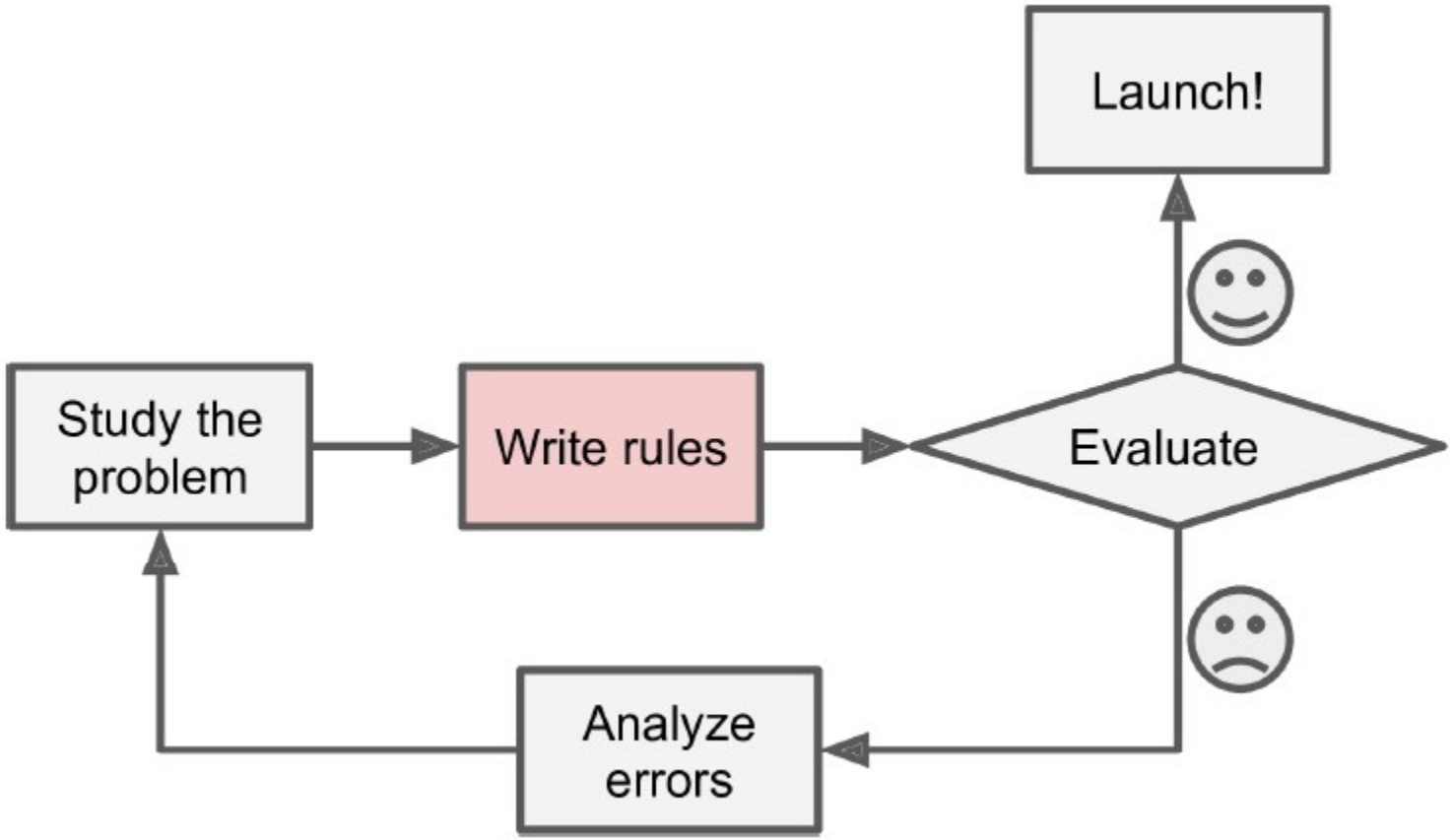
CSE 445 Lecture 1

Machine Learning Intro

What is machine learning?

- Machine Learning is the science (and art) of programming computers so they can *learn from data*
- Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed
- For example, spam filter is a Machine Learning program
  - Flag spam given examples of spam emails (e.g., flagged by users)
  - And examples of regular (non-spam) emails
  - The examples that the system uses to learn are called the *training set*
  - Each training example is called a *training instance* (or *sample*).

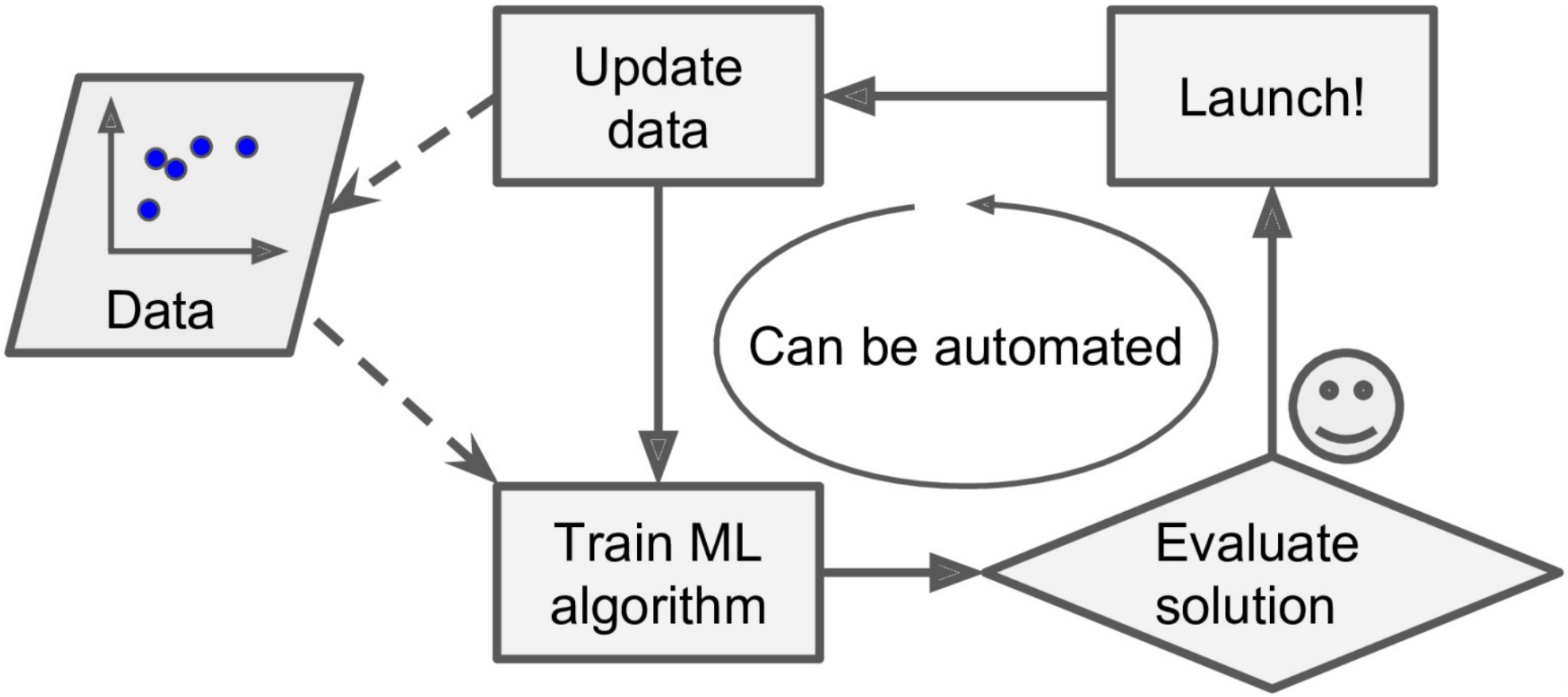
Solution without machine learning



Why use machine learning?

- How would we write a spam filter using traditional programming techniques?
- First you would look at what spam typically looks like
  - Some words or phrases (such as “4U,” “credit card,” “free,” and “amazing”) tend to come up a lot in the subject
  - Perhaps a few other patterns in the sender’s name, the email’s body, and so on.
- We would write a detection algorithm for each of the patterns that we noticed
- Our program would flag emails as spam if a number of these patterns are detected
- Since the problem is not trivial, your program will likely become a long list of complex rules— hard to maintain

Why machine learning? Adapts to changes.



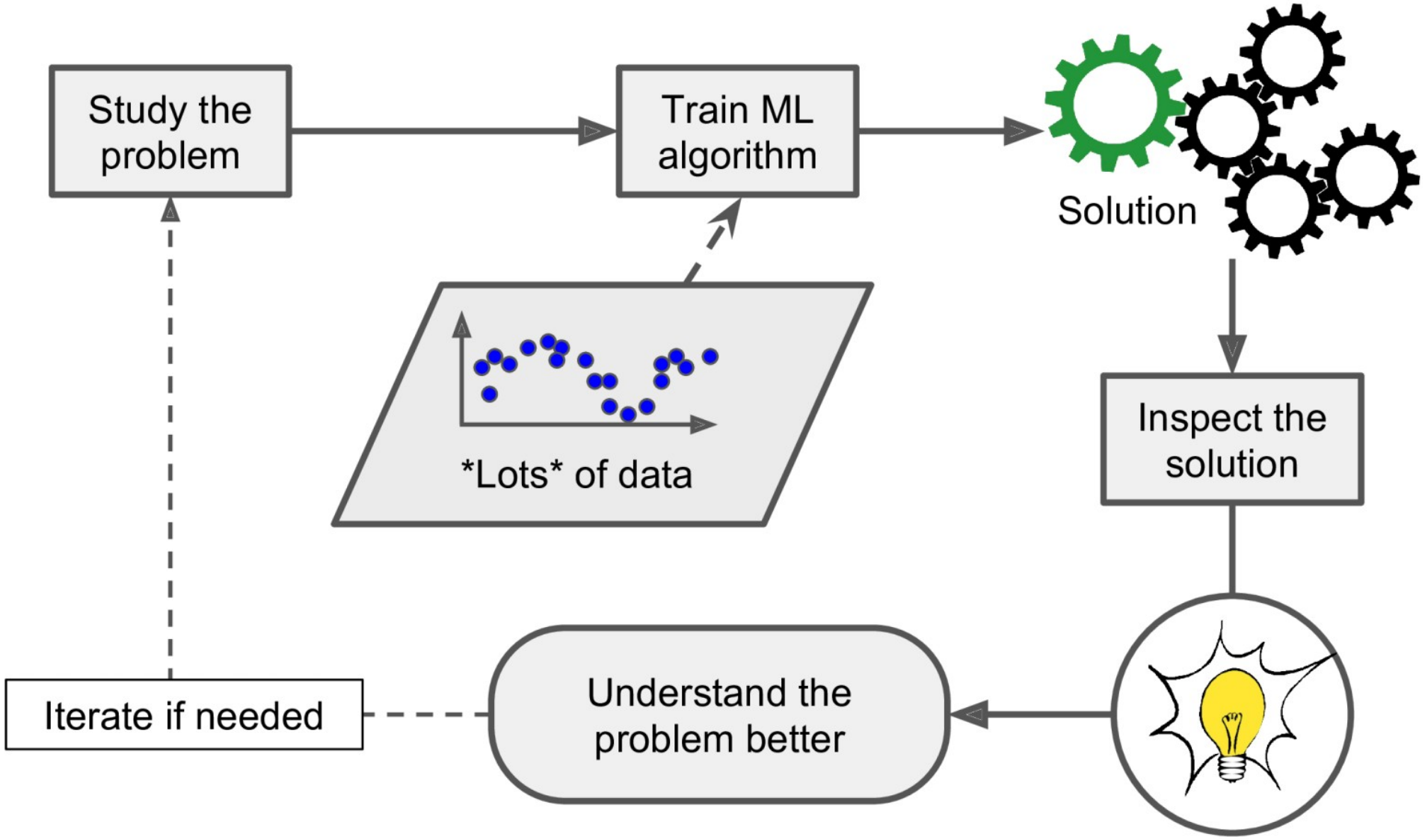
Why use machine learning?

- A spam filter based on Machine Learning techniques automatically learns which words and phrases are good predictors
- Moreover, if spammers notice that all their emails containing “4U” are blocked, they might start writing “For U” instead
- A spam filter using traditional programming techniques would need to be updated to flag “For U” emails
- In contrast, a spam filter based on Machine Learning techniques automatically notices that “For U” has become unusually frequent in spam flagged by users, and it starts flagging them without your intervention

Why machine learning?

- Some problems are too complex to program using traditional approaches or have no known algorithm, for example speech recognition
- Suppose we want to start simple and write a program capable of distinguishing the words “one” and “two”
- You might notice that the word “two” starts with a high-pitch sound (“T”), so you could hardcode an algorithm that measures high-pitch sound intensity and use that to distinguish ones and twos
- Obviously, this technique will not scale to thousands of words spoken by millions of very different people in noisy environments and in dozens of languages

Why machine learning? Gain insights.



Why machine learning?

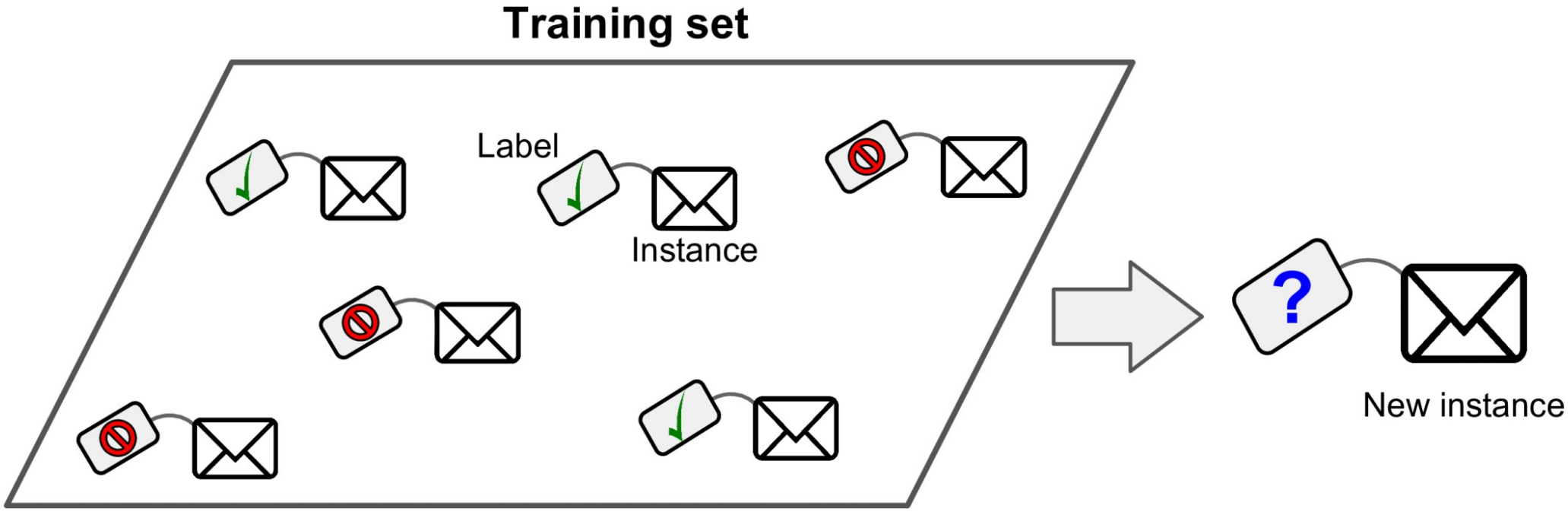
- To summarize, Machine Learning is great for:
  - Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better
  - Complex problems for which there is no good solution at all using a traditional approach: the best Machine Learning techniques can find a solution
  - Fluctuating environments: a Machine Learning system can adapt to new data
  - Getting insights about complex problems and large amounts of data.

Types of machine learning systems

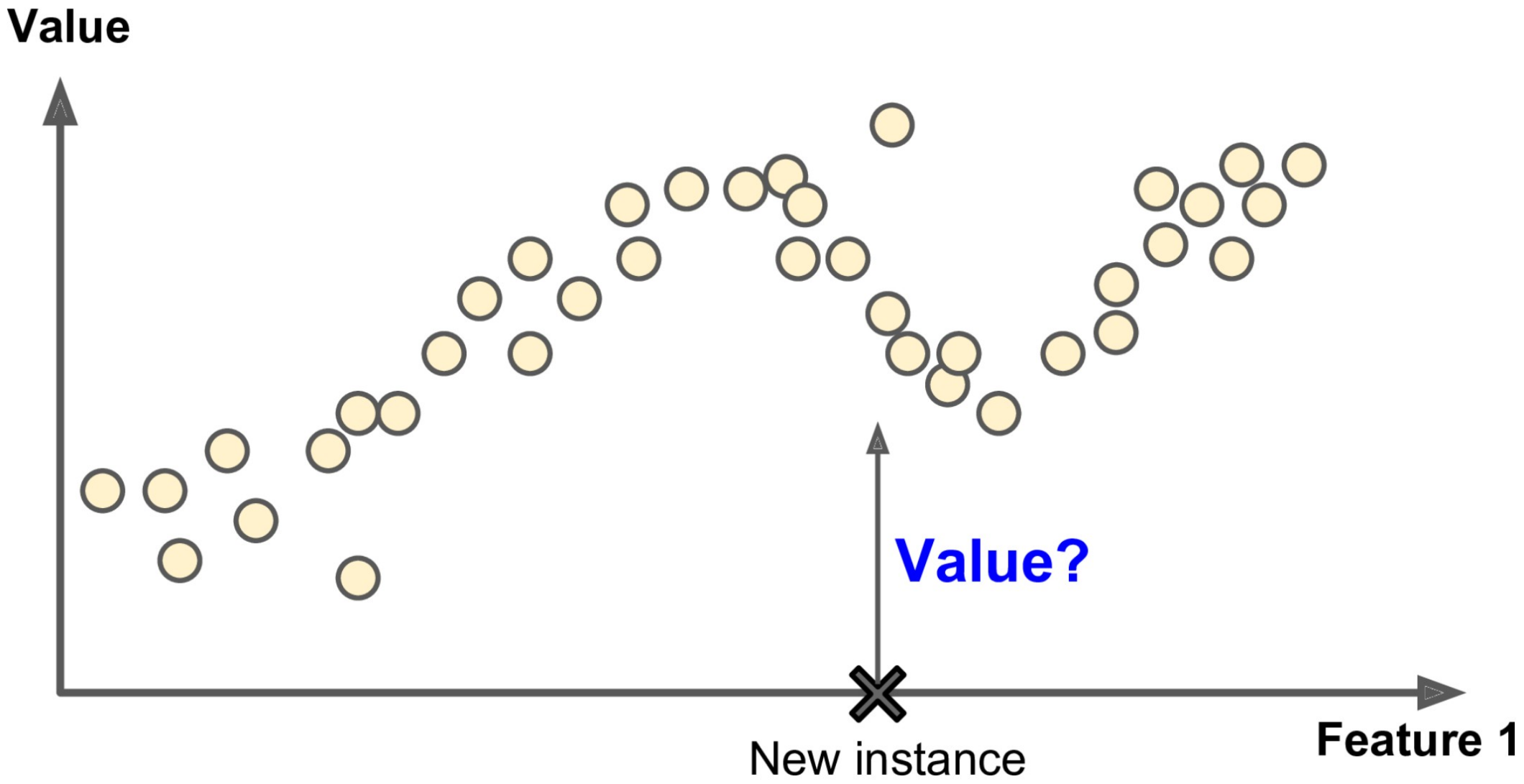
- Whether or not they are trained with human supervision
  - Supervised
  - Unsupervised
  - Semi-supervised
  - Reinforcement Learning
- Whether or not they can learn incrementally on the fly
  - Online learning
  - Batch learning
- Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do
  - Instance-based learning
  - Model-based learning

Supervised learning

- In *supervised learning*, the training data you feed to the algorithm includes the desired solutions, called *labels*



Supervised learning – Regression problem

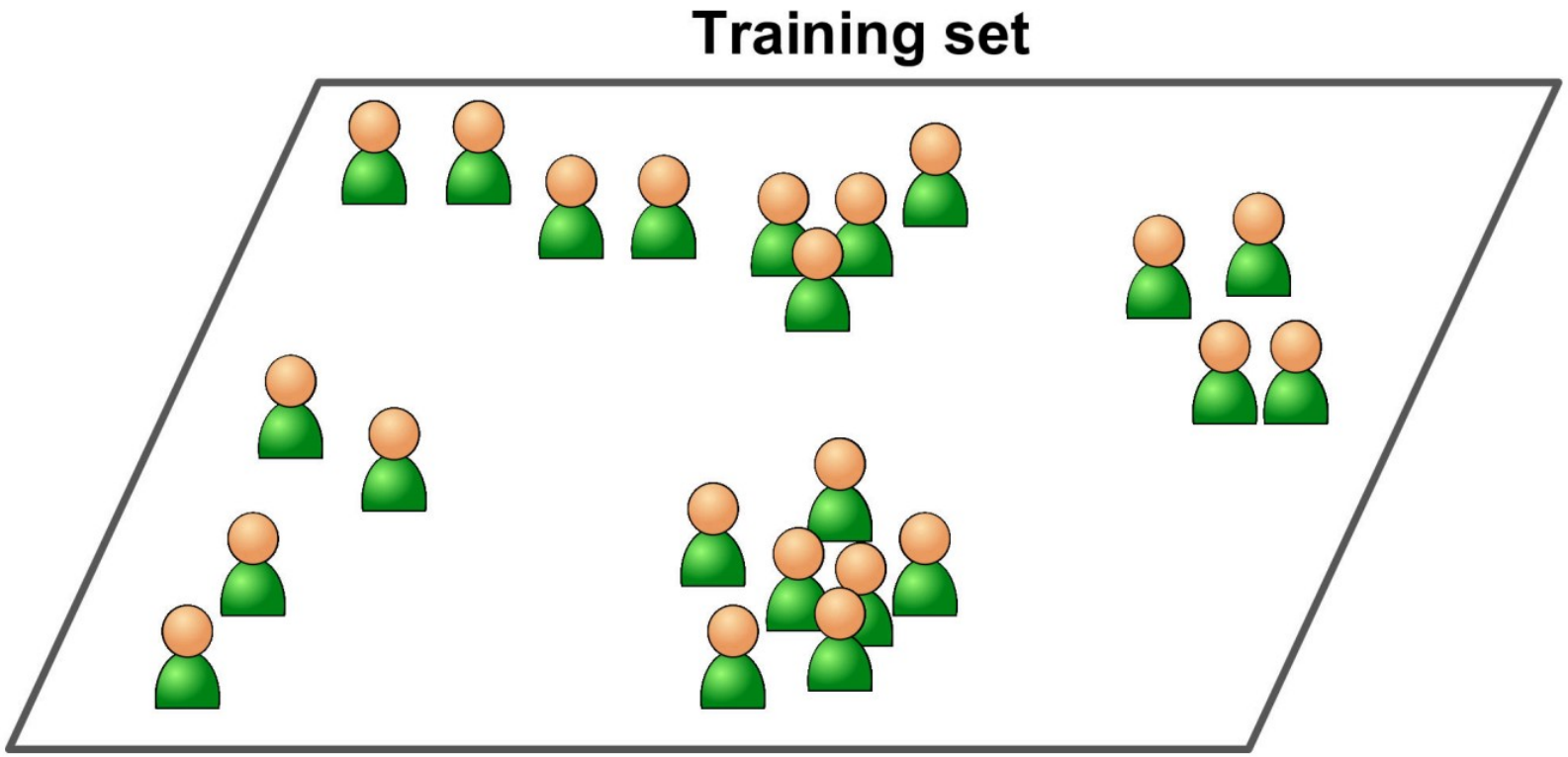


Supervised learning algorithms

- Here are some of the supervised learning algorithms
  - Linear Regression
  - Logistic Regression
  - Support Vector Machines (SVMs)
  - Decision Trees and Random Forests
  - Neural networks

Unsupervised learning

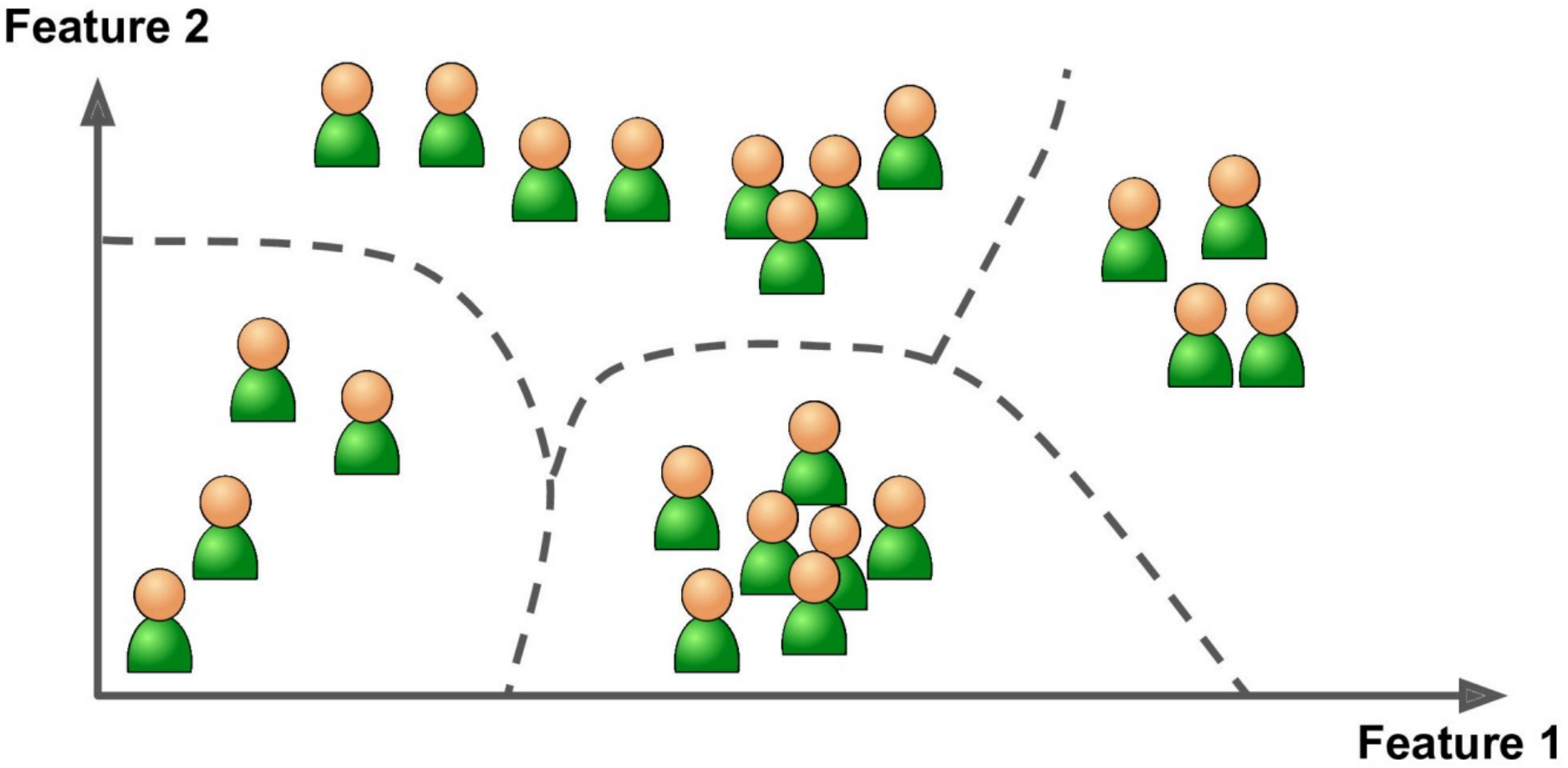
- In *unsupervised learning*, the training data is unlabeled
- The system tries to learn without a teacher



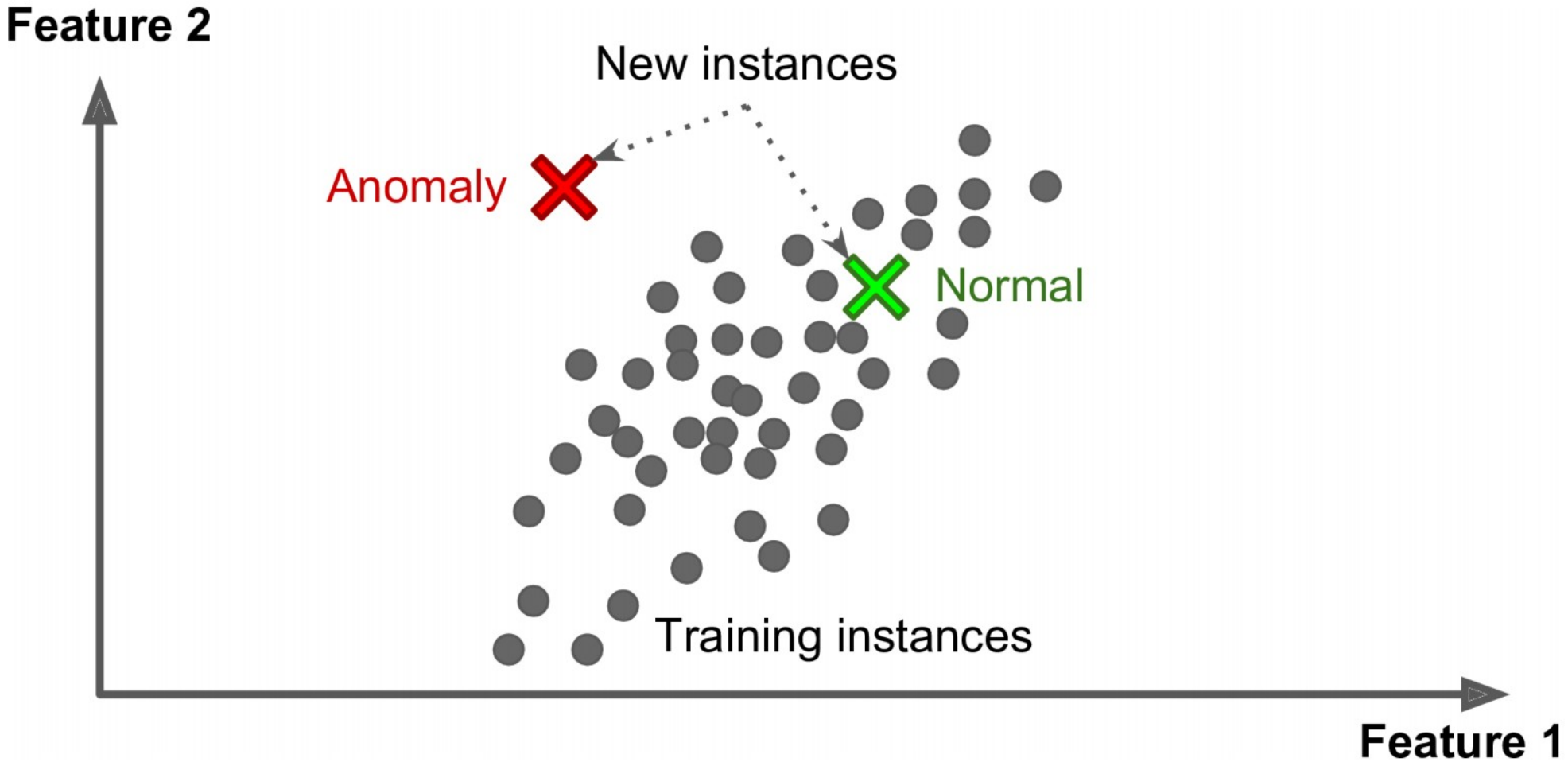
Unsupervised learning algorithms

- In unsupervised learning, the goal is to identify meaningful patterns in the data
  - To accomplish this, the machine must learn from an unlabeled data set
  - The model has no hints how to categorize each piece of data and must infer its own rules for doing so
- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally-Linear Embedding (LLE)
  - t-distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori
  - Eclat

Unsupervised learning – clustering



Unsupervised learning – anomaly detection

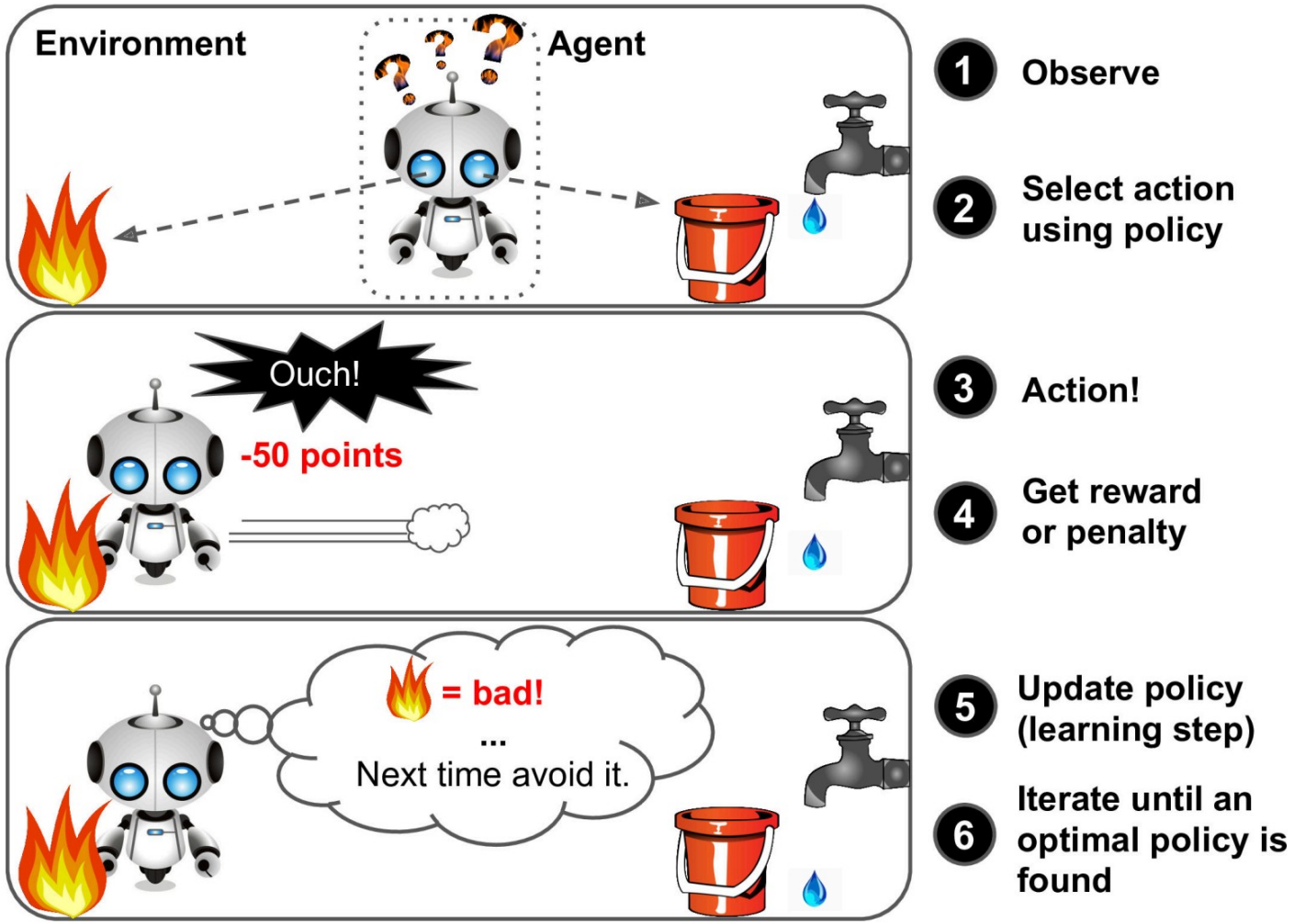


Semi-supervised learning

- Some algorithms can deal with partially labeled training data, usually a lot of unlabeled data and a little bit of labeled data
- This is called *semisupervised learning*
- For example, photo sharing service
- Once we upload our family photos to the service, it automatically recognizes that the same person A shows up in photos 1, 5, and 11
- While another person B shows up in photos 2, 5, and 7: this is the unsupervised part of the algorithm (clustering)
- Now all the system needs is for you to tell it who these people are
- Just one label per person, 4 and it is able to name everyone in every photo, which is useful for searching photos

Reinforcement learning

The learning system, called an *agent* in this context, can observe the environment, select and perform actions, and get *rewards* in return

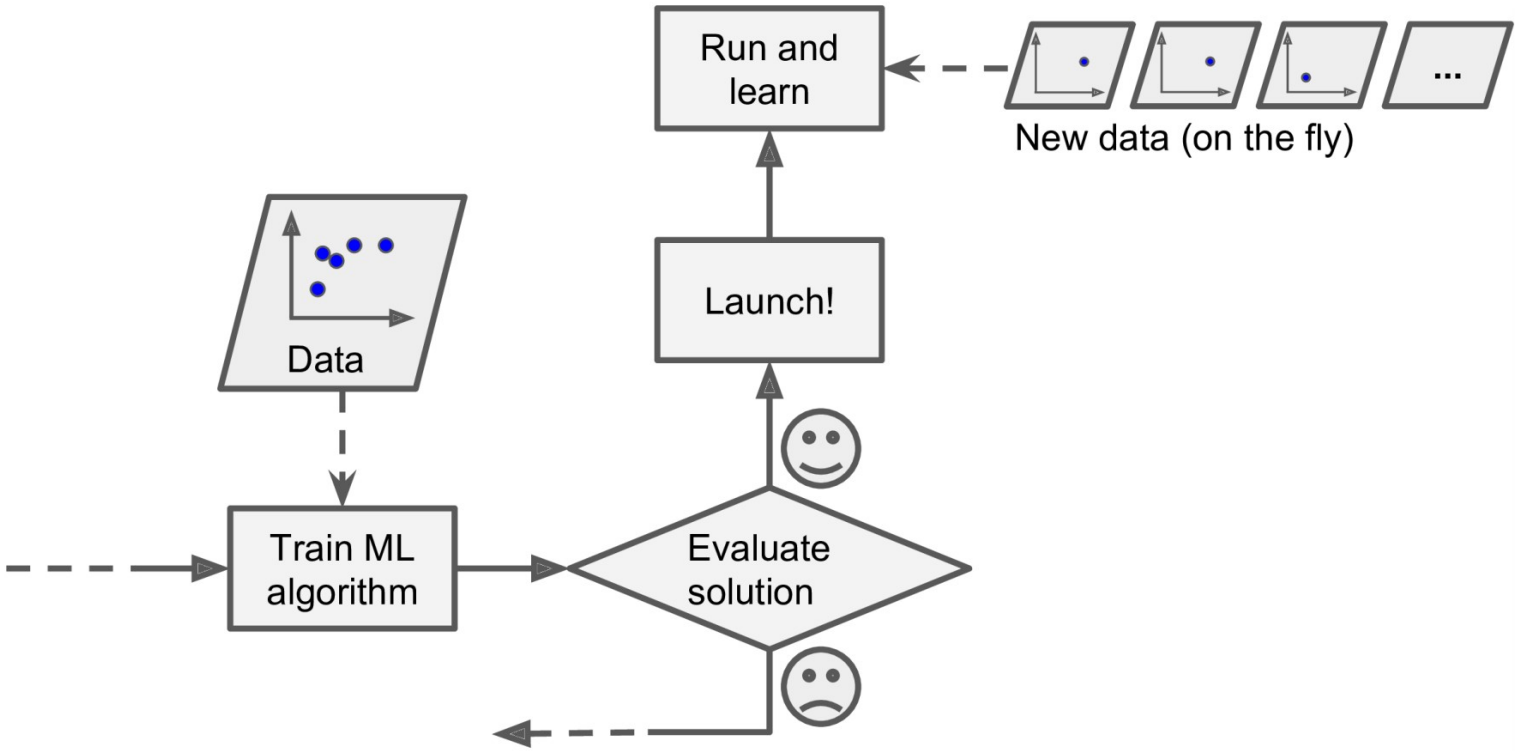


Batch learning

- In *batch learning*, the system is incapable of learning incrementally: it must be trained using all the available data
- This will generally take a lot of time and computing resources, so it is typically done offline
- First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned
- This is called *offline learning*

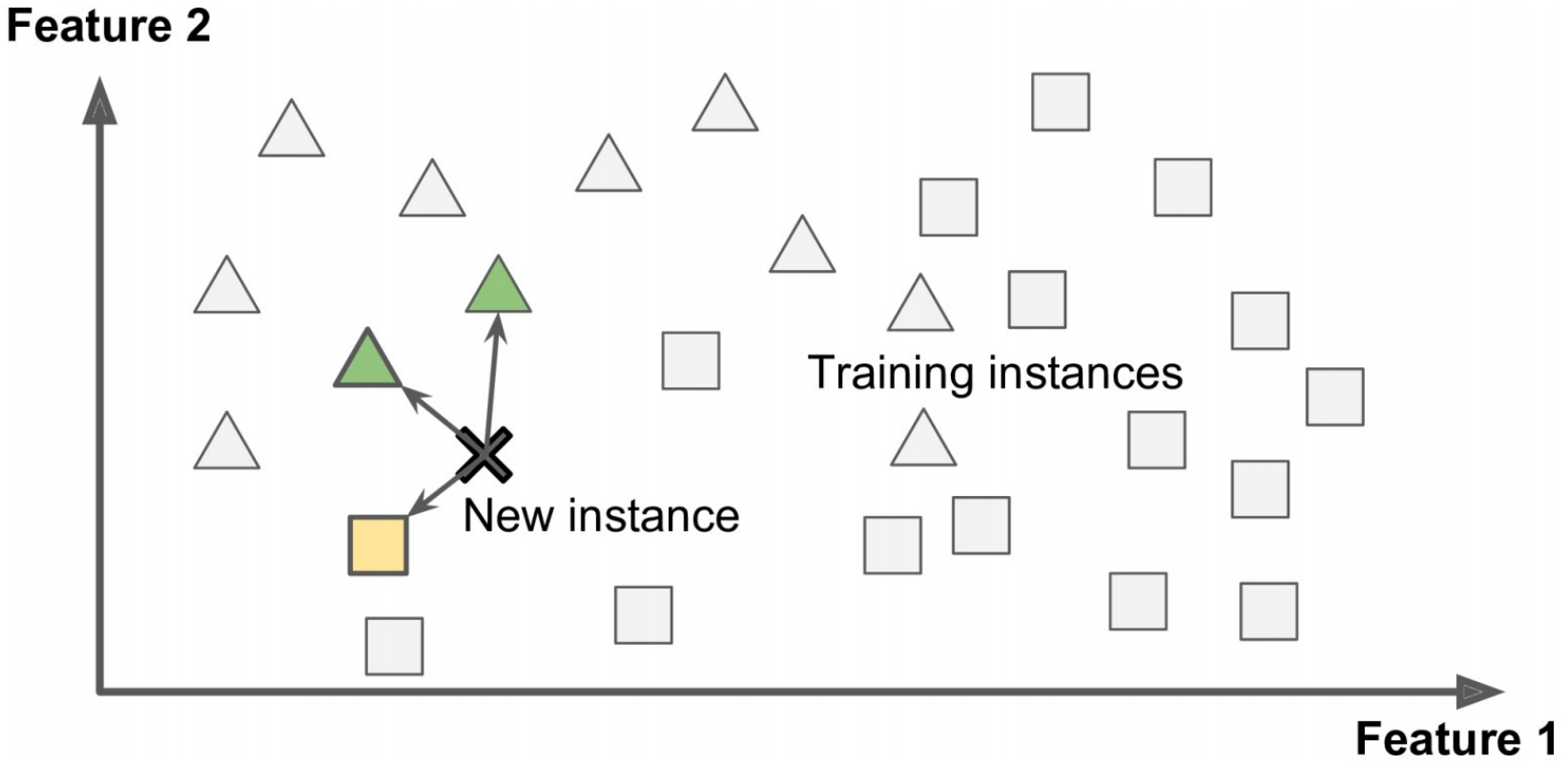
Online learning

- In *online learning*, the system is trained incrementally by feeding it data instances sequentially, either individually or by small groups called *mini-batches*
- Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives



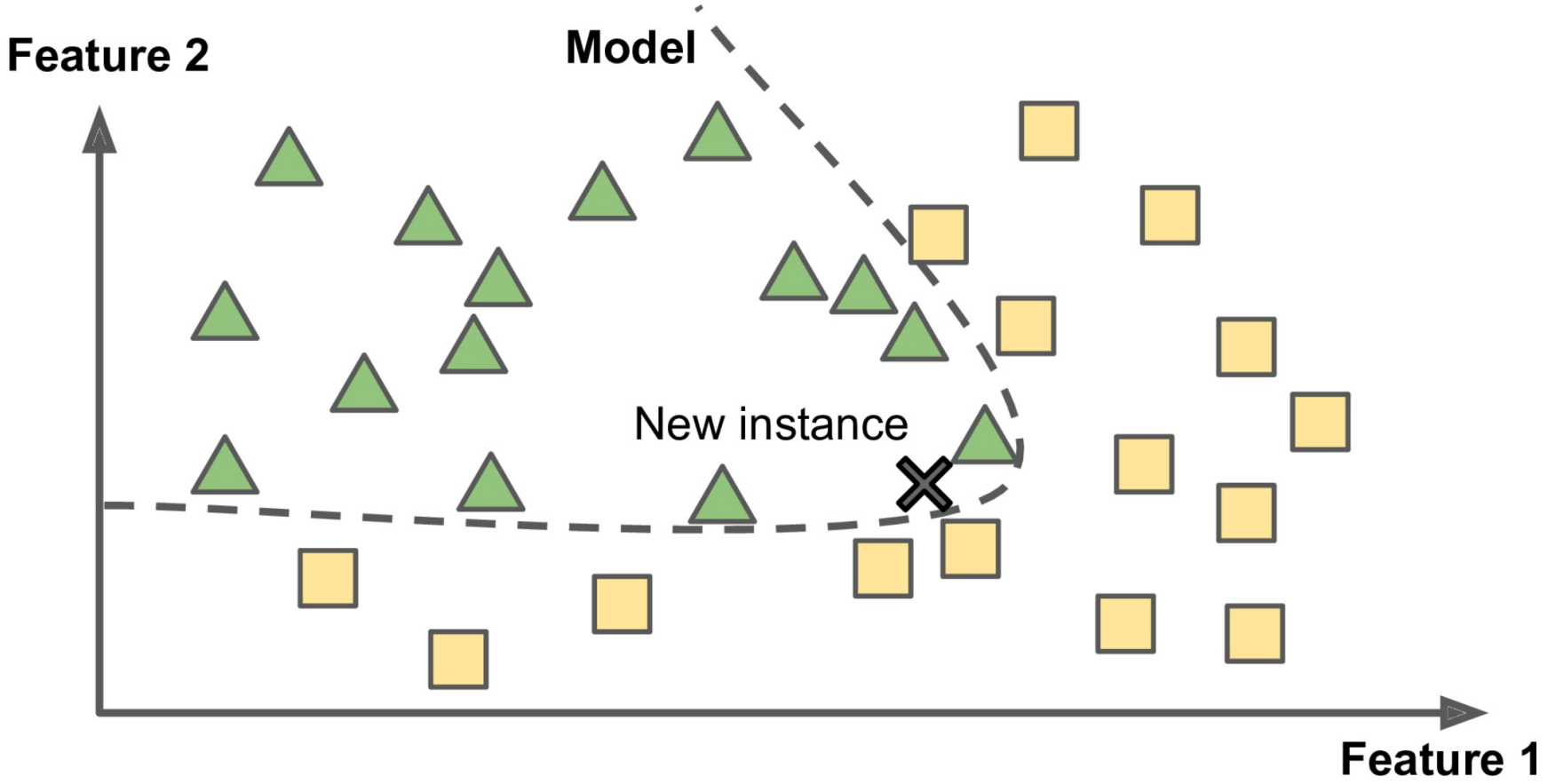
Instance based learning

- *In instance-based learning* the system learns the examples by heart, then generalizes to new cases by comparing them to the learned examples, or a subset of them, using a similarity measure



Model based learning

- Another way to generalize from a set of examples is to build a model of these examples, then use that model to make *predictions*
- This is called *model-based learning*



Problems of machine learning

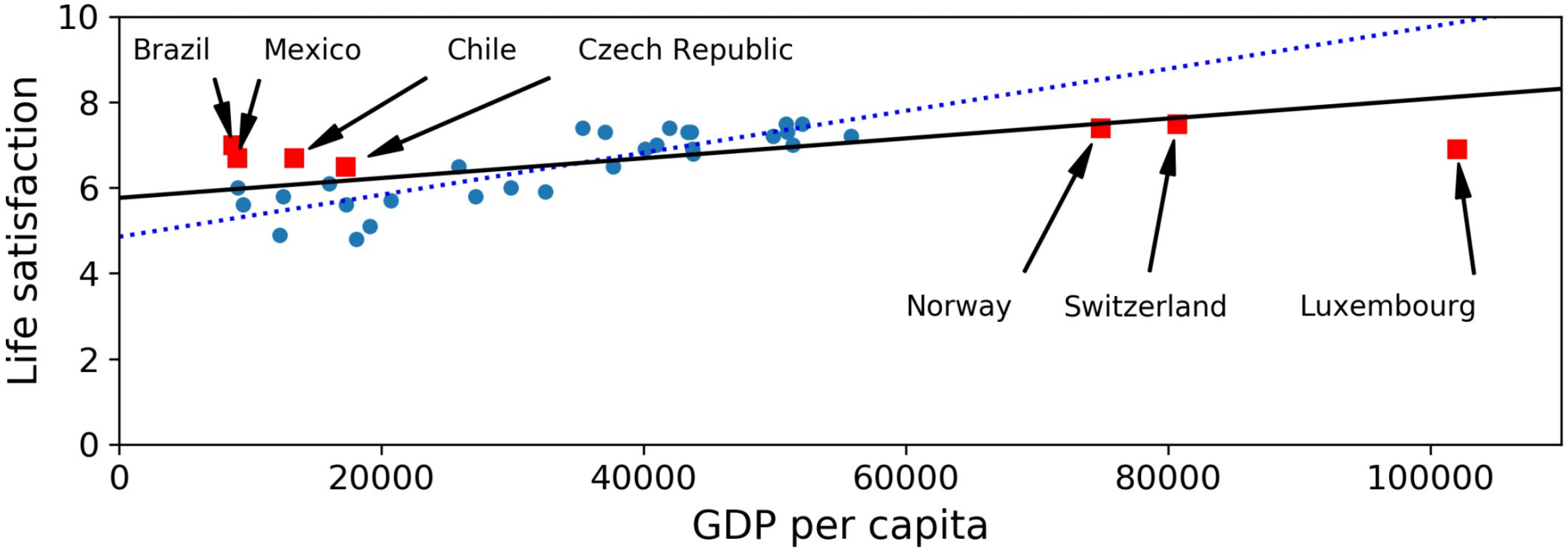
- Insufficient quantity of training data
- Nonrepresentative training data
- Poor-quality data
- Irrelevant features
- Overfitting the training data
- Underfitting the training data

Insufficient quantity of training data

- Machine Learning takes a lot of data
- Even for very simple problems requires thousands of examples
- For complex problems such as image or speech recognition requires millions of examples
  - Unless we can reuse parts of an existing model

Nonrepresentative training data

- In order to generalize well, it is crucial that training data be representative of the new cases to generalize to
- This is true whether you use instance-based learning or model-based learning



Poor-quality data

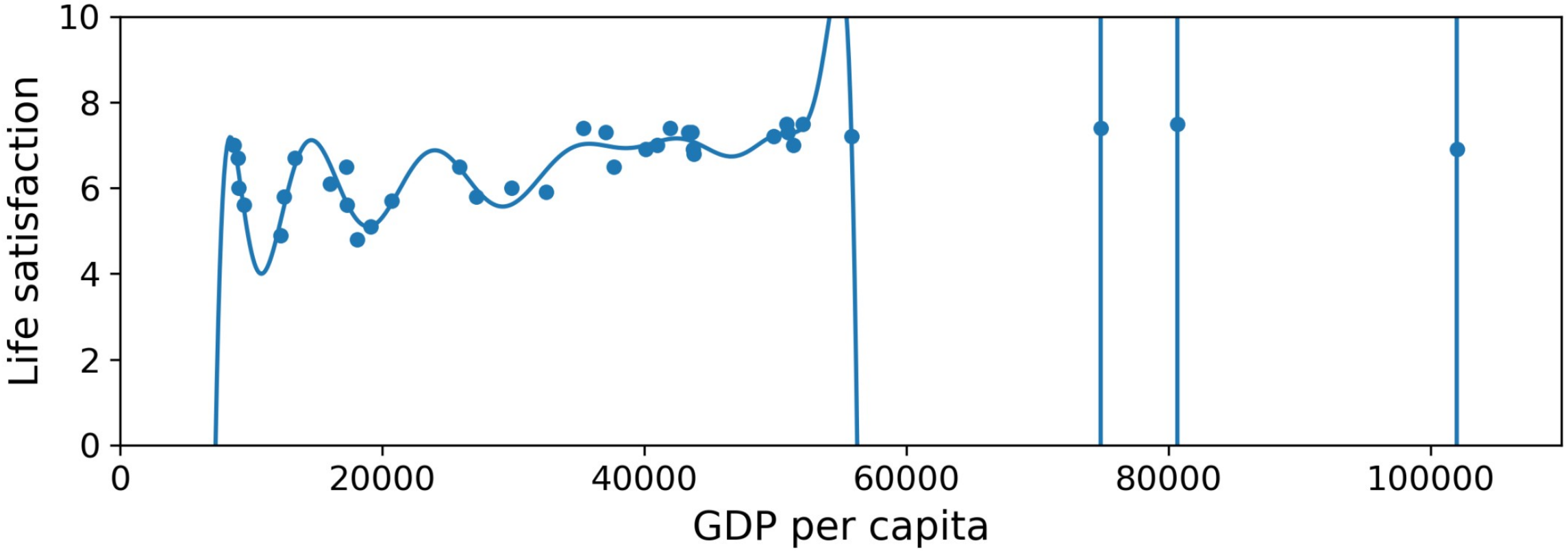
- If training data is full of errors, outliers, and noise (e.g., due to poor quality measurements), it will make it harder for the system to detect the underlying patterns, so the system is less likely to perform well
- It is often well worth the effort to spend time cleaning up the training data
- Most data scientists spend a significant part of their time doing just that
- For example:
  - If some instances are clearly outliers, it may help to simply discard them or try to fix the errors manually
  - If some instances are missing a few features (e.g., 5% of the customers did not specify their age), what do we do?
    - Ignore this attribute altogether?
    - Ignore these instances?
    - Fill in the missing values? (e.g., with the median age)
    - Train one model with the feature and one model without it, and so on

Irrelevant features

- As the saying goes: garbage in, garbage out
- A system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones
- A critical part of the success of a Machine Learning project is to coming up with a good set of features to train on
  - *Feature selection*: selecting the most useful features to train on among existing features
  - *Feature extraction*: combining existing features to produce more useful ones
  - Creating new features by gathering new data.

Overfitting the training data

- Generalize – do not find the perfect model for the training set only



Underfitting the training data

- *Underfitting* is the opposite of overfitting: it occurs when model is too simple to learn the underlying structure of the data
- The main options to fix this problem are:
  - Selecting a more powerful model, with more parameters
  - Feeding better features to the learning algorithm (feature engineering)
  - Reducing the constraints on the model (e.g., reducing the regularization hyper-
    parameter)

Hyperparameter Tuning and Model Selection

- So evaluating a model is simple enough: just use a test set?
- How do we choose one of the two models?
  - Say between a linear model and a polynomial model
- One option is to train both and compare how well they generalize using
  the test set
- Now suppose that the linear model generalizes better, but you want to apply some regularization to avoid overfitting
- The question is: how do you choose the value of the regularization hyperparameter?