Principles of Machine Learning and Artificial Intelligence

Machine learning

Thanks to machine learning, modern AI systems are able to

- recognise faces in digital photographs
- produce and understand natural language
- control machines, vehicles, and robots
- offer decision support in healthcare
- detect fraudulent bank transactions and cyberattacks
- suggest products, songs, and films
- help us find information on the web

What do we mean by learning?

- Informally speaking, a machine learning algorithm is an algorithm that is able to learn from data.
- ‘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 tasks in T, as measured by P, improves with E.’

  Mitchell (1997)

Different learning experiences

- **Supervised learning**

  The system has access to both the input and the target output.

  classification, regression

- **Unsupervised learning**

  The system has access to the input but not the target output.

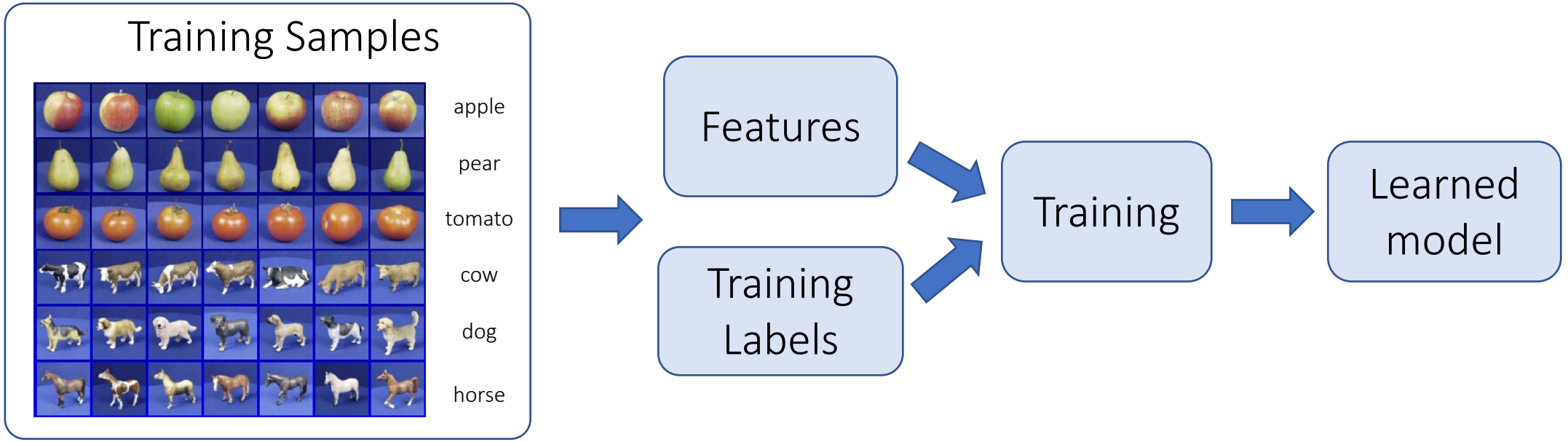
  topic models, clustering

Supervised Statistical Learning

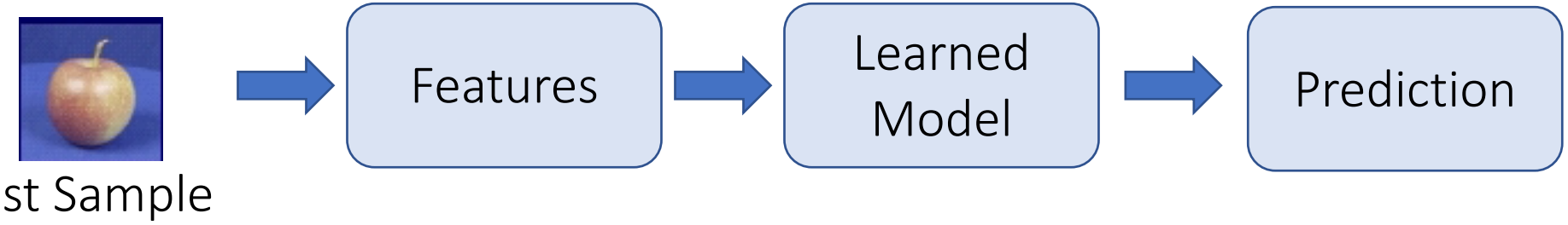
- Given inputs (data-label pairs), learn a model to predict output
- Require training data to learn

Basic Supervised Learning Framework

Training time



Testing time



Credit: Lazebnik

Basic Supervised Learning Formulation

$$y = f(\mathbf{x})$$

output — prediction function — input

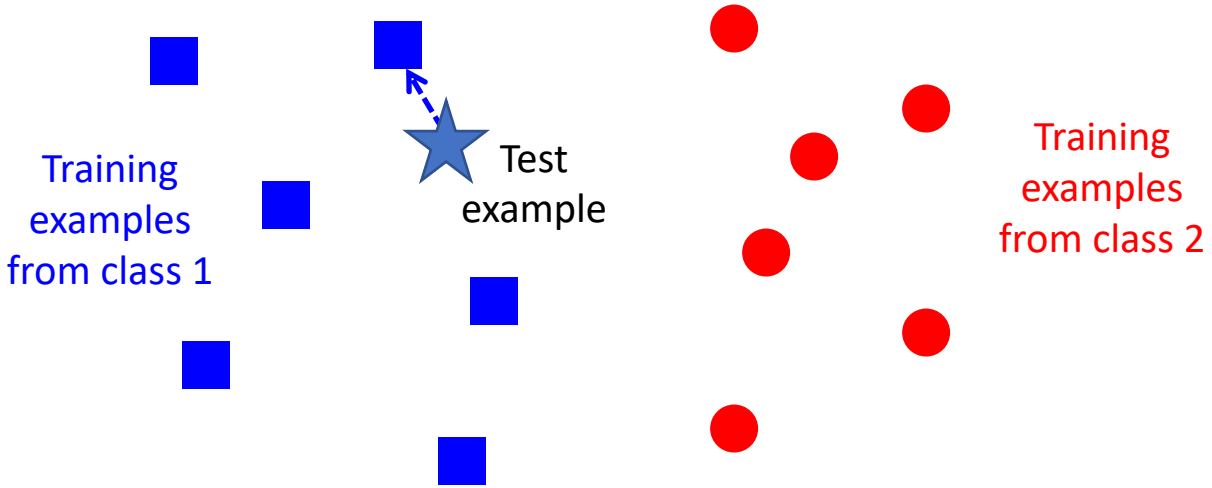
Formulation:

- Given training data: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$,
- Find $y = f(\mathbf{x}) \in \mathcal{H}$ using training data
- such that f is correct on test data

Loss measures in next class

Credit: Lazebnik, Liang

Simple Classifier



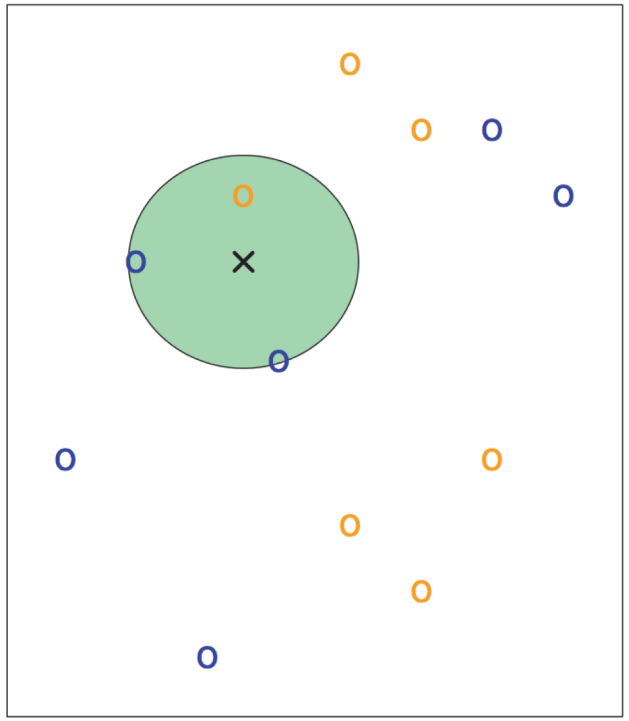
f(**x**) = label of the training example nearest to **x**

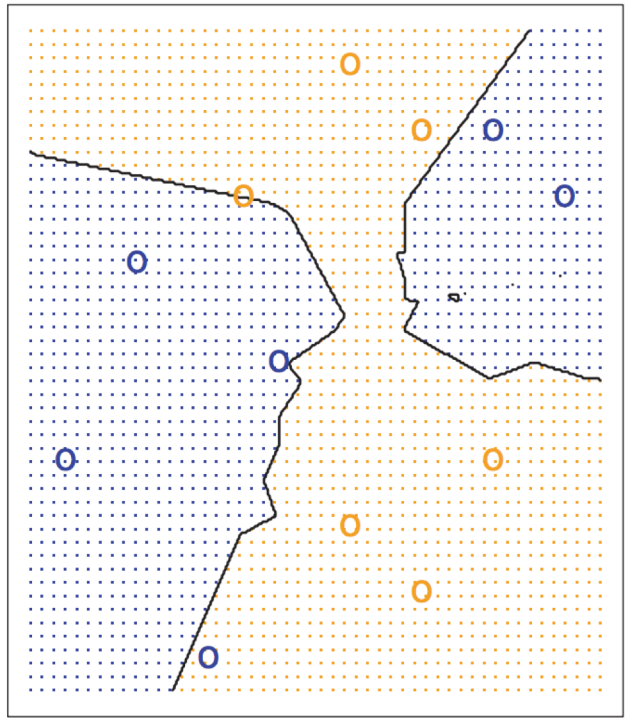
- All we need is a distance function for our inputs
- No training required!

Credit: Lazebnik

K-Nearest Neighbor (kNN) Classifier

k = 3

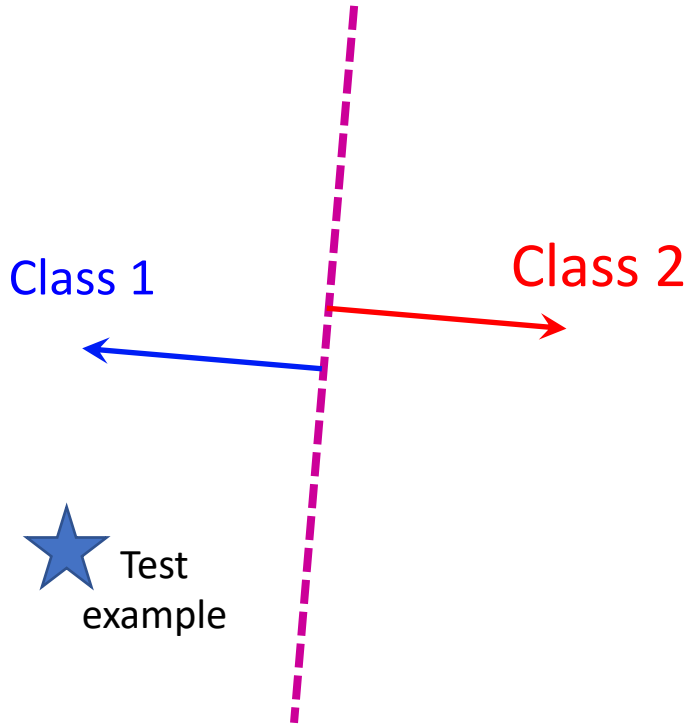




- Find k nearest points to **x**
- f(**x**) = vote for class labels with labels of the k nearest points

Credit: An Introduction to Statistical Learning

Linear Classifier



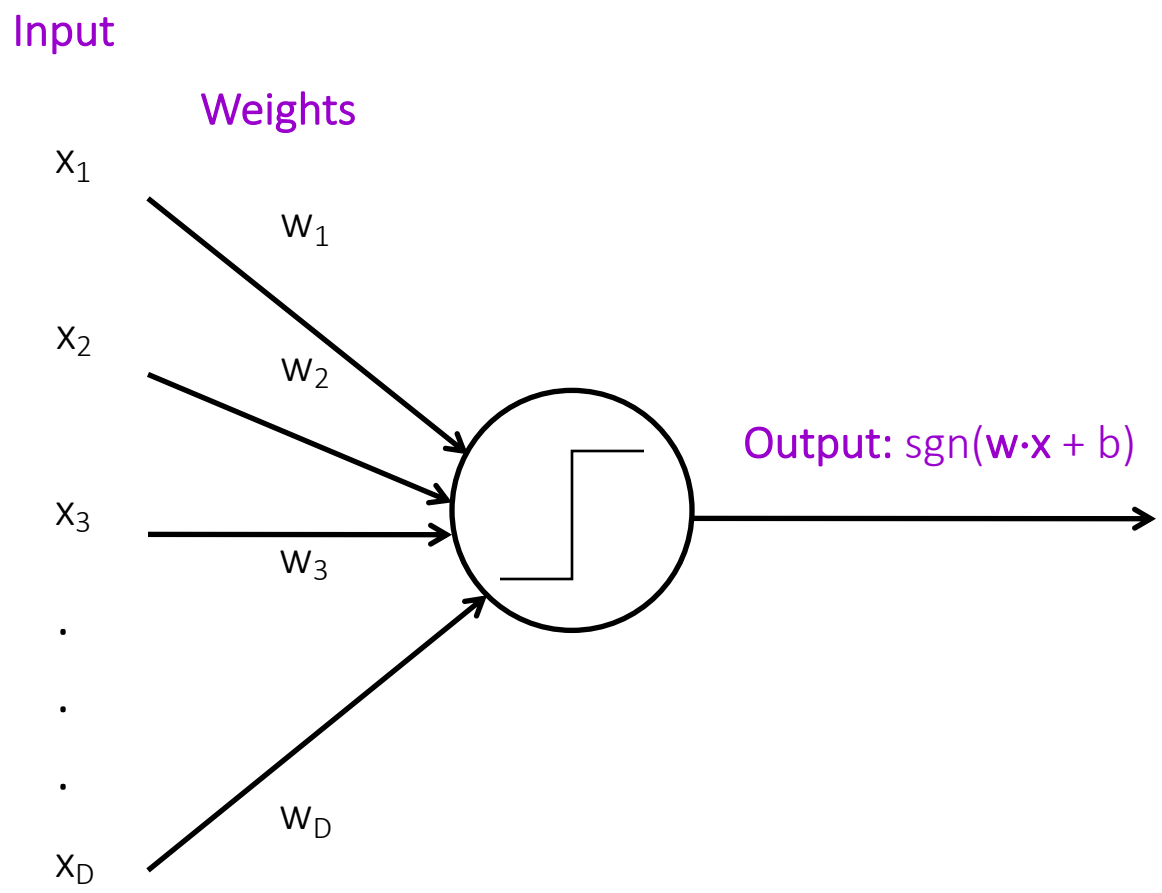
Find *linear function* that separates the classes

$$f(\mathbf{x}) = \text{sgn}(w_1x_1 + w_2x_2 + \ldots + w_Dx_D + b) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

$$f(\mathbf{x}) = \text{sgn}(w_1x_1 + w_2x_2 + \ldots + w_Dx_D + w_0x_0) = \text{sgn}(\mathbf{w} \cdot \mathbf{x})$$

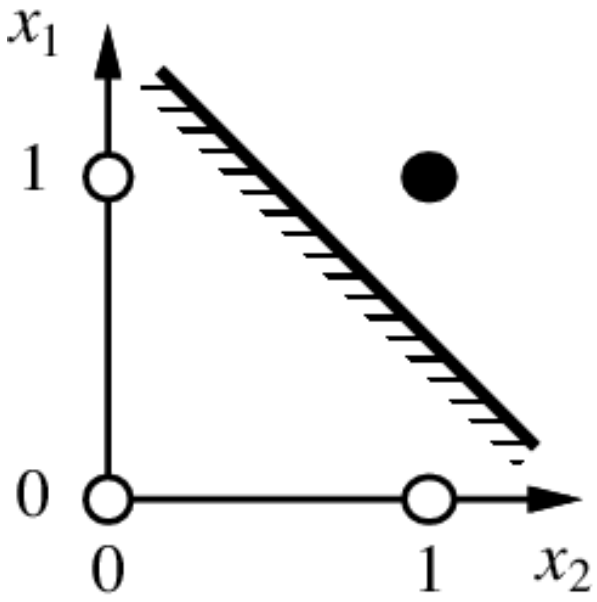
Credit: Lazebnik

Linear Classifier: Perceptron view

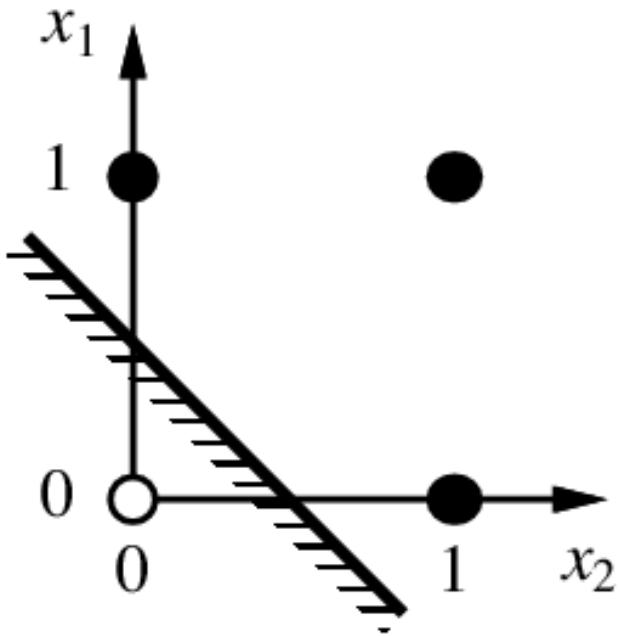


Credit: Lazebnik

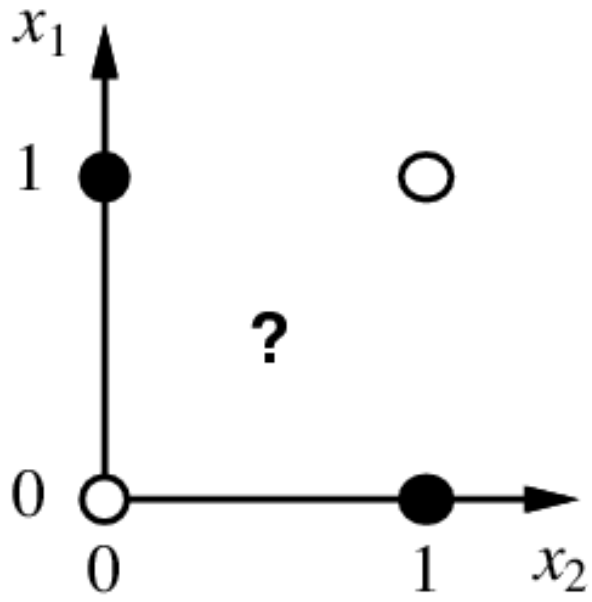
Perceptrons, linear separability, and Boolean functions



x_1 **and** x_2



x_1 **or** x_2



x_1 **xor** x_2

Credit: Lazebnik

Learning problem

- Choose parameters such that the total distance between the corresponding hyperplane and the data points is minimal.

  as measures by mean squared error

- This problem has an exact solution that can be found using the **method of least squares**.

- An inexact (numerical) but more general method to solve the problem is to use **gradient descent**.

Recall: A brief history of Neural Networks

Frank Rosenblatt
1928–1969

Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

Snippet credit: PRML 2006, Bishop

Slide credit: Lazebnik

kNN vs. Linear Classifiers

kNN Pros

+ Simple to implement
+ Decision boundaries not necessarily linear
+ Works for any number of classes
+ *Nonparametric* method

kNN Cons

– Need good distance function
– Slow at test time

Linear Pros

+ Low-dimensional *parametric* representation
+ Easy to learn (more later)
+ Very fast at test time

Linear Cons

– Works for two classes (?)
– How to train the linear function?
– What if data is not linearly separable?

Credit: Lazebnik

Multi-class Classification – Softmax

- Probability
  - Positive
  - Sums to 1

$$P(c) = \frac{\exp(z_c)}{\sum_k \exp(z_k)}$$

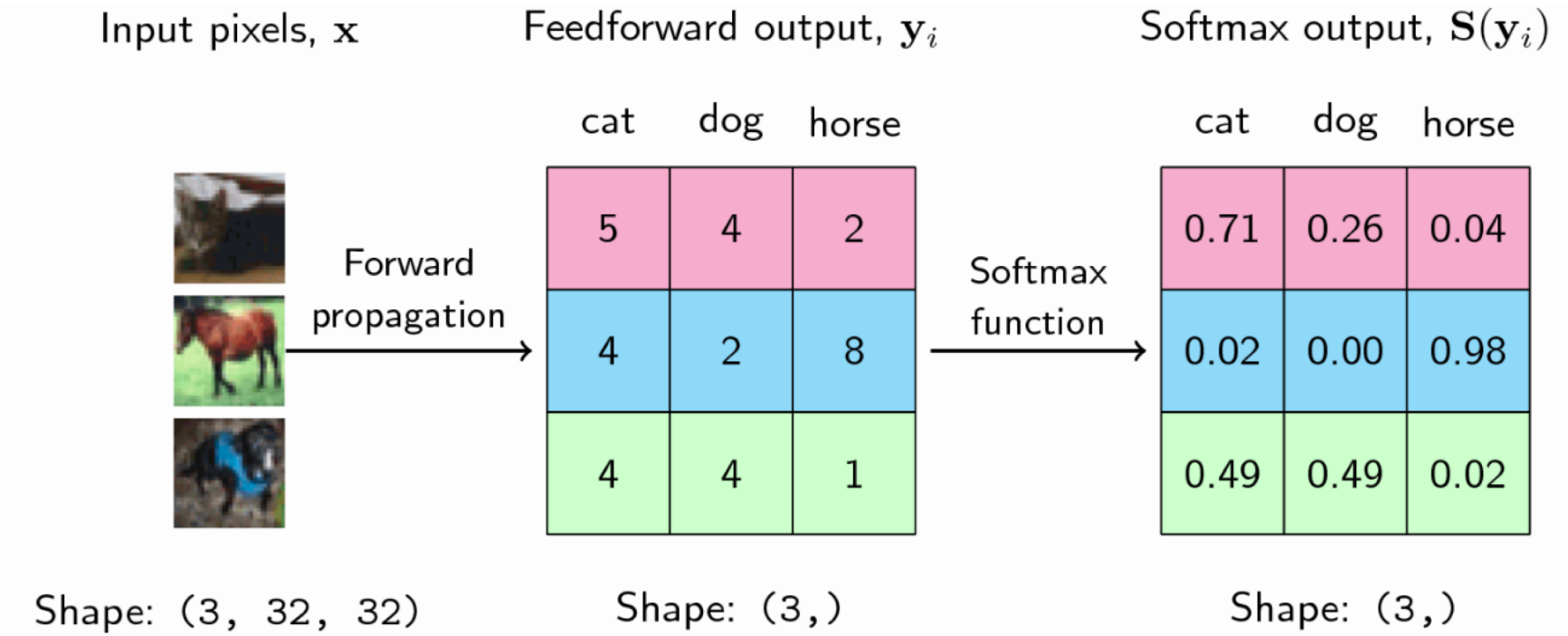
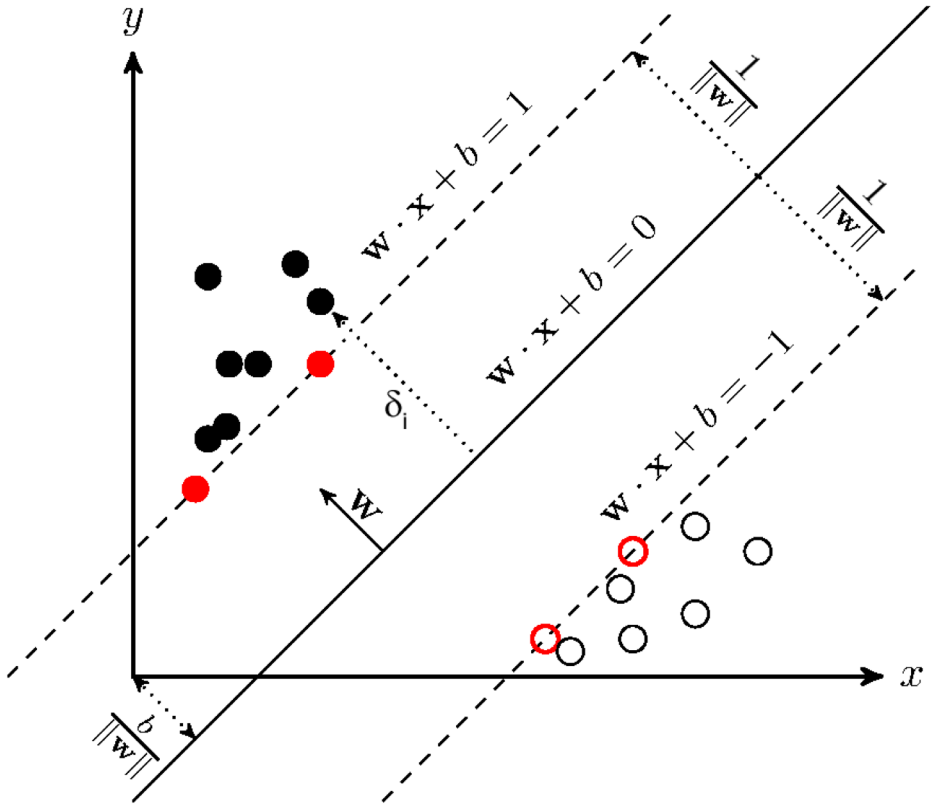


Figure: *Softmax Computation for three classes*

Credit: Krähenbühl; refer to blog

Classification (max-margin)

SVM Loss



Credit: Krähenbühl, image credit

Training error and generalisation error

- We train a model by minimising its error on the training data.

  optimisation

- The **training error** is different from the **generalisation error** – the expected value of the error on previously unseen inputs.

- We can estimate the generalisation error of a model by measuring its **test error** – the error on a held-out test set.

How can we hope to perform well on the test set?

- **Assumption 1:** The examples in the training set and the test set are mutually independent.
- **Assumption 2:** The examples in the training set and the test set are identically distributed.

  sampled from the same data generating distribution

The relation between training error and test error

- Probabilistic story behind supervised machine learning:
  - sample the training set
  - use the training set to set the parameters of the model
  - sample the test set and compute the test error
- Under this process, the expected test error is greater than or equal to the expected training error.

Underfitting and overfitting

- **Underfitting**

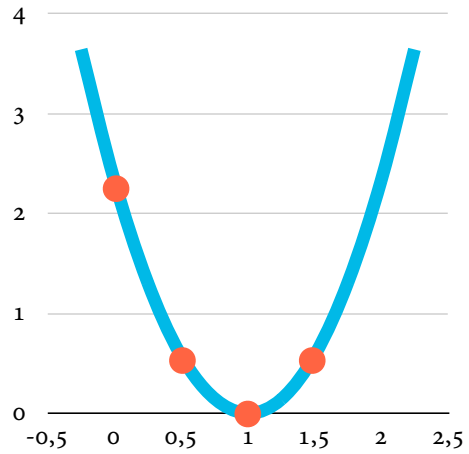
  The model is unable to obtain a sufficiently low error on the training set. The model is not expressive enough.

- **Overfitting**

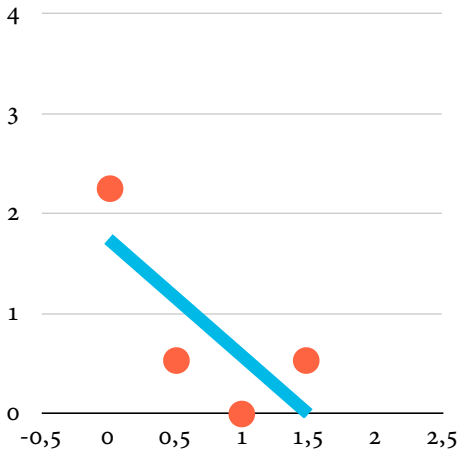
  The gap between the training error and the test error is too large. The model is over-optimised for the training data.

  memorises noise

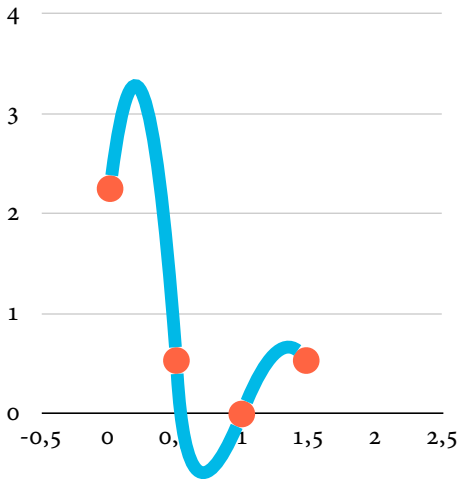
Underfitting and overfitting



appropriate



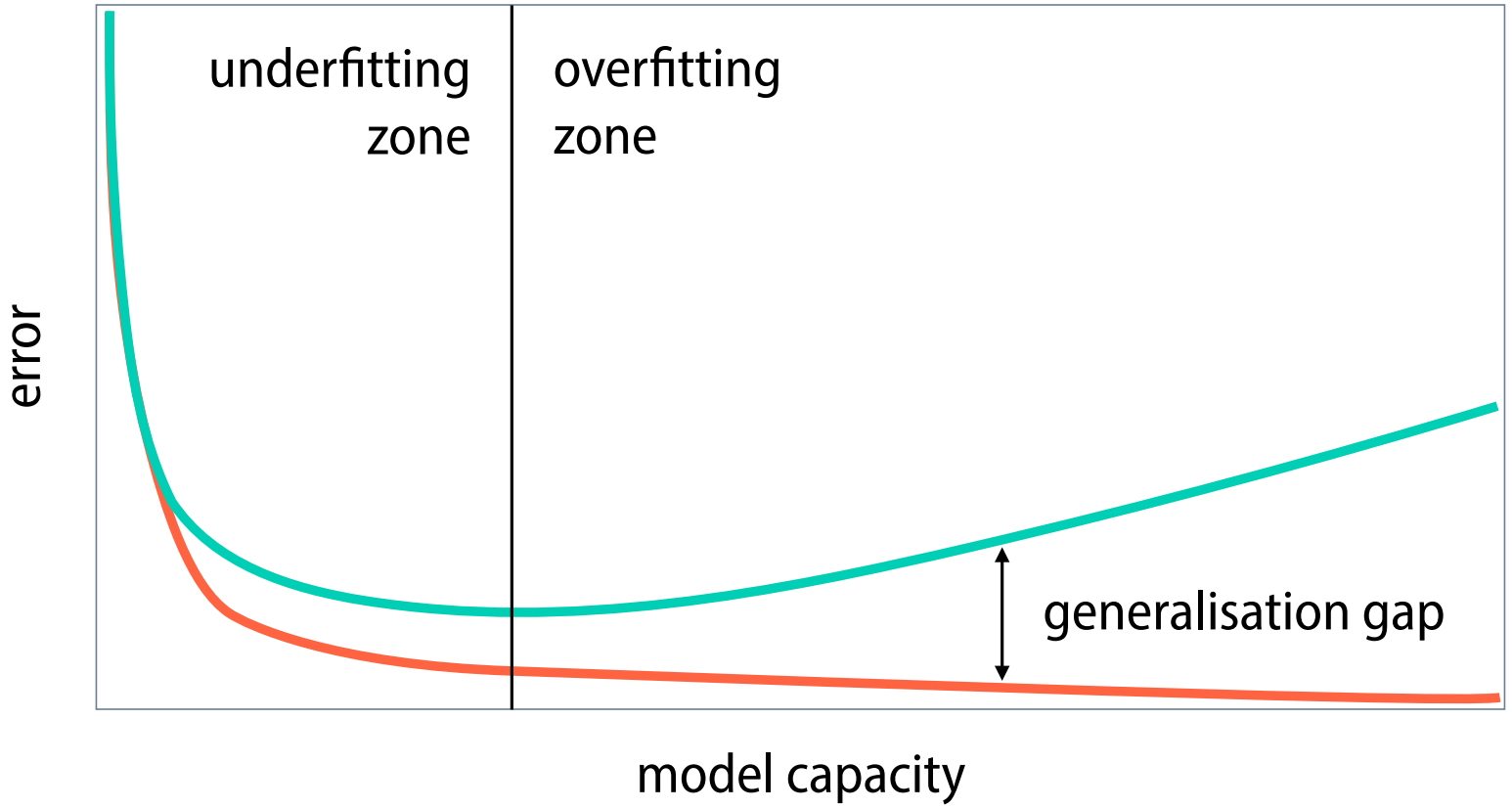
underfitting



overfitting

Kuhlmann

Relationship between model capacity and error



Kuhlmann

‘No free lunch’ theorems

- Averaged over all possible data-generating distributions, every learning algorithm has the same generalisation error.

  Wolpert (1996)

- This means that there is no universal learning algorithm or absolute best learning algorithm.

  Not even neural networks!

- We need to make assumptions about the kinds of data-generating distributions we encounter in practice.

Regularisation

- One way to tailor a learning algorithm to a specific task and to prevent it from overfitting is to use **regularisation**.
- Regularisation refers to modifications intended to reduce the generalisation error but not the training error of an algorithm.
- A standard example is **L2-regularisation**, where we give preference to parameter vectors with smaller Euclidean norms.

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \rho\,\boldsymbol{\theta}^{\top}\boldsymbol{\theta}$$

Hyperparameters and validation sets

- A setting of a machine learning algorithm that is not adapted by the algorithm itself is called a **hyperparameter**.

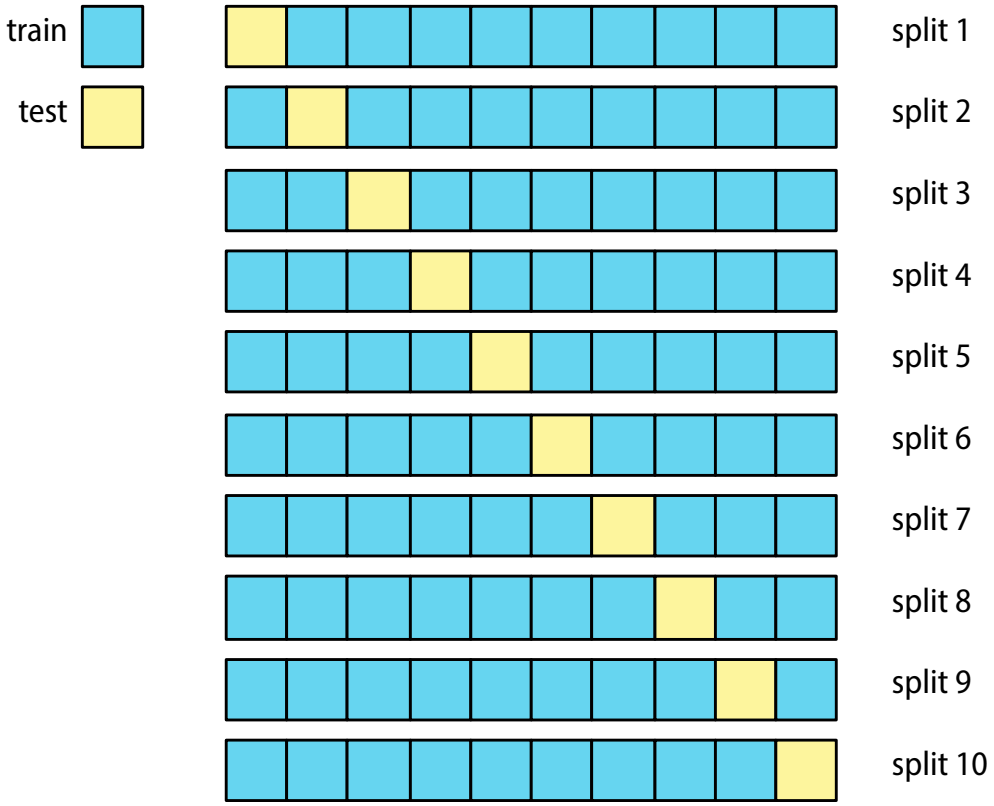
  typical example: learning rate

- Some settings need to be hyperparameters because adapting them during training would lead to overfitting.

  such as parameters related to the model's capacity

- To tune hyperparameters, we need a separate validation set, or need to use cross-validation.

Cross-validation



Kuhlmann

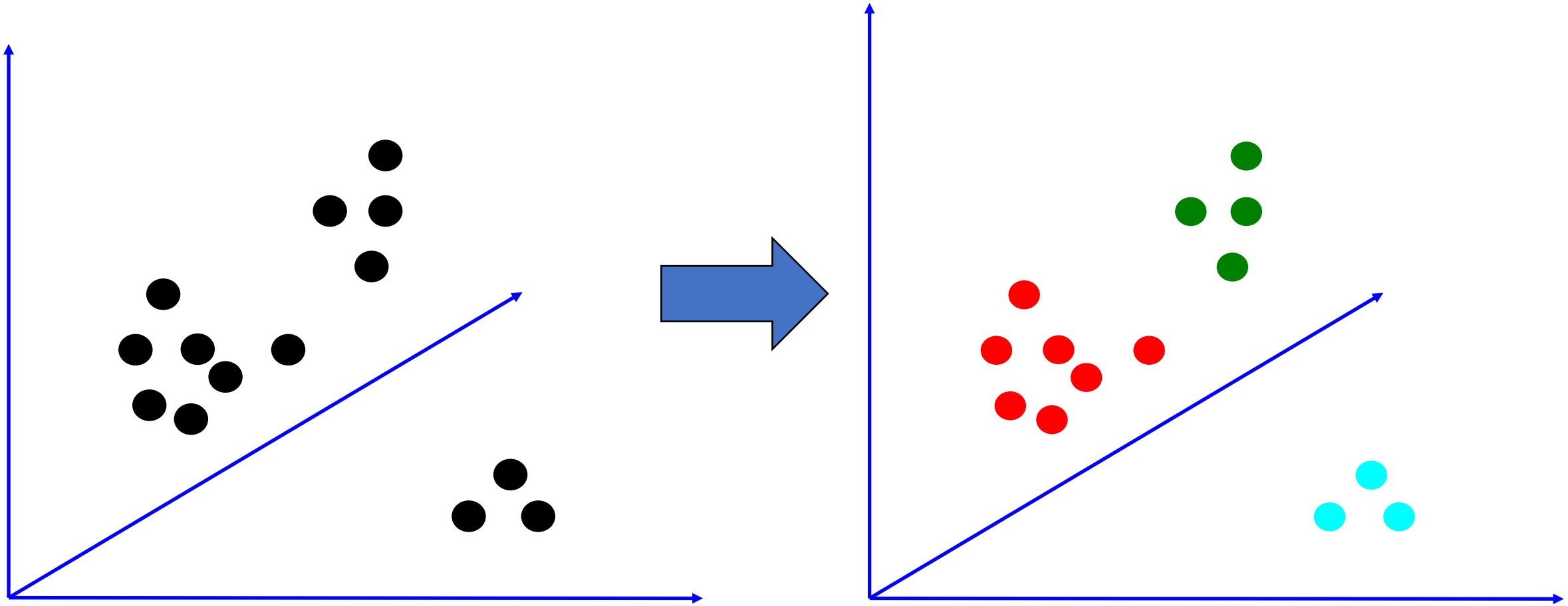
Unsupervised Learning

- Given just data as input (no labels), learn some sort of *underlying structure*..
  - Goal is often vague or subjective (compared to supervised learning, where labels define the goals)
  - Also known as exploratory/descriptive data analysis

Credit: Lazebnik

Unsupervised Learning: Clustering

- Discover groups of “similar” data points

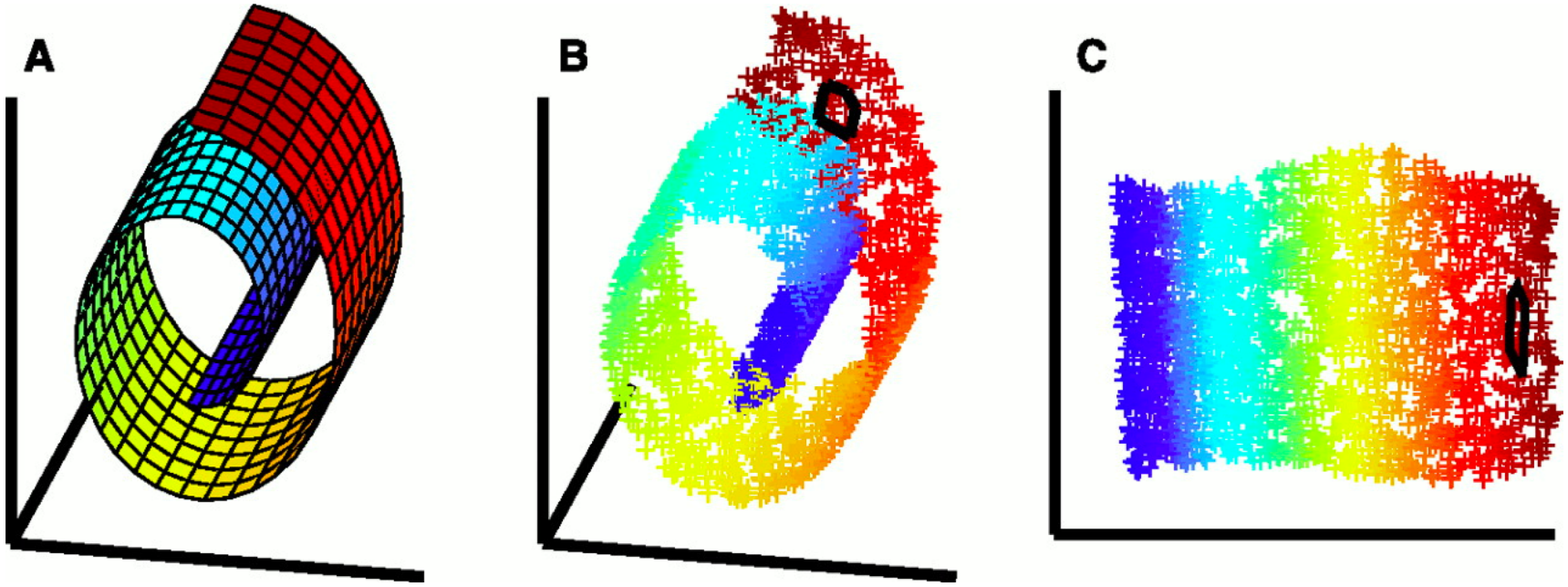


Credit: Lazebnik

Unsupervised Learning: Dimensionality Reduction

Dimensionality reduction, manifold learning

- Discover a lower-dimensional surface on which the data lives

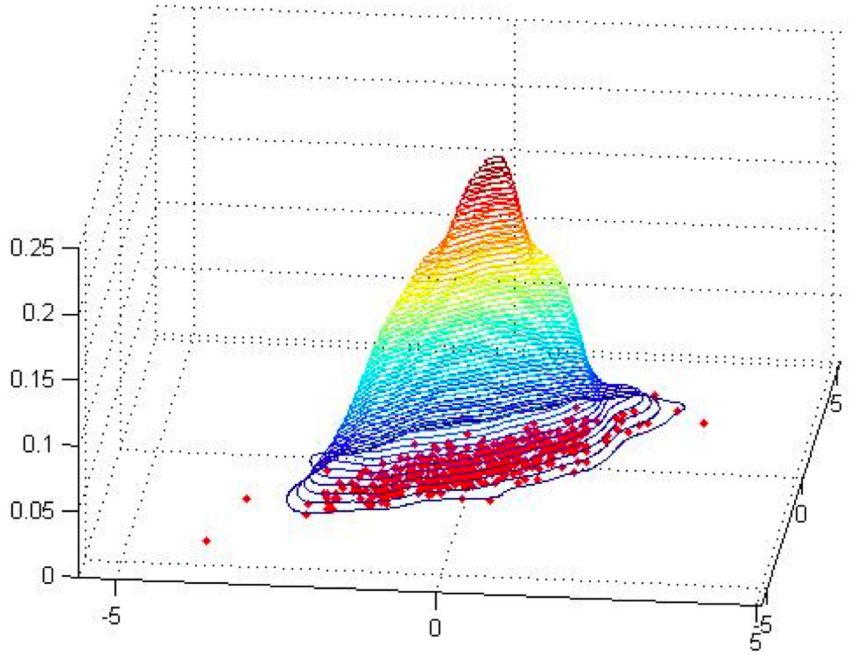


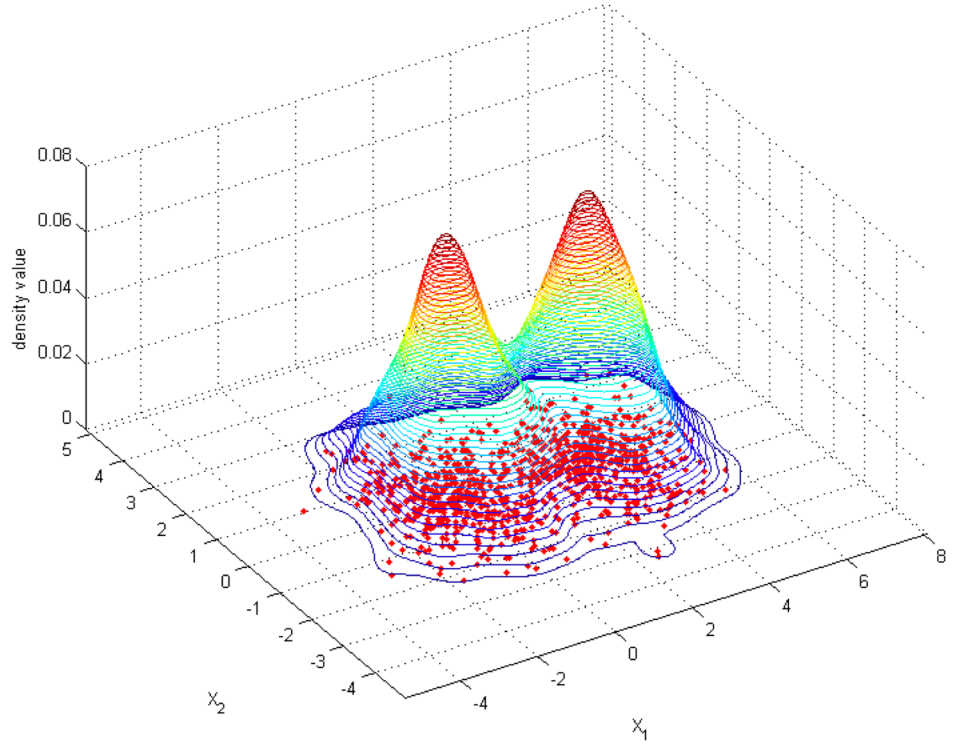
Credit: Lazebnik, Image source1, source2

Unsupervised Learning: Learning Data Distribution

Density Estimation

- Find a function that approximates the probability density of the data (i.e., value of the function is high for “typical” points and low for “atypical” points)
- Can be used for *anomaly detection*





Credit: Lazebnik, Image source1, source2

Weakly-Supervised Learning

- Noisy or incomplete labels: Extracting labels (e.g., captions to labels)

Figure 1. Six randomly picked photos from the Flickr 100M dataset and the corresponding descriptions.

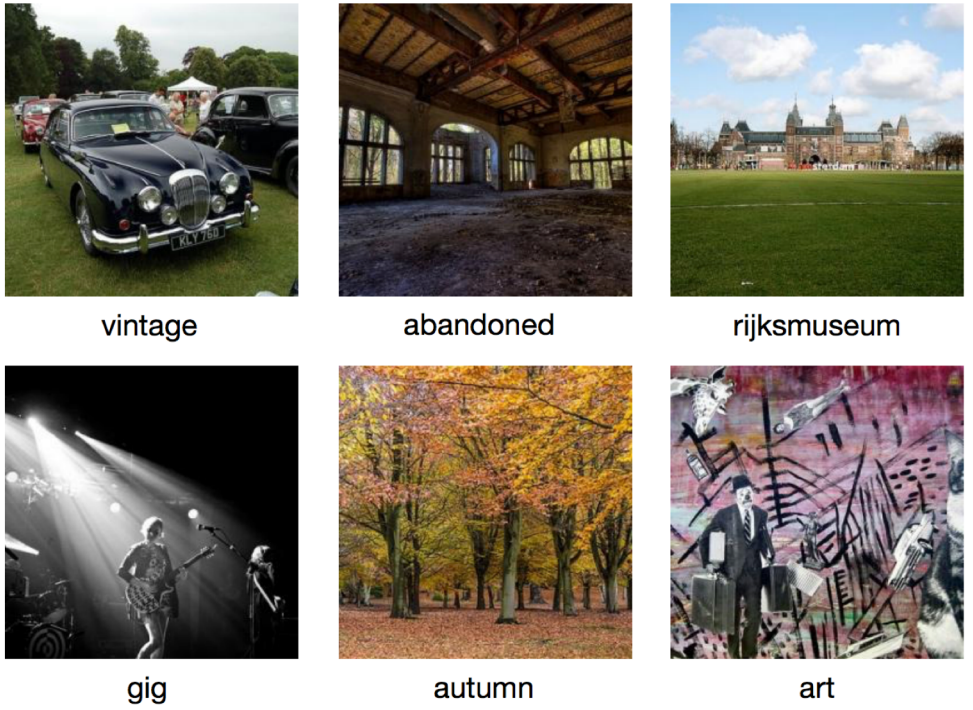


Figure 3. Six test images with high scores for different words. The scores were computed using an AlexNet trained on the Flickr dataset with a dictionary size of $K = 100,000$.

Figures: Joulin et al.

(Some) Learning Paradigms

Supervised Learning	Weakly-Supervised Learning	Semi-Supervised Learning	Unsupervised Learning
• Needs nice, clean labels • All data needs to be labeled • Labels correspond to task of interest	• **Noisy, often incorrect, labels** • All data needs to be labeled • **Labels might not correspond to task of interest, but need to be related**	• Needs nice, clean labels • **Small portion of data needs to be labeled** • Labels correspond to task of interest	• No labels • No labels • How does this work?

(Some) Learning Paradigms