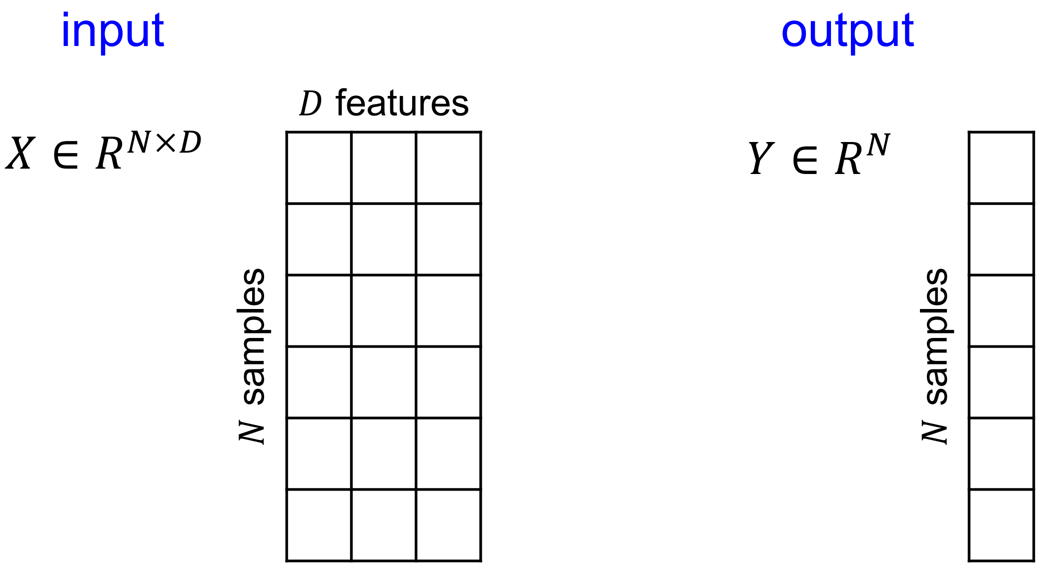
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

key class: datasets

- data formats:
  - matrix as a NumPy ndarray or a Pandas DataFrame/Series
  - each row of these matrices: one instance of the dataset
  - each column: a quantitative piece of information (each instance or features)



key class: models

- models include
  - sklearn.tree, sklearn.neighbors, sklearn.svm, etc

```
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.ensemble import RandomForestClassifier, GradientBoostingRegressor
from sklearn.svm import SVC, SVR
from sklearn.linear_model import LinearRegression, LogisticRegression
```

```
model = KNeighborsClassifier(n_neighbors=5)
print(model)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

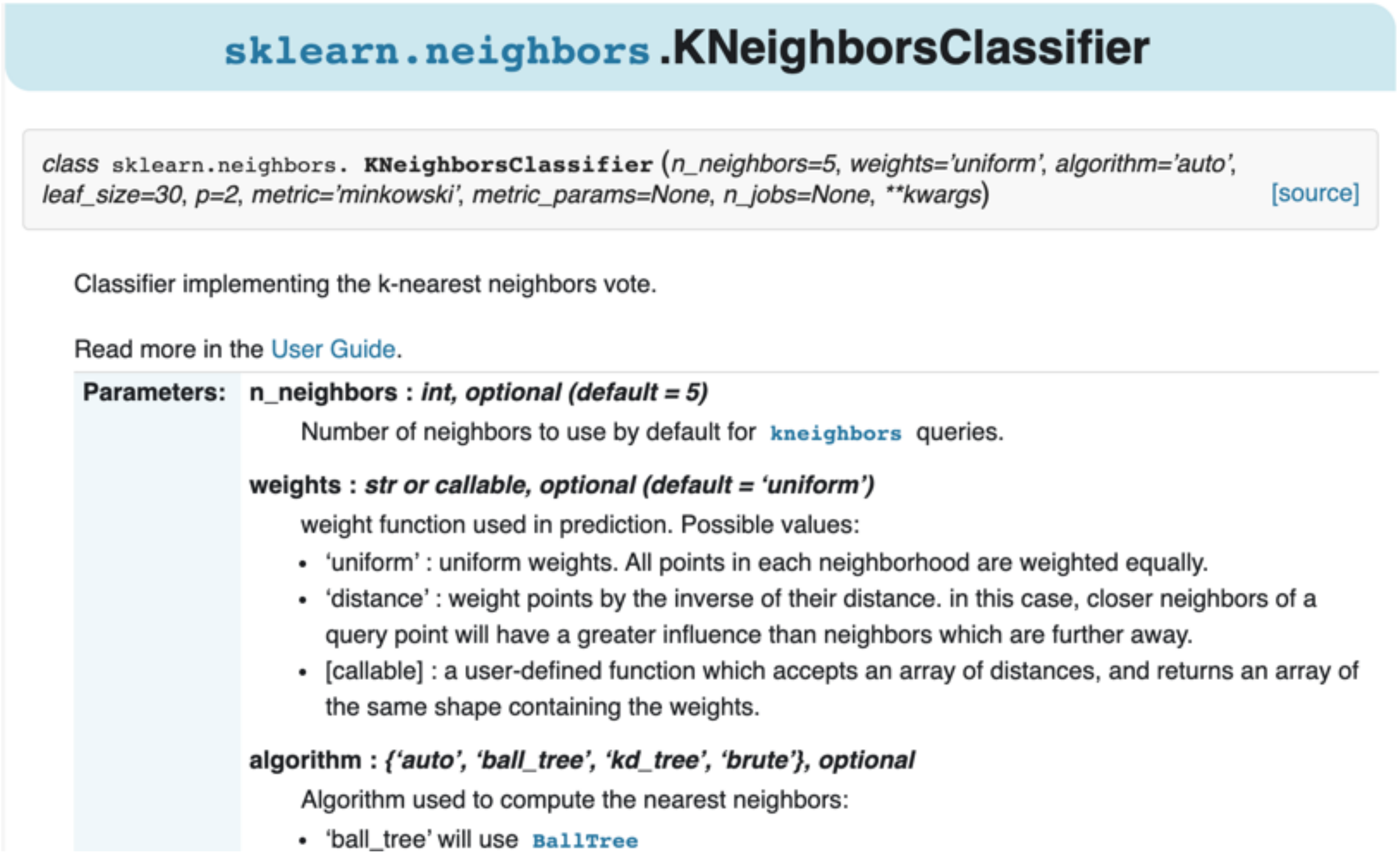
key class: models

- available models (sklearn.tree, sklearn.neighbors, …)
  - for supervised learning
    - linear models (logistic regression)
    - support vector machines
    - tree-based methods (decision trees, random forests)
    - nearest neighbors
    - neural networks
    - Gaussian process
    - feature selection
  - for unsupervised learning
    - clustering
    - matrix decomposition
    - manifold learning
    - outlier detection

adapted from Charmgil Hong's slide

key class: models

- models are well documented at https://scikit-learn.org/



sklearn.neighbors.KNeighborsClassifier

class `sklearn.neighbors.` **`KNeighborsClassifier`** (*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Parameters: **n_neighbors : *int, optional (default = 5)***

Number of neighbors to use by default for `kneighbors` queries.

weights : *str or callable, optional (default = 'uniform')*

weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : *{'auto', 'ball_tree', 'kd_tree', 'brute'}, optional*

Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`

key class: models

- classification models have these member functions.

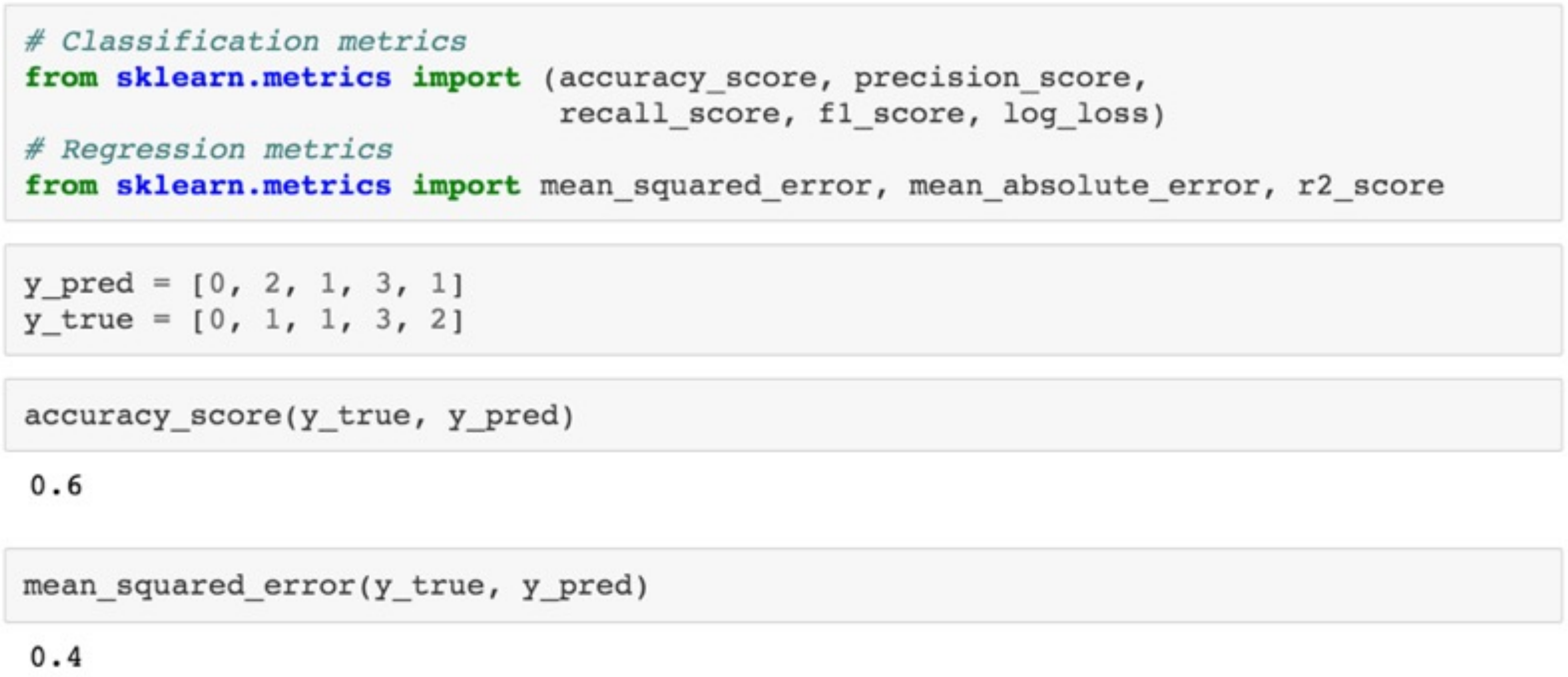
```
class Estimator(BaseClass):

    def __init__(self, **hyperparameters):
        # Setup Estimator here

    def fit(self, X, y):
        # Implement algorithm here

        return self

    def predict(self, X):
        # Get predicted target from trained model
        # Note: fit must be called before predict

        return y_pred
```

```
# Create the model
model = KNeighborsClassifier(n_neighbors=5)

# Fit the model
model.fit(X, y)

# Get model predictions
y_pred = model.predict(X)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 2, 2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 2, 1, 2, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 1, 2,
       2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2,
       2, 2, 1, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1])
```

key class: evaluation

- evaluation metrics (sklearn.metrics)



```python
# Classification metrics
from sklearn.metrics import (accuracy_score, precision_score,
                             recall_score, f1_score, log_loss)
# Regression metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

```python
y_pred = [0, 2, 1, 3, 1]
y_true = [0, 1, 1, 3, 2]
```

```python
accuracy_score(y_true, y_pred)
```

```
0.6
```

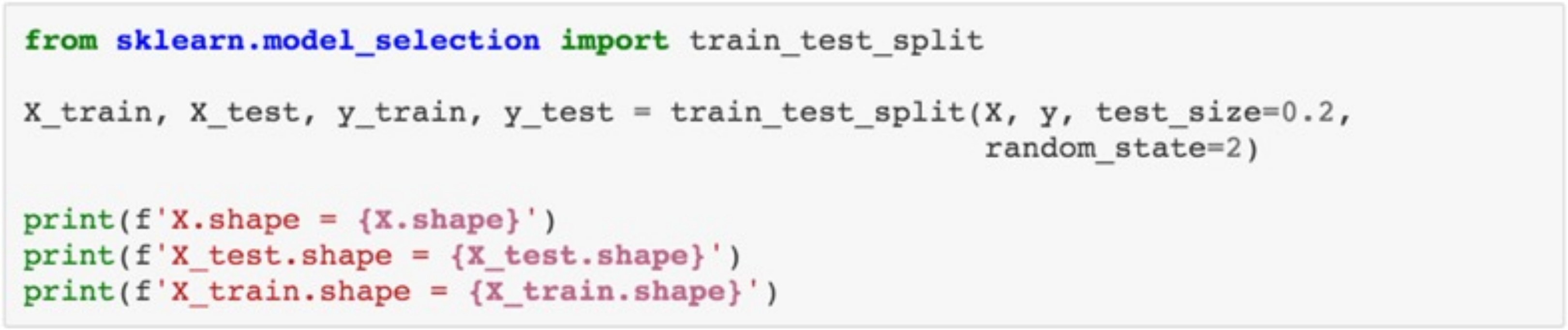
```python
mean_squared_error(y_true, y_pred)
```

```
0.4
```

key class: experiments

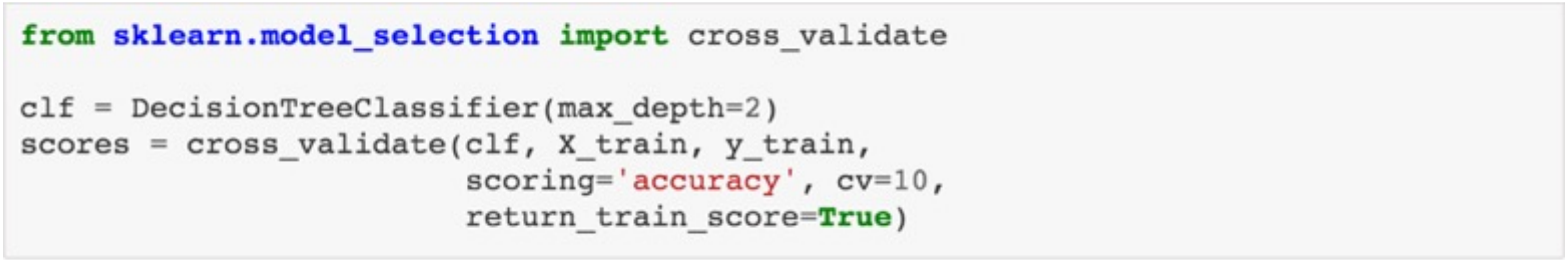
- experiment (sklearn.model_selection)

- data split



```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=2)

print(f'X.shape = {X.shape}')
print(f'X_test.shape = {X_test.shape}')
print(f'X_train.shape = {X_train.shape}')
```

- cross validation



```python
from sklearn.model_selection import cross_validate

clf = DecisionTreeClassifier(max_depth=2)
scores = cross_validate(clf, X_train, y_train,
                        scoring='accuracy', cv=10,
                        return_train_score=True)
```

example with random forest

- data split: train and test

```
X_tr, X_ts, y_tr, y_ts = train_test_split(X, y, test_size=0.3, random_state=777)
```

- cross-validation and learning

```
clf = RandomForestClassifier()
parameters = {'n_estimators': [100, 150, 200],
              'criterion': ['gini', 'entropy']}  # 6 configurations of hyper-parameters
gridsearch = GridSearchCV(clf, parameters, scoring='accuracy', cv=5)
gridsearch.fit(X_tr, y_tr)
print(f'gridsearch.best_params_ = {gridsearch.best_params_}')

best_clf = gridsearch.best_estimator_
best_clf
```

best_params_: hyper-parameters
best_estimator_: parameters

- testing

```
y_pred = best_clf.predict(X_ts)
test_acc = accuracy_score(y_ts, y_pred)
print(f'test_acc = {test_acc}')
```

- final training

```
final_model = RandomForestClassifier(**gridsearch.best_params_)
final_model.fit(X, y)
```