DataSci 420

lesson 7: support vector machines

Seth Mottaghinejad

today's agenda

- SVM **pros and cons**
- linear separability and **wide-margin classifiers**
- non-linear separability
- the **kernel trick**
- **soft-margin classifiers**
- **multi-class classification** with SVMs
- **cross validation** for **hyper-parameter tuning**

SVMs in a nutshell

- world-class until the advent of deep learning
- they can run through **a lot of compute**, although the **kernel trick** makes the computation much more efficient
- less affected by outliers (because the separation boundary only depends on the **support vectors**)
- can still be used when there are more features than samples
- not great for **multi-class** classfication: **one-vs-one**, **one-vs-all**
- can also be used for regression (SVR instead of SVM) with some slight modifications to the algorithm

SVM classifier

- there are many lines that offer **linear separability**
- the one that maximizes the **margin** is the best one
- SVM are called **wide-margin classifires**
- the model is explained by its **support vectors**

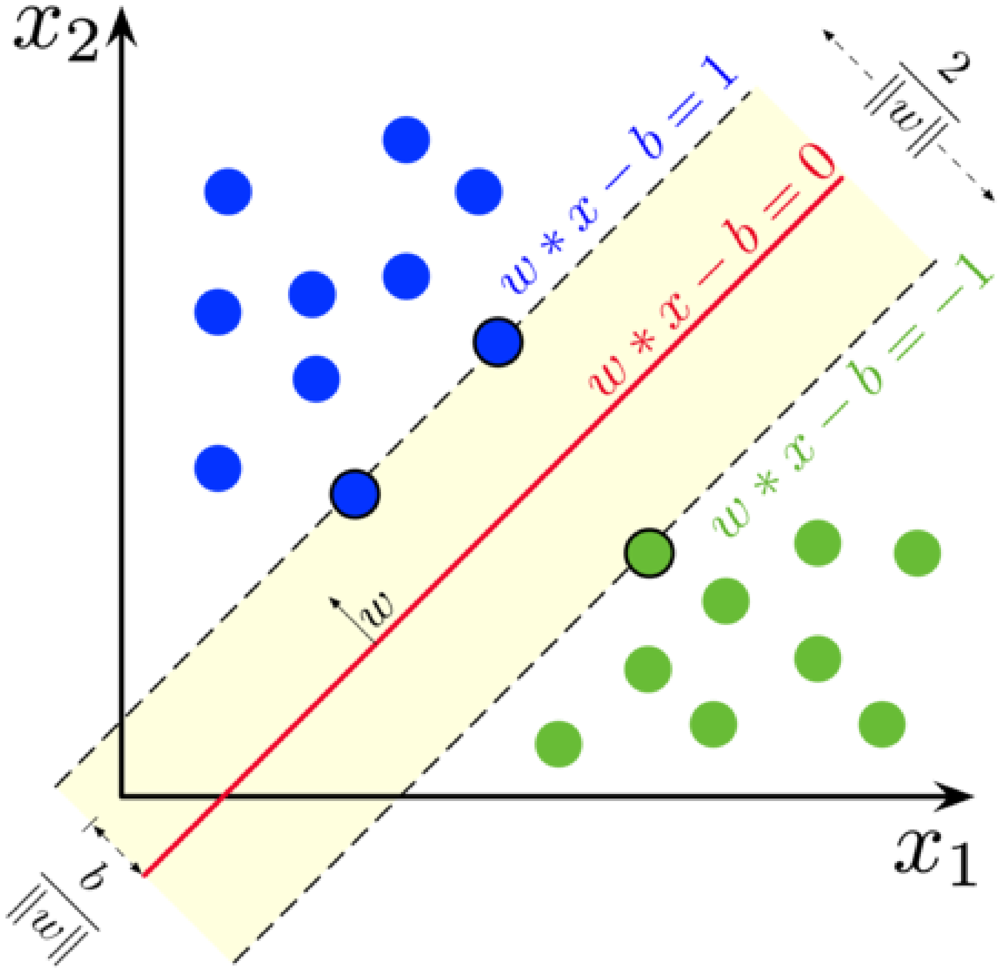

image source: [wikipedia.org]

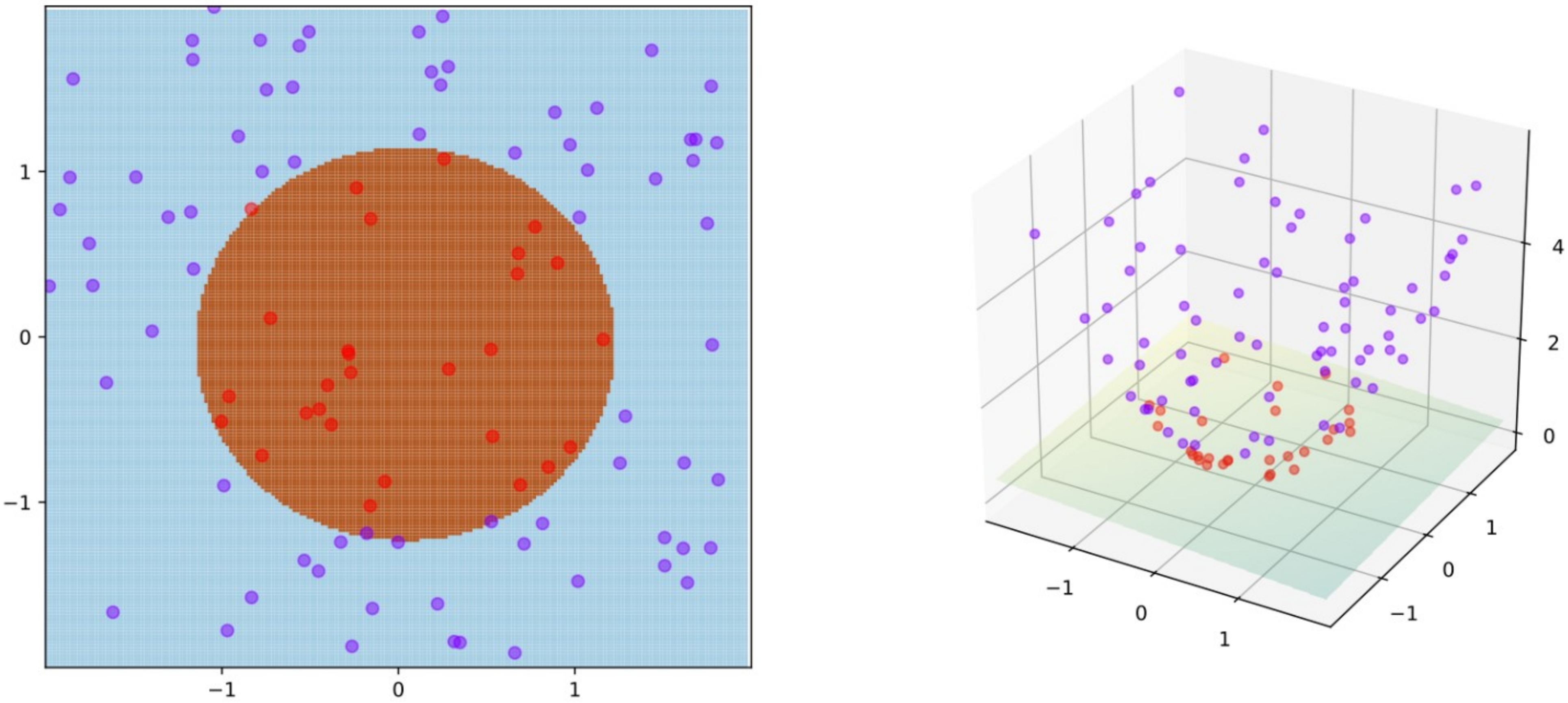

image source: [wikipedia.org]

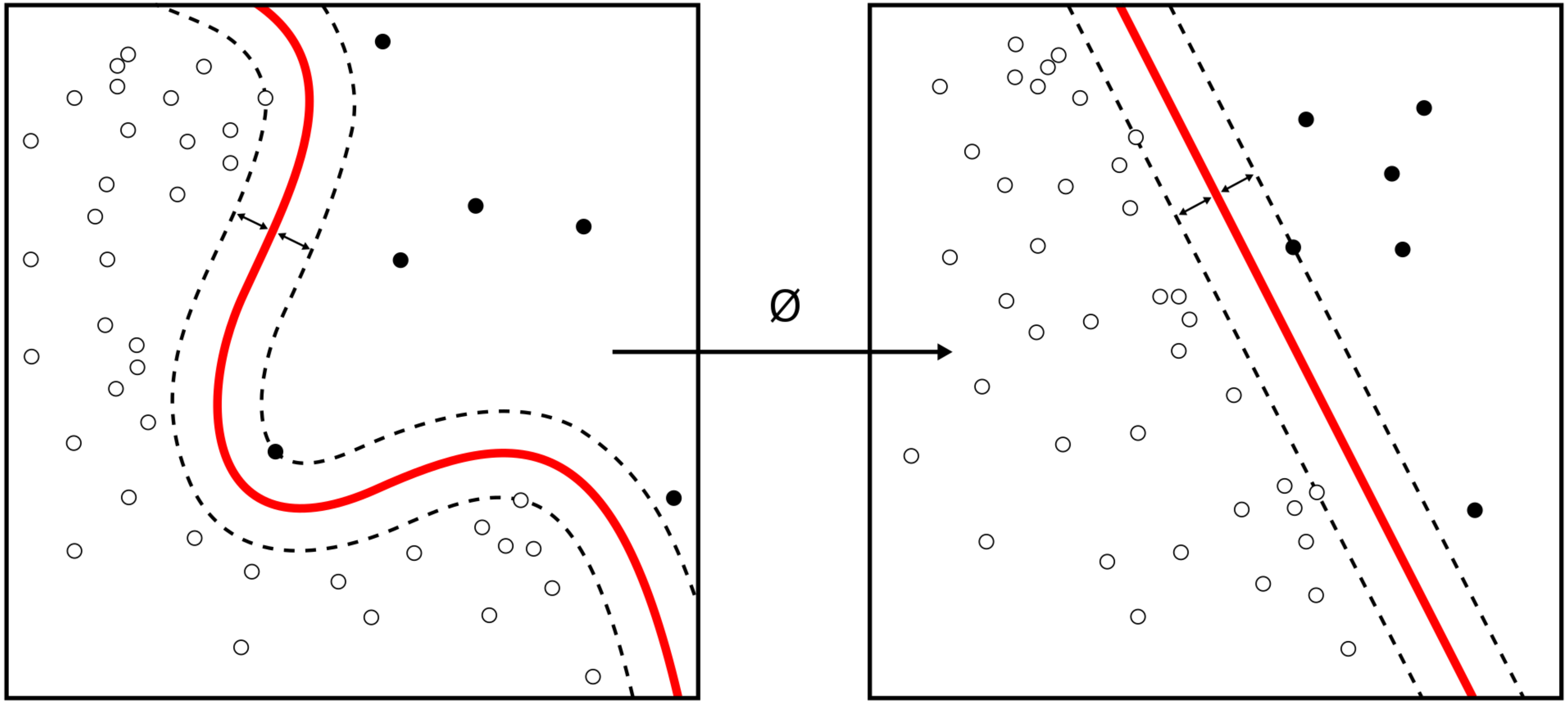

image source: [wikipedia.org]

SVM motivation

- if data is **linearly separable** (by a hyper-plane), then a **wide-margin classifier** is a better classifier
- when data is not linearly separable, project it to a **higher dimension** ($\phi : X \to Z$) in which the labels are linearly separable
- in Z space, **decision boundary** is linear, pinned down only by a few data points called **support vectors**
- the **pre-image** of decision boundary in X space can look complex, but it's the pre-image of a hyper-plane in Z space

break time

the kernel trick

- the math for SVMs can be challenging: linear algebra including some abstract concepts
- the prediction equation is $g(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^T\mathbf{z} + b)$
- we need to calculate $\mathbf{z}_n^T\mathbf{z}_m$ to find a solution
- with the kernel trick we can do it **without explicitly finding the mappings $\mathbf{x}_i \mapsto \mathbf{z}_i$**
- instead we use the kernel K: $\mathbf{z}_n^T\mathbf{z}_m = K(\mathbf{x}_n, \mathbf{x}_m)$

types of kernels

depending on the choice, kernels introduce new hyper-parameters (such as γ and d)

- **linear:** $K(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{x}_n^T \mathbf{x}_m$ is just the standard **dot product**
- **polynomial:** $K(\mathbf{x}_n, \mathbf{x}_m) = (\gamma \mathbf{x}_n^T \mathbf{x}_m + r)^d$ where $\gamma > 0$
- **radial or gaussian:** $K(\mathbf{x}_n, \mathbf{x}_m) = \exp(-\gamma ||\mathbf{x}_n - \mathbf{x}_m||^2)$ where $\gamma > 0$ (which corresponds to an infinite dimensional Z space if we look at its Taylor series expansion)

kernel trick example

- let $\mathbf{x}_n = (a, b)$ and $\mathbf{x}_m = (v, w)$ be vectors that represent two data points (2D in this case, i.e. we have two features)
- then $K(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^T \mathbf{x}_m)^2 = (1 + av + bw)^2$ expands into $(1 + 2av + 2bw + a^2v^2 + b^2w^2 + 2avbw)$
- let $\mathbf{z}_n = (1, \sqrt{2}a, \sqrt{2}b, a^2, b^2, \sqrt{2}ab)$ and
- let $\mathbf{z}_m = (1, \sqrt{2}v, \sqrt{2}w, v^2, w^2, \sqrt{2}vw)$
- then $K(\mathbf{x}_n, \mathbf{x}_m) = \mathbf{z}_n^T \mathbf{z}_m$, where the X space is 2D, but the Z space is 6D, but the left side requires fewer calculations (that's why it's called the kernel **trick**)

soft-margin classifiers

- **hard-margin classifiers** expect perfect separability, but we can add a **slack variable** and get a **soft-margin classifier**
- when the data is not linearly separable, we can adjust the trade-off between margin width and the classification error using the C **hyper-parameter**
- C is the penalty on data points that are on the wrong side of the decision boundary:
  - smaller C: wider margins and lower training accuracy
  - larger C: smaller margins but higher training accuracy

multi-class classification

- let k be the number of classes
- SVMs can only give us binary classifiers but we can still use them to do multi-class classification:
  - **one vs one:** builds $\binom{k}{2}$ classifiers
  - **one vs rest:** (also called. **one vs all**), builds k classifers
- unlike SVMs, neural networks can train multi-class classifiers with a single instance of training
  - logistic regression is like a NN too and can do the same

break time

hyper-parameter tuning

- we can do a **three-way split**:
  - **training data** is for **learning**, **validation data** is for **model selection**, **test data** is for **evaluating final model**
- we can do a **two-way split** and **cross validation**:
  - training data is divided into k folds:
    - $k-1$ folds are for learning, and the kth fold for validation
    - we repeat this k times, one for each fold
  - test data is for **evaluating final model**

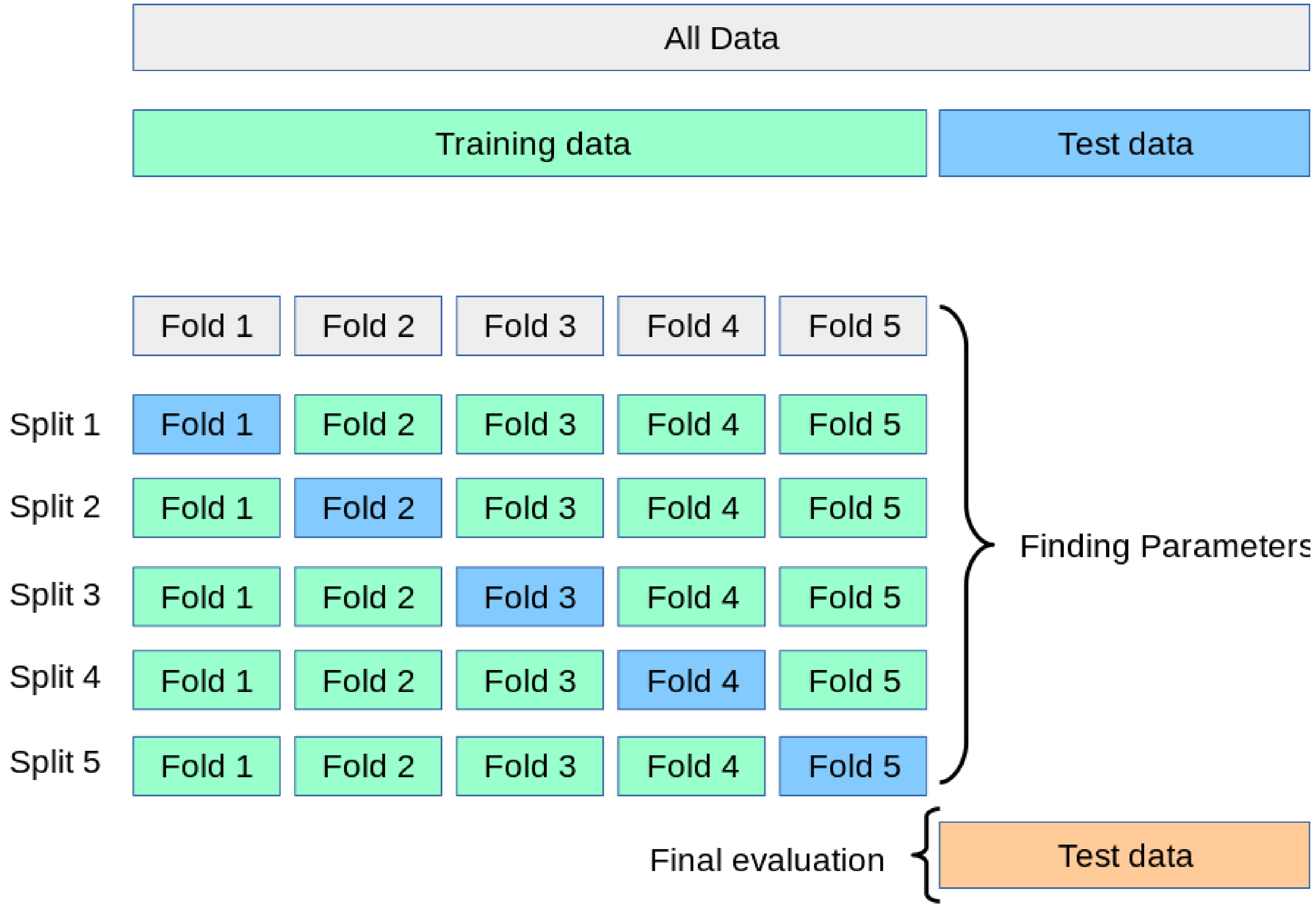

image source: scikit-learn.org

searching the hyper-parameter space

- we search the HP space by training models with a different configurations of HPs and choosing the one with the best cross-validated score
    - **grid search** consists of training a model using **every combination** of HPs
    - **random search** picks a few random combination of HPs and trains a model for each
    - **Bayesian search** picks the next combination of HPs to try based on **exploration-vs-exploitation trade-off**

the end