wtech
Teknolojide Kadın Derneği

Introduction to Machine Learning with Python

Supervised Learning: k-NN and Linear Models

Dr. Süha Tuna
İTÜ Informatics Institute

Classification and Regression

What is Classification?

- **Objective**: Predict a **class label** from a predefined set of categories.
- **Examples**:
  - Classifying **irises** into one of three species.
  - Determining the **language** of a website based on its text.
- **Types of Classification**:
  - **Binary Classification**:
    - Two classes only (e.g., *Spam vs Not Spam*).
    - Answers a **yes/no** type question.
  - **Multiclass Classification**:
    - More than two possible classes.
    - E.g., **English**, **French**, **German**, etc.

What is Regression?

- **Objective**: Predict a **continuous numerical value** (i.e., a real or floating-point number).
- **Examples**:
  - Predicting a **person's income** based on education, age, and location.
  - Estimating **corn yield** using weather, workforce, and past yields.

Key Distinction: Continuity of Output

- **Regression**:
  - Outputs are **continuous**.
  - Small differences in prediction are tolerable.
  - E.g., $40,000 vs $40,001 is **not significant**.
- **Classification**:
  - Outputs are **discrete labels**.
  - No in-between or gradual difference.
  - E.g., a website is **either English or French**, not partially both.

Classification and Regression: Summary

Feature	Classification	Regression
Output Type	Discrete Class Labels	Continuous Numerical Values
Example Task	Spam Detection	Income Prediction
Continuity in Output	No	Yes
Error Tolerance	Exact Class Required	Small deviations acceptable
Evaluation Metric	Accuracy, Precision, Recall, F1	MSE, RMSE, MAE

Generalization, Overfitting and Underfitting

Goal of Supervised Learning
- Train a model using labeled training data.
- Use the model to make accurate predictions on unseen data with similar characteristics.
- Success metric: the model's ability to generalize from training data to test data.

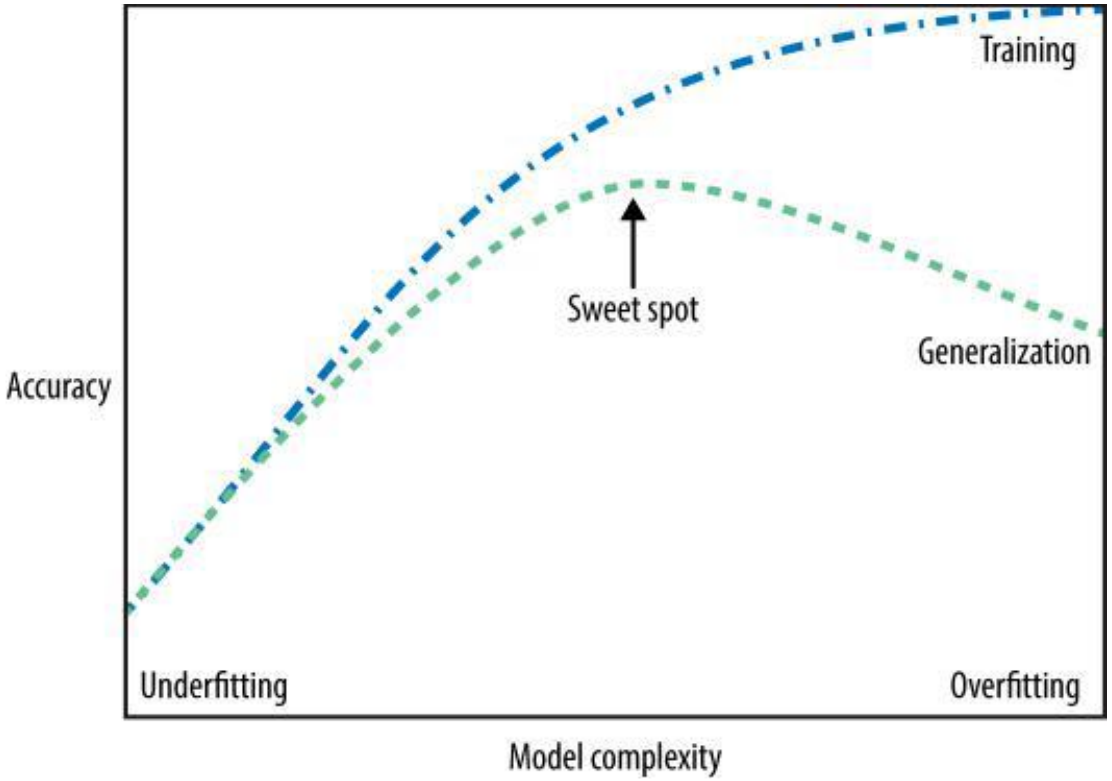
Generalization Defined
- A model generalizes well if it performs accurately on new, unseen data.
- Generalization is crucial for real-world applicability.

Training Accuracy vs Generalization
- Models are often trained to maximize accuracy on the training set.
- Assumption: If the training and test data are sufficiently similar, a well-trained model will perform well on both.

The Problem with Overly Complex Models
- Complex models can achieve very low training error.
- But high complexity may lead to overfitting:
- - The model memorizes the training data.
- - It loses the ability to generalize to new examples.
- Overfitting results in poor test performance, despite perfect training accuracy.

Finding the Right Model Complexity
- There exists a sweet spot of model complexity:
- - Too simple → Underfitting (poor training and test performance).
- - Too complex → Overfitting (great training, poor test performance).
- The optimal model balances:
- - Sufficient complexity to capture patterns.
- - Enough simplicity to maintain generalization.

Generalization, Overfitting and Underfitting



Concept	Description
Generalization	Ability to perform well on **unseen** data
Overfitting	Model is too complex and memorizes training data
Underfitting	Model is too simple to capture underlying patterns
Ideal Model	Balances complexity and accuracy for best generalization

Some sample datasets

In[2]:

```
# generate dataset
X, y = mglearn.datasets.make_forge()
# plot dataset
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.legend(["Class 0", "Class 1"], loc=4)
plt.xlabel("First feature")
plt.ylabel("Second feature")
print("X.shape: {}".format(X.shape))
```

Out[2]:

```
X.shape: (26, 2)
```

In[3]:

```
X, y = mglearn.datasets.make_wave(n_samples=40)
plt.plot(X, y, 'o')
plt.ylim(-3, 3)
plt.xlabel("Feature")
plt.ylabel("Target")
```

In[4]:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
print("cancer.keys(): \n{}".format(cancer.keys()))
```

Out[4]:

```
cancer.keys():
dict_keys(['feature_names', 'data', 'DESCR', 'target', 'target_names'])
```

In[9]:

```
X, y = mglearn.datasets.load_extended_boston()
print("X.shape: {}".format(X.shape))
```

Out[9]:

```
X.shape: (506, 104)
```

The Models to be Covered

- k-Neigbors Classification
- k-Neigbors Regression
- Linear Regression
- Ridge, LASSO, Elastic-Net
- Linear Models for Classification
- Multiclass Classification