Case Study: Classification

Machine Learning

Course Outline

Assignments are submitted at the same date as the next assignment comes out

Wk	Lecture 1	Lecture 2	Individual Assignment	Project Assignment
1	Case Study: Regression	Case Study: Classification	A1: Car Price Prediction	
2	Batch Gradient Descent	Stochastic / Mini-Batch Gradient Descent		
3	Regularization	Binary Logistic Regression	A2: TBD	
4	Multinomial Logistic Regression	Gaussian Naive Bayes		
5	Multinomial Naive Bayes	K-Nearest Neighbors	A3: TBD	
6	Support Vector Machine	Support Vector Machine II		
7	**No class**	**Midterm Exam**		Phase 1: Reading paper round 1 (KDD)
8	Decision Tree	Bagging / Random Forests		Phase 1: Reading paper round 2 (KDD)
9	Ada Boosting / Gradient Boosting	K-Means Clustering		Phase 2: Proposal - Paper writing (Intro, Related Work, Method)
10	Gaussian Mixture	Principal Component Analysis		
11	PyTorch Linear Regression	**Project Proposal Presentation**		Phase 3: Experiment - Paper writing (Intro, Related Work, Method, Results)
12	PyTorch Logistic Regression	Convolutional Neural Network		
13	Recurrent Neural Network	Reinforcement Learning		
14	Q-learning	**Project Progress Presentation**		Phase 4: Conclusion - Paper writing (Abstract, Intro, Related Work, Method, Results, Discussion, Conclusion)
15	**No class**	**Final Exam**		
16	**No class**	**Final Project Presentation**		

Background - Classification

- Predict **cancer or no cancer** using **sizes?**
- Predict **positive or negative** given **a sentence?**
- Predict **car brands** using **images?**

Notice all the **labels** here are **categorical (discrete)** values?

Classification is a **supervised** algorithm to *predict* **categorical (discrete)** values

- Labels must be categorical; features can be categorical or continuous
- Labels can be binary (two class) or multiclass
- *Supervised* - has both features and labels; *Unsupervised* - only has features

Big picture of ML

1. Load data -> 2. Exploratory Data Analysis -> 3. Feature engineering -> 4. Feature selection -> 5. Preprocessing

1. Load data
- CSV, JSON, Database
- Renaming
- Label encoding

2. Exploratory Data Analysis
- Countplot
- Distribution plot
- Boxplot
- Scatter plot
- Correlation Matrix
- Predictive Power Score

3. Feature engineering
- Dimensionality reduction
- Feature splitting (e.g., date)
- Creating features (e.g., some equation)

4. Feature selection
- Train / dev / test set
- Select your X (features) and y (target)
- In ML, it's better to choose X
- In DL, we usually just input all features

5. Preprocessing
- Null values
- Outliers
- Typos / Entry errors / Duplicates / IDs
- Scaling (min-max; standardize)

6. Model selection -> 7. Testing -> 8. Analysis -> 9. Inference -> 10. Deployment

6. Model selection

Supervised
- Regression
- Classification

Unsupervised
- Clustering
- Dimensionality reduction

Reinforcement
- PPO
- Q-learning

- **Cross-validation**
- **Grid search**

7. Testing

Apply your best model on your test set

8. Analysis

Analyze your model, e.g., **feature importance**

9. Inference

Apply your best model on some unseen data, and see whether it makes sense

10. Deployment
- Flask
- Django
- FastAPI

- Docker

METRIC
- **Regression** (r^2, MSE)
- **Classification** (recall, precision, f1)
- **Clustering** (inertia)
- **Dimensionality reduction** (mean squared distance between the original data and the reconstructed data)
- **Reinforcement learning** (cumulative rewards)

- MLFlow
- Wandb
- Tensorboard

TOOLS
- **Python**, R - programming tool
  - **NumPy** (matrix manipulation), **Pandas** (Excel-like), **Matplotlib/Seaborn** (visualization), **Sklearn** (machine learning), **PyTorch** (deep learning)
- **Tableau**, **Power BI** - Business Intelligence (BI) tools
- **Microsoft Azure, Rapidminer, Weka** - data science and machine learning tools
- **SPSS, SAS, JASP** - statistical tool

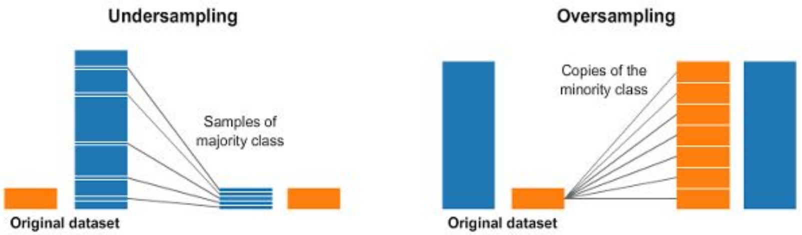
TOP VENUES
1. *ML (KDD)*
2. *DL (ICML, NIPS, ICMR)*
3. *NLP (ACL, EMNLP)*
4. *CV (CVPR, ICCV)*

One-hot encoding

- Recall label encoding which we turn categories into 0, 1, 2 etc.
- When we have **more than two categories**, if we encode into 0, 1, 2
  - we create a unintentional order, i.e., the model "may" think that 0 < 1 < 2
- Possible solution: **one hot encoding**
  - E.g., Male, Female, Unknown ⇒ [1, 0, 0] if male; [0, 1, 0] if female
  - Limitation
    - what if we have like 5000 categories....
    - one hot encode this will result in 5000 columns --> too much! -> Two choices:
      - Group these categories into bigger categories, and then one-hot encode
      - Do label encoding anyway......but note the possible order effect
- **Tips**: one thing you need to know is that you can always cut down one column
  - [1, 0, 0], [0, 1, 0], [0, 0, 1] is same as [1, 0], [0, 1], [0, 0] by setting `'drop_first=True'`

Class imbalance

- In classification, it's important to check the **class imbalance**
- For example, if you want to predict cat or dog, but you have 100 images of cat, but 1000 images of dogs
- Ways to deal with class imbalance
    - Sample randomly 100 images of dogs (**downsampling**) - loss of information
    - Randomly augment 900 more images of cat (**upsampling**) - redundant info
- Further, two types of sampling methods:
    - **Offline** - treat sampling as preprocessing step
    - **Online** - during the epoch, downsampling or upsampling on the fly (much better)
- If class imbalance exists, you MUST not use "accuracy", use recall / precision / f1-score instead




```
# import library
from imblearn.over_sampling import SMOTE

smote = SMOTE()

# fit predictor and target variable
x_smote, y_smote = smote.fit_resample(x, y)

print('Original dataset shape', Counter(y))
print('Resample dataset shape', Counter(y_ros))
```

```
Original dataset shape Counter({0: 9000, 1: 492})
Resample dataset shape Counter({1: 9000, 0: 9000})
```

Classification metrics

Given:

y = [0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]
ypred = [1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1]

Confusion Matrix

		Actual P	Actual N
Predict	P	7	2
	N	1	1

accuracy = (TP + TN) / (TP + FP + FN + TN)
accuracy = (7 + 1) / (7 + 2 + 1 + 1) = 8 / 11 = 72%

Bad for imbalance y

recall = TP / (TP + FN)
recall = (7) / (7 + 1) = 7 / 8 = 87.5%

Good for minimizing FN, e.g., cancer

precision = TP / (TP + FP)
precision = (7) / (7 + 2) = 7 / 9 = 77%

Good for minimizing FP, e.g., search engine

f1 = 2 * ((recall * precision) / (precision + recall))
f1 = 2 * ((87.5 * 77) / (87.5 + 77)) = 81.9%

Balance

We called 3 as **True Positive**, 2 as **False Positive**, 1 as **False Negative**, and the bottom right one as **True Negative**