

Support Vector Machines and Kernels

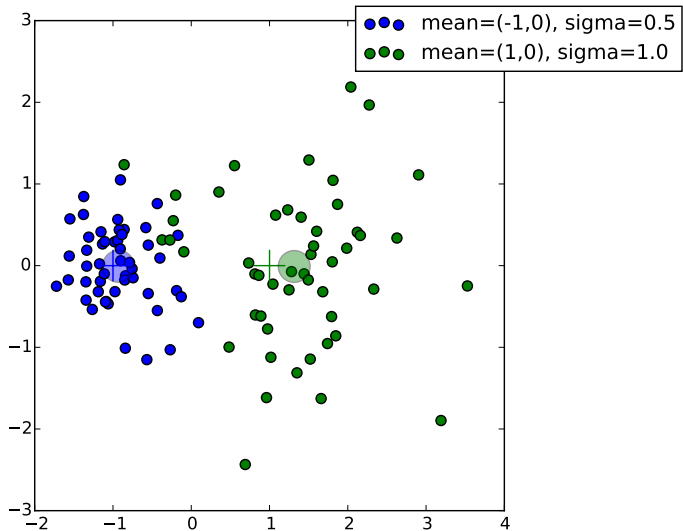
Krzysztof Czarnowski

v2, May 22, 2017

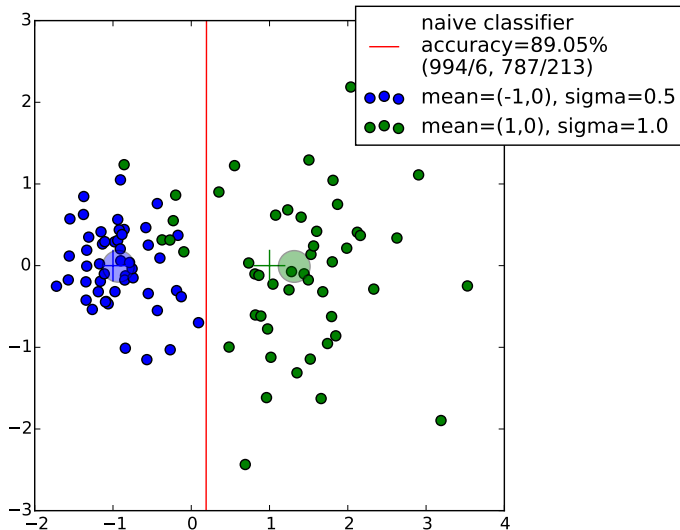
- Linear classification and linear SVM.
- Convex optimization and duality.
- Features, kernels and kernel trick.
- Nonlinear (kernel) SVM.
- Other kernel methods
- Examples
- Some diversions. . . (?)
- Not today:
 - In depth theory of kernels and RKHS.
 - VC-dimension and statistical theory of learning
 - More omissions. . .

- Naive, based on class means
— why it's not the best possible one.
- Fisher classifier — the celebrated classics! LDA.
- Linear SVM.
- Underlying optimization scheme.

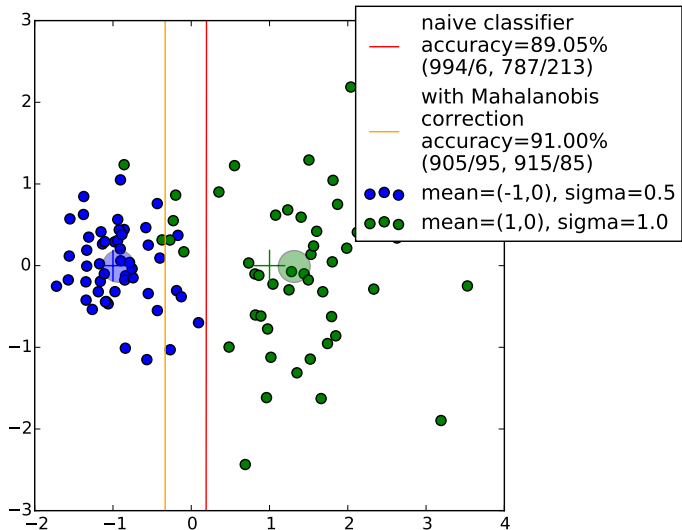
Example 1: symmetric gaussian distributions



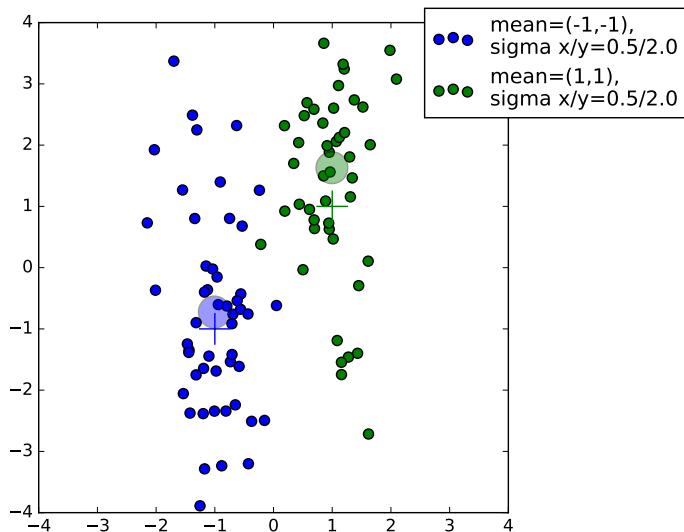
Example 1: naive attempt is not so bad ...



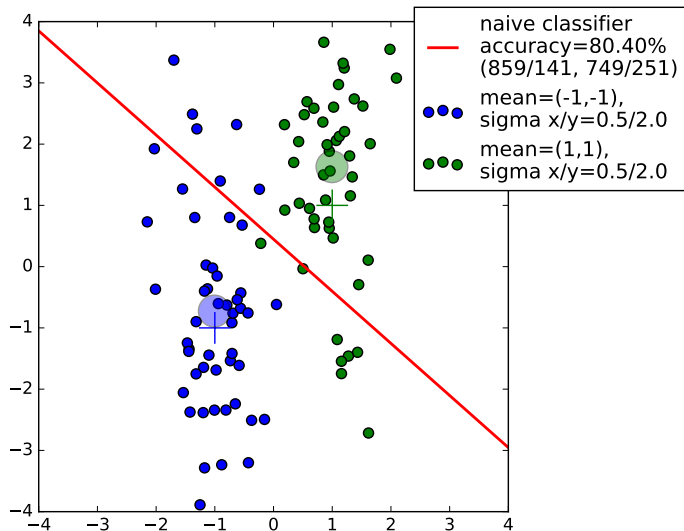
Example 1: better with correction for pdf widths'



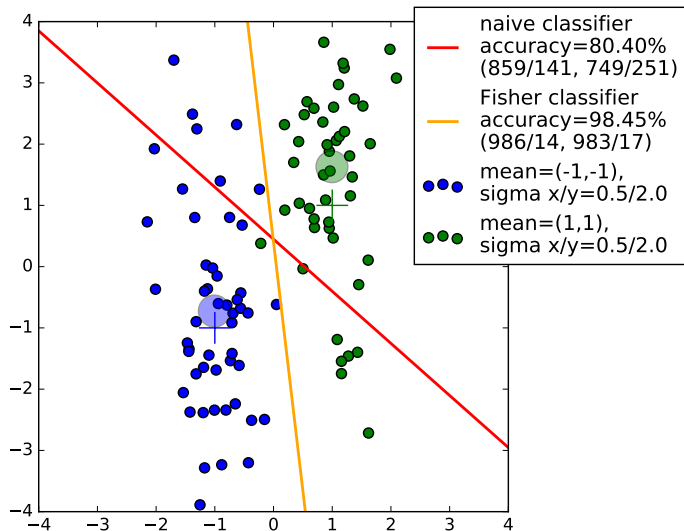
Example 2: asymmetric gaussian distributions



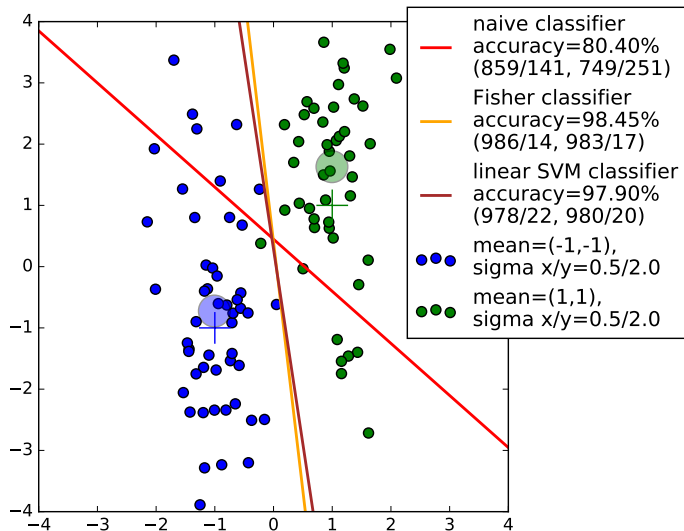
Example 2: naive attempt is worse.



Example 2: classical Fisher classifier much better

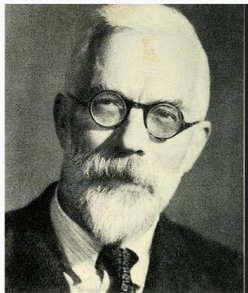


Example 2: linear SVM classifier almost as good



Fisher classifier

The idea was introduced by Ronald Fisher in his classic paper from 1936 on discriminating three Iris flower species based on four anatomical measurements (Iris data)



Ronald Fisher, 1890-1962

- Consider the two class problem and seek the best discriminating direction \mathbf{w} in the observation space.
- Let \mathbf{m}_1 and \mathbf{m}_2 denote estimated class means. It's natural to choose \mathbf{w} to maximize *inter-class separation* along \mathbf{w}

$$\mathbf{w} \cdot (\mathbf{m}_2 - \mathbf{m}_1)$$

- But then obviously $\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1$ — we just get the naive classifier! So another optimization criterium is added...

Graphics: https://en.wikipedia.org/wiki/Ronald_Fisher

Fisher classifier, continued

- Let S_1, S_2 denote the corresponding estimated in-class covariance matrices.
- Another natural objective while choosing \mathbf{w} is to minimize both of the *in-class variances* along \mathbf{w} , or just the sum of these:

$$\mathbf{w} \cdot S_1 \mathbf{w} + \mathbf{w} \cdot S_2 \mathbf{w} = \mathbf{w} \cdot (S_1 + S_2) \mathbf{w}$$

- The final optimization criterium is

$$\arg \max_{\mathbf{w}} \frac{(\mathbf{w} \cdot \mathbf{m}_{12})^2}{\mathbf{w} \cdot S_{12} \mathbf{w}} \quad (1)$$

where $\mathbf{m}_{12} = \mathbf{m}_2 - \mathbf{m}_1$ and $S_{12} = S_1 + S_2$.

- This can be interpreted as maximizing signal-to-noise ratio in the training data!
- (Why do we absolutely need the square in the nominator?)

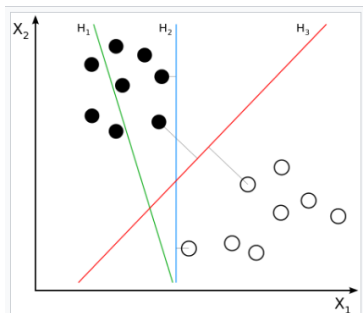
- The solution is found by equating the derivative to zero (some math involved):


$$\mathbf{w} \propto \mathbf{S}_{12}^{-1} \mathbf{m}_{12}$$

- *Remark:* The idea behind (1) may probably be applied in many new contexts. I like the appeal...

Linear SVM

- A different approach to optimum separation. . .
- The idea of *maximum-margin separating hyperplane* was developed by Vladimir Vapnik and others around 1964-65.
- The optimally separating hyperplane is based on *extreme* training observations which are called *support vectors* here.



H_1 does not separate the classes. 
 H_2 does, but only with a small margin.
 H_3 separates them with the maximum margin.

- *Side note:* classical perceptron learning algorithm may as well end up in H_2 -like classifier. (Learning stops when all training examples are classified correctly.)

Linear SVM continued

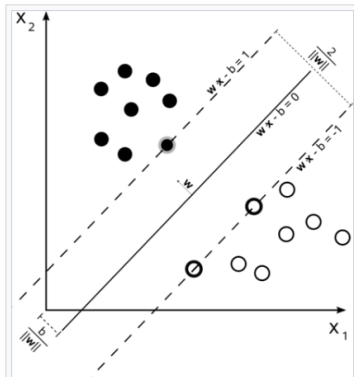
- Let $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{-1, 1\}$, $i = 1, 2, \dots, N$ denote the training data and targets.
- We assume that the training data is linearly separable, so there exists a hyperplane


$$H : \mathbf{w} \cdot \mathbf{x} + b = 0, \quad \|\mathbf{w}\| > 0$$

such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0,$$

$$i = 1, 2, \dots, N$$



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors. 

- By appropriately scaling \mathbf{w} and b we can assume that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N$$

where the equality takes place for at least one observation from both the negative and positive class.

- The margin for hyperplane H is then equal to $2/\|\mathbf{w}\|$. (Recall that the distance from a point \mathbf{x} to H is given by $|\mathbf{w} \cdot \mathbf{x} + b|/\|\mathbf{w}\|$.)
- So the optimal H is found by solving the following constrained minimization problem

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \tag{2}$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, N \tag{3}$$

- However, there are two issues with (2-3).
 - the training set may not be linearly separable in the first place
 - susceptibility to outliers
- So we allow to relax the constraint for individual training vectors and introduce a penalty for this

$$\arg \min_{\xi, \mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_i \xi_i \quad (4)$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, \dots, N \quad (5)$$

- C balances the cost of decreasing $\|\mathbf{w}\|^2$ by introducing a positive ξ_i for some training vectors. (This is kind of l_1 regularization.)

- For now we shall stick with non-regularized version, for simplicity.
- Problem (2-3) is a quadratic programming problem (minimization of a quadratic function subject to constraints involving inequalities with linear functions) for which solvers are available.
- So problem solved. . .
- Kind of. . . but a re-statement is beneficial!
- For an even more general class of *convex problems*:

$$\arg \min_{\mathbf{w}} f(\mathbf{w}) \quad (6)$$

$$g_i(\mathbf{w}) \leq 0, \quad i = 1, 2, \dots, N \quad (7)$$

where all functions f and all g_i are convex, the problem can be converted to another — a *dual problem*.

- We define the Lagrangian for problem (6-7):

$$\mathcal{L}(\mathbf{w}, \alpha) = f(\mathbf{w}) + \sum_i \alpha_i g_i(\mathbf{w}), \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, N$$

- This is like Lagrangian for a problem with equality constraints (part of standard university calculus course), but here with additional constraints on the multipliers α_i .
- Consider the following quantity:

$$\Theta(\mathbf{w}) = \max_{\alpha: \forall_i \alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \alpha)$$

- The point is that (obvious!):

$$\Theta(\mathbf{w}) = \begin{cases} +\infty, & \text{if } \mathbf{w} \text{ violates any of the conditions (7)} \\ f(\mathbf{w}), & \text{otherwise} \end{cases}$$

- So we can replace problem (6-7) by the following:

$$\min_{\mathbf{w}} \Theta(\mathbf{w}) = \min_{\mathbf{w}} \max_{\alpha: \forall_i \alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \alpha) \quad (= \Theta_*)$$

- How about changing the order of “min” and “max”? The following inequality holds generally:

$$(\theta^* =) \quad \max_{\alpha: \forall_i \alpha_i \geq 0} \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \alpha) \leq \min_{\mathbf{w}} \max_{\alpha: \forall_i \alpha_i \geq 0} \mathcal{L}(\mathbf{w}, \alpha) \quad (= \Theta_*)$$

- It's just because it holds *really* generally

$$\max_x \min_y F(x, y) \leq \min_y \max_x F(x, y)$$

for any function of two variables. Easy!

- However the inverse inequality does not hold in this generality (consider $2xy/(x^2 + y^2)$, $0 < x, y < 1$), so the inequality may in fact be strong for some function and variable domains.
- Yet, for the problem (2-3), since minimized function is convex and constraints are affine, we have the equality $\theta^* = \Theta_*$.

- So the problem (2-3) has an associated *dual problem*

$$\arg \max_{\alpha} \theta(\alpha) \quad (8)$$

$$\alpha_i \geq 0, \quad i = 1, 2, \dots, N \quad (9)$$

where

$$\theta(\alpha) = \min_{\mathbf{w}, b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \sum_i \alpha_i [1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)] \right\}$$

- The minimum with respect to \mathbf{w} can be computed by following standard ways. Equating the (\mathbf{w}, b) -derivative to zero and solving for \mathbf{w} and b we get:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad (10)$$

$$\sum_i \alpha_i y_i = 0 \quad (11)$$

- Substituting back we get the following expression for θ :

$$\theta(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j, \quad \sum_i \alpha_i y_i = 0 \quad (12)$$

- and the formulation of the dual problem:

$$\arg \max_{\alpha} \theta(\alpha), \quad \sum_i \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, N \quad (13)$$

- and the decision criterium for unseen \mathbf{x} :

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b \quad (14)$$

$$b = -(1/2) \left(\max_{i: y_i = -1} \mathbf{w} \cdot \mathbf{x}_i + \min_{i: y_i = 1} \mathbf{w} \cdot \mathbf{x}_i \right)$$

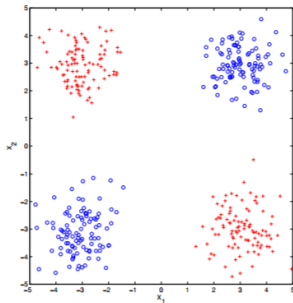
- Since the solution of (13) yields all α_i zero except for a small number of support vectors, the sum and min/max in (14) are restricted to only a small number of indexes, say $i \in S$.

Why bother about duality?

- But finally, what's the real gain in moving from primary to dual problem in SVM?
- There is a specialized and very effective algorithm for solving this particular QP problem: the sequential minimal optimization (SMO) algorithm. (Not part of this presentation.)
- But I also like an answer I saw somewhere in the Web.
 - short one: *kernels!*
 - longer: *keernerneeeels!*
- The nice property of the dual problem and decision criterium is that the training data appears exclusively in the form of scalar products $\mathbf{x}_i \cdot \mathbf{x}_j$.
- This allows for easy replacement of raw observations by *features*. . . Possibly very high-dimensional ones.
- And features are connected to kernels. . .

Nonlinear classification problems

- Some classification problems can't be solved with linear methods — notably the XOR problem.
- Regardless that the training data is not linearly separable in the original *observation space* this is “easily” overcome by introducing an extra *feature* $\phi(\mathbf{x}) = x_1x_2$.



- When mapped to a higher dimensional feature space with

$$(x_1, x_2) \mapsto (X_1, X_2, X_3) = \Phi(x_1, x_2) = (x_1, x_2, x_1x_2)$$

the data is separable with $X_3 = 0$ hyperplane (or some other which may be even more optimal for the data).

- The resulting classification boundary in the observation space is nonlinear.

- Assume that we have feature set $\mathbb{R}^d \ni \mathbf{x} \mapsto \Phi(\mathbf{x}) \in \mathbb{R}^D$.
- There may be a lot of features (D much bigger than d), but they are great in separating training data!
- To run SVM with features rather than original observations, it suffices to replace \mathbf{x} 's by $\Phi(\mathbf{x})$'s in (12), (13) and (14).
- Note that we now need no structure in the observation space — no geometry, distance, scalar products, hyperplanes, . . . So it can be just a plain set \mathcal{X} instead of \mathbb{R}^d . Only the feature space counts now and the performance of features.
- Two *little* problems. . .
- A lot of features result in more workload, like $O(D^2)$.
- How to develop good feature set?
- Interestingly both of them can be addressed with *kernels*. . .

- With a feature map $\Phi : \mathcal{X} \rightarrow \mathbb{R}^D$ we can define a 2-variable function:

$$K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}, \quad K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}') \quad (15)$$

- Obviously K is symmetric:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} \quad K(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}', \mathbf{x}) \quad (16)$$

- And the following property is quite easy to verify:

$$\forall_n \forall_{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n \in \mathcal{X}} \forall_{\alpha_1, \alpha_2, \dots, \alpha_n \in \mathbb{R}} \sum_{i,j=1}^n \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0 \quad (17)$$

- (The sum is equal to $\mathbf{X} \cdot \mathbf{X}$, where $\mathbf{X} = \sum_i \alpha_i \Phi(\mathbf{x}_i)$.)
- We call K a (symmetric, positive-definite) *kernel* on \mathcal{X} .

- Assume K is a kernel on \mathcal{X} , so that (16-17) hold. Then, with some additional more theoretical assumptions, there exists a mapping $\Phi : \mathcal{X} \rightarrow H$, where H is a space with a scalar product, such that

$$K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}')$$

- The little catch here is that the space H may be infinite dimensional and \cdot is the scalar product in H — a kernel is generally associated with infinitely many features.
- So H is a *Hilbert* space... But do we care?
- No! Because we don't have to see features and compute the scalar products explicitly — the kernel computes them for us!
- About the “additional assumptions”... These may be: let \mathcal{X} be a compact topological space with a strictly positive finite Borel measure and K be continuous. Or just $X = [a, b]$ and K continuous. For the curious: see *Mercer's theorem*.

$$\theta(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{ij} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j), \quad \sum_i \alpha_i y_i = 0$$

$$\arg \max_{\alpha} \theta(\alpha), \quad \sum_i \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, N$$

$$f(\mathbf{x}) = \sum_{i \in S} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

$$b = -(1/2) \left(\max_{i \in S: y_i = -1} f_0(\mathbf{x}_i) + \min_{i \in S: y_i = 1} f_0(\mathbf{x}_i) \right),$$

$$f_0(\mathbf{x}_i) = \sum_{j \in S} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i)$$

- Note that $K(\mathbf{x}_i, \mathbf{x}_j) = K_{ij}$ is just a matrix (symmetrical and positive-semidefinite).

- Kernel SVM was developed by Vapnik and co-workers in 1992-95:
 - Initial paper introducing the algorithm: Boser, Guyon and Vapnik, 1992
 - Matched performance on MNIST with CNN at the time: Cortes and Vapnik, 1995
- Results on MNIST in 1995:
 - Test error for SVM: 1.1%. LeNet-1: 1.7, LeNet-4: 1.1.
 - However LeNet-4 was carefully handcrafted based on the errors made by a long standing state-of-the-art LeNet-1 (since 1989)
 - SVM was not engineered in a similar fashion, citation from LeCun et. al., 1998:

The SVM has excellent accuracy, which is most remarkable, because unlike the other high performance classifiers it does not include a priori knowledge about the problem. In fact, this classifier would do just as well if the image pixels were permuted with a fixed mapping and lost their pictorial structure.

- Interestingly SVM (and kernel methods in general) became the mainstream machine learning algorithm in late 1990's early 2000's and the interest in ANN's decreased considerably.
- Things changed again around 2010.
- Maybe it's good to remember this?
- A little curiosity from speech domain, not so old: Huang, Avron, Sainath, Sindhvani, Ramabhadran, *Kernel methods match deep neural networks on TIMIT*, 2014.
(Most of the authors from IBM T.J. Watson Research Center. It's not about SVM but *randomized approximate feature maps* generated with kernels... But not today!)

What kernel?

- The intuition is that the kernel is a *similarity measure* in the observation space. This similarity measure implies features.
- Anyway, the choice of the kernel is the tricky part! Apart from regularization weight. . . Some example kernels:

- Polynomial kernel with degree p :

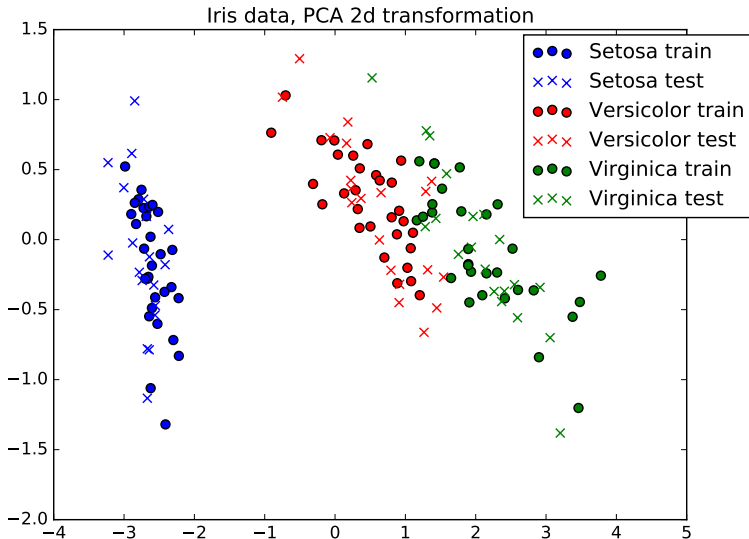
$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^p, \quad c \geq 0, \quad p \in \mathbb{N}$$

- Radial basis function (or gaussian) kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2), \quad \sigma > 0$$

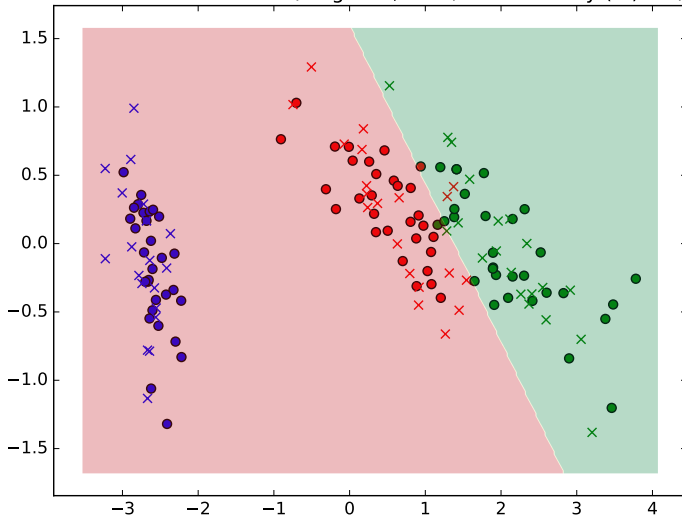
- And of course many more, also of a different taste: string kernels for text classification, graph kernels, Fisher information kernels, locality improved kernels, . . . One can also construct kernels from other kernels.
- Some try to learn the kernel from the data. . .

Example 3: Iris data, PCA transformed



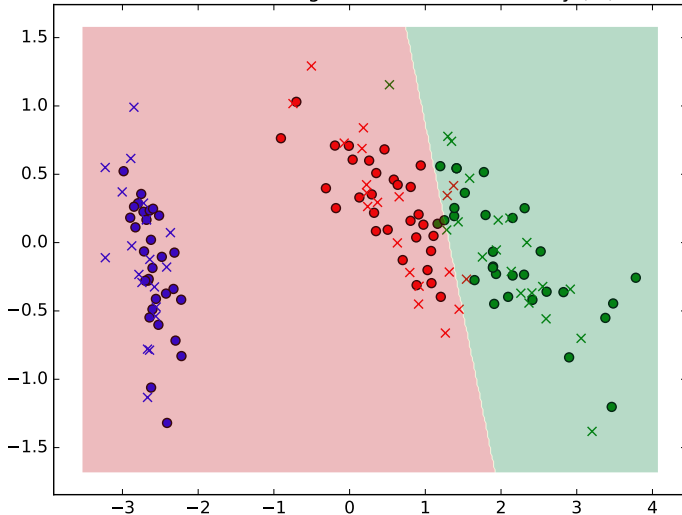
Example 3: Versicolor vs Virginica, Fisher classifier

Fisher classifier Versicolor/Virginica, train/test accuracy (%): 97/93



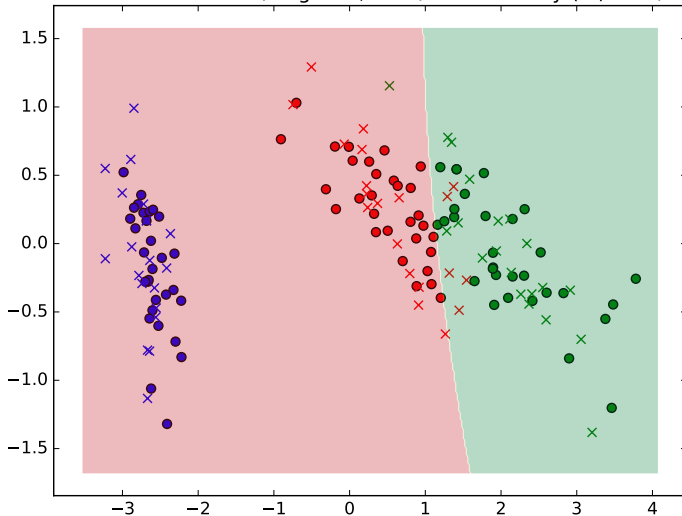
Example 3: Versicolor vs Virginica, linear SVM

Linear SVC Versicolor/Virginica, train/test accuracy (%): 98/93



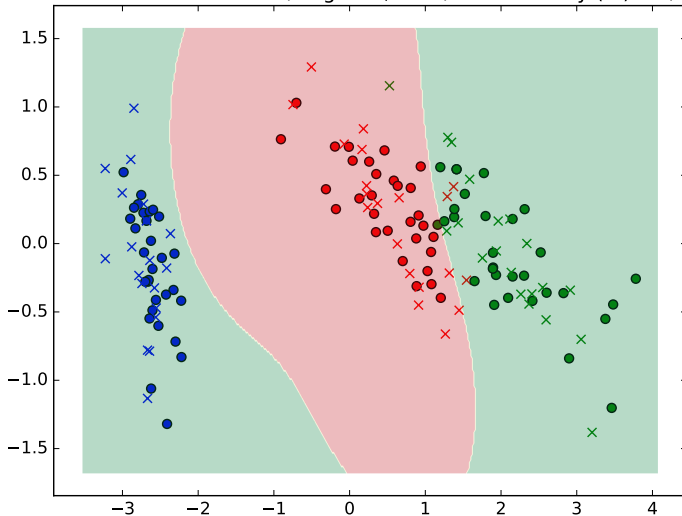
Example 3: Versicolor vs Virginica, cubic SVM

Cubic SVC Versicolor/Virginica, train/test accuracy (%): 100/85

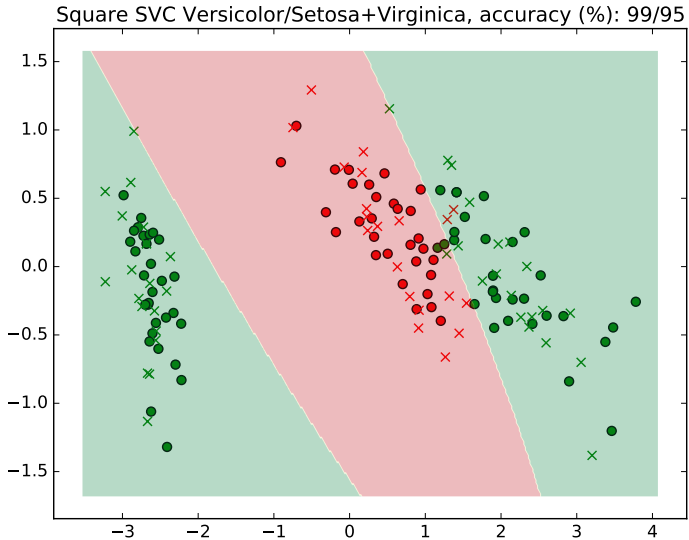


Example 3: Versicolor vs Virginica, gaussian SVM

Gaussian SVC Versicolor/Virginica, train/test accuracy (%): 98/93



Example 3: Versicolor vs Setosa+Virginica, square SVM



Kernel methods: high level view

- The basic idea is:
 - Choose a kernel $K(\mathbf{x}, \mathbf{x}')$ and thus the implicit feature map $\Phi(\mathbf{x})$. No need to see/derive any explicit formula for Φ !
 - Solve the underlying *linear* problem in the feature space. As long as the solution involves only scalar products of features of training data, the kernel K will suffice.
 - Map the solution back to original observation space — done!
- Many examples: kernel SVM, kernel PCA, kernel DA (kernelized LDA), ...
- Consider PCA. The linear classic works like this: assume that we have zero centered training data $(1/N) \sum_n \mathbf{x}_n = 0$ we compute the covariance matrix $\mathbb{C} = (1/N) \sum_n \mathbf{x}_n \mathbf{x}_n^T$, which is symmetric. Then we solve the eigenvalue problem $\mathbb{C}\mathbf{x} = \lambda\mathbf{x}$. Eigenvectors of largest eigenvalues indicate principal directions.
- Let's take a look at the kernelized version... (Some details are skipped. For example: how to ensure zero centering in the feature space? See Schölkopf et.al 1998)

Kernel PCA, the algorithm

- For the kernel version the underlying feature space problem is

$$\mathbb{C}\mathbf{V} = \lambda\mathbf{V}, \quad \mathbb{C} = \frac{1}{N} \sum_n \mathbf{v}_n \mathbf{v}_n^T, \quad \mathbf{v}_n = \Phi(\mathbf{x}_n)$$

- Since $\mathbb{C}\mathbf{V} = (1/N) \sum_n (\mathbf{v}_n \cdot \mathbf{V}) \mathbf{v}_n$ the eigenvectors are spanned by \mathbf{v}_n so it's sufficient to compute the coefficients $\mathbf{V} = \sum_n a_n \mathbf{v}_n$. We rewrite the eigenvalue problem like so:

$$\mathbf{v}_m \cdot \mathbb{C}\mathbf{V} = \frac{1}{N} \sum_n (\mathbf{v}_n \cdot \mathbf{v}_m) (\mathbf{v}_n \cdot \mathbf{V}) = \frac{1}{N} \sum_n \mathbb{K}_{nm} a_n$$

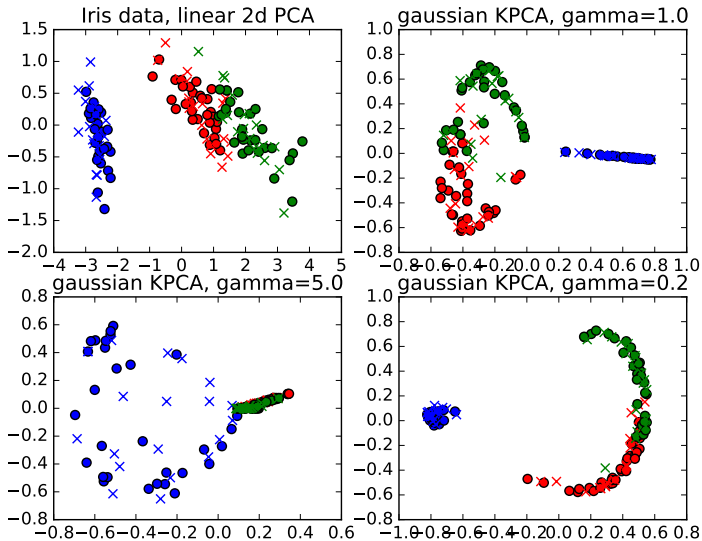
where $\mathbb{K}_{mn} = \mathbf{v}_m \cdot (\mathbf{V})_n = K(\mathbf{x}_m, \mathbf{x}_n)$ and since the right hand side is then λa_m we get $(1/N) \sum_n \mathbb{K}_{nm} a_n = \lambda a_m$. Finally (\mathbb{K} is symmetric):

$$\mathbb{K}\mathbf{a} = N\lambda\mathbf{a}$$

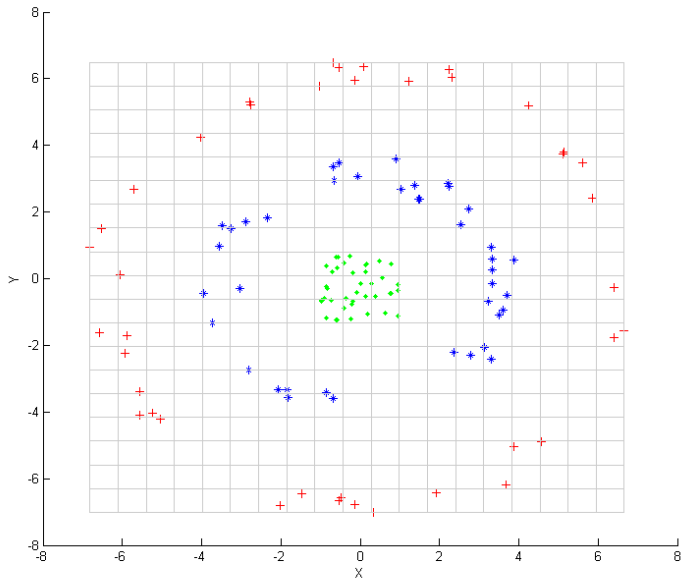
- Projection onto eigenvector (corresponding to) \mathbf{a} is given by

$$\left(\sum_n a_n \mathbf{v}_n \right) \cdot \mathbf{V} = \sum_n a_n K(\mathbf{x}_n, \mathbf{x})$$

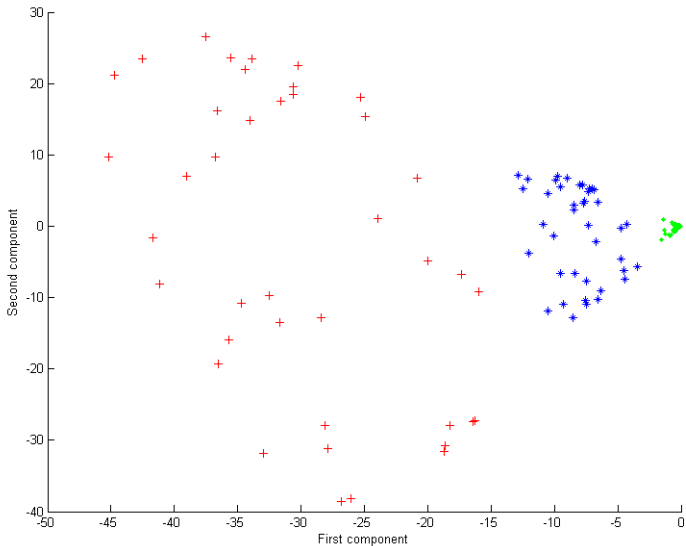
Example 4: Iris data, kernel PCA



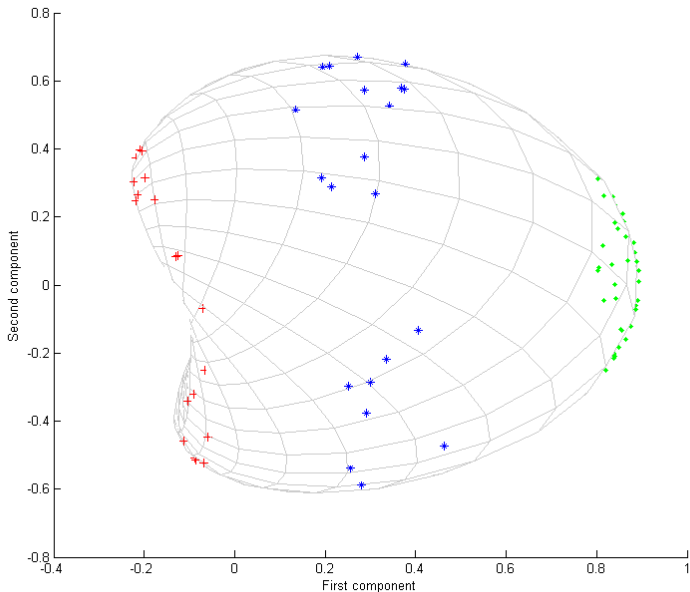
Example 5: KPCA example from Wikipedia



Example 5: KPCA with square polynomial kernel



Example 5: KPCA with gaussian kernel



<https://commons.wikimedia.org/w/index.php?curid=42169073>
by Flickr commons <https://www.flickr.com/photos/internetarchivebookimages/20150531109/>, CC BY 2.0

<https://commons.wikimedia.org/w/index.php?curid=22877598>
by User:ZackWeinberg, based on PNG version by User:Cyc, CC BY-SA 3.0

<https://commons.wikimedia.org/w/index.php?curid=3566688>
by Cyc — Own work, Public Domain

<https://commons.wikimedia.org/w/index.php?curid=3936385>
by Petter Strandmark — Own work, CC BY 3.0

<https://commons.wikimedia.org/w/index.php?curid=3936390>
by Petter Strandmark — Own work, CC BY 3.0

<https://commons.wikimedia.org/w/index.php?curid=3936753>
by Petter Strandmark — Own work, CC BY 3.0