Statistical Machine Learning: Neural Networks and Kernel Methods

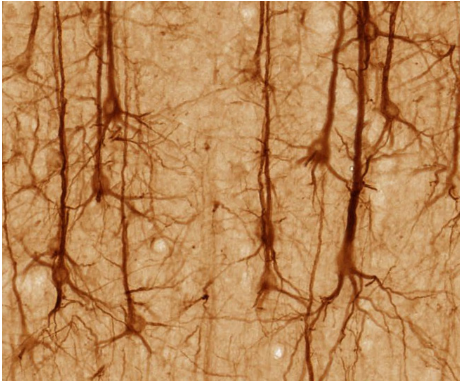
Dino Sejdinovic
Department of Statistics
University of Oxford

22-24 June 2015, Novi Sad
slides available at:
http://www.stats.ox.ac.uk/~sejdinov/talks.html

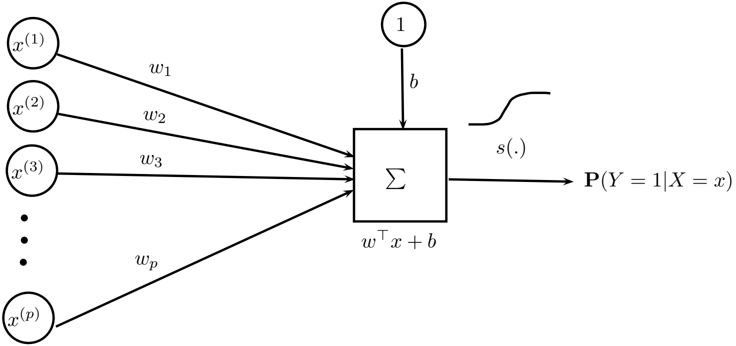
Neural Networks

Biological inspiration

- Basic computational elements: neurons.
- Receives signals from other neurons via dendrites.
- Sends processed signals via axons.
- Axon-dendrite interactions at synapses.
- $10^{10} - 10^{11}$ neurons.
- $10^{14} - 10^{15}$ synapses.

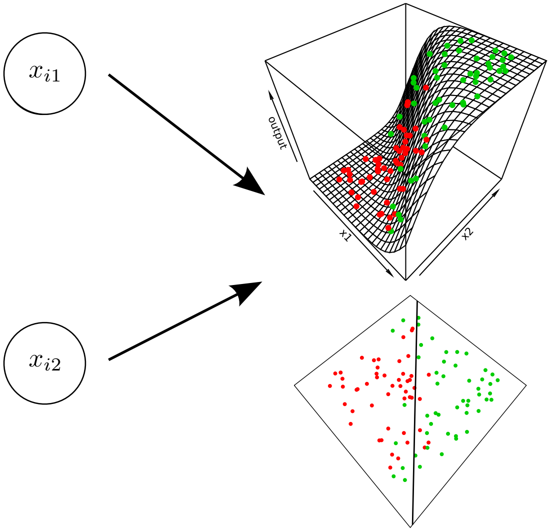


Single Neuron Classifier

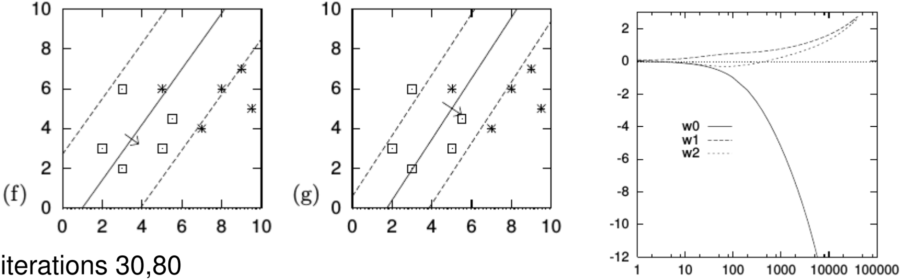


- **activation** $w^\top x + b$ (linear in **inputs** x)
- **activation/transfer function** s gives the **output/activity** (potentially nonlinear in x)
- common nonlinear activation function $s(a) = \frac{1}{1+e^{-a}}$: **logistic regression**
- learn w and b via gradient descent

Single Neuron Classifier



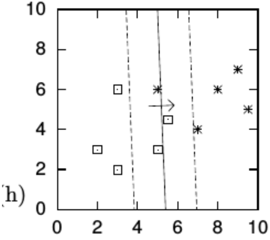
Overfitting

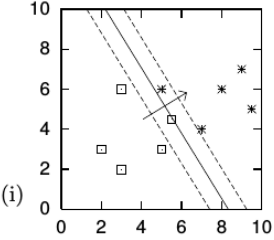


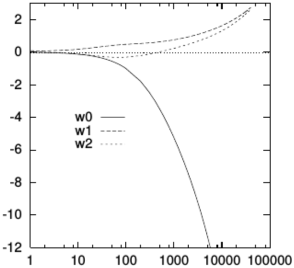
iterations 30,80

Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

Overfitting





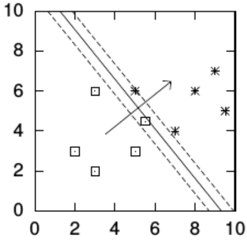


iterations 500,3000

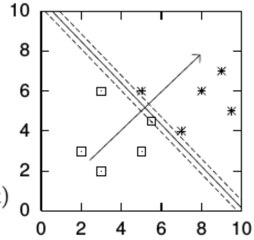
Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

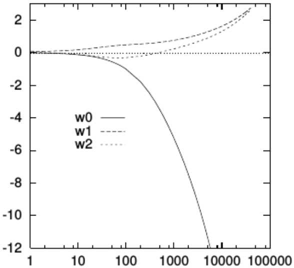
Overfitting

(j)



(k)





iterations 10000,40000

prevent overfitting by:

- **early stopping**: just halt the gradient descent
- regularization: L_2-regularization called **weight decay** in neural networks literature.

Figures from D. MacKay, **Information Theory, Inference and Learning Algorithms**

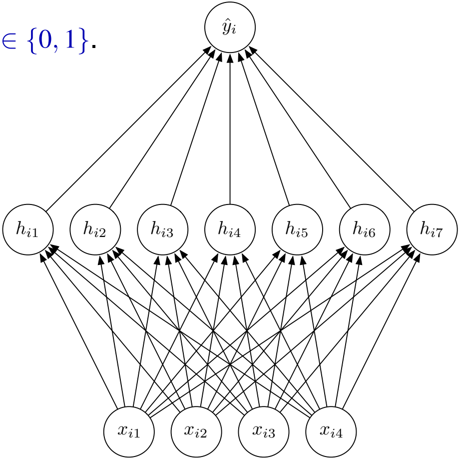
Multilayer Networks

- Data vectors $x_i \in \mathbb{R}^p$, binary labels $y_i \in \{0, 1\}$.
- **inputs** $x_{i1}, \ldots, x_{ip}$
- **output** $\hat{y}_i = \mathbb{P}(Y = 1|X = x_i)$
- **hidden unit activities** $h_{i1}, \ldots, h_{im}$
  - Compute **hidden unit activities**:

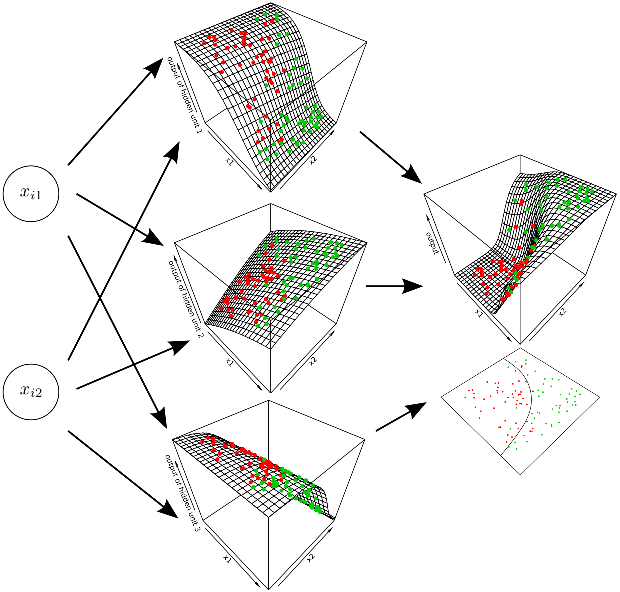
$$h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij}\right)$$

  - Compute **output probability**:

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_k^o h_{il}\right)$$



Multilayer Networks



Training a Neural Network

- Objective function: L_2-regularized log-loss

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) + \frac{\lambda}{2} \left(\sum_{jl} (w_{jl}^h)^2 + \sum_{l} (w_l^o)^2 \right)$$

where

$$\hat{y}_i = s\left(b^o + \sum_{l=1}^{m} w_l^o h_{il}\right) \qquad h_{il} = s\left(b_l^h + \sum_{j=1}^{p} w_{jl}^h x_{ij}\right)$$

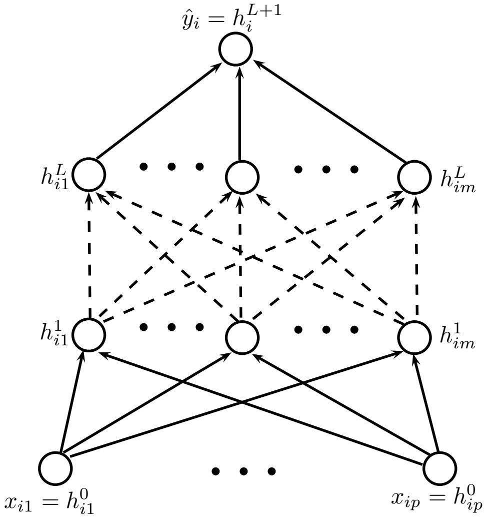
- Optimize parameters $\theta = \{b^h, w^h, b^o, w^o\}$, where $b^h \in \mathbb{R}^m$, $w^h \in \mathbb{R}^{p \times m}$, $b^o \in \mathbb{R}$, $w^o \in \mathbb{R}^m$ with gradient descent.

$$\frac{\partial J}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial w_l^o} = \lambda w_l^o + \sum_{i=1}^{n} (\hat{y}_i - y_i) h_{il},$$

$$\frac{\partial J}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial h_{il}} \frac{\partial h_{il}}{\partial w_{jl}^h} = \lambda w_{jl}^h + \sum_{i=1}^{n} (\hat{y}_i - y_i) w_l^o h_{il} (1 - h_{il}) x_{ij}.$$

- L_2-regularization often called **weight decay**.
- Multiple hidden layers: **Backpropagation** algorithm

Multiple hidden layers

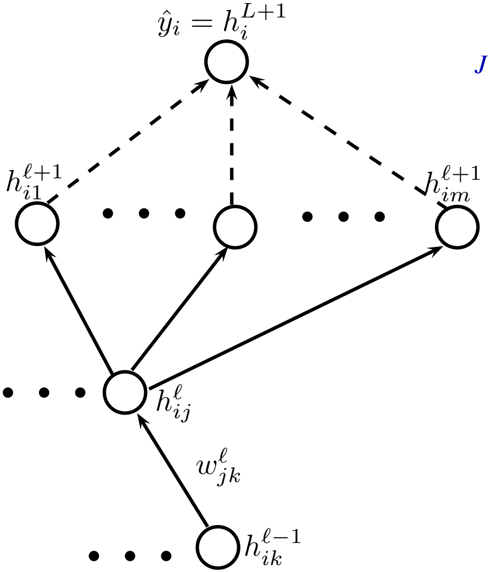


$$h_i^{\ell+1} = \underline{s}\left(W^{\ell+1} h_i^\ell\right)$$

- $W^{\ell+1} = \left(w_{jk}^\ell\right)_{jk}$: weight matrix at the $(\ell+1)$-th layer, weight w_{jk}^ℓ on the edge between $h_{ik}^{\ell-1}$ and h_{ij}^ℓ
- $\underline{s}$: entrywise (logistic) transfer function

$$\hat{y}_i = \underline{s}\left(W^{L+1} \underline{s}\left(W^L \left(\cdots \underline{s}\left(W^1 x_i\right)\right)\right)\right)$$

Backpropagation



$$J = -\sum_{i=1}^{n} y_i \log h_i^{L+1} + (1-y_i)\log(1-h_i^{L+1})$$

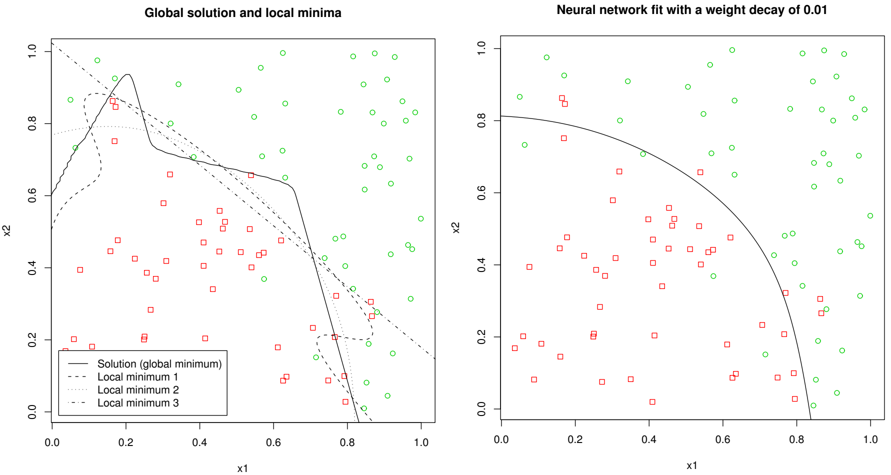
- Gradients wrt h_{ij}^{ℓ} computed by recursive applications of chain rule, and propagated through the network backwards.

$$\frac{\partial J}{\partial h_i^{L+1}} = -\frac{y_i}{h_i^{L+1}} + \frac{1-y_i}{1-h_i^{L+1}}$$

$$\frac{\partial J}{\partial h_{ij}^{\ell}} = \sum_{r=1}^{m} \frac{\partial J}{\partial h_{ir}^{\ell+1}} \frac{\partial h_{ir}^{\ell+1}}{\partial h_{ij}^{\ell}}$$

$$\frac{\partial J}{\partial w_{jk}^{\ell}} = \sum_{i=1}^{n} \frac{\partial J}{\partial h_{ij}^{\ell}} \frac{\partial h_{ij}^{\ell}}{\partial w_{jk}^{\ell}}$$

Neural Networks



R package implementing neural networks with a single hidden layer: `nnet`.

Neural Networks – Discussion

- Nonlinear hidden units introduce modelling flexibility.
- In contrast to user-introduced nonlinearities, features are global, and can be learned to maximize predictive performance.
- Neural networks with a single hidden layer and sufficiently many hidden units can model arbitrarily complex functions.
- Optimization problem is **not convex**, and objective function can have many local optima, plateaus and ridges.
- On large scale problems, often use **stochastic gradient descent**, along with a whole host of techniques for optimization, regularization, and initialization.
- Recent developments, especially by Geoffrey Hinton, Yann LeCun, Yoshua Bengio, Andrew Ng and others. See also `http://deeplearning.net/`.

Dropout Training of Neural Networks

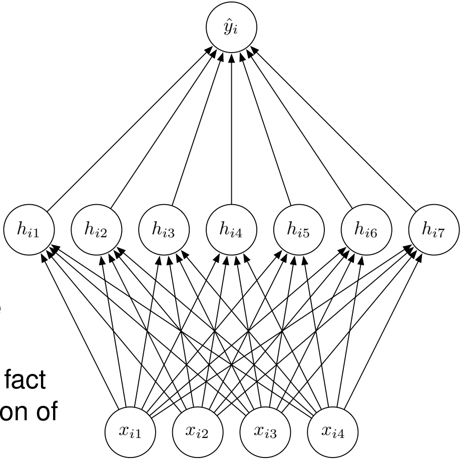
- Neural network with single layer of hidden units:
  - **Hidden unit activations**:

$$h_{ik} = s\left(b_k^h + \sum_{j=1}^{p} W_{jk}^h x_{ij}\right)$$

  - **Output probability**:

$$\hat{y}_i = s\left(b^o + \sum_{k=1}^{m} W_k^o h_{ik}\right)$$

- Large, overfitted networks often have co-adapted hidden units.
- What each hidden unit learns may in fact be useless, e.g. predicting the negation of predictions from other units.
- Can prevent co-adaptation by randomly **dropping out** units from network.

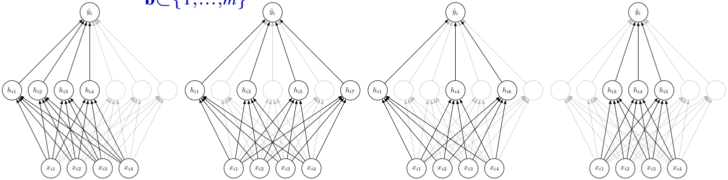


Hinton et al (2012).

Dropout Training of Neural Networks

- Model as an ensemble of networks:

$$p(y_i = 1|x_i, \theta) = \sum_{\mathbf{b} \subset \{1,\ldots,m\}} q^{|\mathbf{b}|}(1-q)^{m-|\mathbf{b}|} p(y_i = 1|x_i, \theta, \text{drop out units } \mathbf{b})$$



- **Weight-sharing** among all networks: each network uses a subset of the parameters of the full network (corresponding to the retained units).
- Training by stochastic gradient descent: at each iteration a network is sampled from ensemble, and its subset of parameters are updated.
- Biological inspiration: 10^{14} weights to be fitted in a lifetime of 10^{9} seconds
  - Poisson spikes as a regularization mechanism which prevents co-adaptation: Geoff Hinton on Brains, Sex and Machine Learning

Dropout Training of Neural Networks

Classification of phonemes in speech.

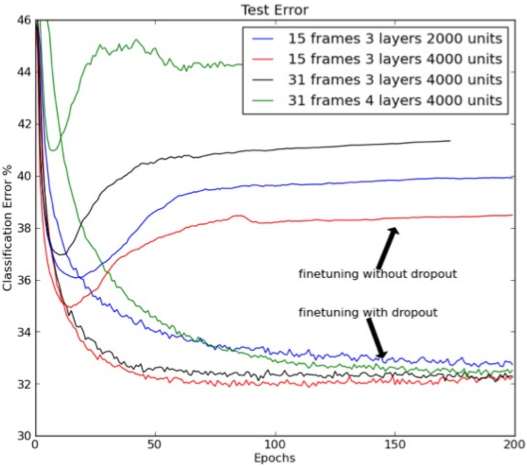
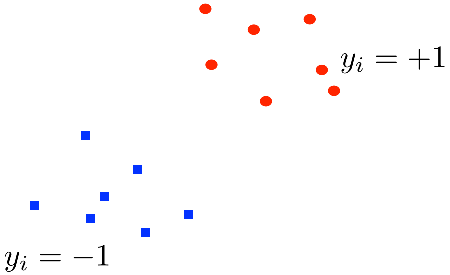


Figure from Hinton et al.

Support Vector Machines

Linearly separable points

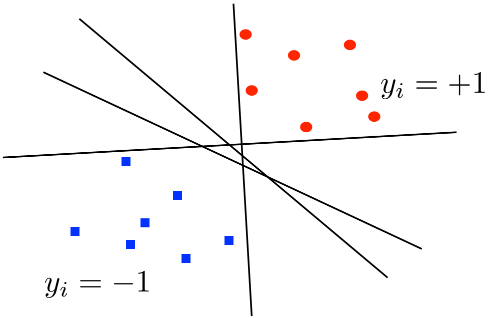
Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.



Data given by $\{x_i, y_i\}_{i=1}^{n}$, $x_i \in \mathbb{R}^p$, $y_i \in \{-1, +1\}$

Linearly separable points

Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.

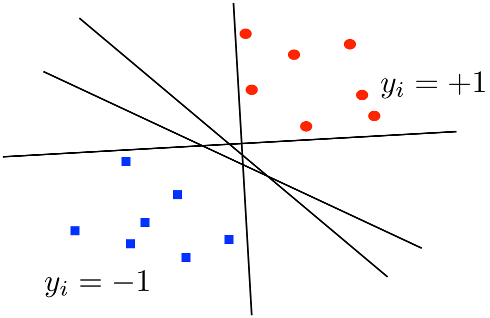


Hyperplane equation $w^\top x + b = 0$. Linear discriminant given by

$$f(x) = \text{sign}(w^\top x + b)$$

Linearly separable points

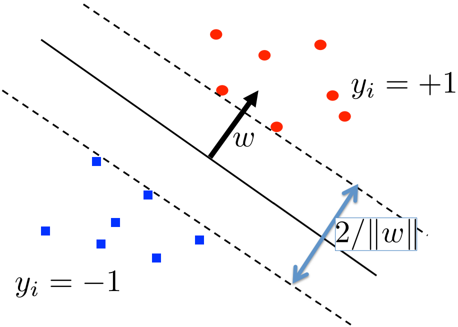
Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.



For a datapoint close to the decision boundary, a small change leads to a change in classification. Can we make the classifier more robust?

Linearly separable points

Classify two clouds of points, where there exists a hyperplane which linearly separates one cloud from the other without error.



Smallest distance from each class to the separating hyperplane $w^\top x + b$ is called the **margin**.

Maximum margin classifier, linearly separable case

This problem can be expressed as follows:

$$\max_{w,b} (\text{margin}) = \max_{w,b} \left(\frac{1}{\|w\|} \right)$$

subject to

$$\begin{cases} w^\top x_i + b \geq 1 & i : y_i = +1, \\ w^\top x_i + b \leq -1 & i : y_i = -1. \end{cases}$$

The resulting classifier is

$$f(x) = \text{sign}(w^\top x + b),$$

We can rewrite to obtain a **quadratic program**:

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

subject to

$$y_i(w^\top x_i + b) \geq 1.$$

Maximum margin classifier: with errors allowed

Allow “errors”: points within the margin, or even on the wrong side of the decision boudary. Ideally:

$$\min_{w,b}\left(\frac{1}{2}\|w\|^2+C\sum_{i=1}^{n}\mathbb{I}[y_i\left(w^\top x_i+b\right)<0]\right),$$

where C controls the tradeoff between maximum margin and loss.
Replace with **convex upper bound**:

$$\min_{w,b}\left(\frac{1}{2}\|w\|^2+C\sum_{i=1}^{n}h\left(y_i\left(w^\top x_i+b\right)\right)\right).$$

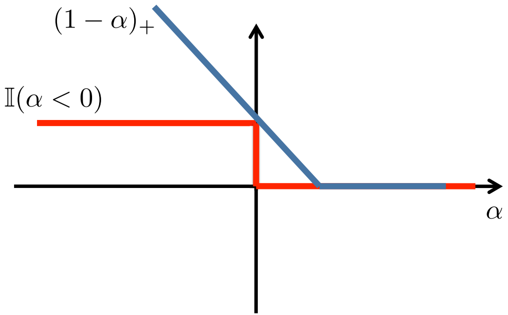
with hinge loss,

$$h(\alpha)=(1-\alpha)_+=\begin{cases}1-\alpha, & 1-\alpha>0\\ 0, & \text{otherwise.}\end{cases}$$

Hinge loss

Hinge loss:

$$h(\alpha) = (1-\alpha)_+ = \begin{cases} 1-\alpha, & 1-\alpha > 0 \\ 0, & \text{otherwise.} \end{cases}$$



Support vector classification

Substituting in the hinge loss, we get

$$\min_{w,b}\left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} h\left(y_i\left(w^\top x_i + b\right)\right)\right).$$

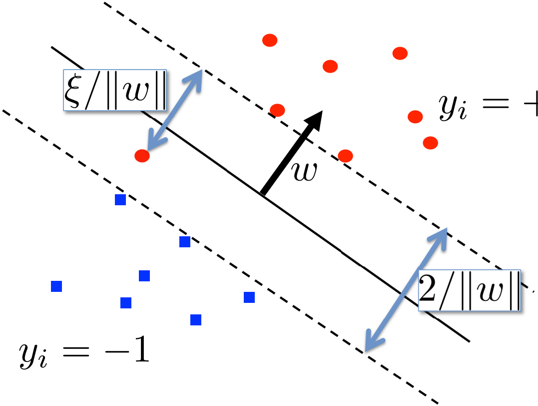
To simplify, use substitution $\xi_i = h\left(y_i\left(w^\top x_i + b\right)\right)$:

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

subject to

$$\xi_i \geq 0 \qquad y_i\left(w^\top x_i + b\right) \geq 1 - \xi_i$$

Support vector classification



Does strong duality hold?

1. Is the optimization problem convex wrt the variables w, b, ξ?

$$\begin{aligned} \text{minimize} \quad & f_0(w,b,\xi) := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & f_i(w,b,\xi) := 1 - \xi_i - y_i\left(w^\top x_i + b\right) \leq 0,\ i = 1,\ldots,n \\ & f_i(w,b,\xi) := -\xi_i \leq 0,\ i = n+1,\ldots,2n \end{aligned}$$

   Each of $f_0, f_1, \ldots, f_n$ are convex. No equality constraints.

2. Does Slater's condition hold? Yes (trivially) since inequality constraints **affine**.

Thus **strong duality** holds, the problem is differentiable, hence the KKT conditions hold at the global optimum.

Support vector classification: Lagrangian

The Lagrangian: $L(w, b, \xi, \alpha, \lambda) =$

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(w^\top x_i + b\right)\right) + \sum_{i=1}^{n}\lambda_i(-\xi_i)$$

with dual variable constraints

$$\alpha_i \geq 0, \qquad \lambda_i \geq 0.$$

Minimize wrt the primal variables w, b, and ξ.
Derivative wrt w:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0 \qquad w = \sum_{i=1}^{n}\alpha_i y_i x_i.$$

Derivative wrt b:

$$\frac{\partial L}{\partial b} = \sum_i y_i \alpha_i = 0.$$

Support vector classification: Lagrangian

Derivative wrt ξ_i:

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \lambda_i = 0 \qquad \alpha_i = C - \lambda_i.$$

Since $\lambda_i \geq 0$,

$$\alpha_i \leq C.$$

Now use complementary slackness:

Non-margin SVs (margin errors): $\alpha_i = C > 0$:

1. We immediately have $y_i \left(w^\top x_i + b\right) = 1 - \xi_i$.
2. Also, from condition $\alpha_i = C - \lambda_i$, we have $\lambda_i = 0$, so $\xi_i \geq 0$

Margin SVs: $0 < \alpha_i < C$:

1. We again have $y_i \left(w^\top x_i + b\right) = 1 - \xi_i$.
2. This time, from $\alpha_i = C - \lambda_i$, we have $\lambda_i > 0$, hence $\xi_i = 0$.

Non-SVs (on the correct side of the margin): $\alpha_i = 0$:

1. From $\alpha_i = C - \lambda_i$, we have $\lambda_i > 0$, hence $\xi_i = 0$.
2. Thus, $y_i \left(w^\top x_i + b\right) \geq 1$

The support vectors

We observe:

1. The solution is sparse: points which are neither on the margin nor "margin errors" have $\alpha_i = 0$
2. The support vectors: only those points on the decision boundary, or which are margin errors, contribute.
3. Influence of the non-margin SVs is bounded, since their weight cannot exceed C.

Support vector classification: dual function

Thus, our goal is to maximize the dual,

$$
\begin{aligned}
g(\alpha, \lambda) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - y_i\left(w^\top x_i + b\right) - \xi_i\right) \\
&\quad + \sum_{i=1}^{n}\lambda_i(-\xi_i) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j \\
&\quad \underbrace{-b\sum_{i=1}^{n}\alpha_i y_i}_{0} + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i - \sum_{i=1}^{n}(C-\alpha_i)\xi_i \\
&= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^\top x_j.
\end{aligned}
$$

Support vector classification: dual problem

Maximize the dual,

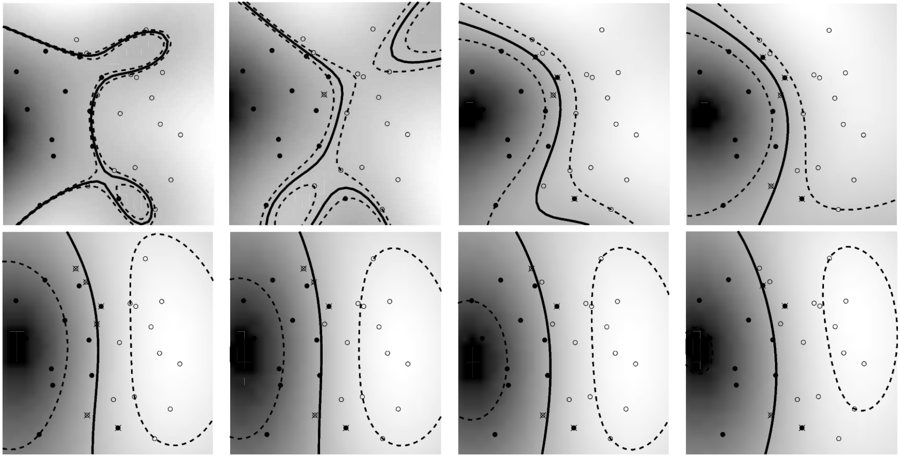
$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j,$$

subject to the constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

This is a quadratic program. From α, obtain the hyperplane with
$w = \sum_{i=1}^{n} \alpha_i y_i x_i$
Offset b can be obtained from any of the margin SVs: $1 = y_i \left(w^\top x_i + b \right)$.

Support vector classification: kernel version



Taken from Schoelkopf and Smola (2002)

Support vector classification: kernel version

Maximum margin classifier in RKHS: write the hinge loss formulation

$$\min_w \left(\frac{1}{2}\|w\|_{\mathcal{H}}^2 + C\sum_{i=1}^n \theta\left(y_i \langle w, k(x_i,\cdot)\rangle_{\mathcal{H}}\right)\right)$$

for the RKHS $\mathcal{H}$ with kernel $k(x,x')$. Maximizing the margin equivalent to minimizing $\|w\|_{\mathcal{H}}^2$: for many RKHSs a smoothness constraint (e.g. Gaussian kernel).

Support vector classification: kernel version

Maximum margin classifier in RKHS: write the hinge loss formulation

$$\min_{w}\left(\frac{1}{2}\|w\|_{\mathcal{H}}^{2}+C\sum_{i=1}^{n}\theta\left(y_i\left\langle w,k(x_i,\cdot)\right\rangle_{\mathcal{H}}\right)\right)$$

for the RKHS $\mathcal{H}$ with kernel $k(x,x')$. Maximizing the margin equivalent to minimizing $\|w\|_{\mathcal{H}}^{2}$: for many RKHSs a smoothness constraint (e.g. Gaussian kernel).

Optimization over an infinitely dimensional space!

Support vector classification: kernel version

Dual in the linear case:

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j,$$

subject to the constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

Support vector classification: kernel version

Dual in the linear case:

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j,$$

subject to the constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

Dual in the kernel case:

$$\max_{\alpha} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \right),$$

subject to the constraints

$$0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

Convex in α since K is positive definite.

Primal and the representer theorem

After solving the dual we can obtain the decision function

$$w(\cdot) = \sum_{i=1}^{n} y_i \alpha_i k(x_i, \cdot).$$

which lies in a finite dimensional subspace of $\mathcal{H}$, i.e., it is a (sparse) linear combination of the features (representer theorem).
Thus, we can also derive the finite-dimensional primal by setting $w(\cdot) = \sum_{i=1}^{n} \beta_i k(x_i, \cdot)$.

$$\min_{\beta,\xi} \left(\frac{1}{2} \beta^\top K \beta + C \sum_{i=1}^{n} \xi_i \right) \tag{1}$$

where the matrix K has i, jth entry $K_{ij} = k(x_i, x_j)$, subject to

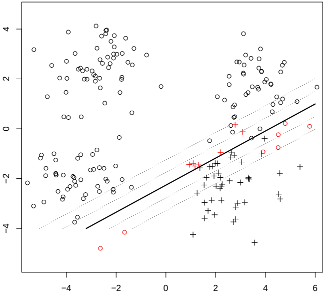
$$\xi_i \geq 0 \qquad y_i \sum_{j=1}^{n} \beta_j k(x_i, x_j) \geq 1 - \xi_i.$$

What is an advantage of the dual?

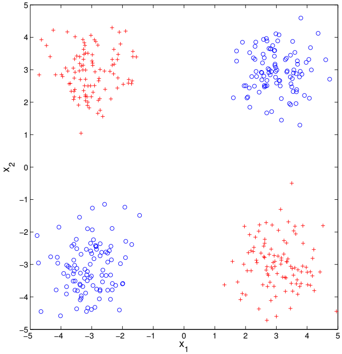
Kernel Methods

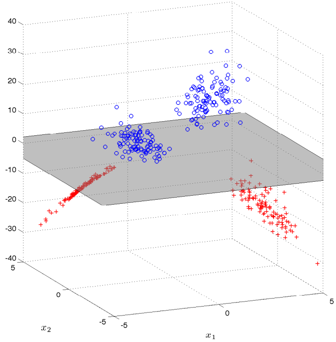
Non-linear methods

- Linear methods (LDA, logistic regression, naïve Bayes) are simple and effective techniques to learn from data "to first order".
- To capture more intricate information from data, non-linear methods are often needed:
  - Explicit non-linear transformations $x \mapsto \varphi(x)$.
  - Local methods like kNN.
- **Kernel methods**: introduce non-linearities through **implicit** non-linear transforms, often local in nature.



XOR example





- No linear classifier separates red from blue.
- Linear separation after mapping to a **higher dimensional feature space**:

$$\mathbb{R}^2 \ni \begin{pmatrix} x^{(1)} & x^{(2)} \end{pmatrix}^\top = x \;\mapsto\; \varphi(x) = \begin{pmatrix} x^{(1)} & x^{(2)} & x^{(1)}x^{(2)} \end{pmatrix}^\top \in \mathbb{R}^3$$

Kernel SVM

- Back to the dual C-SVM with explicit non-linear transformation $x \mapsto \varphi(x)$:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \varphi(x_i)^\top \varphi(x_j) \quad \text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \preceq \alpha \preceq C \end{cases}$$

- Suppose $p = 2$, and we would like to introduce quadratic non-linearities,

$$\varphi(x) = \left(1, \sqrt{2}x^{(1)}, \sqrt{2}x^{(2)}, \sqrt{2}x^{(1)}x^{(2)}, \left(x^{(1)}\right)^2, \left(x^{(2)}\right)^2\right)^\top$$

  Then

$$\begin{aligned} \varphi(x_i)^\top \varphi(x_j) &= 1 + 2x_i^{(1)}x_j^{(1)} + 2x_i^{(2)}x_j^{(2)} + 2x_i^{(1)}x_i^{(2)}x_j^{(1)}x_j^{(2)} \\ &\quad + \left(x_i^{(1)}\right)^2 \left(x_j^{(1)}\right)^2 + \left(x_i^{(2)}\right)^2 \left(x_j^{(2)}\right)^2 = (1 + x_i^\top x_j)^2 \end{aligned}$$

- Since only dot-products are needed in the objective function, non-linear transform need not be computed explicitly - inner product between features is often a simple function (**kernel**) of x_i and x_j: $k(x_i, x_j) = \varphi(x_i)^\top \varphi(x_j) = (1 + x_i^\top x_j)^2$
- Generally, m-order interactions can be implemented simply by $k(x_i, x_j) = (1 + x_i^\top x_j)^m$ (**polynomial kernel**).

Kernel SVM: Kernel trick

- Kernel SVM with $k(x_i, x_j)$. Non-linear transformation $x \mapsto \varphi(x)$ still present, but **implicit** (coordinates of the vector $\varphi(x)$ are never computed).

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \quad \text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \preceq \alpha \preceq C \end{cases}$$

- Prediction? $f(x) = \text{sign}\left(w^\top \varphi(x) + b\right)$, where $w = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)$ and offset b obtained from a margin support vector x_j with $\alpha_j \in (0, C)$.
  - No need to compute w either! Just need

$$w^\top \varphi(x) = \sum_{i=1}^{n} \alpha_i y_i \varphi(x_i)^\top \varphi(x) = \sum_{i=1}^{n} \alpha_i y_i k(x_i, x).$$

  - Get offset from

$$b = y_j - w^\top \varphi(x_j) = y_j - \sum_{i=1}^{n} \alpha_i y_i k(x_i, x_j)$$

    for any margin support-vector x_j ($\alpha_j \in (0, C)$).
- Fitted a separating hyperplane in a high-dimensional feature space without ever mapping explicitly to that space.

Kernel trick in general

- In a learning algorithm, if only inner products $x_i^\top x_j$ are explicitly used, rather than data items x_i, x_j directly, we can replace them with a kernel function $k(x_i, x_j) = \langle \varphi(x_i), \varphi(x_j) \rangle$, where $\varphi(x)$ could be **nonlinear, high- and potentially infinite-dimensional** features of the original data.
  - Kernel ridge regression
  - Kernel PCA
  - Kernel K-means
  - Kernel FDA

Gram matrix

- The **Gram matrix** is the matrix of dot-products, $\mathbf{K}_{ij} = \varphi(x_i)^\top \varphi(x_j)$.

$$\mathbf{K} = \begin{pmatrix} - \ \varphi(x_1)^\top \ - \\ \vdots \\ - \ \varphi(x_i)^\top \ - \\ \vdots \\ - \ \varphi(x_n)^\top \ - \end{pmatrix} \cdot \begin{pmatrix} | & & | & & | \\ \varphi(x_1) & \cdots & \varphi(x_j) & \cdots & \varphi(x_n) \\ | & & | & & | \end{pmatrix}$$

- Since $\mathbf{K} = \Phi\Phi^\top$, it is symmetric and positive semidefinite.
- Recall: Gram matrix closely related to the distance matrix (MDS)
- Assuming features are centred, the sample covariance of features is $\Phi^\top\Phi$.
- Many kernel methods, e.g. kernel PCA, make use of the duality between the Gram and the sample covariance matrix.

Kernel: an inner product between feature maps

Definition (kernel)

Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **kernel** if there exists a **Hilbert space** and a map $\varphi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

$$k(x, x') := \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}} .$$

- Almost no conditions on $\mathcal{X}$ (eg, $\mathcal{X}$ itself need not have an inner product, e.g., documents).
- Think of kernel as **similarity measure between features**

What are some simple kernels? E.g., for text documents? For images?

- A single kernel can correspond to multiple sets of underlying features.

$$\varphi_1(x) = x \qquad \text{and} \qquad \varphi_2(x) = \begin{pmatrix} x/\sqrt{2} & x/\sqrt{2} \end{pmatrix}^\top$$

Positive semidefinite functions

If we are given a "measure of similarity" with two arguments, $k(x, x')$, how can we determine if it is a valid kernel?

1. Find a feature map?
   - Sometimes not obvious (especially if the feature vector is infinite dimensional)
2. A simpler direct property of the function: positive semidefiniteness.

Positive semidefinite functions

Definition (Positive semidefinite functions)

A symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semidefinite if $\forall n \geq 1,\ \forall (a_1, \ldots a_n) \in \mathbb{R}^n,\ \forall (x_1, \ldots, x_n) \in \mathcal{X}^n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \kappa(x_i, x_j) \geq 0.$$

- Kernel $k(x, y) := \langle \varphi(x), \varphi(y) \rangle_{\mathcal{H}}$ for a Hilbert space $\mathcal{H}$ is positive semidefinite.

$$\begin{aligned}\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle a_i \varphi(x_i), a_j \varphi(x_j) \rangle_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^{n} a_i \varphi(x_i) \right\|_{\mathcal{H}}^2 \geq 0.\end{aligned}$$

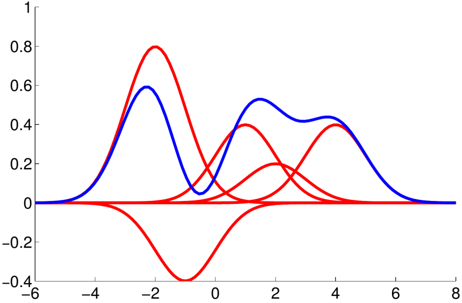
Positive semidefinite functions are kernels

Moore-Aronszajn Theorem

Every positive semidefinite function is a kernel for some Hilbert space $\mathcal{H}$.

- $\mathcal{H}$ is usually thought of as a space of functions (**Reproducing kernel Hilbert space - RKHS**)

Gaussian RBF kernel $k(x, x') = \exp\left(-\frac{1}{2\gamma^2}\|x - x'\|^2\right)$ has an infinite-dimensional $\mathcal{H}$ with elements $h(x) = \sum_{i=1}^{m} a_i k(x_i, x)$ (recall that $w^\top \varphi(x)$ in SVM has exactly this form!).



Reproducing kernel

Definition (Reproducing kernel)

Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ defined on a non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called **a reproducing kernel** of $\mathcal{H}$ if it satisfies

- $\forall x \in \mathcal{X}, \;\; k_x = k(\cdot, x) \in \mathcal{H}$,
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \;\; \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$ (the reproducing property).

Reproducing kernel

Definition (Reproducing kernel)

Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$ defined on a non-empty set $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called **a reproducing kernel** of $\mathcal{H}$ if it satisfies

- $\forall x \in \mathcal{X}, \quad k_x = k(\cdot, x) \in \mathcal{H}$,
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \quad \langle f, k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$ (the reproducing property).

In particular, for any $x, y \in \mathcal{X}$, $k(x, y) = \langle k(\cdot, y), k(\cdot, x) \rangle_{\mathcal{H}} = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}}$.

RKHS

Definition (Reproducing kernel Hilbert space)

A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, defined on a non-empty set $\mathcal{X}$ is said to be a Reproducing Kernel Hilbert Space (RKHS) if evaluation functionals $\delta_x : \mathcal{H} \to \mathbb{R}$, $\delta_x f = f(x)$ are continuous $\forall x \in \mathcal{X}$.

RKHS

Definition (Reproducing kernel Hilbert space)

A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, defined on a non-empty set $\mathcal{X}$ is said to be a Reproducing Kernel Hilbert Space (RKHS) if evaluation functionals $\delta_x : \mathcal{H} \to \mathbb{R}$, $\delta_x f = f(x)$ are continuous $\forall x \in \mathcal{X}$.

Theorem (Norm convergence implies pointwise convergence)

If $\lim_{n\to\infty} \|f_n - f\|_{\mathcal{H}} = 0$, *then* $\lim_{n\to\infty} f_n(x) = f(x)$, $\forall x \in \mathcal{X}$.

RKHS

Definition (Reproducing kernel Hilbert space)

A Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \to \mathbb{R}$, defined on a non-empty set $\mathcal{X}$ is said to be a Reproducing Kernel Hilbert Space (RKHS) if evaluation functionals $\delta_x : \mathcal{H} \to \mathbb{R}$, $\delta_x f = f(x)$ are continuous $\forall x \in \mathcal{X}$.

Theorem (Norm convergence implies pointwise convergence)

If $\lim_{n\to\infty} \|f_n - f\|_{\mathcal{H}} = 0$, *then* $\lim_{n\to\infty} f_n(x) = f(x), \forall x \in \mathcal{X}$.

If two functions $f, g \in \mathcal{H}$ are close in the norm of $\mathcal{H}$, then $f(x)$ and $g(x)$ are close for all $x \in \mathcal{X}$

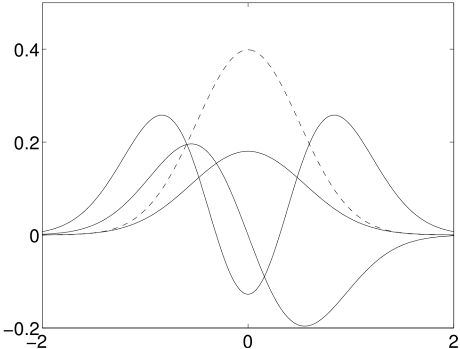
RKHS of a Gaussian RBF kernel

Gaussian kernel

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\gamma^2}\right) = \sum_{j=1}^{\infty}\left(\sqrt{\lambda_j e_j(x)}\right)\left(\sqrt{\lambda_j e_j(x')}\right)$$

$$\lambda_j \propto b^j \qquad b < 1$$

$$e_j(x) \propto \exp(-(c-a)x^2)H_j(x\sqrt{2c}),$$



a, b, c are functions of γ, and H_j is the j-th order Hermite polynomial.

$$\|f\|^2_{\mathcal{H}_k} = \sum_{j=1}^{\infty}\frac{a_j^2}{\lambda_j}$$

(Figure from Rasmussen and Williams)

Examples of kernels

- **Linear**: $k(x, x') = x^\top x'$.
- **Polynomial**: $k(x, x') = (c + x^\top x')^m$, $c \in \mathbb{R}$, $m \in \mathbb{N}$.
- **Gaussian RBF**: $k(x, x') = \exp\left(-\frac{1}{2\gamma^2}\|x - x'\|^2\right)$, $\gamma > 0$.
- **Laplacian**: $k(x, x') = \exp\left(-\frac{1}{2\gamma^2}\|x - x'\|\right)$, $\gamma > 0$.
- **Rational quadratic**: $k(x, x') = \left(1 + \frac{\|x - x'\|^2}{2\alpha\gamma^2}\right)^{-\alpha}$, $\alpha, \gamma > 0$.
- **Brownian covariance**: $k(x, x') = \frac{1}{2}\left(\|x\|^\gamma + \|x\|^\gamma - \|x - x'\|^\gamma\right)$, $\gamma \in [0, 2]$.

New kernels from old: sums, transformations

The great majority of useful kernels are built from simpler kernels.

Lemma (Sums of kernels are kernels)

Given $\alpha > 0$ and k, k_1 and k_2 all kernels on $\mathcal{X}$, then αk and $k_1 + k_2$ are kernels on $\mathcal{X}$.

To prove this, just check inner product definition. A difference of kernels may not be a kernel (**why?**)

Lemma (Mappings between spaces)

Let $\mathcal{X}$ and $\widetilde{\mathcal{X}}$ be sets, and define a map $s : \mathcal{X} \to \widetilde{\mathcal{X}}$. Define the kernel k on $\widetilde{\mathcal{X}}$. Then the kernel $k(s(x), s(x'))$ is a kernel on $\mathcal{X}$.

Example: $k(x, x') = x^2 \left(x'\right)^2$.

New kernels from old: products

Lemma (Products of kernels are kernels)

Given k_1 on $\mathcal{X}_1$ and k_2 on $\mathcal{X}_2$, then $k_1 \times k_2$ is a kernel on $\mathcal{X}_1 \times \mathcal{X}_2$.

Proof.

Sketch for finite-dimensional spaces only. Assume $\mathcal{H}_1$ corresponding to k_1 is $\mathbb{R}^m$, and $\mathcal{H}_2$ corresponding to k_2 is $\mathbb{R}^n$. Define:

- $k_1 := u^\top v$ for $u, v \in \mathbb{R}^m$ (e.g.: kernel between two images)
- $k_2 := p^\top q$ for $p, q \in \mathbb{R}^n$ (e.g.: kernel between two captions)

Is the following a kernel?

$$K\left[(u,p);(v,q)\right] = k_1 \times k_2$$

(e.g. kernel between one image-caption pair and another)

$\square$

New kernels from old: products

Proof.

(continued)

$$\begin{aligned} k_1 k_2 &= \left(u^\top v\right)\left(q^\top p\right) \\ &= \mathrm{trace}(u^\top v q^\top p) \\ &= \mathrm{trace}(p u^\top v q^\top) \\ &= \langle A, B \rangle , \end{aligned}$$

where $A := pu^\top$, $B := qv^\top$ (features of image-caption pairs) Thus $k_1 k_2$ is a valid kernel, since inner product between $A, B \in \mathbb{R}^{m \times n}$ is

$$\langle A, B \rangle = \mathrm{trace}(AB^\top).$$

□

Kernel methods at scale

- Expressivity of kernel methods (rich, often infinite-dimensional hypothesis classes) comes with a cost that scales at least quadratically in the number of observations (due to needing to compute, store and often invert the Gram matrix).
- Problematic when we have a lot of observations (and this is exactly when we want to use a rich expressive model!)
- Scaling up kernel methods is a very active research area (Rahimi & Recht 2007; Le, Sarlos, Smola, 2013; Wilson et al, 2014; Dai et al, 2014; Sriperumbudur, Szabo, 2015).

Random Fourier features

Bochner's representation: any positive definite **translation-invariant** kernel on $\mathbb{R}^p$ can be written as

$$\begin{aligned} k(x,y) &= \int_{\mathbb{R}^p} d\Gamma(\omega) \exp\left(i\omega^\top(x-y)\right) \\ &= 2\int_{\mathbb{R}^p} d\Gamma(\omega) \int_0^{2\pi} db \cos\left(\omega^\top x + b\right) \cos\left(\omega^\top y + b\right) \end{aligned}$$

for some positive measure (w.l.o.g. a probability distribution) Γ.

- Idea: for a given kernel k, compute its inverse Fourier transform and sample m frequencies $\omega_i \sim \Gamma$, $b_i \sim \text{Unif}[0, 2\pi]$ and use a Monte Carlo estimator of the kernel function:

$$\begin{aligned} k(x,y) &= \frac{2}{m}\sum_{i=1}^{m} \cos\left(\omega_i^\top x + b_i\right) \cos\left(\omega_i^\top y + b_i\right) \\ &= \langle \phi_{\omega,\mathbf{b}}(x), \phi_{\omega,\mathbf{b}}(y) \rangle, \end{aligned}$$

with an explicit set of features
$x \mapsto \left[\sqrt{\frac{2}{m}} \cos\left(\omega_1^\top x + b_1\right), \ldots, \sqrt{\frac{2}{m}} \cos\left(\omega_m^\top x + b_m\right)\right] \in \mathbb{R}^m$, allowing running algorithms in the primal and reducing quadratic cost in n to quadratic cost in m. (Rahimi & Recht 2007), (Le, Sarlos, Smola, 2013)

Inducing variables

- Directly approximate the $n \times n$ Gram matrix K_{XX} of a set of inputs $\{x_i\}_{i=1}^n$ with
$$\hat{K}_{XX} = K_{XZ} K_{ZZ}^{-1} K_{ZX}$$
where K_{ZZ} is $m \times m$ on “inducing” inputs $\{z_i\}_{i=1}^m$.
- Corresponds to explicit feature representation $x \mapsto K_{xZ} K_{ZZ}^{-1/2}$.
- Surrogate kernel $\hat{k}(x, x') = \langle k_|(\cdot, x), k_|(\cdot, x') \rangle$, where $k_|(\cdot, x)$ is a projection of $k(\cdot, x)$ to span $\{k(\cdot, z_1), \ldots, k(\cdot, z_m)\}$
- Often used in regression with Gaussian processes: with the use of Sherman-Morrison-Woodbury identity, reduces $O(n^3)$ cost to $O(nm^2)$.
(Quiñonero-Candela and Rasmussen, 2005), (Snelson and Ghahramani, 2006)

Kernel Methods – Discussion

- Kernel methods allows for very flexible and powerful machine learning models.
- **Nonparametric** method: parameter space (e.g., of parameter w in SVM) can be infinite-dimensional
- Kernels can be defined over more complex structures than vectors, e.g. graphs, strings, images, probability distributions.
- Computational cost at least quadratic in the number of observations, often $O(n^3)$ computation and $O(n^2)$ memory (various approximations with good scaling up properties)
- Further reading:
  - Bishop, Pattern Recognition and Machine Learning, Chapter 6.
  - Schölkopf and Smola, Learning with Kernels, 2001.
  - Rasmussen and Williams, Gaussian Processes for Machine Learning, 2006.