



Biological Network Analysis

Tutorial 1

Nino Shervashidze

nino.shervashidze@tuebingen.mpg.de.

Bioinformatics Group
MPIs Tübingen



Some preliminaries

- **Gram matrix:** given a function $k : \mathcal{X}^2 \rightarrow \mathbb{R}$ and examples $x_1, x_2, \dots, x_n \in \mathcal{X}$, the $n \times n$ matrix K with elements

$$K_{ij} = k(x_i, x_j)$$

is called the **Gram matrix** (or **kernel matrix**) of k with respect to x_1, x_2, \dots, x_n .

- **Positive definite matrix:** A real $n \times n$ matrix K satisfying

$$\sum_{ij} c_i c_j K_{ij} \geq 0$$

for all $c_i \in \mathbb{R}$ is called positive definite (strictly speaking, positive semi-definite, but 'semi' is often omitted in machine learning context).



(Positive Definite) Kernel

A function k on $\mathcal{X} \times \mathcal{X}$ is called a **positive definite kernel**, if for all $n \in \mathbb{N}$ and $\{x_1, x_2, \dots, x_n \in \mathcal{X}\}$ the corresponding Gram matrix is positive definite. It is often simply referred to as a **kernel**.

Mercer's theorem

Any continuous, symmetric, positive semi-definite kernel function k on $\mathcal{X} \times \mathcal{X}$ can be expressed as an **inner product** in a high-dimensional space.

Examples/exercises

Is $\forall x, \forall x', k(x, x') = 0$ a kernel?

Is $\forall x, \forall x', k(x, x') = 1$ a kernel? And -1 ?

Is $k(x, x') = \langle x, x' \rangle$ a kernel?



Closure properties of kernels

- **sum of kernels:** if k_1 and k_2 are kernels, and $\alpha_1, \alpha_2 \geq 0$, then

$$\alpha_1 k_1 + \alpha_2 k_2$$

is a kernel

- **pointwise product:** if k_1 and k_2 are kernels, then $k_1 k_2$, defined by

$$(k_1 k_2)(x, x') = k_1(x, x') k_2(x, x'),$$

is a kernel



Some prominent kernels

- linear kernel

$$k(x_i, x_j) = \sum_{l=1}^n x_{i_l} x_{j_l} = x_i^\top x_j,$$

- polynomial kernel

$$k(x_i, x_j) = (x_i^\top x_j + c)^d,$$

where $c, d \in \mathbb{R}$,

- Gaussian RBF kernel

$$k(x_i, x_j) = -\frac{1}{2\sigma^2} \|x_i - x_j\|^2,$$

where $\sigma \in \mathbb{R}$.

...So, why can we say that polynomial kernel is an inner product?



How to measure the performance of a classification algorithm?

Confusion matrix

| | | Predicted labels | |
|-------------|----------|------------------------|------------------------|
| | | Positive | Negative |
| True labels | Positive | True Positives | False Negatives |
| | Negative | False Positives | True Negatives |

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

precision and **recall** are used if **FPs** and **FNs** have different meaning or cost (e.g. drug discovery)

accuracy=1 → no errors

precision=1 → all that is classified positive is really positive

recall=1 → all positives are classified positive

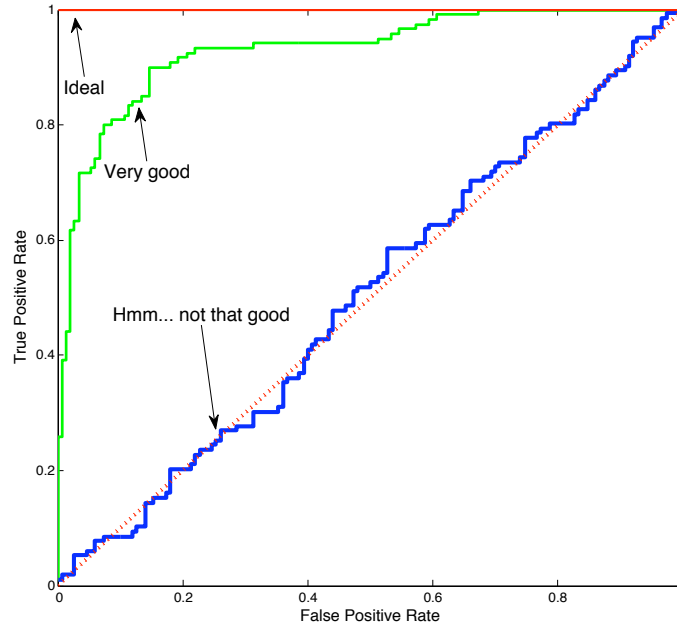


Area Under ROC Curve

- ROC stands for Receiver operating characteristic (signal detection theory)
- ROC curve is most meaningful if the classification function outputs **not binary labels**, but some **real values** y (these values are then converted into binary labels: the label is 1 if $y >$ some threshold, -1 otherwise. e.g. **SVM**)
- ROC curve shows TP rate vs FP rate for each possible threshold
- AUC does not depend on the threshold, for computing it you only need the original labels and real values y from above
- Informally, AUC is the probability of a classifier to rank positive examples above negative ones



Area Under ROC Curve



● AUC=1

● AUC=0.93

● AUC=0.50



Independent Evaluation Scheme

Divide the data into **training** and **test** sets X_{tr} and X_{te}

The set of parameter values to be tested: p_1, \dots, p_k

For $i = 1..k$

- Split the training set into c (usually between 2 and 10) subsets of equal size $X_{tr_1}, \dots, X_{tr_c}$
- For $j = 1, \dots, c$
 - Learn on $X_{tr} \setminus X_{tr_j}$ using p_i
 - Predict on X_{tr_j} . Let acc_j denote the obtained accuracy
- $meanAccuracy_i = \frac{\sum_{j=1}^c acc_j}{c}$

$bestP = arg \max_{i=1}^k meanAccuracy_i$

Learn on X_{tr} with $bestP$. Predict on X_{te} .

The whole procedure can be repeated for more confidence.



This loop...

For $i = 1..k$

- Split the training set into c (usually between 2 and 10) subsets of equal size $X_{tr_1}, \dots, X_{tr_c}$
- For $j = 1, \dots, c$
 - Learn on $X_{tr} \setminus X_{tr_j}$ using p_i
 - Predict on X_{tr_j} . Let acc_j denote the obtained accuracy
- $meanAccuracy_i = \frac{\sum_{j=1}^c acc_j}{c}$

...is called c -fold cross-validation on training data.



Download from <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.

libSVM options

- t k_type: set type of kernel function (default 2)
 - 0 -- linear: $u \cdot v$
 - 1 -- polynomial: $(\gamma u \cdot v + \text{coef0})^{\text{degree}}$
 - 2 -- radial basis function: $\exp(-\gamma |u - v|^2)$
 - 3 -- sigmoid: $\tanh(\gamma u \cdot v + \text{coef0})$
 - 4 -- input your own kernel matrix K instead of the data matrix (but first add a column $(1:\text{size}(K,1))'$ to K)
- d degree : set degree in kernel function (default 3)
- g gamma : set gamma in kernel function (default $1/k$)
- r coef0 : set coef0 in kernel function (default 0)
- c cost : set the parameter C (error cost, default 1)