Learning with kernels and SVMs

Théomé Borck, *February 8th, 2021*

Seminar “Mathematics for Data Science”

References

- Schölkopf, B and Smola, A. (2002) Learning with kernels, Support Vector Machines, Regularization, Optimization and Beyond. *MIT Press, Cambridge, MA.*
- Poggio T. and Smale, S. (2003) The Mathematics of Learning: Dealing with Data. *Notices of the American Mathematical Society (AMS), Vol. 50, No. 5, 537-544.*

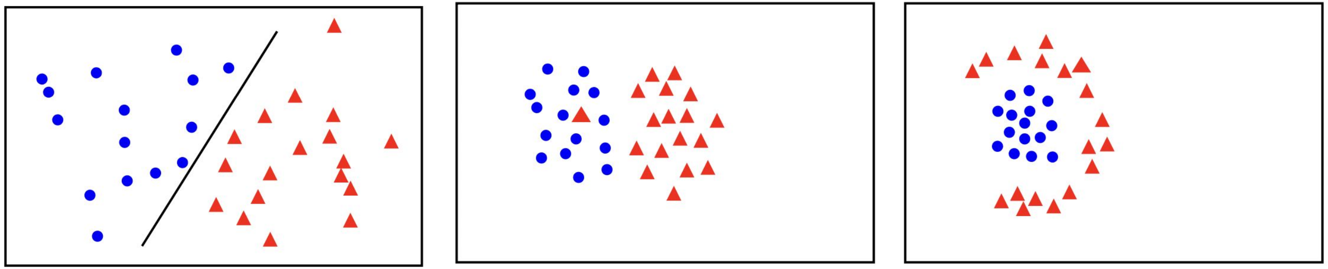
Additional references

- Zisserman, A. (2015) Lecture 3: SVM dual, kernels and regression. *C19 Machine Learning, Oxford University, Oxford.*
- Günnemann, S. (2019) Lecture 8: SVM and kernels. *IN2064 Machine Learning WS2019, TUM, Munich.*

Table of Contents

Binary classification

= given training data, learn a **classifier** to discriminate data instances



Linearly separable **Non-linearly separable**

$$(x_i, y_i), i \in [N], y_i \in \{-1, 1\} \rightarrow \begin{array}{l} f(x_i) = w^t x_i + b \\ y_i f(x_i) > 0 \end{array}$$

→ **Perceptron** algorithm

Converges slowly, only works with linearly-separable data and no margin interpretation

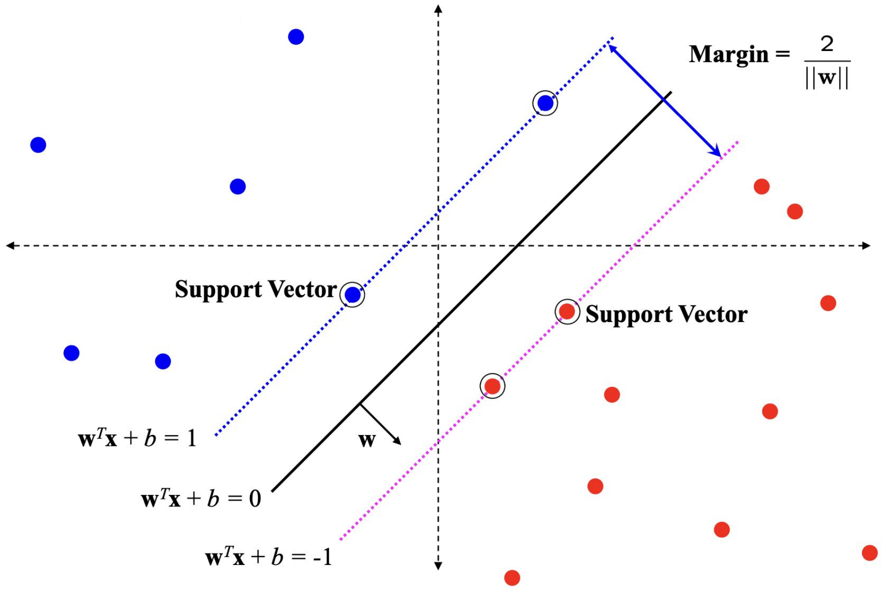
1. Support Vector Machine

The intuition

Objective = Find a hyperplane that separates both classes with the maximum margin

→ more stable model given perturbations of the training data
→ better generalization

Actual rigorous motivation
→ Statistical Learning Theory (Vapnik, 1995)



An optimization problem

The SVM objective comes down to:

L2-norm

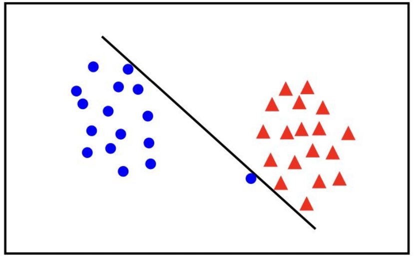
Minimize $f_0(\boldsymbol{w}, b) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}$

subject to $f_i(\boldsymbol{w}, b) = y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 \geq 0 \quad \text{for } i = 1, \ldots, N\,.$

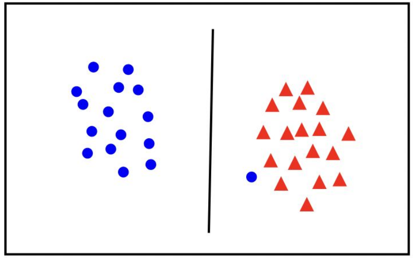
This is a convex and quadratic optimization problem, subject to linear constraints.

→ a global optimal solution can be obtained, i.e. a unique minimizer.

Soft-SVM: introducing slack variables



Points can be linearly separated but the margin is very narrow.



The larger margin solution is better, but one constraint is violated.

We introduce a "trade-off" between the margin and the number of mistakes on the training data

Soft-SVM: introducing slack variables

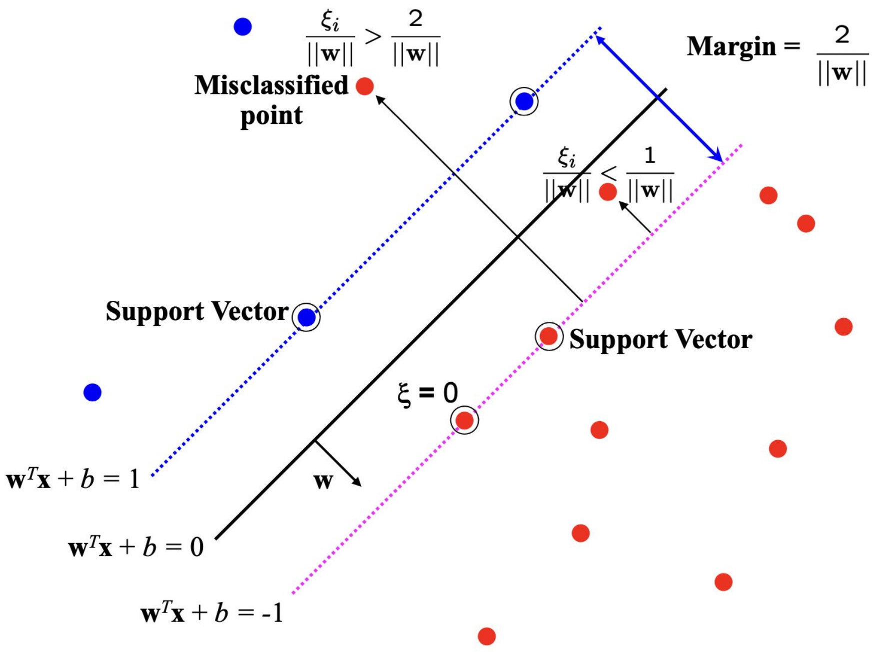
Idea = Relax the constraints but punish as much as necessary the relaxation of a constraint

$\longrightarrow$ slack variable $\xi_i \geq 0$

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i \qquad \text{for all } i\,.$$

For $0 < \xi_i \leq 1$, the instance is between margin and correct side of the hyperplane.
$\longrightarrow$ margin violation

For $\xi_i > 1$, point is misclassified.



Soft-SVM: introducing slack variables

The SVM objective becomes:

Minimize $$f_0(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i$$

subject to $$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \geq 0 \qquad i = 1, \dots, N\,,$$
$$\xi_i \geq 0 \qquad i = 1, \dots, N\,.$$

C > 0 determines how heavy is the violation punished → regularization parameter.

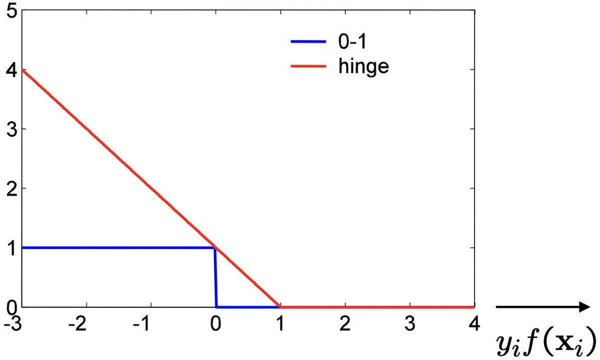
The best C is often found through hyperparameter search.

SGD to solve the SVM problem - Pegasos algorithm

The slack variables can be concisely rewritten as: $$\xi_i = \begin{cases} 1 - y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b), & \text{if } y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) < 1 \\ 0 & \text{else} \end{cases}$$

Hence, the problem becomes an unconstrained optimization problem:

$$\min_{\boldsymbol{w},b} \quad \underbrace{\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}}_{\text{Regularization}} + C\sum_{i=1}^{N} \underbrace{\max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\}}_{\text{Hinge loss}}$$



→ can be optimized using standard gradient-based methods such as Stochastic Gradient Descent.

From primal to dual problem

Using the Lagrangian multipliers method (or through the Representer Theorem), we can convert our problem:

Primal problem

Minimize $$f_0(\boldsymbol{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i$$

parameters = vector space dim. = d

subject to $$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1 + \xi_i \geq 0 \qquad i = 1, \ldots, N\,,$$
$$\xi_i \geq 0 \qquad i = 1, \ldots, N\,.$$

Dual problem

Maximize $$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boxed{\boldsymbol{x_i}^T\boldsymbol{x_j}}$$

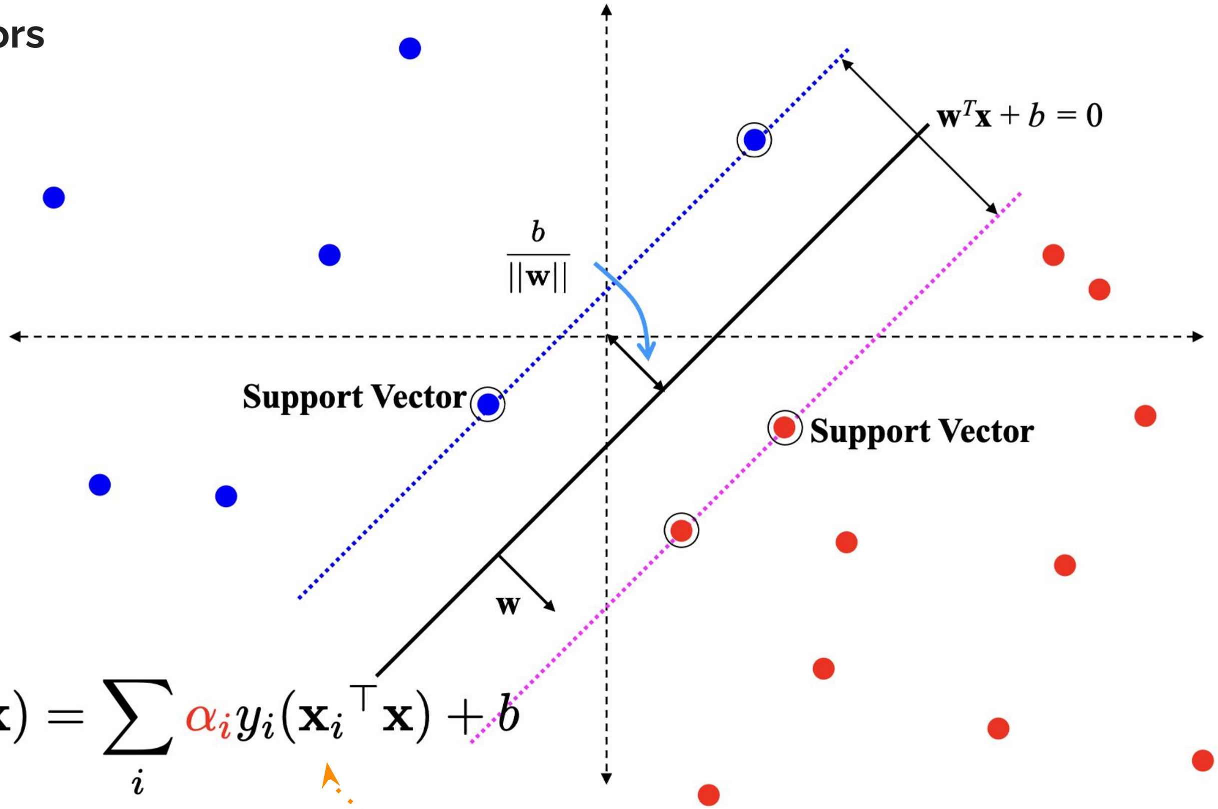
parameters = N

subject to $$\sum_{i=1}^{N}\alpha_i y_i = 0 \qquad 0 \leq \alpha_i \leq C$$

Inner product of data instances

1. Support Vector Machine

Support vectors



$$f(\mathbf{x}) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

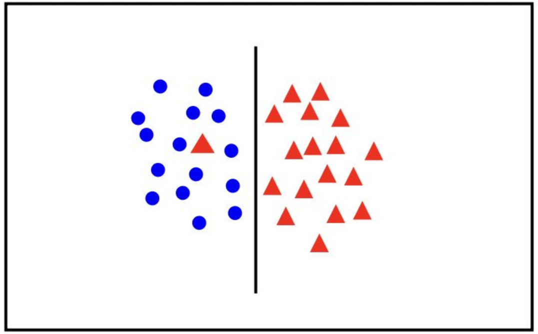
Support vectors

1. Support Vector Machine

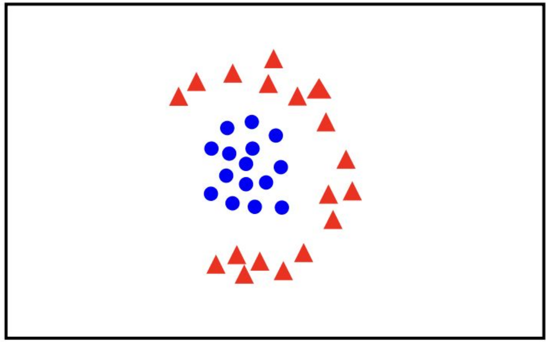
Where do we stand ?

- We have a linear binary classifier which maximizes the margin between two classes.
- Our objective function is convex so we can optimize using gradient descent.
- We derived the SVM through a dual formulation, solvable using standard QP libraries.

But how do we handle the more complex non-linearly separable cases ?



Handled by slack variables



Underlying non-linear distribution

2. Kernels

Feature map projection

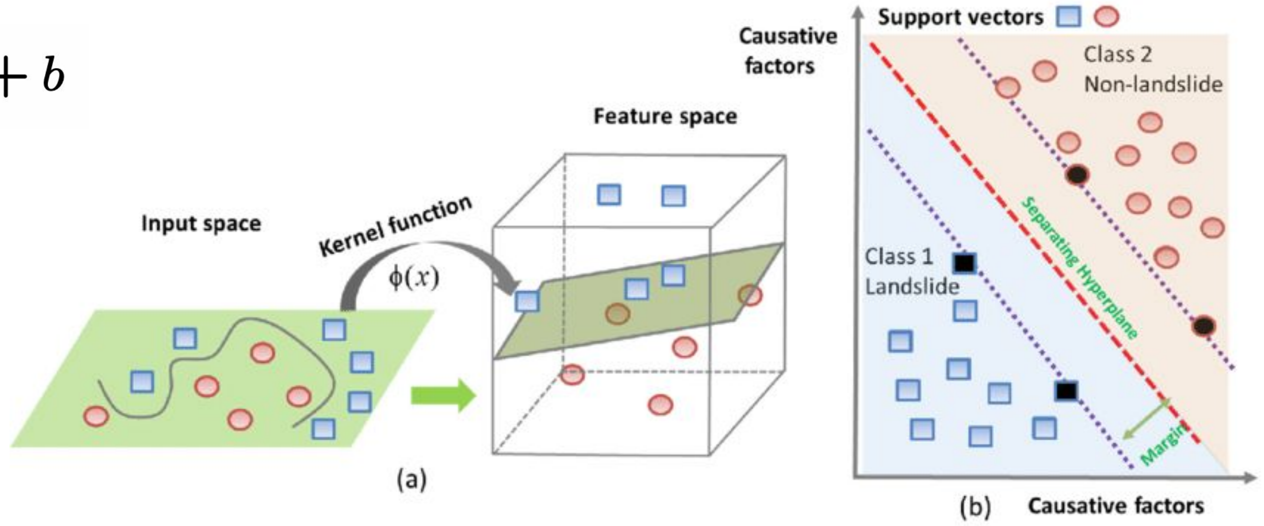
To find a linearly-separable space, we project to a higher-dimensional space.

$$\Phi : \mathbf{x} \rightarrow \Phi(\mathbf{x}) \qquad \mathbb{R}^d \rightarrow \mathbb{R}^D$$

The linear classifier to learn is now in this projected space.

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b$$

Feature map



The “kernel trick”

With the feature projection, our dual formulation

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

becomes

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$$

Defining the kernel function:

$$k : R^d \times R^d \rightarrow R$$

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) := \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$$

We can rewrite the dual as

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

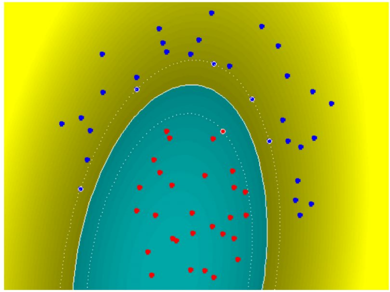
The classifier can be learnt and applied without explicitly computing the feature map projection.
Complexity of learning only depends on N and not on D.

2. Kernels

Examples

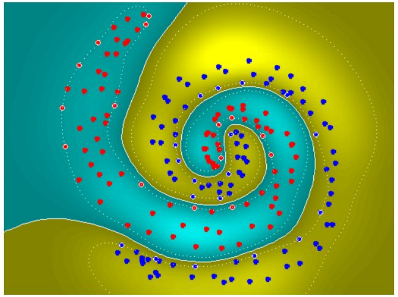
Linear kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a}^T \boldsymbol{b}$

Polynomial kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a}^T \boldsymbol{b})^p$ or $(\boldsymbol{a}^T \boldsymbol{b} + 1)^p$



→ contains all polynomials terms up to degree p

Gaussian kernel: $k(\boldsymbol{a}, \boldsymbol{b}) = \exp\left(-\frac{\|\boldsymbol{a}-\boldsymbol{b}\|^2}{2\sigma^2}\right)$



→ projection to an infinite dimensional feature space

$$exp(-\frac{1}{2}\|x-y\|_2^2) = \underbrace{exp(-\frac{1}{2}\|x\|_2^2)}_{c(x)}\underbrace{exp(-\frac{1}{2}\|y\|_2^2)}_{c(y)} \sum_{k=0}^{\infty} \frac{(x,y)^k}{k!} = \sum_{k=0}^{\infty} \left\langle \sqrt[k]{\frac{c(x)}{k!}}x, \sqrt[k]{\frac{c(y)}{k!}}y \right\rangle^k$$

Extending kernels to non-numerical data

Mercer's theorem:

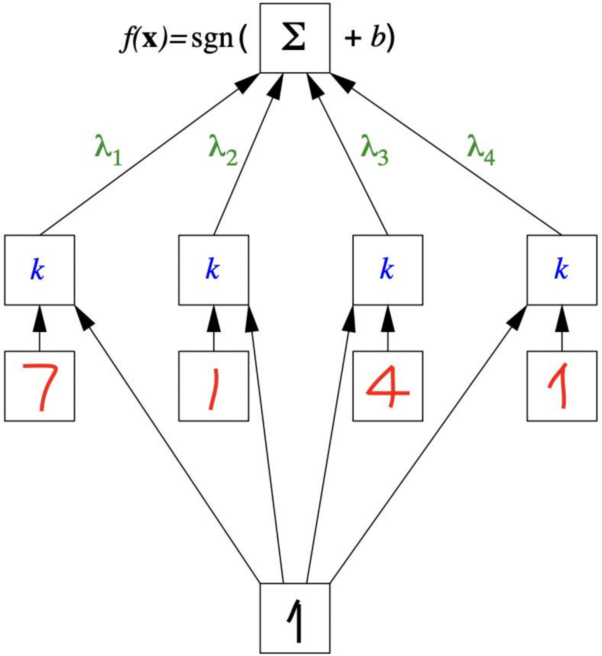
A kernel is valid if it gives rise to a symmetric, positive semidefinite kernel matrix **K** for any input data **X**.

$$\boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & k(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_N) \\ k(\boldsymbol{x}_2, \boldsymbol{x}_1) & k(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_2, \boldsymbol{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_N, \boldsymbol{x}_1) & k(\boldsymbol{x}_N, \boldsymbol{x}_2) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{pmatrix}$$

Hence, we can use kernels to encode similarity with non-numerical data such as strings or graphs.

→ *k("Coca", "Pepsi") = -5*   *k("Coca", "Wine") = 8*

$$f(\mathbf{x}) = \mathrm{sgn}\left(\sum \lambda_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b \right)$$



Classification

Weights

Comparison → *K(X, Xi)*

Support vectors

Input data

SVM for multi-classification

Two versions:

One-vs-rest = Train C SVM models for C classes, where each SVM is being trained for classification of one class against all the remaining ones.
→ The predicted class is then the one, where the distance from the hyperplane is maximal.

One-vs-one
= Train a classifier for all possible pairings and evaluate all.
→ The predicted class is the one with the majority vote
(the votes are weighted according to the distance from the margin).

3. SVM in practice

Applications

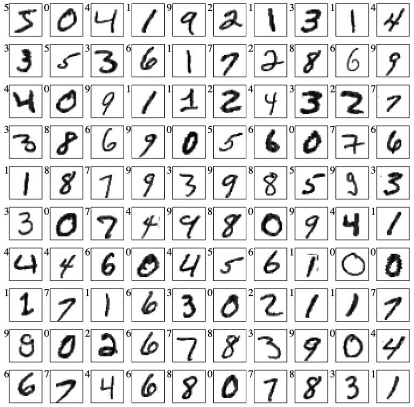
- Face detection
- Image classification
- Handwriting recognition
- Bioinformatics - protein/gene classification
- News classification
- … and many other classification problems.

MNIST Benchmark

60000 training examples

10000 test examples

28x28



Classifier	test error	reference
linear classifier	8.4%	[7]
3-nearest-neighbour	2.4%	[7]
SVM	1.4%	[12]
Tangent distance	1.1%	[57]
LeNet4	1.1%	[38]
Boosted LeNet4	0.7%	[38]
Translation invariant SVM	0.56%	[18]

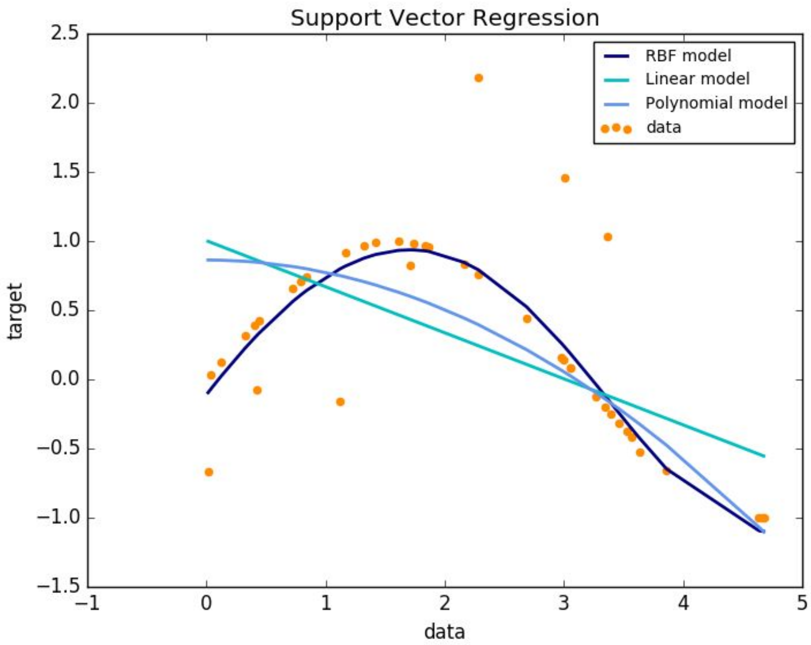
SVM for regression

We want to find a prediction line (or an hyperspace in a high-dimensional feature space) which deviates less than $\boldsymbol{\varepsilon}$ with respect to the ground truth $\boldsymbol{y}$.

$$\min_{w,\xi_i} \underbrace{\frac{1}{2}\|w\|_2^2}_{\text{Regularization}} + C\sum_{i=1}^{N}|\xi_i|$$

subject to

$$|y_i - w^t x_i| \leq \varepsilon + |\xi_i|$$



A key algorithm to learning theory

"The Mathematics of Learning: Dealing with Data." from Poggio and Smale, written in 2003.
→ Outlines the mathematical foundations of learning theory and describe a key algorithm of it.

Supervised learning = systems trained, instead of programmed, with a set of examples.
Training = synthesize a function that best represents the relation between inputs and outputs.
Learning = a principle method for distilling predictive and hence scientific theories from the data.

A key algorithm to learning theory

1. Start with data $(x_i, y_i)_{i=1}^m$
2. Choose a symmetric, positive definite function $K_x(x') = K(x, x')$, continuous on $X \times X$. A kernel $K(t, s)$ is *positive definite* if $\sum_{i,j=1}^n c_i c_j K(t_i, t_j) \geq 0$ for any $n \in \mathbb{N}$ and choice of $t_1, ..., t_n \in X$ and $c_1, ..., c_n \in \mathbb{R}$. An example of such a Mercer kernel is the Gaussian

$$K(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}. \tag{1}$$

restricted to $X \times X$.

3. Set $f : X \to Y$ to

$$f(x) = \sum_{i=1}^m c_i K_{x_i}(x). \tag{2}$$

where $\mathbf{c} = (c_1, ..., c_m)$ and

$$(m\gamma\mathbf{I} + \mathbf{K})\mathbf{c} = \mathbf{y} \tag{3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{K}$ is the square positive definite matrix with elements $K_{i,j} = K(x_i, x_j)$ and $\mathbf{y}$ is the vector with coordinates y_i. The parameter γ is a positive, real number.

→ Equation 2 approximates the unknown function by a weighted superposition of Gaussian “blobs” , each centered at the location of one of the m examples.

→ The algorithm performs well in a number of applications involving regression as well as binary classification.

The derivation of this key algorithm

The algorithm can be derived from Tikhonov regularization.
We use Empirical Risk Minimization to find a function *f* which minimizes:

$$\frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2 + \gamma \|f\|_K^2,$$

Norm in $\mathcal{H}_K$ – the Reproducing Kernel Hilbert Space (RKHS), defined by the kernel K

- Proximal vector machines
- For SVM, we replace the loss function by a more adapted non-quadratic loss $V(f(x,y)) = (1 - yf(x))_+$
  Coefficients are then retrieved by solving the quadratic programming problem.
- Bayesian interpretation of the regularization term

A theoretical approach to the bias/variance trade-off

We can link the algorithm to some basics of the learning theory, which yields:

$$\int_X (f_z - f_\rho)^2 = S(z, \mathcal{H}) + \int_X (f_\mathcal{H} - f_\rho)^2 = S(z, \mathcal{H}) + A(\mathcal{H})$$

S must be estimated in probability over z and the estimate is called the sample error. It is studied thanks to the theory of probability.

A is dealt with via approximation theory and is called the approximation error.

→ Related to the **bias/variance trade-off** in statistics.
→ We can apply theoretical bounds to these errors.

A theoretical framework to understand SVM and its derivation

Basis concept of SVM in linear separable case = *separation by an hyperplane maximizing the margin*.

In the non-separable case, this interpretation loses most of its meaning.

→ A more general and simpler framework for deriving SVM algorithms for classification and regression is to regard them as special cases of regularization and follow the proposed key algorithm.

Maximizing the margin is exactly equivalent to minimizing $||w||^2$ which corresponds to minimizing the RKHS norm.

Conclusion

Support Vector Machine

- A classification or regression algorithm which learns from labeled data → supervised.
- Can be interpreted through the concepts of hyperplanes and margin, <u>but</u> also more theoretically from the learning theory.

Kernels

- Provide a “trick” to help SVMs handle the non-linearly separable data.
- Also take root in a key algorithm derived from the learning theory.
- Choosing a kernel can be challenging → *no free lunch ;)*

Thank you!
Any questions ?