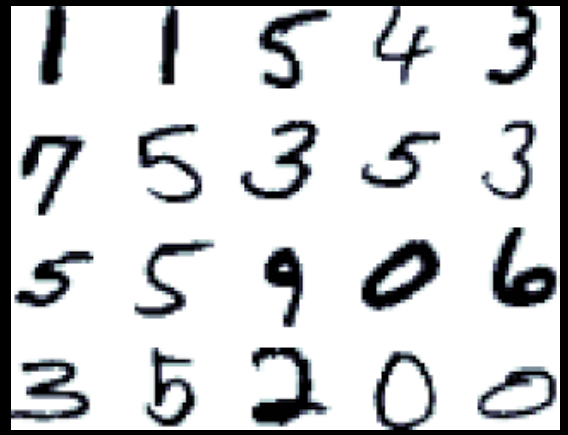
The Machine Learning Landscape

Chapter 1: pp 1 – 34

What is Machine Learning

- It is a field of study that gives the computers the ability to learn without being explicitly programmed – *Arthur Samuel, 1959*



- A computer program is said learn from experience *E* with respect to some task *T* and some performance measure *P*, if its performance on *T*, as measured by *P*, improves with experience *E* – *Tom Mitchell, 1997*

Examples of Applications

- Analyzing images of products to automatically classify them
- Detecting tumors in brain scans
- Automatically classifying new articles
- Automatically flagging offensive comments on discussion forums
- Automatically summarizing long documents
- Forecasting next year’s revenue based on some performance metrics
- Detecting credit card fraud
- Etc …

Why Use Machine Learning?

1). Solution requires too much hand-tuning or too many rules

- Example: How do you write a spam filter for emails containing “4U”, “credit card”, password”?

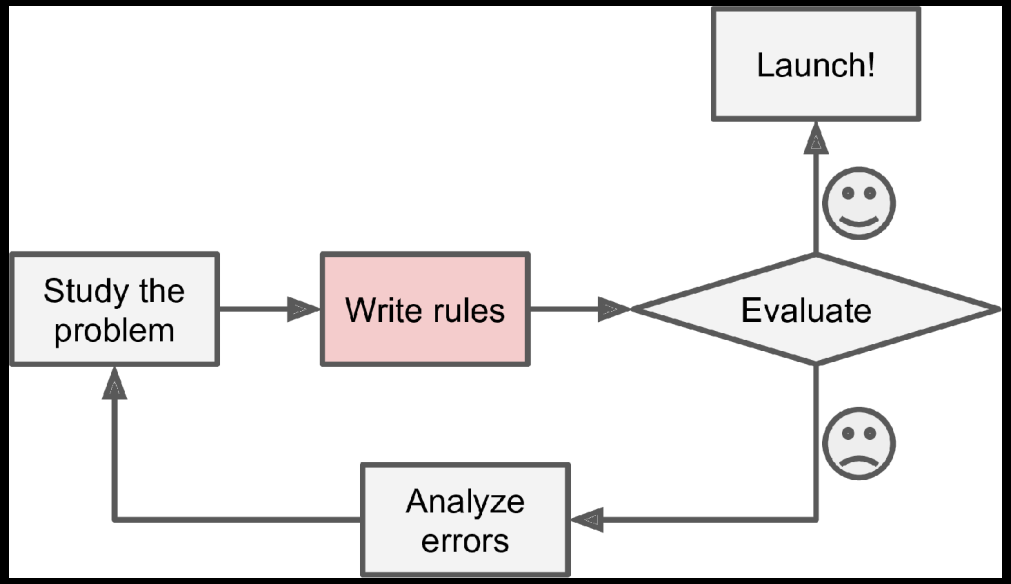


Figure 1-1. The traditional approach

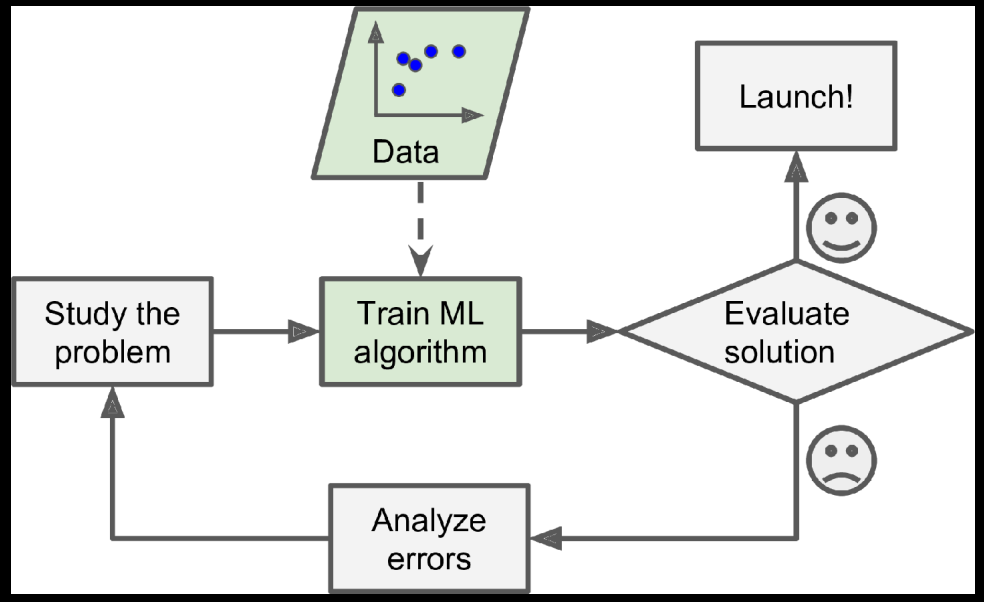


Figure 1-2. Machine Learning approach

Why Use Machine Learning?

2). Fluctuating environments

- What if spammers work around your spam filter (changing “4U” to “For U”)?

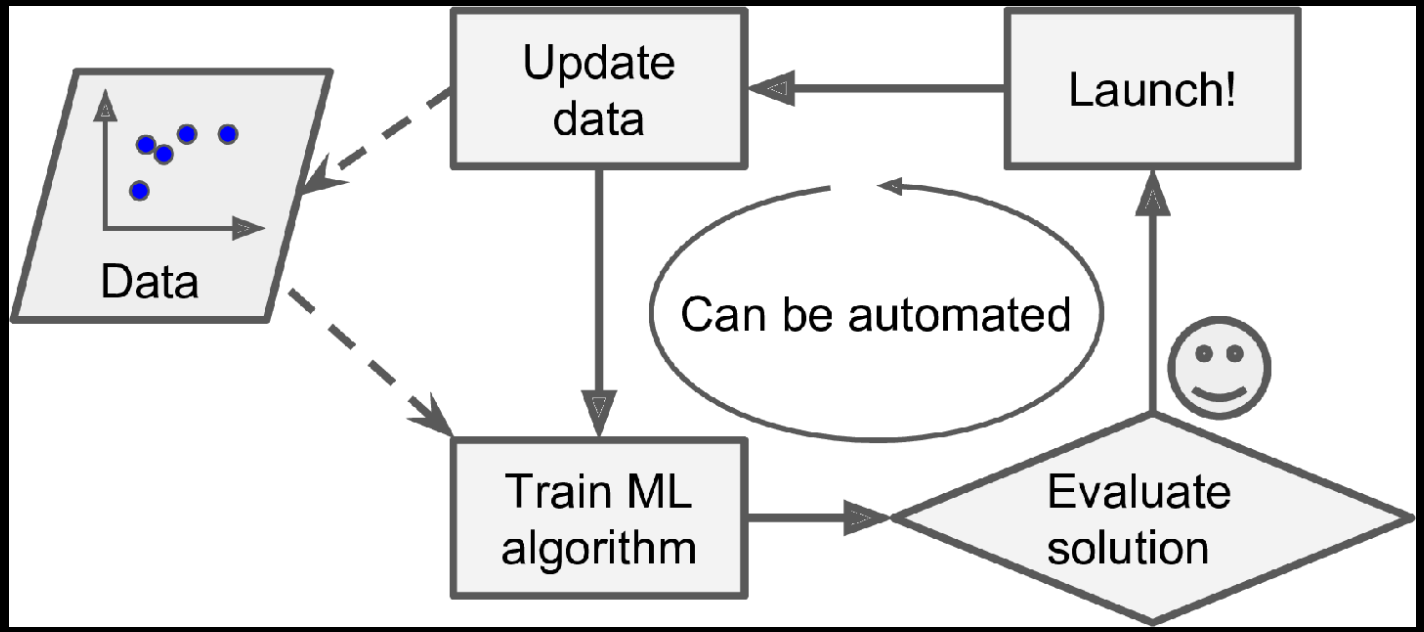


Figure 1-3. Automatically adapting to change

Why Use Machine Learning?

3). To help humans learn

- Example: generate list of words as best predictors

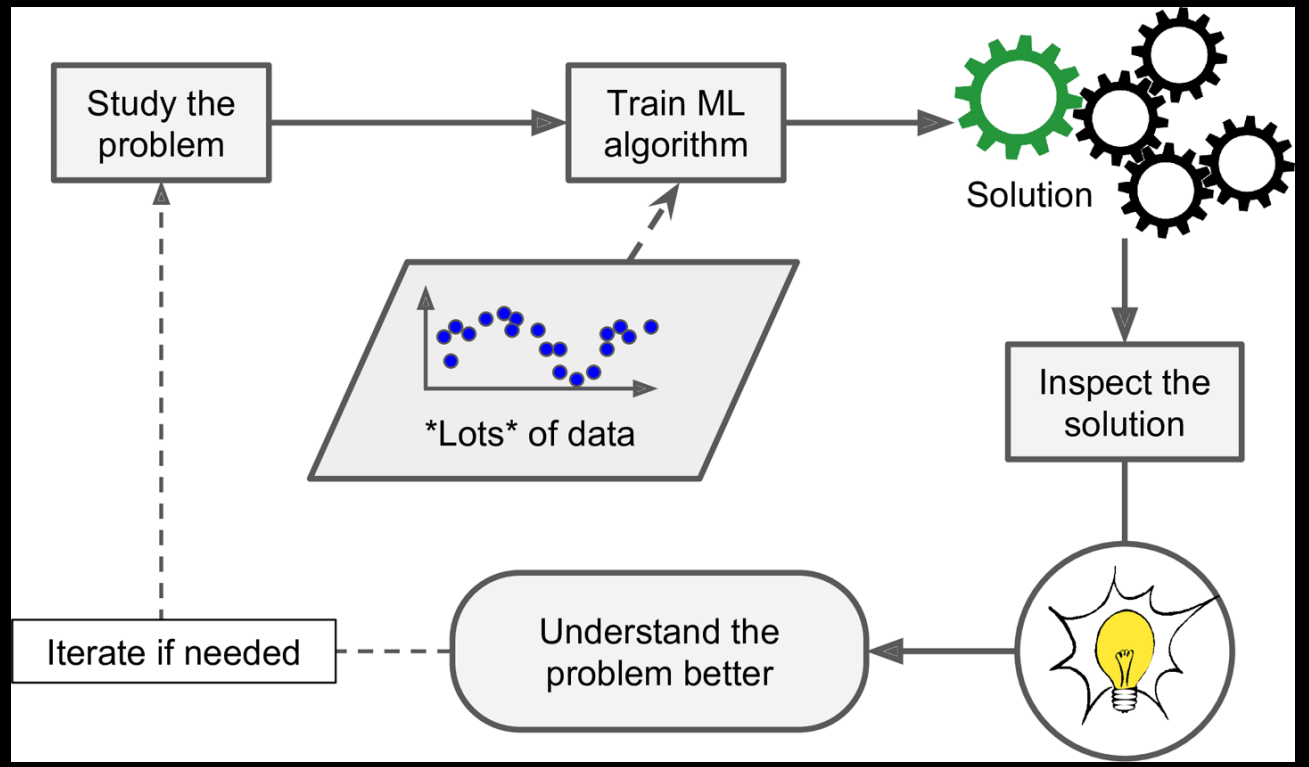
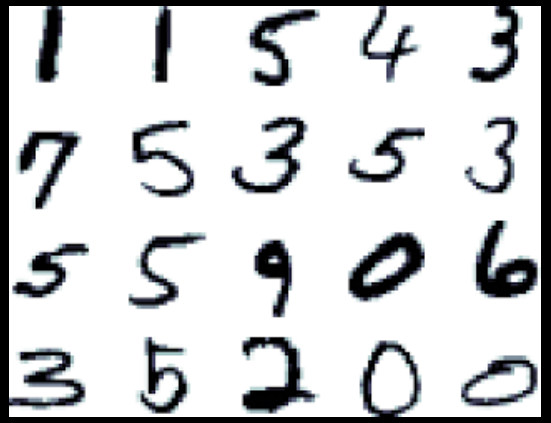


Figure 1-4. Machine Learning can help humans learn

Why Use Machine Learning?

4). Problem is too complex

- Example: hand-written digit recognition



Features & Labels

- Feature / attribute
  - A measurable property of an observable phenomenon
    - Mileage, age, brand etc
  - The number of features determine the dimensionality of the data set
- Label
  - A class identification variable
- Some examples …

Example: Iris flower dataset

- Introduced by Ronald Fisher in 1936 (aka Anderson's Iris dataset)
- measurements in centimeters of the variables sepal length and width and petal length and width for 50 flowers from each of 3 species of iris

Iris Versicolor

Iris Virginica

Iris Setosa

Fisher Iris (sepal length, sepal width, petal length, petal width, class)

5.3000	3.7000	1.5000	0.2000	setosa
5.0000	3.3000	1.4000	0.2000	setosa
7.0000	3.2000	4.7000	1.4000	versicolor
6.9000	3.1000	5.4000	2.1000	virginica

- This is dataset has 4 dimensions

Example: KDD dataset

- Internet traffic collected over some period of time

KDD (duration, protocol type, service, flag, src bytes, dst bytes, …, **class**)

0,tcp,http,SF,266,529,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,1,6,0.00,0.00,0.00,0.00,1.00,0.00,0.67,94,255,1.00,0.00,0.01,0.02,0.00,0.00,0.00,0.00,**normal**
0,tcp,ftp_data,SF,245,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,2,0.00,0.00,0.00,0.00,0.67,0.67,0.00,99,68,0.32,0.06,0.32,0.03,0.00,0.00,0.00,0.00,**normal**
0,tcp,http,SF,295,465,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,12,0.00,0.00,0.00,0.00,1.00,0.00,0.17,69,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,**normal**
0,tcp,gopher,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,44,12,1.00,1.00,0.00,0.00,0.27,0.11,0.00,255,12,0.05,0.08,0.00,0.00,1.00,1.00,0.00,0.00,**neptune**
0,tcp,http,SF,228,6834,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,7,0.00,0.00,0.00,0.00,1.00,0.00,0.57,18,19,1.00,0.00,0.06,0.11,0.00,0.00,0.00,0.00,**normal**
0,icmp,ecr_i,SF,1032,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,511,511,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,212,0.83,0.02,0.83,0.00,0.00,0.00,0.00,0.00,**smurf**

- This is dataset has 41 dimensions

Different Types of Machine Learning Systems

- Do they need human supervision?
  - Supervised Learning
  - Unsupervised Learning
  - Semi-supervised Learning
  - Reinforcement Learning
- Do they learn incrementally on the fly?
  - Online vs batch learning
- Do they make predictions based on comparing data points or based on predictive models?
  - Instance-based vs model-based learning

Different Types of Machine Learning Systems

- Do they need human supervision?
  - Supervised Learning
  - Unsupervised Learning
  - Semi-supervised Learning
  - Reinforcement Learning

Supervised Learning

- Using training data, train the model. Then make predictions.

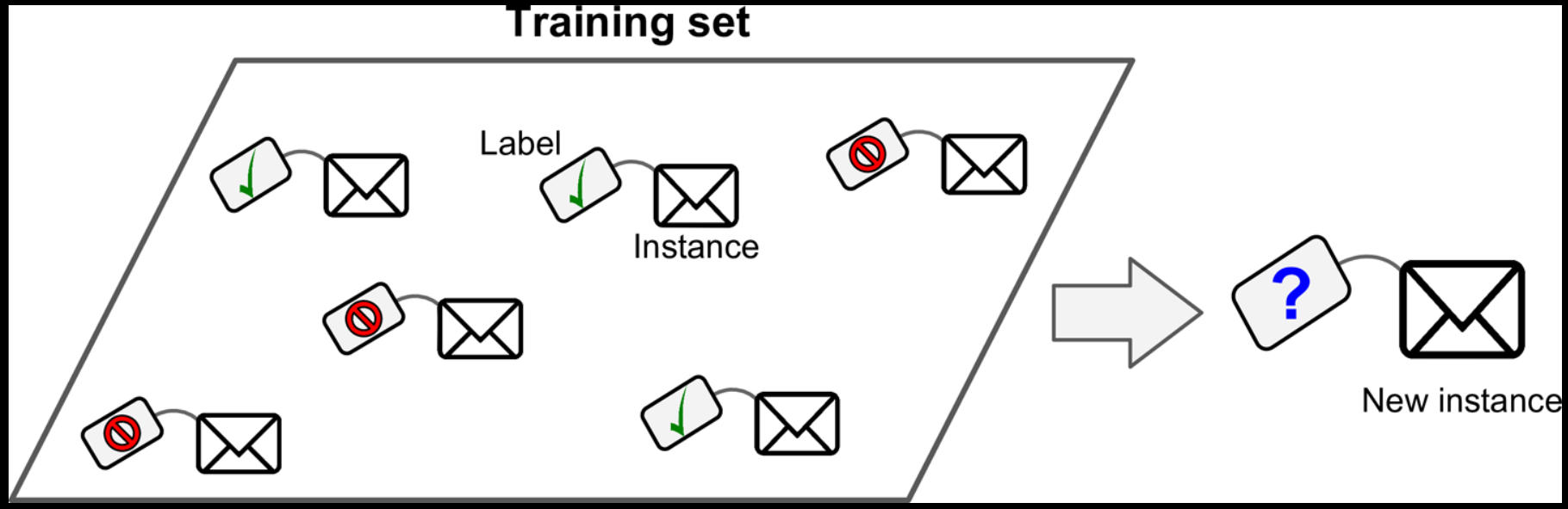


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

Supervised Learning

- Classification: output is discrete

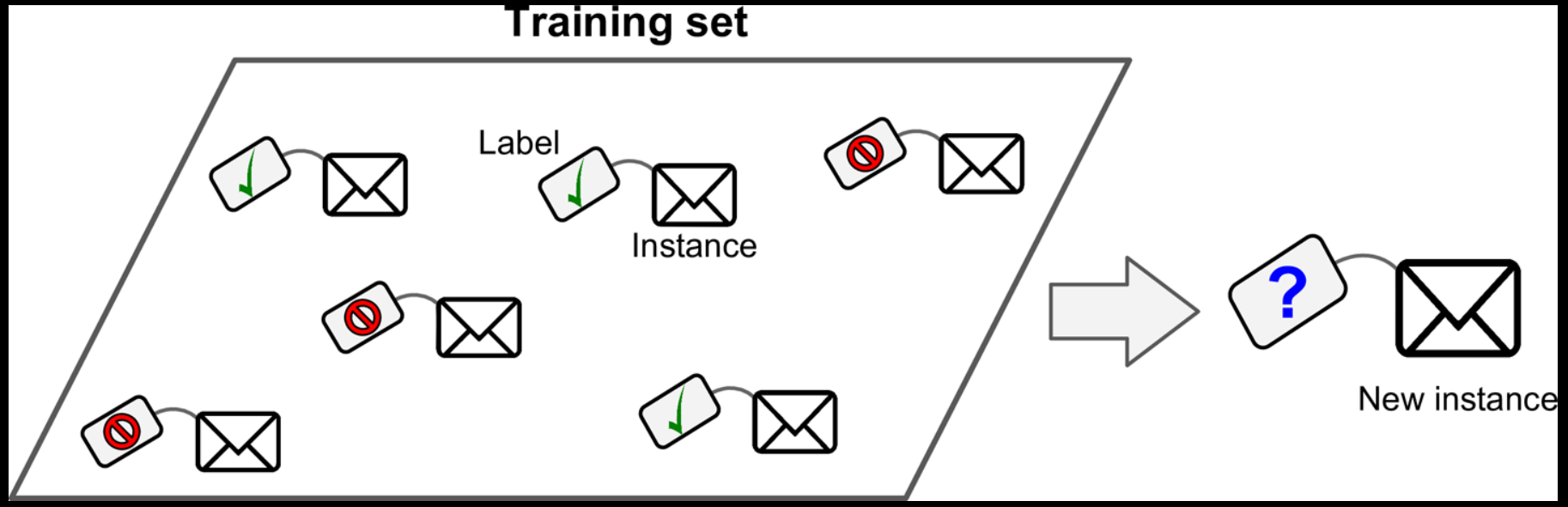


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

Supervised Learning

- Regression: output is continuous

Figure 1-6. Regression

Unsupervised Learning

- Training data is unlabeled

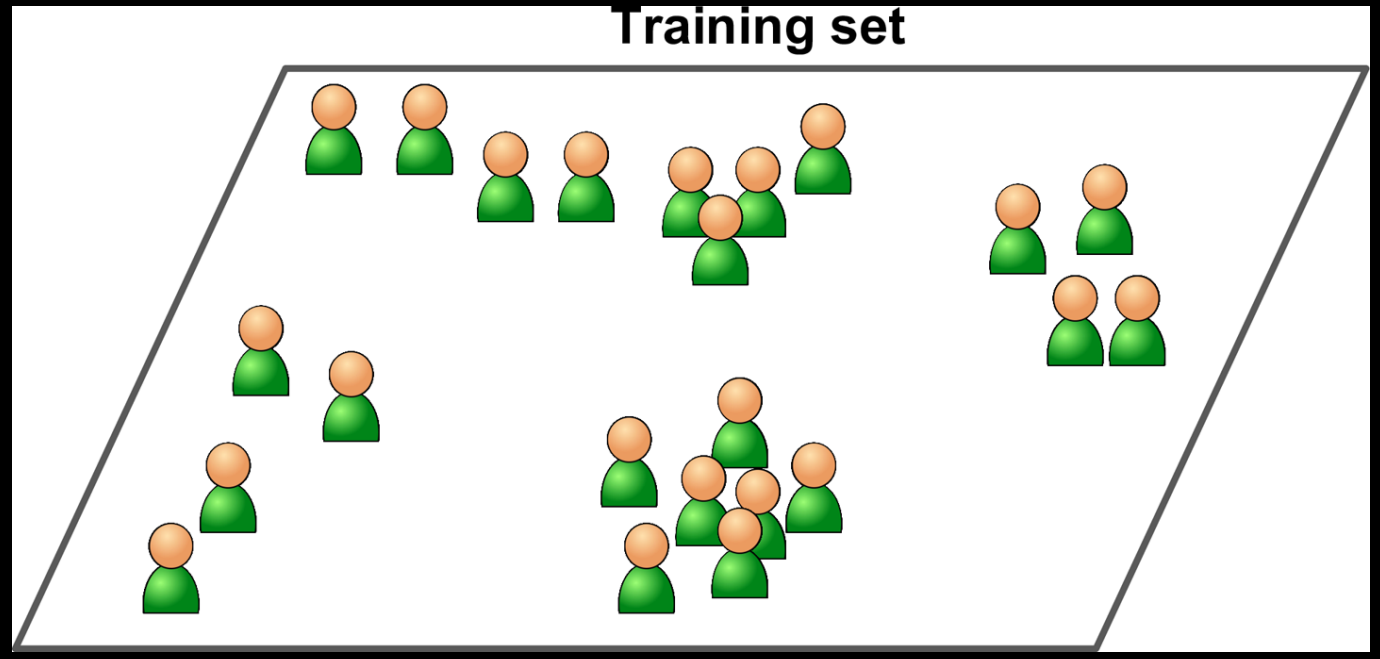


Figure 1-7. An unlabeled training set for unsupervised learning

Unsupervised Learning

- Clustering

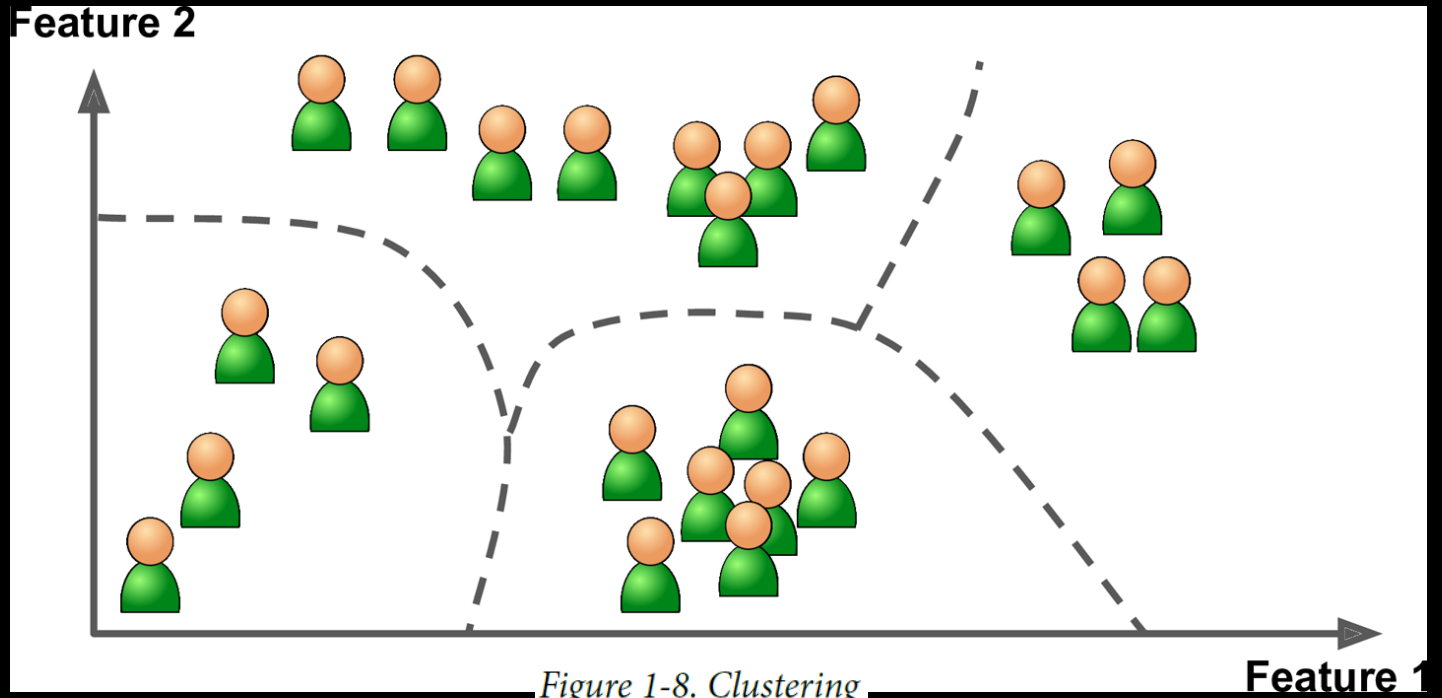


Figure 1-8. Clustering

Unsupervised Learning

- Dimensionality reduction & visualization

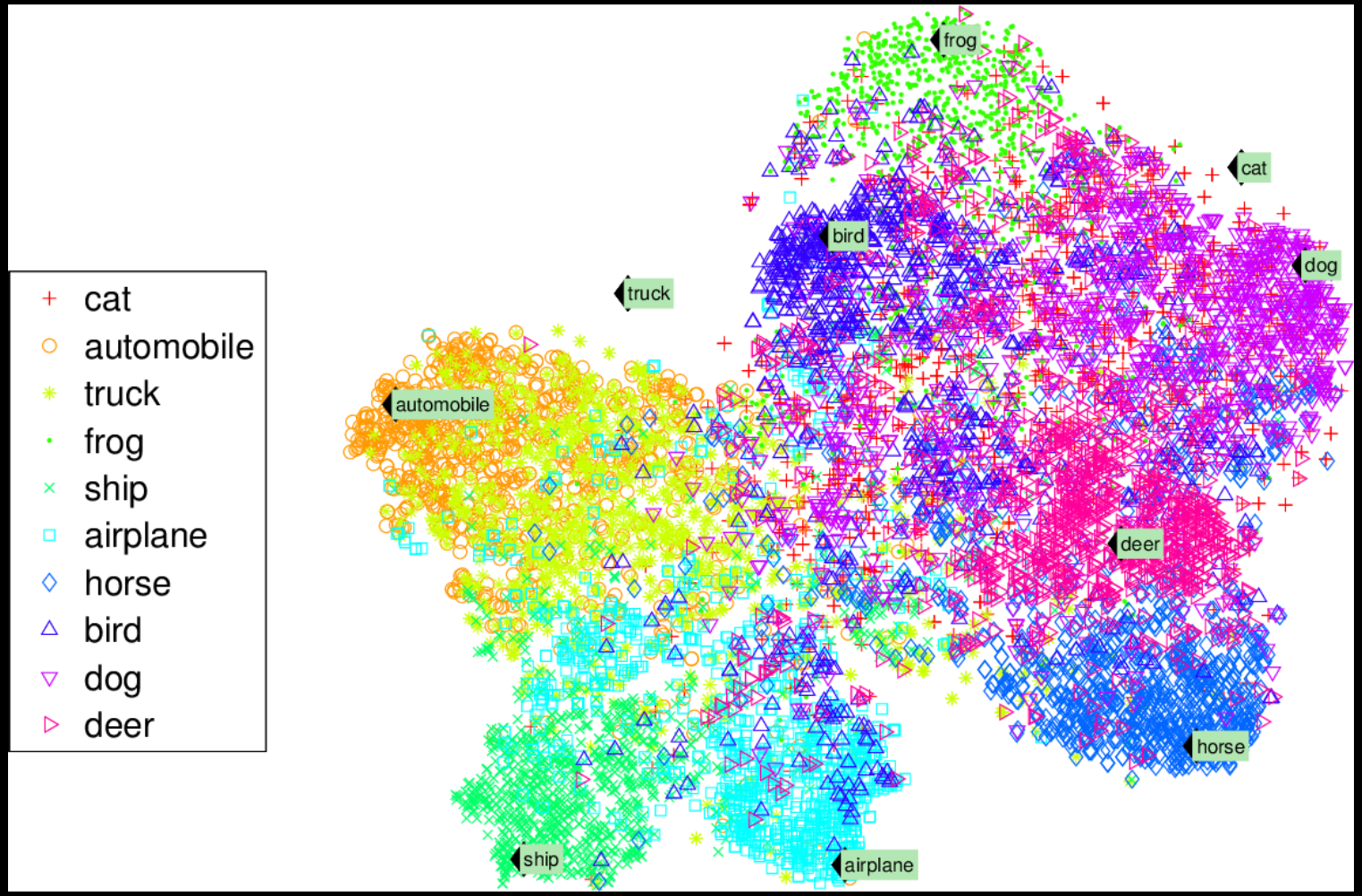


Figure 1-9. Example of a t-SNE visualization highlighting semantic clusters[3]

Unsupervised Learning

- Anomaly detection

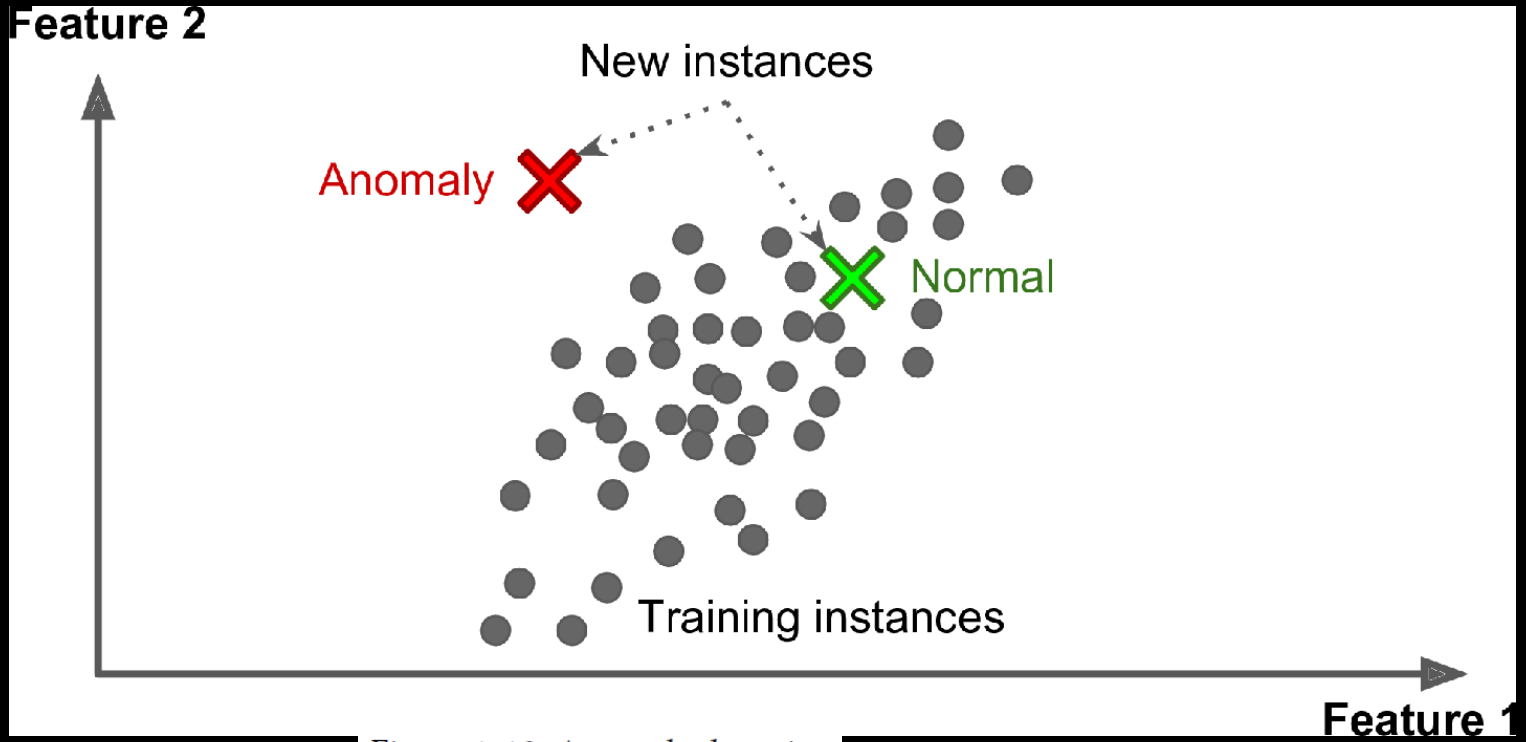


Figure 1-10. Anomaly detection

Unsupervised Learning

- Association rule learning
  - Find relations in large datasets
  - Example: people who buy BBQ sauce and potato chips tend to buy steak

Semi-supervised Learning

- A little bit of labeled data & a lot of unlabeled data

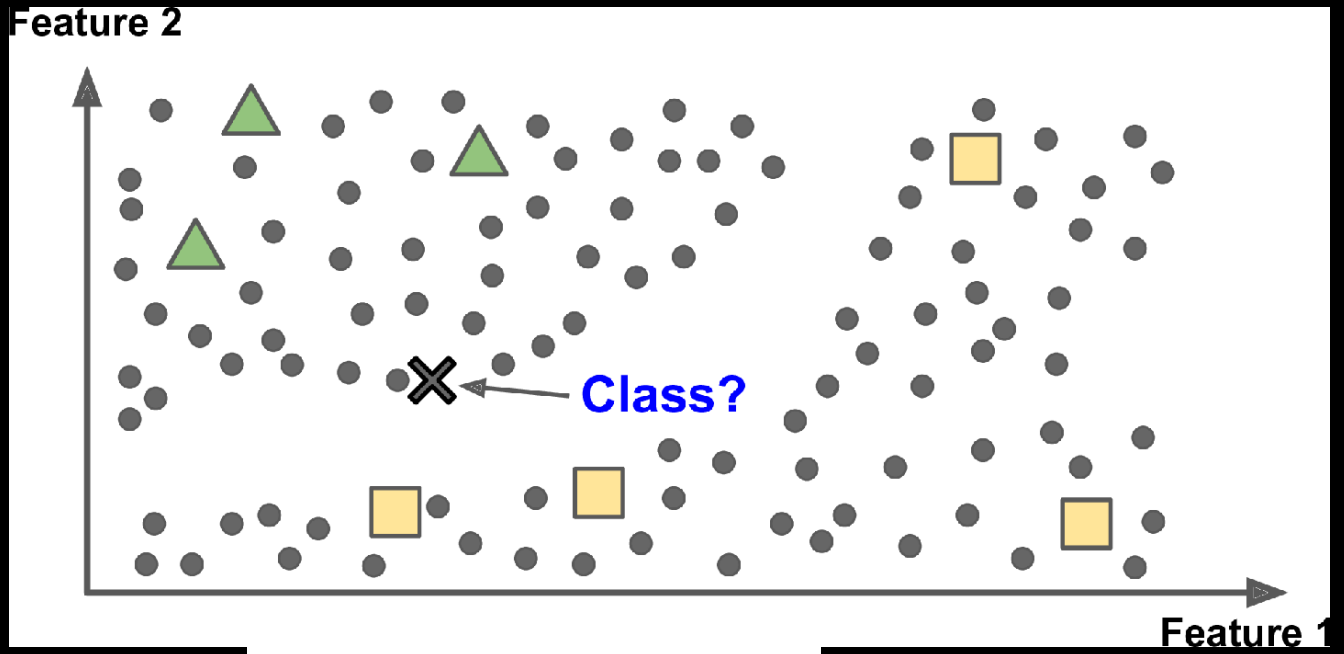


Figure 1-11. Semisupervised learning

Reinforcement Learning

- Agent observes the environment, performs actions based on a policy, gets rewards
- Goal is to learn and maximize the reward over time

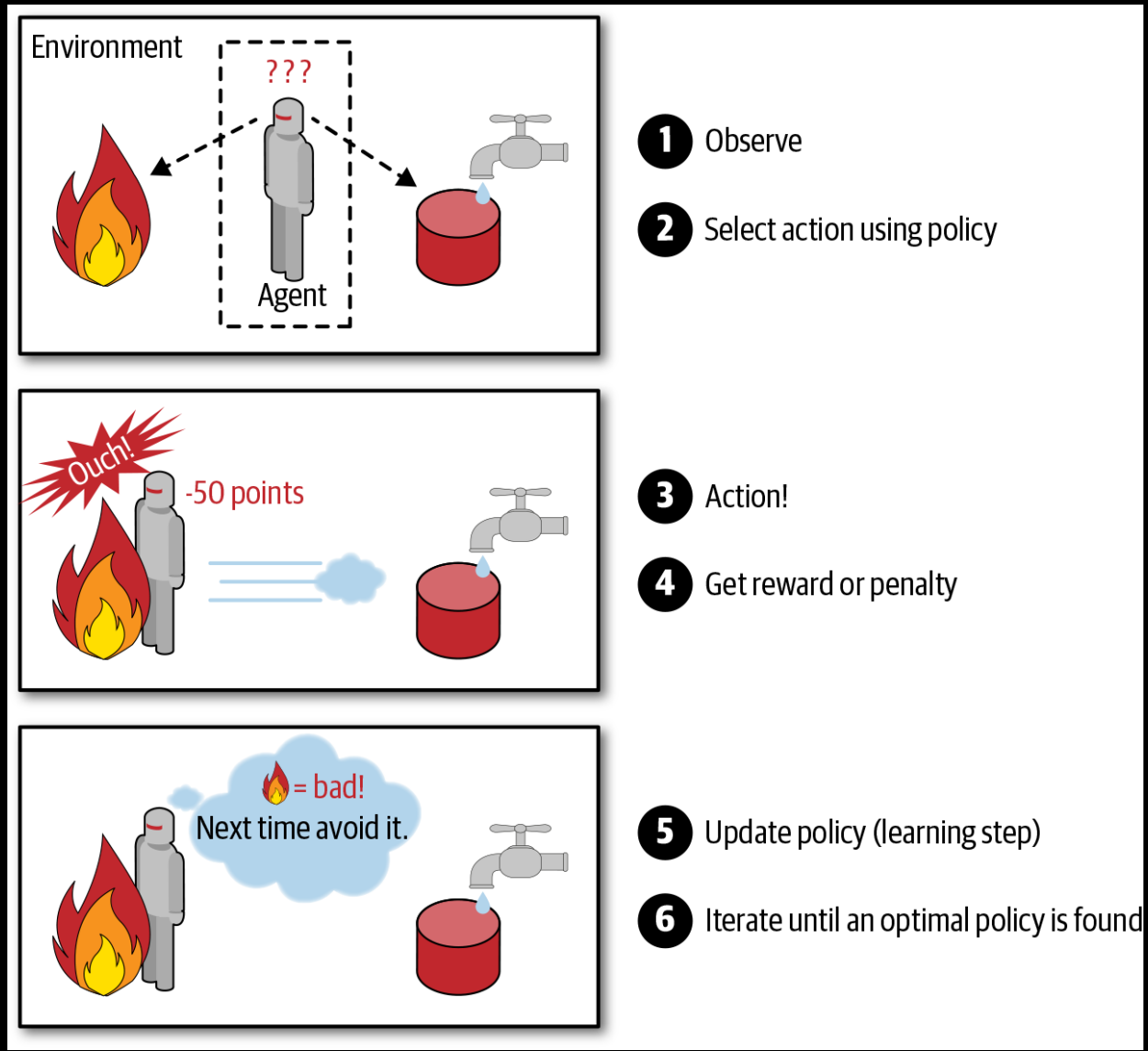


Figure 1-12. Reinforcement Learning

Different Types of Machine Learning Systems

- Do they learn incrementally on the fly?
  - Online vs batch learning

- Batch learning
  - Train the model using all available data all at once
  - Done offline, i.e., called offline learning
- Online learning
  - Train the model by individual example or in small groups (mini-batches)
  - Done on the fly as data arrives

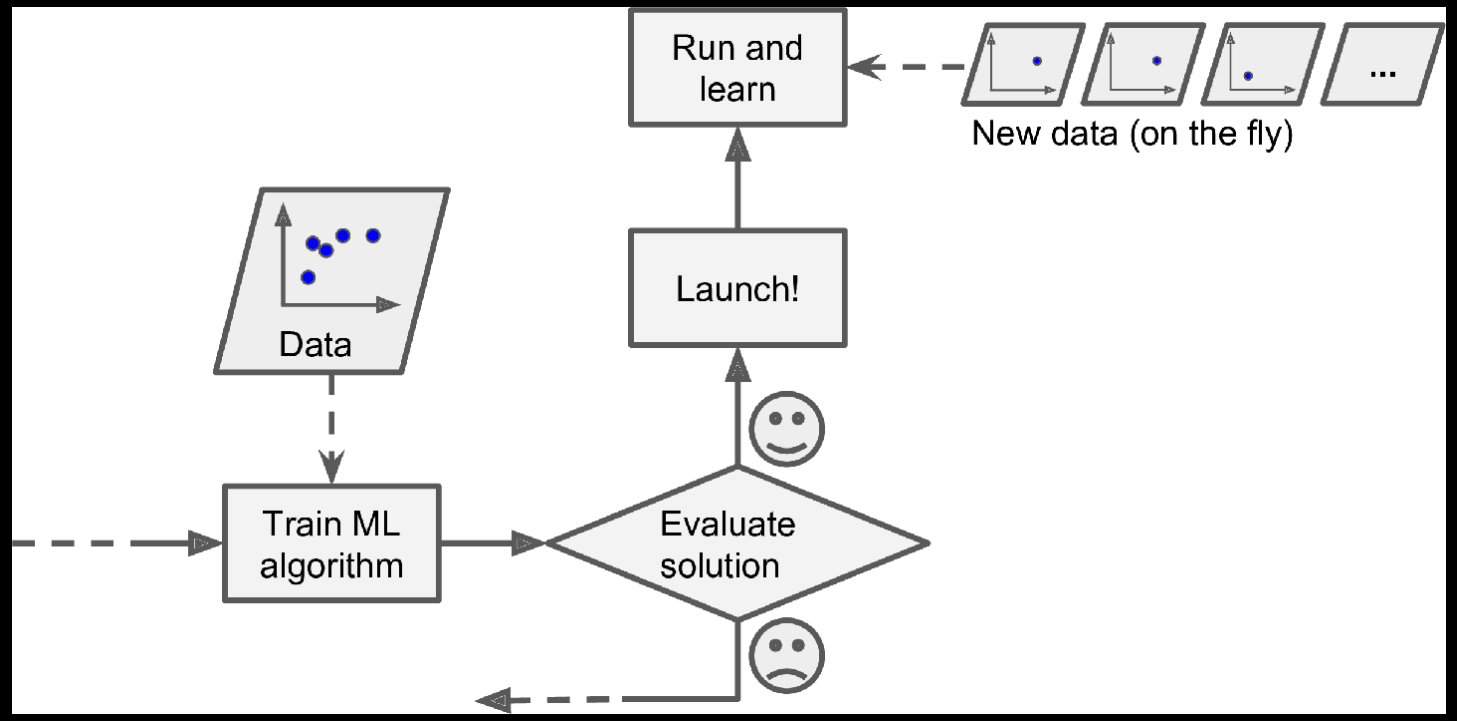


Figure 1-13. Online learning

- Out-of-core learning
  - Done offline
  - Is not online learning

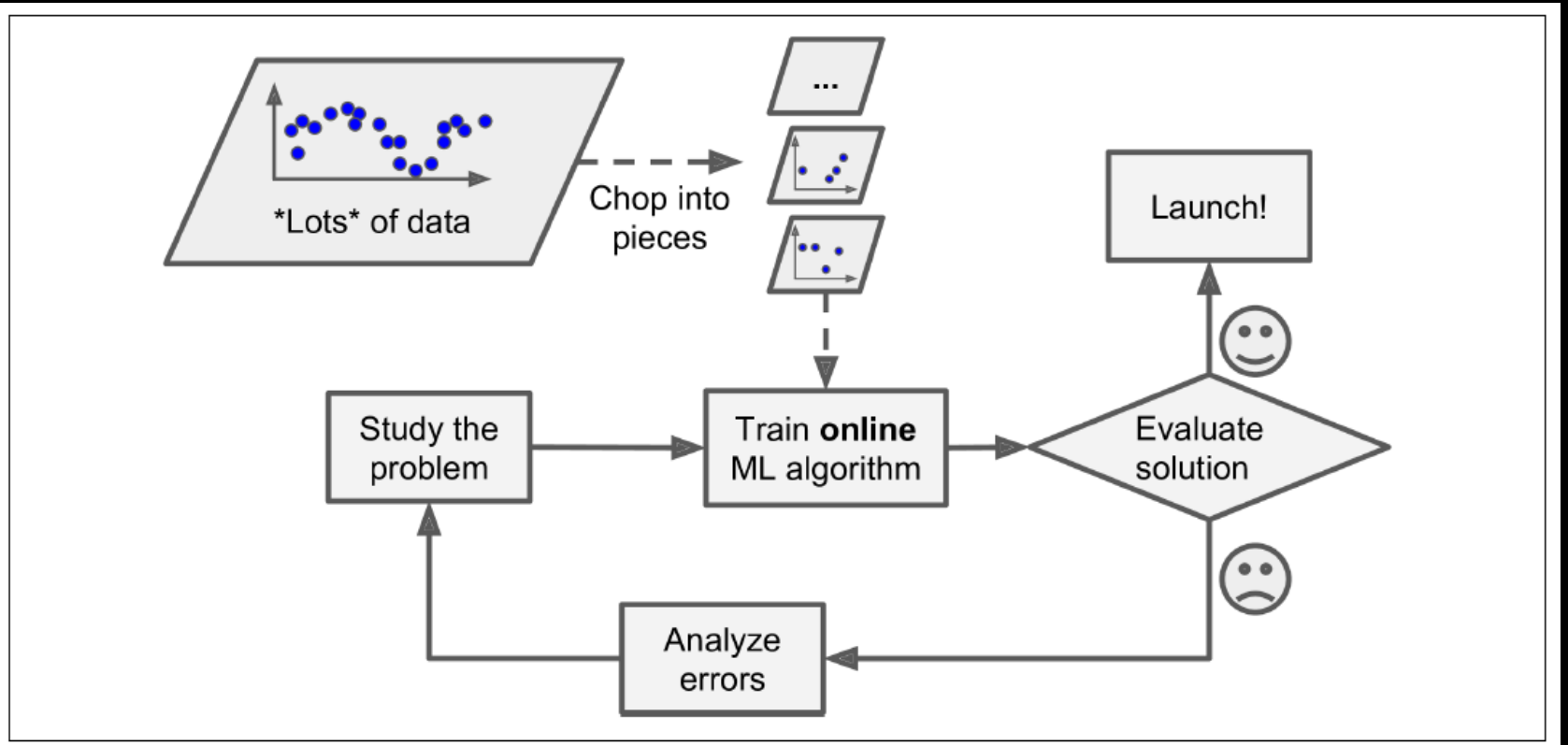


Figure 1-14. Using online learning to handle huge datasets

Different Types of Machine Learning Systems

- Do they make predictions based on comparing data points or based on predictive models?
  - Instance-based vs model-based learning

Instance-based Learning

- Employs a measure of similarity to determine the class label

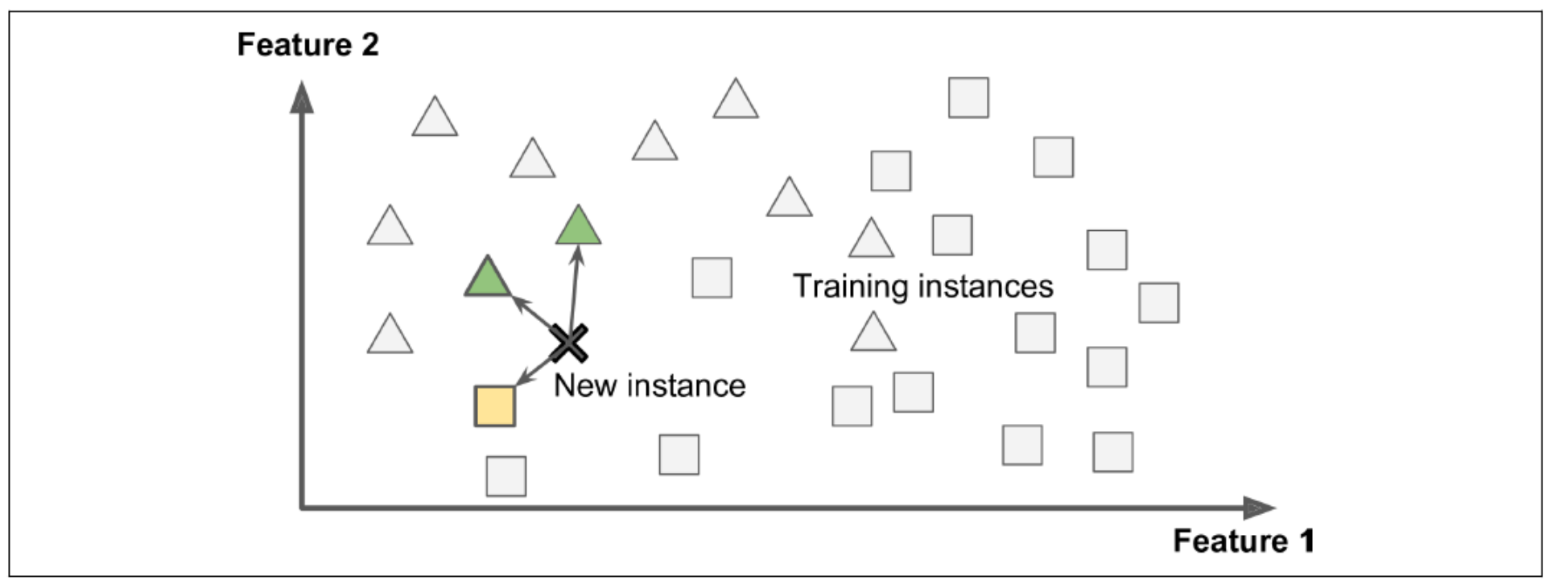


Figure 1-15. Instance-based learning

Model-based Learning

- Builds a model using training examples
- Uses the model to make predictions

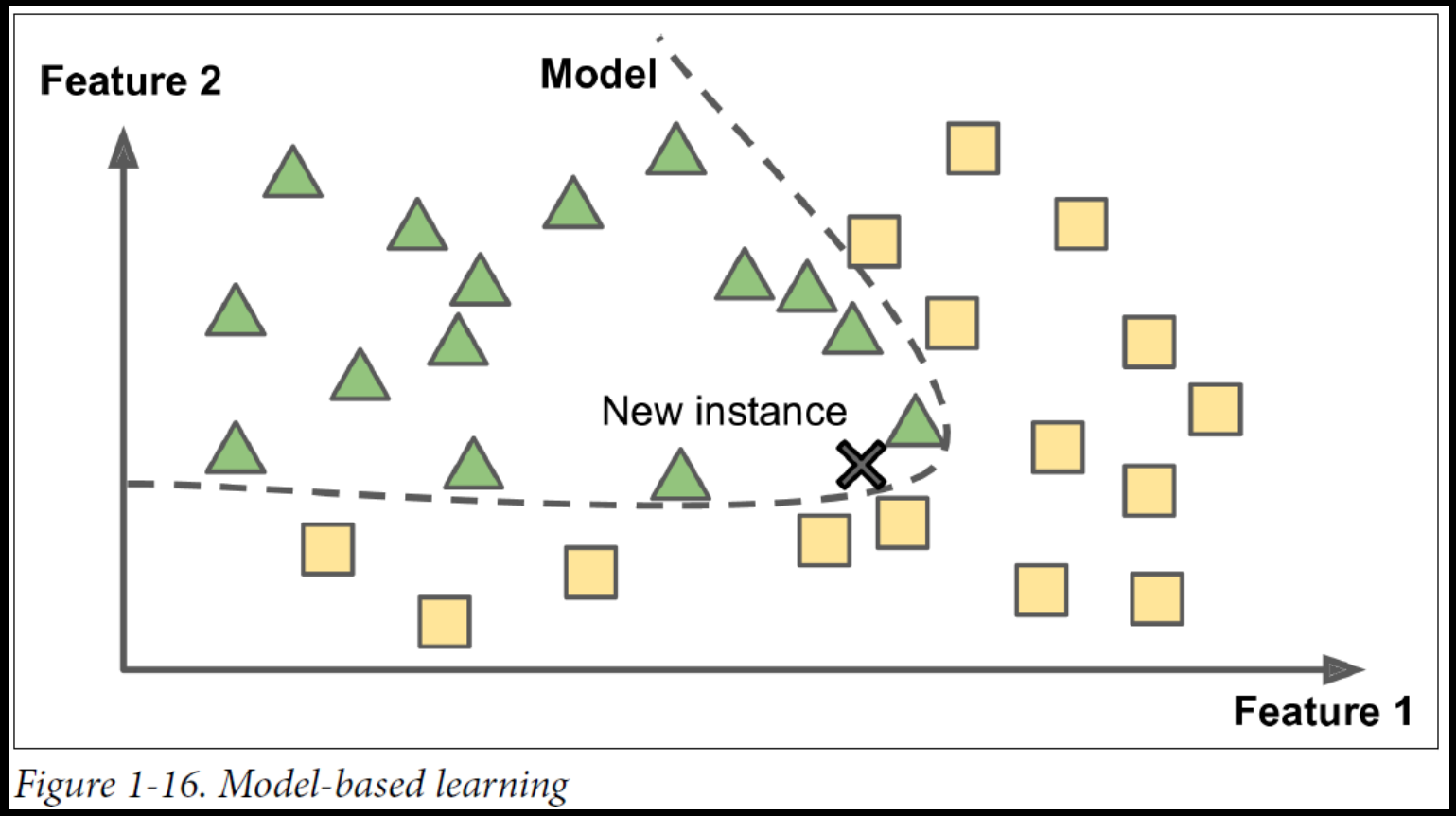


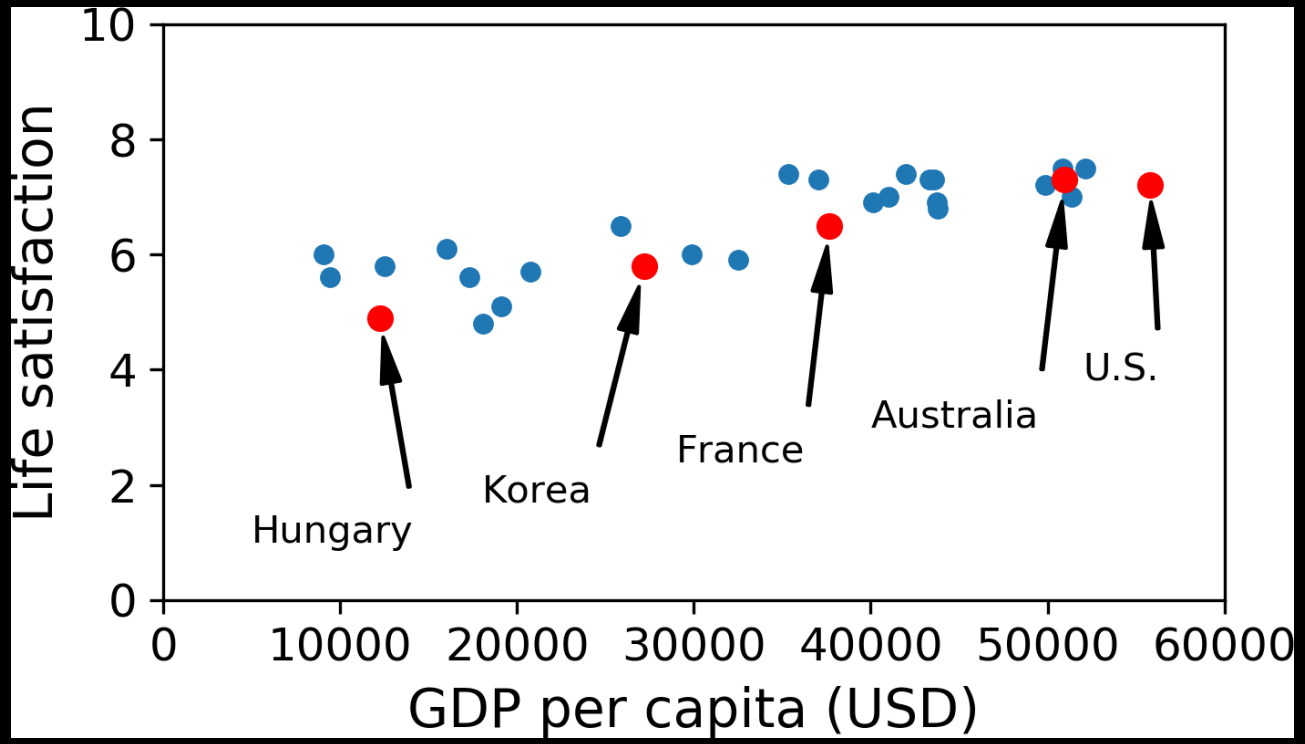
Figure 1-16. Model-based learning

Example: GDP vs Happiness

- Use Better Life Index data from OECD's database
- Is there a trend?

Table 1-1. Does money make people happier?

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2



Linear Regression Algorithm

- Can we build a model?

Equation 1-1. A simple linear model

$$\text{life\_satisfaction} = \theta_0 + \theta_1 \times \text{GDP\_per\_capita}$$

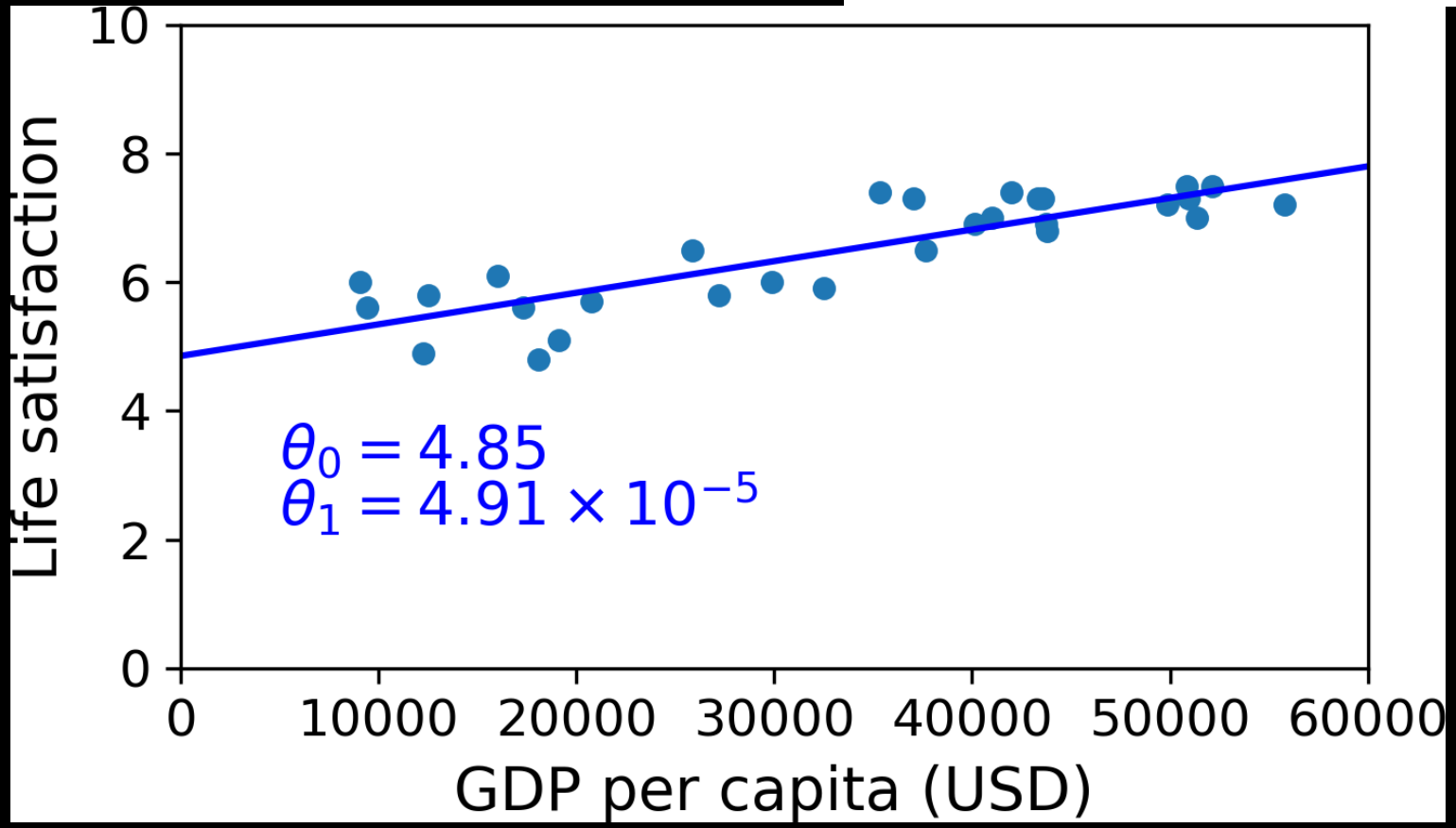


Figure 1-19. The linear model that fits the training data best

- Want to know how happy people in Cyprus are?

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model

# Load the data
oecd_bli = pd.read_csv("oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv("gdp_per_capita.csv",thousands=',',delimiter='\t',
                             encoding='latin1', na_values="n/a")

# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select a linear model
model = sklearn.linear_model.LinearRegression()

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

# Make a prediction for Cyprus
X_new = [[22587]]  # Cyprus' GDP per capita
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2

Main Challenges of Machine Learning: Bad Data vs Bad Algorithms

- Bad Data
  - Insufficient Quantity of Training Data
  - Non-representative Training Data
  - Poor Quality Data
  - Irrelevant Features
- Bad Algorithms
  - Overfitting the Training Data
  - Underfitting the Training Data

Insufficient Quantity of Training Data

- The Unreasonable Effectiveness of Data
  - Data matters more than algorithms for complex problems

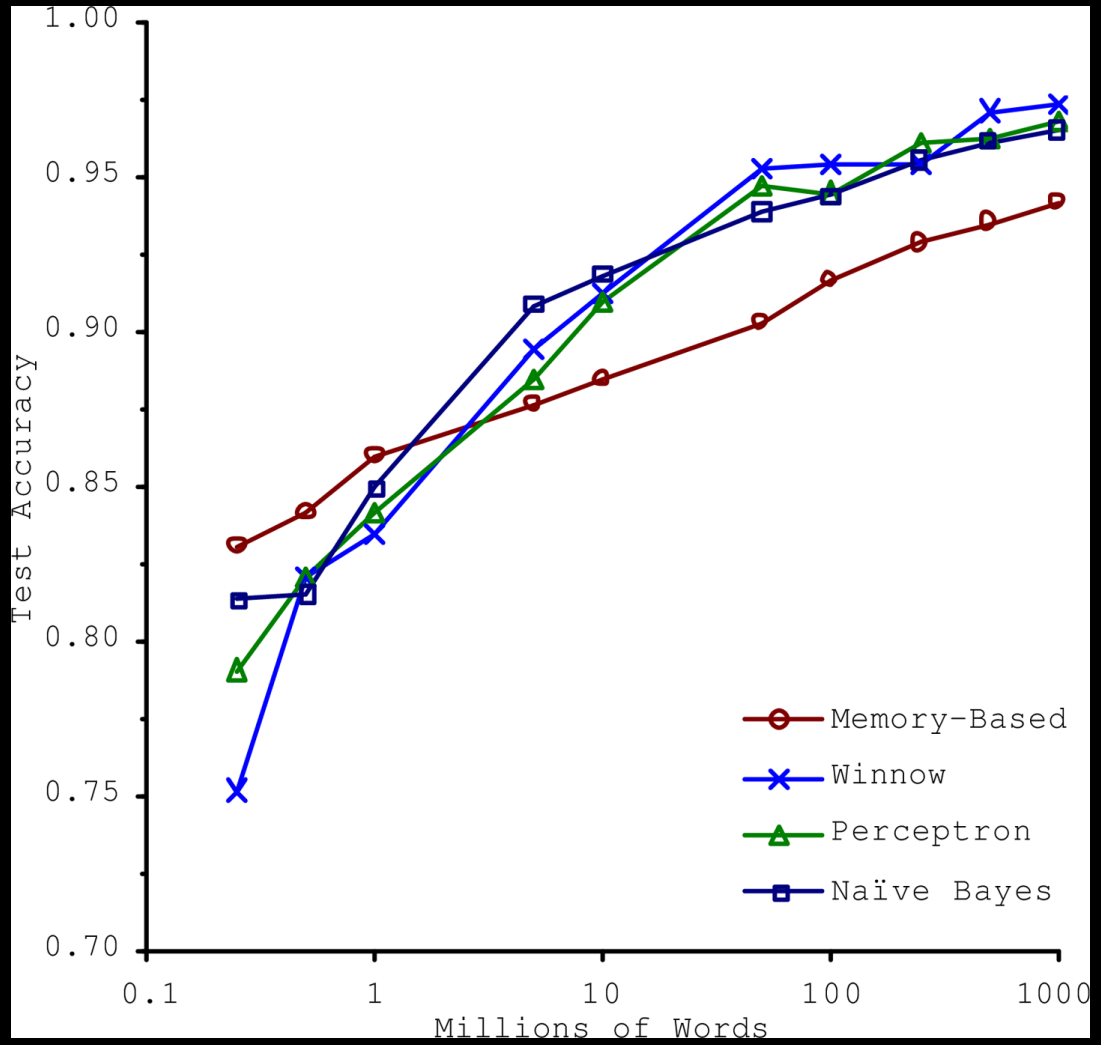


Figure 1-20. The importance of data versus algorithms

Non-representative Training Data

- May lead to a false conclusion
  - Dotted line: missing a few countries (previous model)
  - Solid line: with missing countries included
  - Famous Example of Sampling Bias – Landon vs Roosevelt 1936 Presidential Election

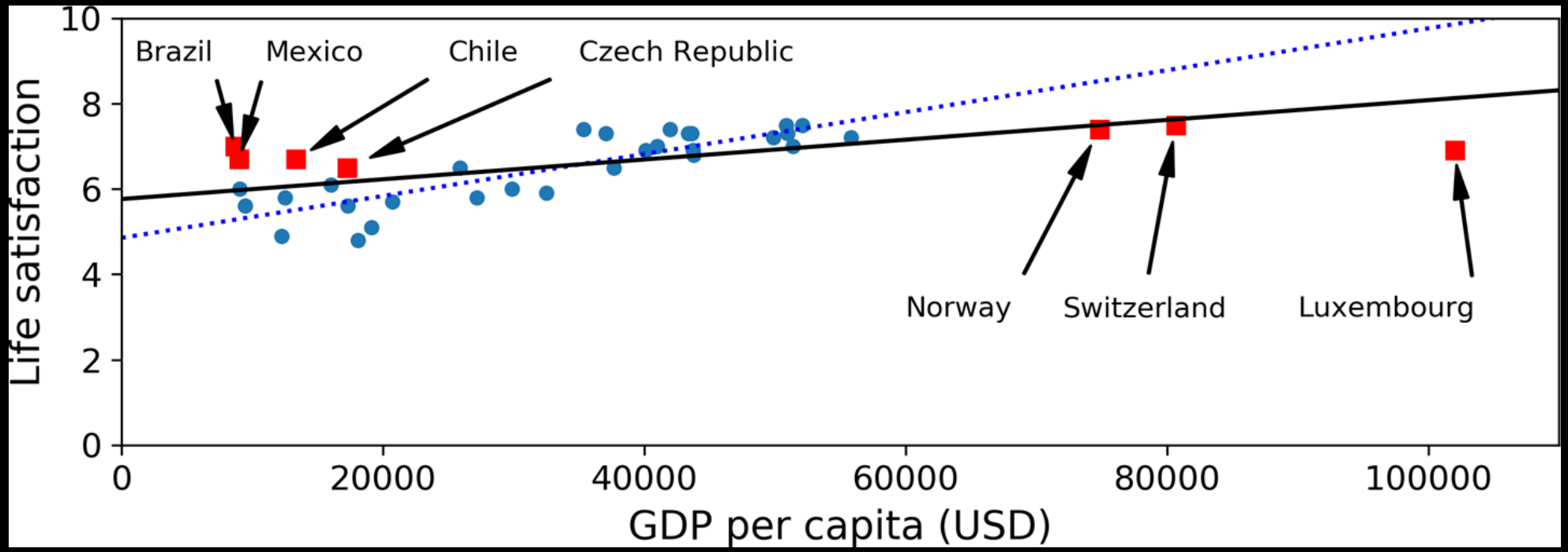


Figure 1-21. A more representative training sample

Poor Quality Data

- Training data is full of errors, outliers and noise
- Missing features from the training data
- Spend time to clean up the data first

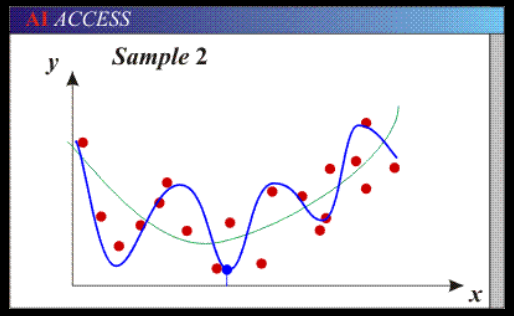
Irrelevant Features

- Garbage In → Garbage Out

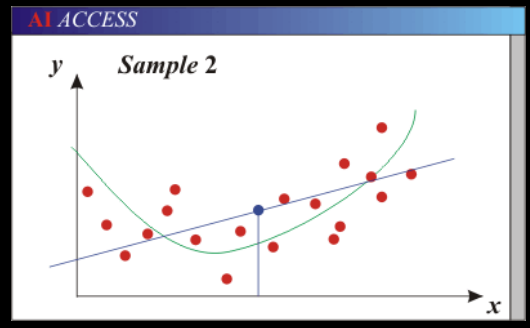
- Feature Engineering
  - Feature Selection
  - Feature Extraction

Bad Models: Overfitting & Underfitting

- Generalization
  - How well does a learned model generalize from the training data to new instances?
  - Causes of error: overfitting & underfitting
  - Overfitting: model is too complex
  - Underfitting: model is too simple



- Overfitting: model is too complex



- Underfitting: model is too simple

Mitigation for Overfitting Problems

- Gather mode training data & reduce the noise
- Simplify the model by reducing the number of features or by regularization

$$\text{life\_satisfaction} = \theta_0 + \theta_1 \times \text{GDP\_per\_capita}$$

  - Allow θ_0 and θ_1 to control the slope but force θ_1 small controlled by a *hyperparameter*

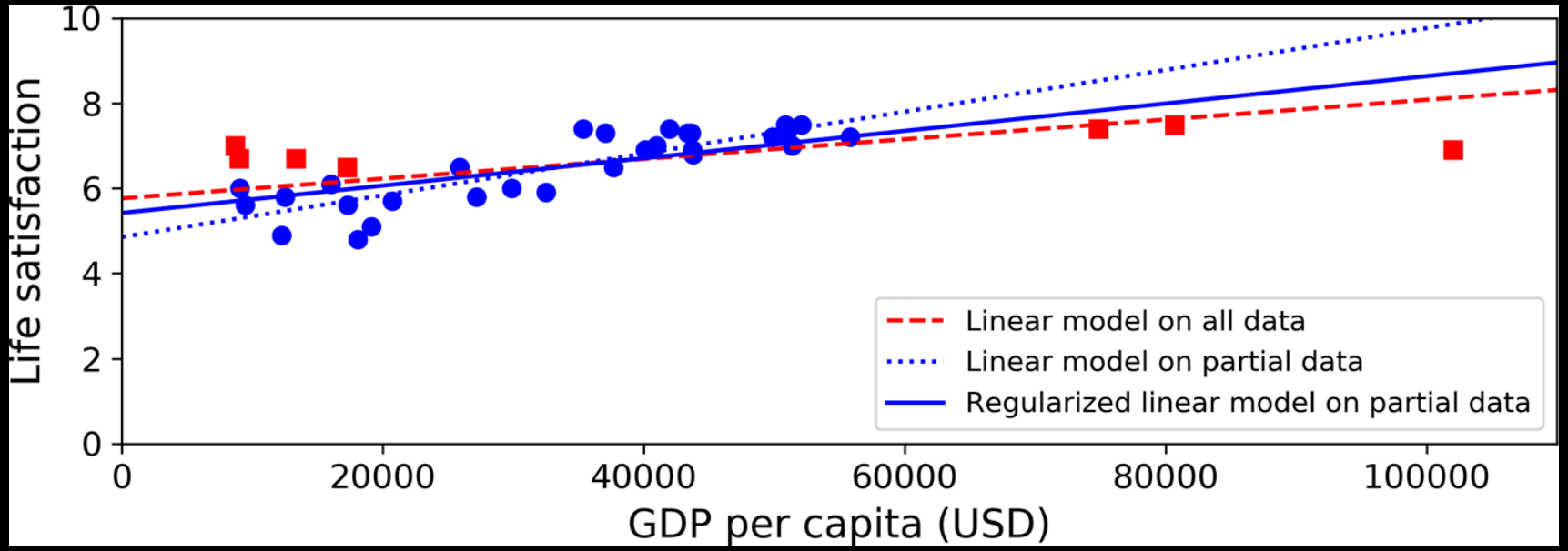


Figure 1-23. Regularization reduces the risk of overfitting

- *Parameter vs hyperparameter*

Mitigation for Underfitting Problems

- Find more complex model
- Feature engineering
- Reduce regularization

Testing and Validation

- How do you know that the model will generalize well to new cases?
  - Try it in the field – bad idea!
- A better option: split data into *training set* and *test set* and measure the *generalization error* (*out-of-sample error*)
  - Common practice 80/20 split
  - Low training error but high generalization error → model overfits training data

Hyperparameter Tuning & Model Selection

- If you have multiple models, how do you choose which one?
  - Look at the generalization error
- Now you want to improve the model you selected using regularization. How do you choose the value of the regularization parameter?
  - Experiment different values and look at the generalization error
  - Is this the best solution?
  - Better solution: apply holdout validation

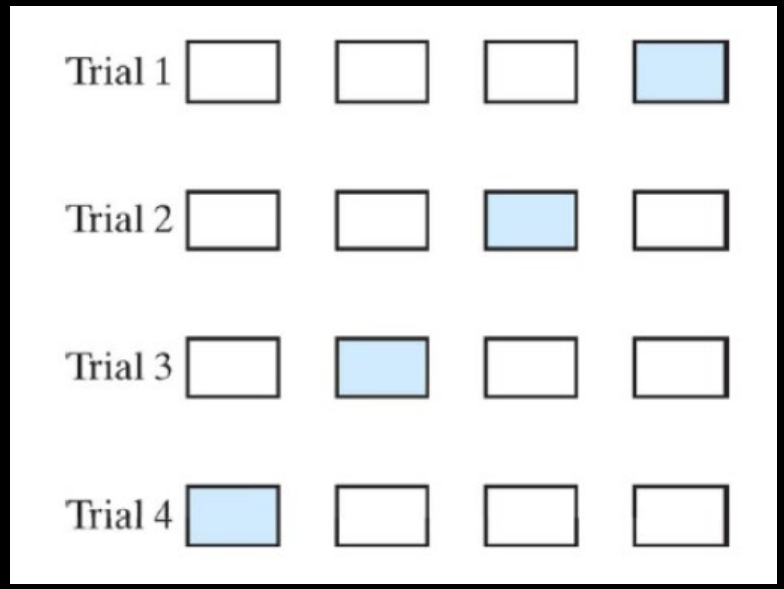
Holdout Validation

- Hold out part of the *training set* as *validation set* (*development set*)
- Train multiple models on reduced *training set*
- Choose the model that performs the best on *validation set*
- Train this model on full *training set*
- Evaluate this model on *test set*

- What if the validation set is too small or too large?
- Solution: perform repeated cross-validation using many small *validation sets*

K-Fold Cross Validation

- Divide training data into K subsets
- 1 subset for validation
- K-1 subsets for training
- Repeat until each subset is used once
- Average validation error



Data Mismatch

- Golden rule: *validation* and *test* sets must be representative of the data used in production
- Suppose we want to create a mobile app to take pictures of flowers and classify their species.
  - Take 10K pictures of flowers (representative samples)
  - Put 50% of app pictures as validation set and 50% as test set
  - Download 1M pictures of flowers from the Web (are they representative?)
  - Train the model on the web pictures
  - Turns out the performance is poor on the validation set
  - How do you know that it's due to overfitting the *training* set or mismatch between the *training* set (web pictures) and the *validation* set (app pictures)?
  - Solution: use *train-dev* set

Train-Dev Set

- Hold out part of the training set (web pictures) as *train-dev* set
- Train the model on the rest of the training set
- Evaluate the model on the train-dev set
  - Good result → model is not overfitting the training set. Problem comes from the data mismatch
  - Poor result →the model overfits the training set

No Free Lunch Theorem (David Wolpert)

- If you make no assumptions about the data, then there is no reason to prefer one model over any other
  - The only way to know which model is best is to evaluate them all
  - In practice, make some reasonable assumptions about the data and evaluate only a few reasonable models
  - For simple problem, evaluate linear models with different regularization levels
  - For complex problem, evaluate various neural networks