Lecture 14. Kernel Methods

COMP90051 Statistical Machine Learning

Semester 2, 2015
Lecturer: Andrey Kan

Based on slides provided by
Ben Rubinstein and
http://kernel-methods.net

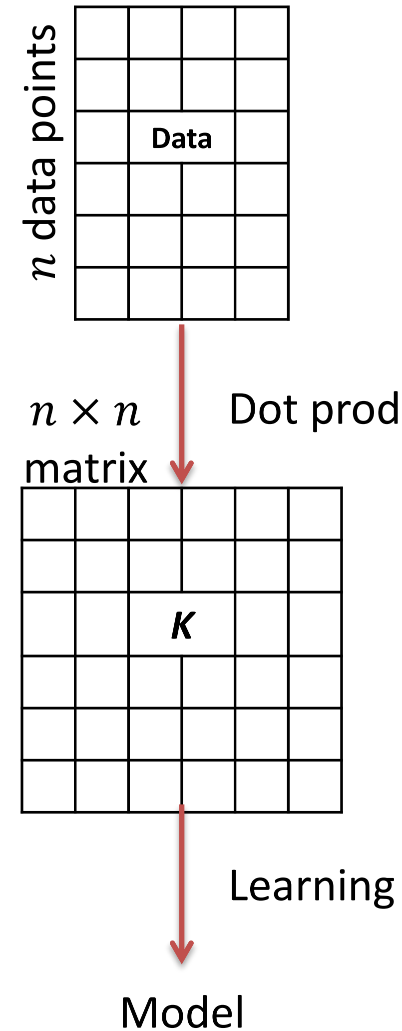
THE UNIVERSITY OF MELBOURNE

Kernel Methods

Very general family of linear techniques that can be made non-linear in a multitude of ways

Overview

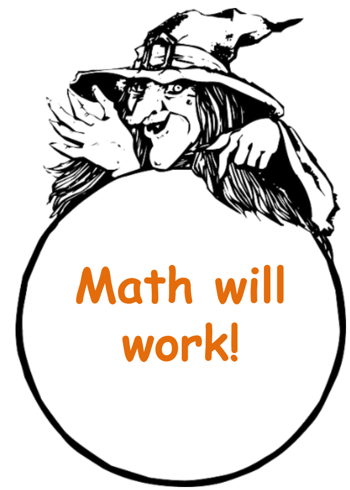
- Kernel matrix $\boldsymbol{K}$
  - Square *n* x *n* matrix measuring pairwise similarity
  - Entries $K_{ij} = \boldsymbol{x}_i \cdot \boldsymbol{x}_j$
- Kernel methods
  - Linear methods relying on training data only through $\boldsymbol{K}$
  - **Make non-linear** by running linear approach in new feature space; but don't need to map the data, just need $\boldsymbol{K}$
- **Modular**: learning algorithm, feature space, decouple
  - Can design general learning algorithms
  - Can design general feature mappings/kernels



Dot products, dot products, …

- How does the SVM depend on the data?

$$\min_{\boldsymbol{w}} \left(\frac{1}{2} \|\boldsymbol{w}\|^2 + \frac{C}{n} \sum_{i=1}^{n} l(1 - y_i \boldsymbol{w} \cdot \boldsymbol{x}_i - b) \right)$$

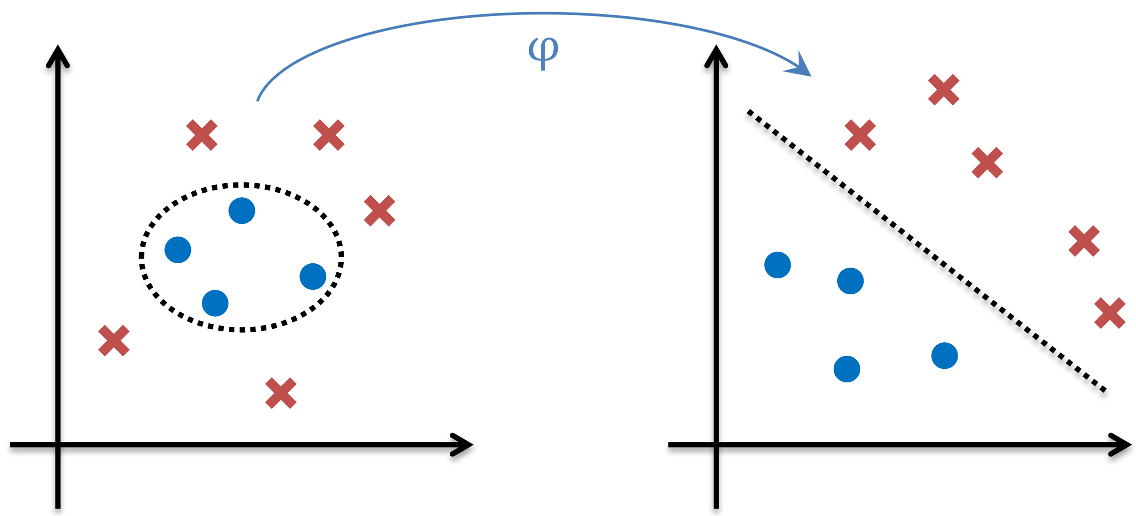


- Famous result: "Representer Theorem"
  * Solution in span of data!! $\boldsymbol{w}^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$
  * Predictions become $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x}$
  * Support vectors are those $\boldsymbol{x}_i$ with $\alpha_i \neq 0$
    → why the SVM is non-parametric!
  * Finding the α_i involves only dot products between data

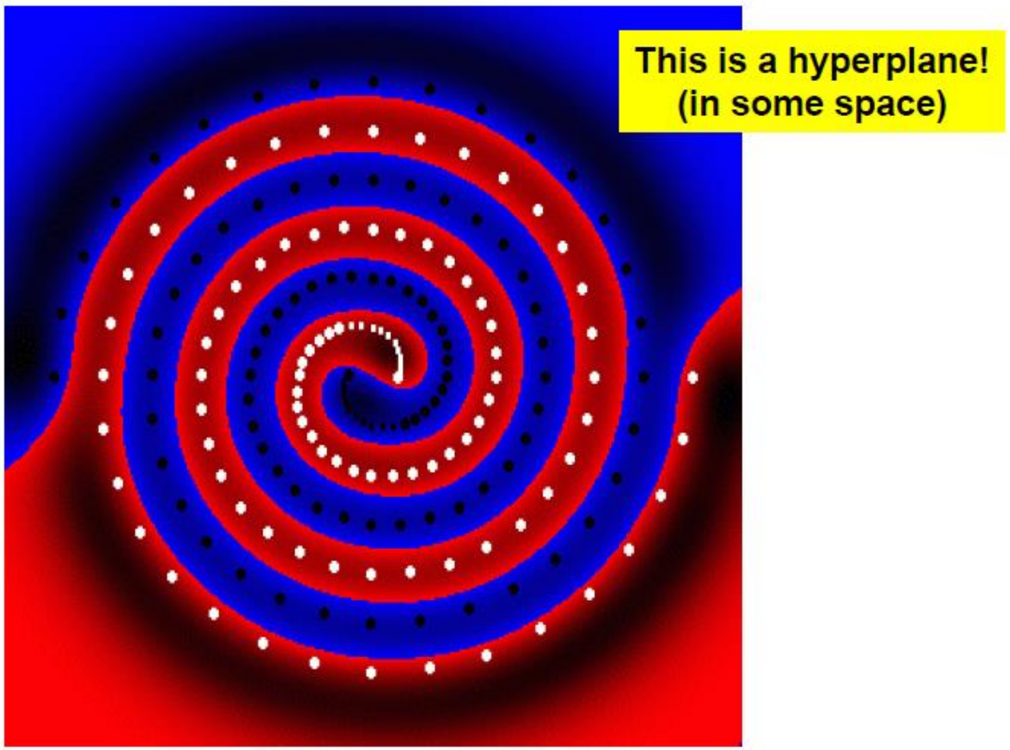
Image: ClkerFreeVectorImages@ pixabay.com (CC0)

Non-linear SVM

- Map data into a new feature space
  - Example: original features and products of pairs
    $\varphi(x_1, \ldots, x_d) = (x_1, \ldots, x_d, x_1 x_1, x_1 x_2, \ldots, x_d x_d)$
- Run linear SVM in new space (use kernel matrix)
- Decision boundary is non-linear in original space



Flexibility of non-linear SVM



www.kernel-methods.net

Blessing of dimensionality

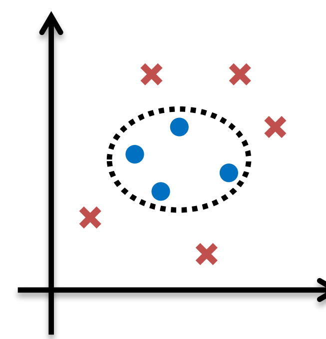
- Kernels implicitly map data to a very high dimensional features space
- Potentially dangerous because it looks like we now have many more parameters than data points
  - That is, $\boldsymbol{w}$ for the transformed features space is high dimensional
  - Curse of dimensionality
  - Danger of overfitting
- Representer theorem: $f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i \cdot \boldsymbol{x}$
  - The number of parameters is at most n, independent of dimensionality
  - Support vectors are those data points with non-zero α_i
- Usually the number of non-zero α_i is smaller than n
  - Sparse kernel machines

Danger!! Potential problem

- Training linear SVM cubic in dimension *d*
- Non-linear → more dimensions → intractable?
- Representer Theorem to the rescue
  - Need only find α_i's which determine weight vector
  - "Only" *n* of them, independent of *d*
    … problematic if "big data"

But wait… what about the kernel matrix?

- *Computing* kernel matrix naïvely expensive
  - Map data to d'-dim feature space, then dot product; takes $O(d'n^2)$
- Computing kernels *directly* can be **cheap as**
- Example: p-degree polynomial kernel
  - $\varphi(x_1, \ldots, x_d) = (x_1, \ldots, x_d, x_1x_1, x_1x_2, \ldots, x_dx_d, \ldots)$
  - $d' = O(d^p)$ is pretty yuck → $O(d^p)$ per matrix entry
  - Trick to cut down to $O(d)$: $\varphi(\boldsymbol{u}) \cdot \varphi(\boldsymbol{v}) = (1 + \boldsymbol{u} \cdot \boldsymbol{v})^p$



- In fact, don't bother with feature spaces just make up matrix
  - Mercer's Theorem: provides a tool for identifying valid kernels

Some popular kernels

Kernel	$k(\boldsymbol{u}, \boldsymbol{v})$	Parameters	Mapping *x* to features
Linear	$\boldsymbol{u} \cdot \boldsymbol{v}$	-	Identity
Polynomial	$(1 + \boldsymbol{u} \cdot \boldsymbol{v})^p$	Integer degree p>1	Degree *p* polynomial in the x_i's
Gaussian aka Radial Basis Function (RBF)	$\exp\left(\frac{-\lVert\boldsymbol{u} - \boldsymbol{v}\rVert^2}{2\sigma^2}\right)$	Width σ	Infinite dimensional, nothing explicit!

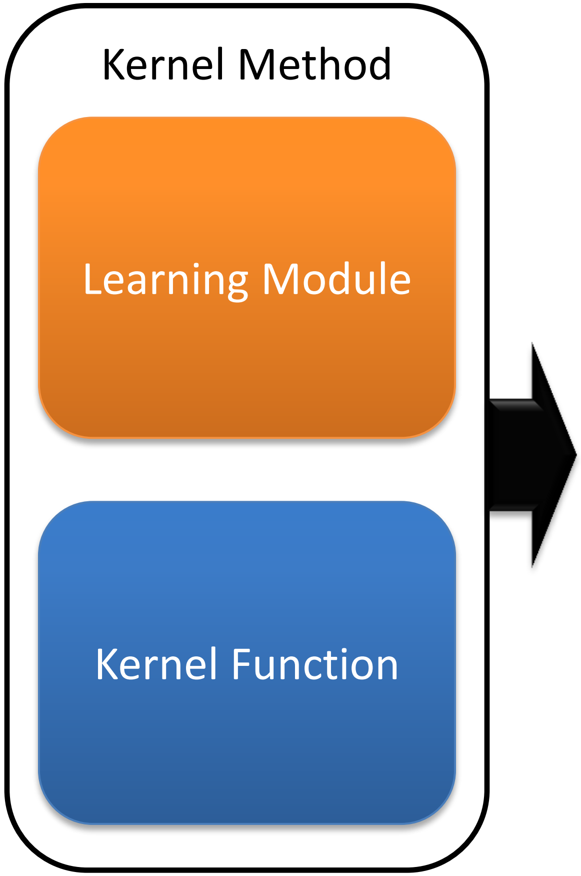
Many more…

Definition 9.1 Polynomial kernel 286
Computation 9.6 All-subsets kernel 289
Computation 9.8 Gaussian kernel 290
Computation 9.12 ANOVA kernel 293
Computation 9.18 Alternative recursion for ANOVA kernel 296
Computation 9.24 General graph kernels 301
Definition 9.33 Exponential difusion kernel 307
Definition 9.34 von Neumann difusion kernel 307
Computation 9.35 Evaluating difusion kernels 308
Computation 9.46 Evaluating randomised kernels 315
Definition 9.37 Intersection kernel 309
Definition 9.38 Union-complement kernel 310
Remark 9.40 Agreement kernel 310
Section 9.6 Kernels on real numbers 311
Remark 9.42 Spline kernels 313
Definition 9.43 Derived subsets kernel 313
Definition 10.5 Vector space kernel 325
Computation 10.8 Latent semantic kernels 332
Definition 11.7 The p-spectrum kernel 342
Computation 11.10 The p-spectrum recursion 343
Remark 11.13 Blended spectrum kernel 344
Computation 11.17 All-subsequences kernel 347
Computation 11.24 Fixed length subsequences kernel 352
Computation 11.33 Naive recursion for gap-weighted subsequences kernel 358
Computation 11.36 Gap-weighted subsequences kernel 360
Computation 11.45 Trie-based string kernels 367
Algorithm 9.14 ANOVA kernel 294
Algorithm 9.25 Simple graph kernels 302
Algorithm 11.20 All–non-contiguous subsequences kernel 350
Algorithm 11.25 Fixed length subsequences kernel 352
Algorithm 11.38 Gap-weighted subsequences kernel 361
Algorithm 11.40 Character weighting string kernel 364
Algorithm 11.41 Soft matching string kernel 365
Algorithm 11.42 Gap number weighting string kernel 366
Algorithm 11.46 Trie-based p-spectrum kernel 368
Algorithm 11.51 Trie-based mismatch kernel 371
Algorithm 11.54 Trie-based restricted gap-weighted kernel 374
Algorithm 11.62 Co-rooted subtree kernel 380
Algorithm 11.65 All-subtree kernel 383
Algorithm 12.8 Fixed length HMM kernel 401
Algorithm 12.14 Pair HMM kernel 407
Algorithm 12.17 Hidden tree model kernel 411
Algorithm 12.34 Fixed length Markov model Fisher kernel 427

www.kernel-methods.net

Modular learning

- More sophisticated learning algorithms, and kernels, exist
- Representer + Mercer Theorems imply their design decouples



$$f(x) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x})$$

…and the algorithms

Computation 2.5 Ridge regression 30
Computation 5.14 Regularised Fisher discriminant 131
Computation 5.15 Regularised kernel Fisher discriminant 133
Computation 6.3 Maximising variance 141
Computation 6.18 Maximising covariance 154
Computation 6.30 Canonical correlation analysis 163
Computation 6.32 Kernel CCA 165
Computation 6.34 Regularised CCA 169
Computation 6.35 Kernel regularised CCA 169
Computation 7.1 Smallest enclosing hypersphere 193
Computation 7.7 Soft minimal hypersphere 199
Computation 7.10 nu-soft minimal hypersphere 202
Computation 7.19 Hard margin SVM 209
Computation 7.28 1-norm soft margin SVM 216
Computation 7.36 2-norm soft margin SVM 223
Computation 7.40 Ridge regression optimisation 229
Computation 7.43 Quadratic e-insensitive SVR 231
Computation 7.46 Linear e-insensitive SVR 233
Computation 7.50 nu-SVR 235
Computation 8.8 Soft ranking 254
Computation 8.17 Cluster quality 261
Computation 8.19 Cluster optimisation strategy 265
Computation 8.25 Multiclass clustering 272
Computation 8.27 Relaxed multiclass clustering 273
Computation 8.30 Visualisation quality 277

Algorithm 5.1 Normalisation 110
Algorithm 5.3 Centering data 113
Algorithm 5.4 Simple novelty detection 116
Algorithm 5.6 Parzen based classifier 118
Algorithm 5.12 Cholesky decomposition or dual Gram–Schmidt 126
Algorithm 5.13 Standardising data 128
Algorithm 5.16 Kernel Fisher discriminant 134
Algorithm 6.6 Primal PCA 143
Algorithm 6.13 Kernel PCA 148
Algorithm 6.16 Whitening 152
Algorithm 6.31 Primal CCA 164
Algorithm 6.36 Kernel CCA 171
Algorithm 6.39 Principal components regression 175
Algorithm 6.42 PLS feature extraction 179
Algorithm 6.45 Primal PLS 182
Algorithm 6.48 Kernel PLS 187
Algorithm 7.2 Samllest hypersphere enclosing data 194
Algorithm 7.8 Soft hypersphere minimisation 201
Algorithm 7.11 nu-soft minimal hypersphere 204
Algorithm 7.21 Hard margin SVM 211
Algorithm 7.26 Alternative hard margin SVM 214
Algorithm 7.29 1-norm soft margin SVM 218
Algorithm 7.32 nu-SVM 221
Algorithm 7.37 2-norm soft margin SVM 225
Algorithm 7.41 Kernel ridge regression 229
Algorithm 7.45 2-norm SVR 232
Algorithm 7.47 1-norm SVR 234
Algorithm 7.51 nu-support vector regression 236
Algorithm 7.52 Kernel perceptron 237
Algorithm 7.59 Kernel adatron 242
Algorithm 7.61 On-line SVR 244
Algorithm 8.9 nu-ranking 254
Algorithm 8.14 On-line ranking 257
Algorithm 8.22 Kernel k-means 269
Algorithm 8.29 MDS for kernel-embedded data 276
Algorithm 8.33 Data visualisation 280

www.kernel-methods.net

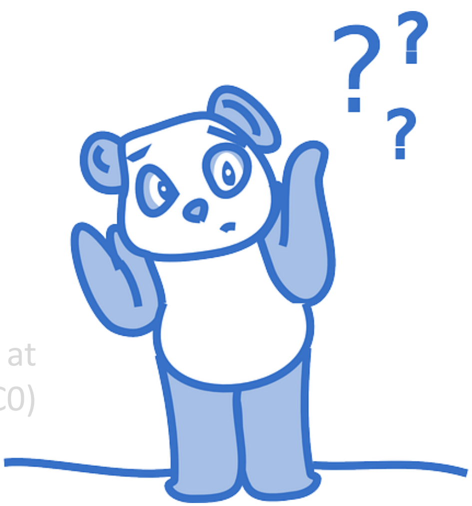
Checkpoint

- Which of the following statements is true?

 Any method that uses a feature space transformation $\varphi(\boldsymbol{x})$ is a kernel method

 Support vectors are points from the training set

 Feature mapping $\varphi(\boldsymbol{x})$ makes data linearly separable



art: OpenClipartVectors at pixabay.com (CC0)

Kernelised SVM

Historically first kernel method

Feature transformations and kernels

- A feature transformation $\varphi(\boldsymbol{x})$ can help transforming data into a linearly separable form
- The kernel trick can be used to perform the transformation implicitly, without actually computing the transformation for each point
- One <u>does not have to use kernels</u>: feature transformation $\varphi(\boldsymbol{x})$ can be also use explicitly
  - If the number of resulting features is reasonable

Solutions to the SVM problem

- Recall the soft-margin SVM problem statement:

$$\min_{\boldsymbol{w}} \left(\frac{1}{2} \|\boldsymbol{w}\|^2 + \frac{C}{2} \sum_{i=1}^{n} \xi_i \right)$$

Subject to $\xi_i \geq 0$, $y_i(\boldsymbol{w}\boldsymbol{x}_i + b) \geq 1 - \xi_i$ for $i = 1, \ldots, n$

- This optimization problem (and many others) can be solved in two ways, called the primal formulation and dual formulation

- The primal approach is to solve the problem as it is stated here
  * If you wish, you can first apply $\varphi(\boldsymbol{x})$ here directly

- The complexity of training is $O(d^3)$, where d is the dimensionality of data
  * Recall that [solving optimization = training]
  * If you have applied $\varphi(\boldsymbol{x})$ then d is the dimensionality of resulting space

The dual formulation

- The idea of the dual formulation is to combine equations for the objective function and constraints into a single new objective function without constraints
  - This can be analytically convenient
  - Also this can lead to a different perspective on the problem
- This method is called *Lagrange multipliers*, and the corresponding new objective function is called *Lagrangian*

Formulating the dual SVM problem

- Soft-margin SVM original problem:

$$\min_{\boldsymbol{w},b} \boxed{\left(\frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i\right)}$$

  - Subject to $\boxed{\xi_i \geq 0}$, $\boxed{y_i(\boldsymbol{w}\boldsymbol{x}_i + b) \geq 1 - \xi_i}$ for $i = 1, \dots, n$

- Lagrangian

$$L_P \stackrel{\text{def}}{=} \boxed{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i} - \sum_{i=1}^{n} \alpha_i \boxed{\left[y_i(\boldsymbol{w}\boldsymbol{x}_i + b) \geq 1 - \xi_i\right]} - \sum_{i=1}^{n} \mu_i \boxed{\xi_i}$$

- Fix α_i, μ_i and minimise Lagrangian with respect to $\boldsymbol{w}, b, \xi$:
  - Set partial derivatives $\frac{\partial L_P}{\partial w_i}, \frac{\partial L_P}{\partial b}, \frac{\partial L_P}{\partial \xi_i}$ to zero

Solving the SVM problem

- Setting the partial derivatives to zero gives
  - $w^* = \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i$
  - $0 = \sum_{i=1}^{n} \alpha_i y_i$
  - $\alpha_i = C/n - \mu_i$
  - $\alpha_i, \mu_i, \xi_i \geq 0$
- Substituting these back into L_P gives
  - $L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j$

Proofs are outside the scope

- L_D gives a lower bound on the original solution
- Therefore we maximize L_D w.r.t. α_i, s.t. $0 = \sum_{i=1}^{n} \alpha_i y_i$ and $0 \leq \alpha_i \leq \frac{C}{n}$
- In addition, Karush-Kuhn-Tucker conditions complete unique characterization of solution

About those alpha's

- Transform

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n} \xi_i$$

into "dual problem"

$$\max_{\boldsymbol{\alpha}} \ \boldsymbol{\alpha} \cdot \mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^T G \boldsymbol{\alpha}$$

subject to $0 \leq \alpha_i \leq \frac{C}{n}$

where $\underbrace{G_{ij}}_{\text{Gram matrix}} = y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j = y_i y_j \underbrace{K_{ij}}_{\text{Kernel matrix}}$

Want different non-linear mapping? Swap out *K*

- Quadratic program but in *n*, not *d* variables
- See how regularisation restricts data influence?

Solutions to the SVM problem

- The complexity of the primal solution is $O(d^3)$, where d is the dimensionality of data (after applying φ)
- The complexity of the dual solution is $O(n^3)$
- For “big data”, and finite-dimensional explicit transformation $\varphi(\boldsymbol{x})$ the primal form can be preferable

SVM: decisions to make

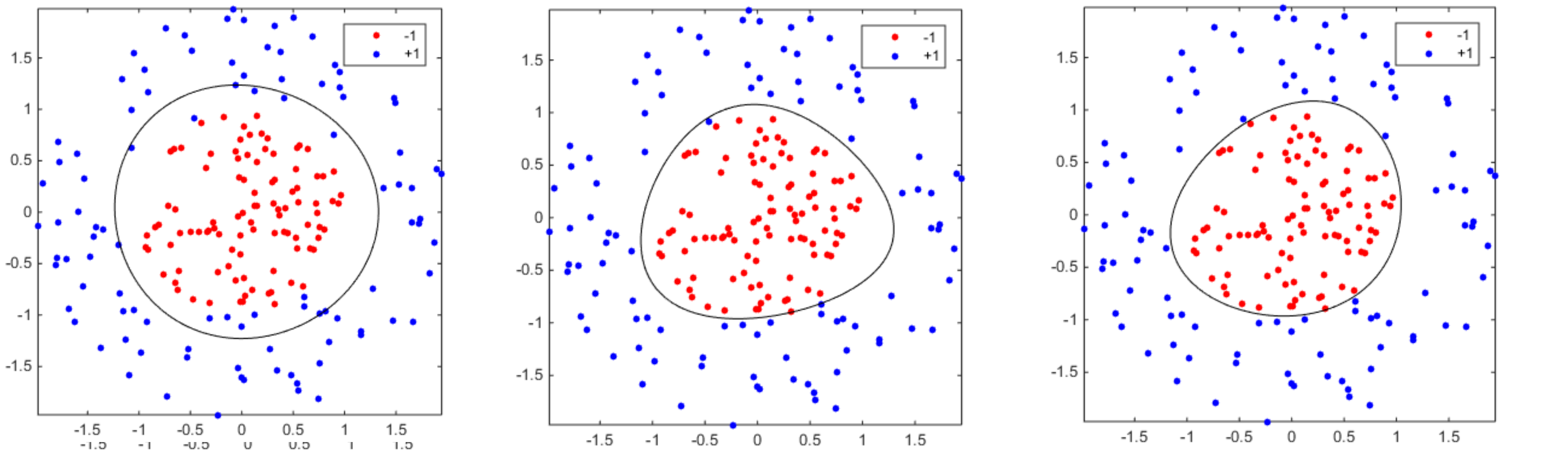
- Regularisation parameter C
- Choice of a kernel
- Note that some kernels introduce additional parameters, e.g., p in polynomial kernel $(1 + \boldsymbol{u} \cdot \boldsymbol{v})^p$

- SVM is a sparse kernel machine
- Regularisation also helps avoid the curse of dimensionality
- In practice: everything is set using cross-validation

Varying regularization parameter

Low C, tendency to underfit

High C, tendency to overfit



Polynomial kernel is used in all cases

SVMs for multiclass classification

- Multiclass classification has to be reduced to binary classification
- ***One-versus-all***: K binary classifications are made for each of the K classes. Each time, classification is "class k" vs "the rest". A new instance is assigned to class for which it has the strongest confidence (furthest away from the boundary).
- ***One-versus-one***: $K(K-1)/2$ classifications are made, covering all possible pairs of classes. A new instance is assigned the most voted class (class that gets most number of "wins").

SVM vs Neural Networks

Kernelised SVM	Feed-forward ANN
Linear SVM applied to non-linear data using kernelisation	Non-linear model by design (non-linear transfer functions)
Variable number of parameters, up to N (data points): predictions are made using support vectors	Parametric model: fixed number of parameters
Inherently batch training (see dual formulation, Kernel matrix)	Naturally amenable to online training (e.g., stochastic gradient descent)
SVM training always finds a global minimum (convex problem)	Global minimum is not guaranteed
Single dimensional output	Naturally adaptable for multiclass classification and multidimensional output

Summary

- Kernel methods
  - Modular decoupling of linear learner and feature mapping
  - Built on Representer Theorem, Mercer’s Theorem
  - Tonnes of kernel functions
  - Tonnes of kernel methods (SVM, PCA)