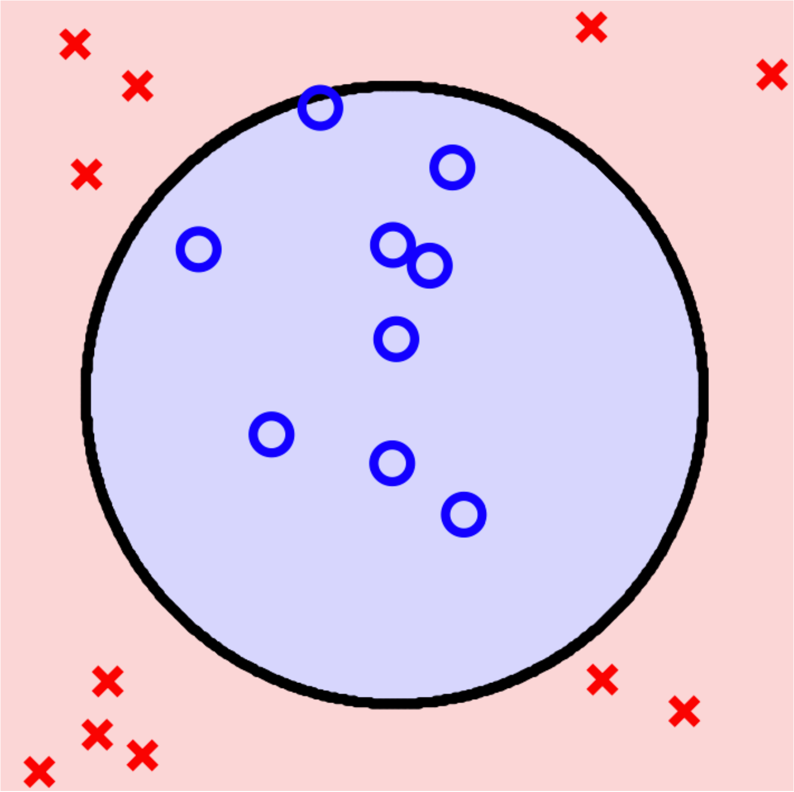
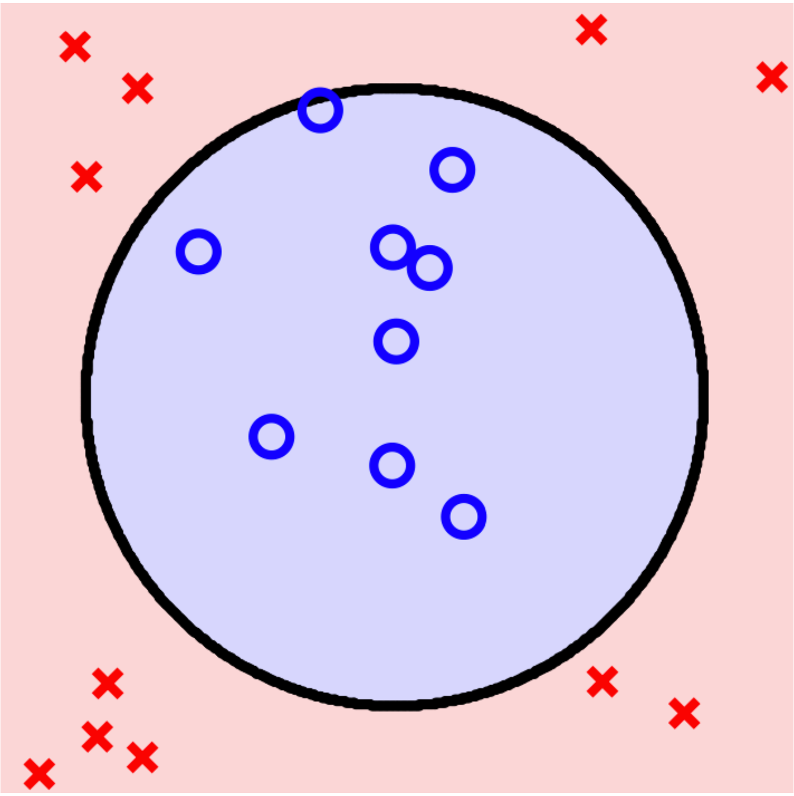
Lecture 7
Kernel Method
(Going Beyond Linear Hypothesis Efficiently)

A simple way to go beyond linear hypothesis space: feature mapping



Not linearly separable in 2 dimensions.
Using logistic regression will also give large error.

A simple way to go beyond linear hypothesis space: feature mapping



Not linearly separable in 2 dimensions.
Using logistic regression will also give large error.

Solution: go to higher dimension via *a feature map!*

$$\Phi(x_1, x_2) = \left(x_1, x_2, x_1^2 + x_2^2\right)$$

This clearly will be linearly separable in 3 dimensions!

A simple way to go beyond linear hypothesis space: feature mapping

We may simply extend beyond a linear hypothesis space with a feature map

$$\Phi : X \to F$$

where typically $|X| = d \ll |F| = p$.

A simple way to go beyond linear hypothesis space: feature mapping

We may simply extend beyond a linear hypothesis space with a feature map

$$\Phi : X \to F$$

where typically $|X| = d \ll |F| = p$.

And we may perform regression based on a much larger, but not arbitrarily large, hypothesis space (linear combination of feature vectors, instead of the original features)

$$\mathscr{H} = \{ f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x}) \} .$$

A simple way to go beyond linear hypothesis space: feature mapping

We may simply extend beyond a linear hypothesis space with a feature map

$$\Phi : X \to F$$

where typically $|X| = d \ll |F| = p$.

And we may perform regression based on a much larger, but not arbitrarily large, hypothesis space (linear combination of feature vectors, instead of the original features)

$$\mathscr{H} = \{f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x})\} .$$

An illustrative simple example could be $\mathbf{x} = \left(x_1, x_2\right) \mapsto \Phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$,

so the hypothesis space is no longer linear, but a polynomial of degree 2.
This is particularly useful, because points that are not classified by a linear model might be easily classified by a linear model in a *feature space*.

A simple way to go beyond linear hypothesis space: feature mapping

We may simply extend beyond a linear hypothesis space with a feature map

$$\Phi : X \to F$$

where typically $|X| = d \ll |F| = p$.

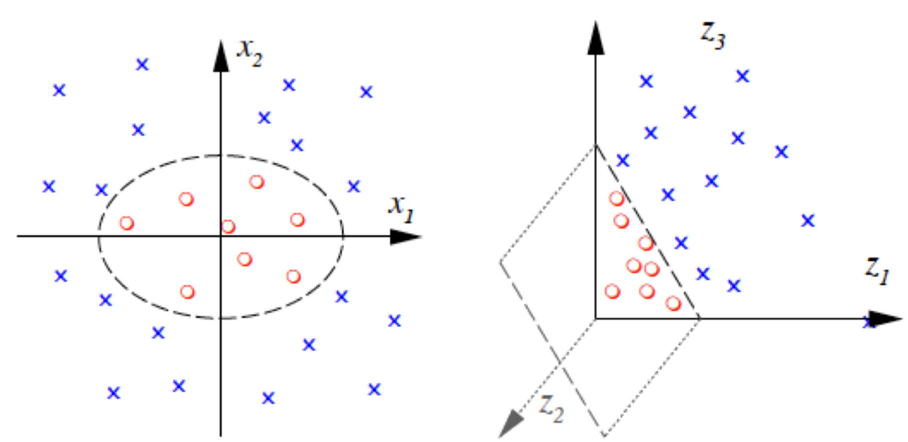
And we may perform regression based on a much larger, but not arbitrarily large, hypothesis space (linear combination of feature vectors, instead of the original features)

$$\mathscr{H} = \{f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x})\} \, .$$

An illustrative simple example could be $\mathbf{x} = \left(x_1, x_2\right) \mapsto \Phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$,

so the hypothesis space is no longer linear, but a polynomial of degree 2.
This is particularly useful, because points that are not classified by a linear model might be easily classified by a linear model in a *feature space*.

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



A simple way to go beyond linear hypothesis space: feature mapping

We may simply extend beyond a linear hypothesis space with a feature map

$$\Phi : X \to F$$

where typically $|X| = d \ll |F| = p$.

And we may perform regression based on a much larger, but not arbitrarily large, hypothesis space (linear combination of feature vectors, instead of the original features)

$$\mathscr{H} = \{f_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \Phi(\mathbf{x})\} .$$

An illustrative simple example could be $\mathbf{x} = \left(x_1, x_2\right) \mapsto \Phi(\mathbf{x}) = \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$,

so the hypothesis space is no longer linear, but a polynomial of degree 2.
This is particularly useful, because points that are not classified by a linear model might be easily classified by a linear model in a *feature space*.

Importantly, it's not difficult to see that solving Tikhonov regularization problem

$$\min_{\mathbf{w} \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \ell \left(y_i, f_{\mathbf{w}}\left(\mathbf{x}_i\right)\right) + \lambda \|\mathbf{w}\|^2$$

is essentially the same, up to replacing $\mathbf{x} \in \mathbb{R}^D$ by the *feature vector* $\Phi(\mathbf{x}) \in \mathbb{R}^p$, as what we have done earlier in this class!

A simple way to go beyond linear hypothesis space: feature mapping

Consider

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \qquad \text{and} \qquad \phi(\mathbf{x}) = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_d \\ x_1 x_2 \\ \vdots \\ x_{d-1} x_d \\ \vdots \\ x_1 x_2 \dots x_d \end{pmatrix}$$

This feature map is *very expressive*, and allows complicated *non-linear decision boundaries*; however, its dimension is $p = 2^d$ and hence is *prohibitively unbearable* (computationally)!

The Kernel Trick

The kernel trick is a way to get around this dilemma by learning a function in a much higher dimensional space, without ever computing a single vector $\Phi(\mathbf{x})$ or $\mathbf{w}$. The magic sauce behind this is the *representer theorem*, which we will not prove here but provides a note for you to read.

The Kernel Trick

The kernel trick is a way to get around this dilemma by learning a function in a much higher dimensional space, without ever computing a single vector $\Phi(\mathbf{x})$ or $\mathbf{w}$. The magic sauce behind this is the *representer theorem*, which we will not prove here but provides a note for you to read.

The representer theorem states that the solution to the Tikhonov regularization problem

$$\min_{\mathbf{w}\in\mathbb{R}^D} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, f_{\mathbf{w}}\left(\mathbf{x}_i\right)\right) + \lambda\|\mathbf{w}\|^2$$

can always be written as a linear combination of input features as

$$\boxed{\hat{\mathbf{w}}^T = \sum_{i=1}^{n} \Phi(\mathbf{x}_i)^T w_i}$$

where w_i is a scalar coefficient.

The Kernel Trick

The kernel trick is a way to get around this dilemma by learning a function in a much higher dimensional space, without ever computing a single vector $\Phi(\mathbf{x})$ or $\mathbf{w}$. The magic sauce behind this is the *representer theorem*, which we will not prove here but provides a note for you to read.

The representer theorem states that the solution to the Tikhonov regularization problem

$$\min_{\mathbf{w}\in\mathbb{R}^D} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, f_{\mathbf{w}}\left(\mathbf{x}_i\right)\right) + \lambda\|\mathbf{w}\|^2$$

can always be written as a linear combination of input features as

$$\boxed{\hat{\mathbf{w}}^T = \sum_{i=1}^{n} \Phi(\mathbf{x}_i)^T w_i}$$

where w_i is a scalar coefficient.

This implies that the empirically optimal hypothesis function can be written as an inner-product

$$\underbrace{\hat{f}_{\hat{\mathbf{w}}}(\mathbf{x})}_{}\overset{O(p^2)!!}{} = \hat{\mathbf{w}}^T\Phi(\mathbf{x}) = \sum_{i=1}^{n} w_i \Phi(\mathbf{x}_i)^T\Phi(\mathbf{x})$$

The solution depends on the input feature only via the inner product!

The Kernel Trick

The kernel trick is a way to get around this dilemma by learning a function in a much higher dimensional space, without ever computing a single vector $\Phi(\mathbf{x})$ or $\mathbf{w}$. The magic sauce behind this is the *representer theorem*, which we will not prove here but provides a note for you to read.

The representer theorem states that the solution to the Tikhonov regularization problem

$$\min_{\mathbf{w}\in\mathbb{R}^D} \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, f_{\mathbf{w}}\left(\mathbf{x}_i\right)\right) + \lambda\|\mathbf{w}\|^2$$

can always be written as a linear combination of input features as

$$\boxed{\hat{\mathbf{w}}^T = \sum_{i=1}^{n} \Phi(\mathbf{x}_i)^T w_i}$$

where w_i is a scalar coefficient.

This implies that the empirically optimal hypothesis function can be written as an inner-product

$O(p^2)!!$

$$\hat{f}_{\hat{\mathbf{w}}}(\mathbf{x}) = \hat{\mathbf{w}}^T\Phi(\mathbf{x}) = \sum_{i=1}^{n} w_i\Phi(\mathbf{x}_i)^T\Phi(\mathbf{x})$$

The solution depends on the input feature only via the inner product!

$$= \sum_{i=1}^{n} w_i \underbrace{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x})\rangle}_{\equiv K(\mathbf{x}_i,\mathbf{x})}$$

$O(n^2)$

Kernel Trick

Replace an inner product with a more general function, with inner product-like property (hence the name *kernel*)

save computational resource!

General Kernels

So Kernel method converts risk minimisation problem in high-dimensional feature space to learning the linear combination

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} w_i K(\mathbf{x}_i, \mathbf{x})$$

where a kernel function $K(\mathbf{x}, \mathbf{x}')$ must behave like an inner-product, that is $K(\mathbf{x}, \mathbf{x}')$ must be symmetric, and positive semi-definite.

Examples

Linear Kernel

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

Similar to a good old linear classifier,
but can be faster if the dimension of data
is large compared to the number of data.

General Kernels

So Kernel method converts risk minimisation problem in high-dimensional feature space to learning the linear combination

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} w_i K(\mathbf{x}_i, \mathbf{x})$$

where a kernel function $K(\mathbf{x}, \mathbf{x}')$ must behave like an inner-product, that is $K(\mathbf{x}, \mathbf{x}')$ must be symmetric, and positive semi-definite.

Examples

Linear Kernel

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

Similar to a good old linear classifier, but can be faster if the dimension of data is large compared to the number of data.

Gaussian Kernel
Radial Basis Function (RBF)

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\| \mathbf{x} - \mathbf{x}' \|^2}{2\sigma^2}\right)$$

The most popular kernel

- It is a universal approximator. (hence the name basis function)

General Kernels

So Kernel method converts risk minimisation problem in high-dimensional feature space to learning the linear combination

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} w_i K(\mathbf{x}_i, \mathbf{x})$$

where a kernel function $K(\mathbf{x}, \mathbf{x}')$ must behave like an inner-product, that is $K(\mathbf{x}, \mathbf{x}')$ must be symmetric, and positive semi-definite.

Examples

Linear Kernel

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

Similar to a good old linear classifier, but can be faster if the dimension of data is large compared to the number of data.

Gaussian Kernel
Radial Basis Function (RBF)

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\| \mathbf{x} - \mathbf{x}' \|^2}{2\sigma^2}\right)$$

The most popular kernel

- It is a universal approximator. (hence the name basis function)

Note that for RBF the feature space of the kernel has an infinite number of dimensions; for $\sigma = 1$, for example, it can be expanded as

$$\exp\left(-\frac{1}{2} \| \mathbf{x} - \mathbf{x}' \|^2\right) = \sum_{j=0}^{\infty} \frac{(\mathbf{x}^\top \mathbf{x}')^j}{j!} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \exp\left(-\frac{1}{2} \| \mathbf{x}' \|^2\right)$$

$$= \sum_{j=0}^{\infty} \sum_{\sum n_i = j} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \frac{x_1^{n_1} \cdots x_k^{n_k}}{\sqrt{n_1! \cdots n_k!}} \exp\left(-\frac{1}{2} \| \mathbf{x}' \|^2\right) \frac{x_1'^{n_1} \cdots x'^{n_k}_k}{\sqrt{n_1! \cdots n_k!}}$$

Kernel Machines and How to Train Them

$$\hat{\mathbf{w}} = \min_{\mathbf{w} \in \mathbb{R}^D} \sum_{i=1}^{n} \left(y_i - f_{\mathbf{w}}\left(\mathbf{x}_i\right) \right)^2 + \lambda \|\mathbf{w}\|^2$$

Kernel Machines and How to Train Them

$$\hat{\mathbf{w}} = \min_{\mathbf{w}\in\mathbb{R}^D} \sum_{i=1}^{n} \left(y_i - f_{\mathbf{w}}\left(\mathbf{x}_i\right)\right)^2 + \lambda\|\mathbf{w}\|^2$$

$$= \min_{\mathbf{w}\in\mathbb{R}^D} \sum_{i,j=1}^{n} \left(y_j - w_i K(\mathbf{x}_i, \mathbf{x}_j)\right)^2 + \lambda\|\mathbf{w}\|^2$$

$\mathbf{K}$ matrix of size n^2

ridge regression

$$\hat{\mathbf{w}} = \left(X^T X + \lambda I\right)^{-1} X^T \mathbf{y}$$

Kernel Machines and How to Train Them

$$\hat{\mathbf{w}} = \min_{\mathbf{w}\in\mathbb{R}^D} \sum_{i=1}^{n} \left(y_i - f_{\mathbf{w}}\left(\mathbf{x}_i\right) \right)^2 + \lambda \|\mathbf{w}\|^2$$

$$= \min_{\mathbf{w}\in\mathbb{R}^D} \sum_{i,j=1}^{n} \left(y_j - w_i \boxed{K(\mathbf{x}_i, \mathbf{x}_j)} \right)^2 + \lambda \|\mathbf{w}\|^2$$

$\mathbf{K}$ matrix of size n^2

ridge regression

$$\hat{\mathbf{w}} = \left(X^T X + \lambda I\right)^{-1} X^T \mathbf{y}$$

kernel ridge regression

$$\hat{\mathbf{w}} = (\mathbf{K} + \lambda I)^{-1} \mathbf{y}$$

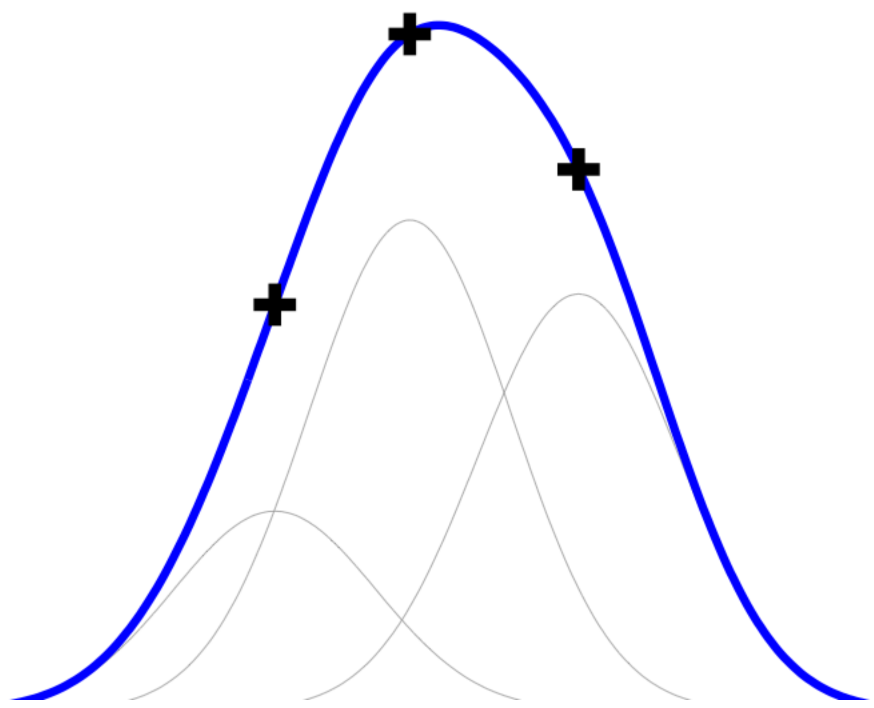
Example usage of Kernel method: RBFs for Regression (Curve-fitting)

$$h(\mathbf{x}) = \sum_{i=1}^{n} w_i \exp\left(- \parallel \mathbf{x} - \mathbf{x}_i \parallel ^2 / 2\sigma^2\right)$$

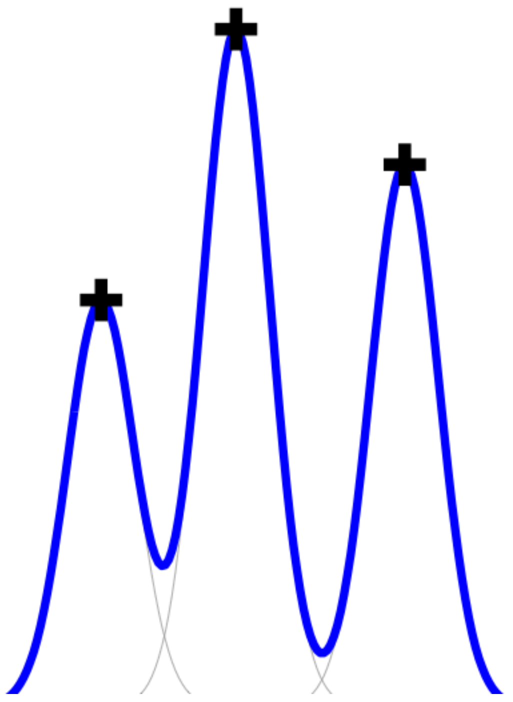
Example usage of Kernel method: RBFs for Regression (Curve-fitting)

$$h(\mathbf{x}) = \sum_{i=1}^{n} w_i \exp\left(- \| \mathbf{x} - \mathbf{x}_i \|^2 / 2\sigma^2\right)$$

$n = 3$



large σ

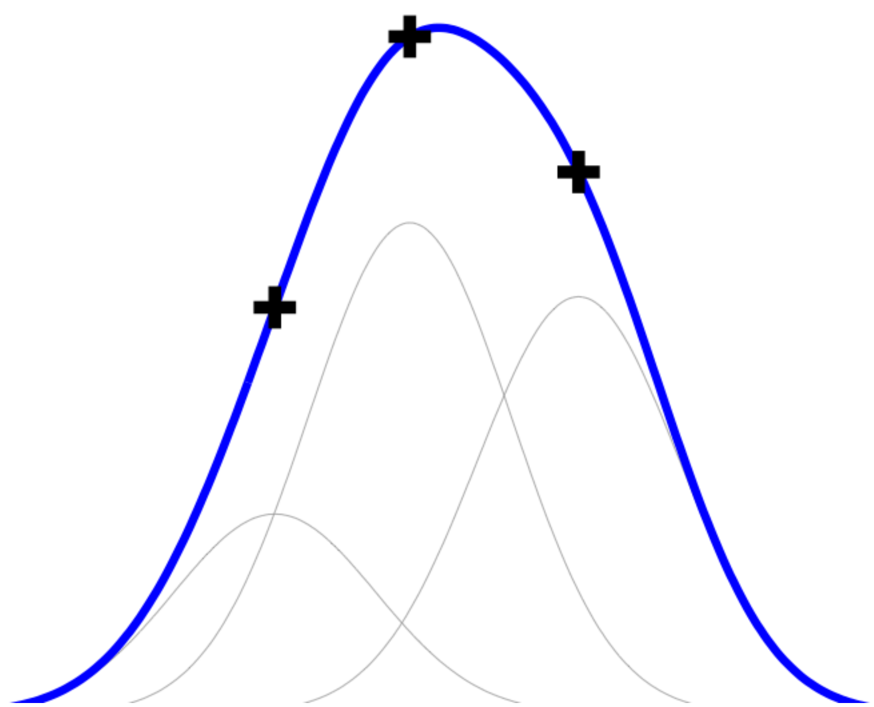


small σ

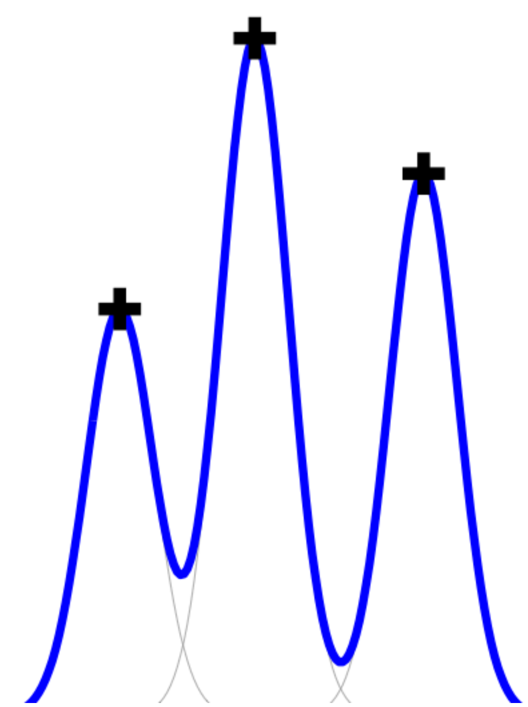
Example usage of Kernel method: RBFs for Regression (Curve-fitting)

$$h(\mathbf{x}) = \sum_{i=1}^{n} w_i \exp\left(- \parallel \mathbf{x} - \mathbf{x}_i \parallel ^2 / 2\sigma^2\right)$$

$n = 3$



large σ

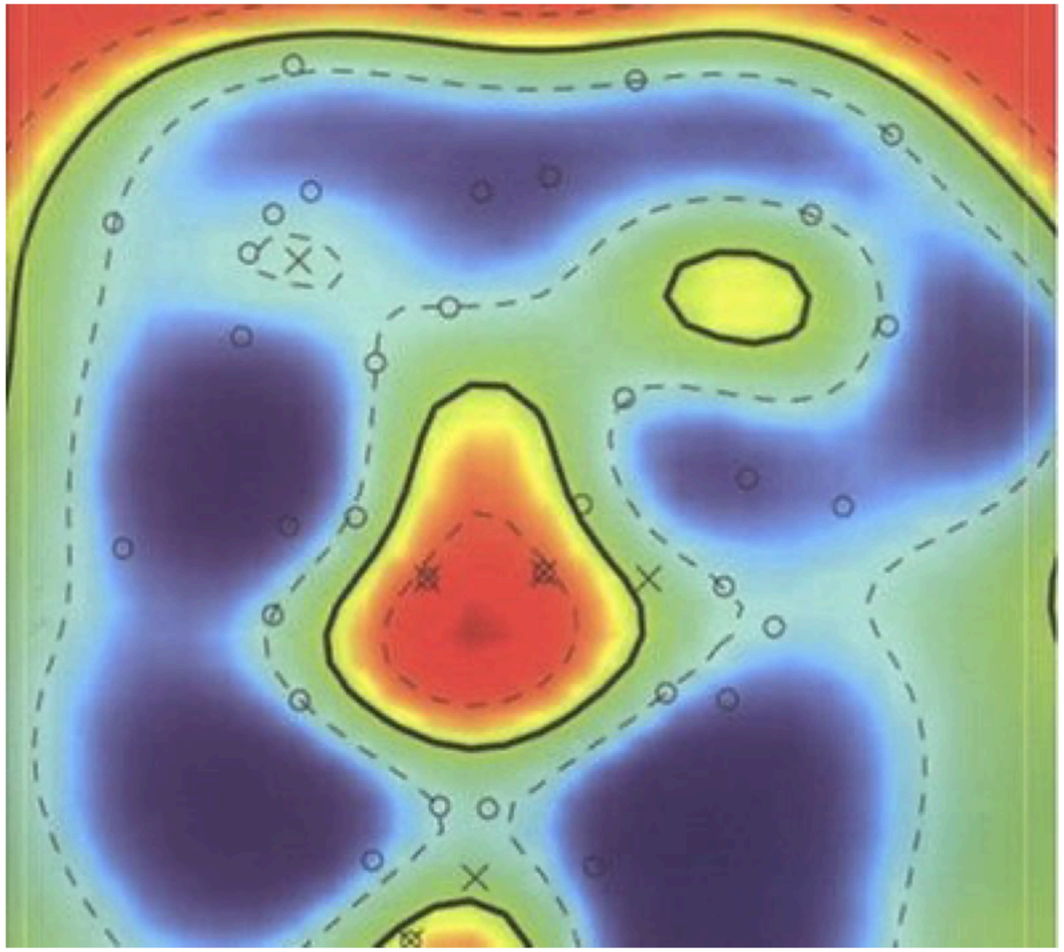


small σ

more expressive, but can overfit

Example usage of Kernel method: RBFs for Classification

$$h(\mathbf{x}) = \text{sign}\left[\sum_{i=1}^{n} w_i \exp\left(- \| \mathbf{x} - \mathbf{x}_i \|^2 / 2\sigma^2\right)\right]$$



Bernhard Schölkopf, Christopher J.C. Burges, and Alexander J. Smola, ADVANCES IN KERNEL METHODS: SUPPORT VECTOR LEARNING, published by the MIT Press. Used with permission.