Machine Learning

Volker Roth

Department of Mathematics & Computer Science
University of Basel

$$R[f_n] \leq R_{\text{emp}}[f_n] + \sqrt{\frac{a}{n}\left(\text{capacity}(\mathcal{H}) + \ln\frac{b}{\delta}\right)}$$

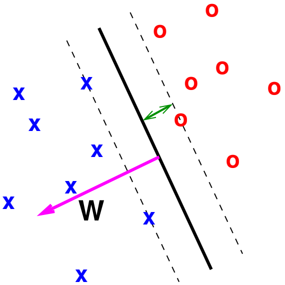
$\rightsquigarrow$ **minimizing capacity($\mathcal{H}$) corresponds to maximizing the margin.**

$\rightsquigarrow$ **Large margin classifiers.**

SVMs

- When the training examples are **linearly separable** we can maximize the margin by minimizing the regularization term

$$\|\boldsymbol{w}\|^2/2 = \sum_{i=1}^{d} w_i^2/2$$



subject to the **classification constraints**

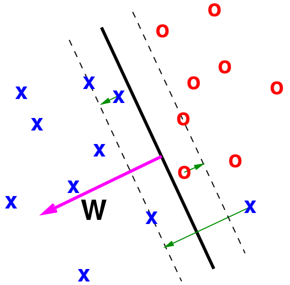
$$y_i[\boldsymbol{x}_i^t \boldsymbol{w}] - 1 \geq 0, \; i = 1, \ldots, n.$$

- The solution is defined only on the basis of a subset of examples or **support vectors.**

SVMs: nonseparable case

- Modify optimization problem slightly by adding a **penalty for violating the classification constraints:**

$$\text{minimize} \quad \|\boldsymbol{w}\|^2/2 + C\sum_{i=1}^{n}\xi_i$$



subject to **relaxed constraints**

$$y_i[\boldsymbol{x}_i^t\boldsymbol{w}] - 1 + \xi_i \geq 0, \ i = 1, \ldots, n.$$

- The $\xi_i \geq 0$ are called **slack variables.**

SVMs: nonseparable case

- We can also write the SVM optimization problem more compactly as

$$C \sum_{i=1}^{n} \overbrace{(1 - y_i[\mathbf{x}_i^t \mathbf{w}])^+}^{\xi_i} + \|\mathbf{w}\|^2/2,$$

  where $(z)^+ = z$ if $z \geq 0$ and zero otherwise.

- This is equivalent to **regularized empirical loss minimization**

$$\underbrace{\frac{1}{n} \sum_{i=1}^{n} (1 - y_i[\mathbf{x}_i^t \mathbf{w}])^+}_{R_{\text{emp}}} + \lambda \|\mathbf{w}\|^2,$$

  where $\lambda = 1/(2nC)$ is the regularization parameter.

SVMs and LOGREG

- When viewed from the point of view of regularized empirical loss minimization, SVM and logistic regression appear quite similar:

$$\text{SVM:}\quad \frac{1}{n}\sum_{i=1}^{n}(1 - y_i[\boldsymbol{x}_i^t \boldsymbol{w}])^+ + \lambda\|\boldsymbol{w}\|^2$$

$$\text{LOGREG:}\quad \frac{1}{n}\sum_{i=1}^{n} -\log \overbrace{\sigma(y_i[\boldsymbol{x}_i^t \boldsymbol{w}])}^{P(y_i|\boldsymbol{x}_i,\boldsymbol{w})} + \lambda\|\boldsymbol{w}\|^2,$$

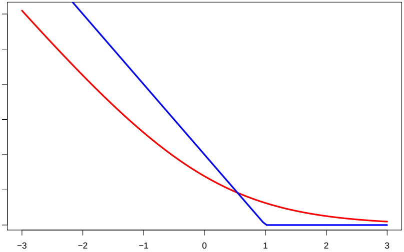
where $\sigma(z) = (1 + e^{-z})^{-1}$ is the logistic function.

SVMs and LOGREG

- The difference comes from how we penalize errors:

$$\text{Both:} \quad \frac{1}{n}\sum_{i=1}^{n} \text{Loss}(\overbrace{y_i[\boldsymbol{x}_i^t \boldsymbol{w}]}^{z}) + \lambda \|\boldsymbol{w}\|^2,$$

- SVM: $\text{Loss}(z) = (1 - z)^+$
- LOGREG: $\text{Loss}(z) = \log(1 + \exp(-z))$



SVMs: solution, Lagrange multipliers

- Back to the separable case: how do we solve

$$\text{minimize}_{\mathbf{w}} \quad \|\mathbf{w}\|^2/2 \quad \text{s.t.} \quad y_i[\mathbf{x}_i^t \mathbf{w}] - 1 \geq 0, \; i = 1, \dots, n.$$

- Represent the constraints as individual loss terms:

$$\sup_{\alpha_i \geq 0} \alpha_i (1 - y_i[\mathbf{x}_i^t \mathbf{w}]) = \begin{cases} 0, & \text{if } y_i[\mathbf{x}_i^t \mathbf{w}] - 1 \geq 0, \\ \infty, & \text{otherwise.} \end{cases}$$

- Rewrite the minimization problem:

$$\begin{aligned} &\text{minimize}_{\mathbf{w}} \quad \|\mathbf{w}\|^2/2 + \sum_{i=1}^{n} \sup_{\alpha_i \geq 0} \alpha_i (1 - y_i[\mathbf{x}_i^t \mathbf{w}]) \\ = &\text{minimize}_{\mathbf{w}} \quad \sup_{\alpha_i \geq 0} \left(\|\mathbf{w}\|^2/2 + \sum_{i=1}^{n} \alpha_i (1 - y_i[\mathbf{x}_i^t \mathbf{w}]) \right) \end{aligned}$$

SVMs: solution, Lagrange multipliers

- Swap maximization and minimization (technically this requires that the problem is convex and feasible $\rightsquigarrow$ **Slater's condition**):

$$\text{minimize}_{\boldsymbol{w}} \quad \left[\sup_{\alpha_i \geq 0} \left(\|\boldsymbol{w}\|^2/2 + \sum_{i=1}^{n} \alpha_i (1 - y_i [\boldsymbol{x}_i^t \boldsymbol{w}]) \right) \right]$$

$$= \text{maximize}_{\alpha_i \geq 0} \quad \left[\min_{\boldsymbol{w}} \underbrace{\left(\|\boldsymbol{w}\|^2/2 + \sum_{i=1}^{n} \alpha_i (1 - y_i [\boldsymbol{x}_i^t \boldsymbol{w}]) \right)}_{J(\boldsymbol{w};\boldsymbol{\alpha})} \right]$$

- We have to minimize $J(\boldsymbol{w}; \boldsymbol{\alpha})$ over parameters $\boldsymbol{w}$ for fixed **Lagrange multipliers $\alpha_i \geq 0$.**
  Simple, because $J(\boldsymbol{w})$ is convex $\rightsquigarrow$ set derivative to zero $\rightsquigarrow$ only one stationary point $\rightsquigarrow$ global minimum.

SVMs: solution, Lagrange multipliers

- Find optimal $\boldsymbol{w}$ by setting the derivatives to zero:

$$\frac{\partial}{\partial \boldsymbol{w}} J(\boldsymbol{w}; \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_i \alpha_i y_i \boldsymbol{x}_i = 0 \;\Rightarrow\; \hat{\boldsymbol{w}} = \sum_i \alpha_i y_i \boldsymbol{x}_i.$$

- Substitute the solution back into the objective and get (after some re-arrangements of terms):

$$\begin{aligned}
&\max_{\alpha_i \geq 0} \min_{\boldsymbol{w}} \left(\|\boldsymbol{w}\|^2/2 + \sum_{i=1}^{n} \alpha_i (1 - y_i[\boldsymbol{x}_i^t \boldsymbol{w}]) \right) \\
=&\max_{\alpha_i \geq 0} \left(\|\hat{\boldsymbol{w}}\|^2/2 + \sum_{i=1}^{n} \alpha_i (1 - y_i[\boldsymbol{x}_i^t \hat{\boldsymbol{w}}]) \right) \\
=&\max_{\alpha_i \geq 0} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^t \boldsymbol{x}_j \right)
\end{aligned}$$

SVMs: summary

- Find optimal Lagrange multipliers $\hat{\alpha}_i$ by maximizing

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \boldsymbol{x}_i^t \boldsymbol{x}_j \quad \text{subject to } \alpha_i \geq 0.$$

- Only $\hat{\alpha}_i$'s corresponding to **support vectors** will be non-zero.
- Make **predictions** on any new example $\boldsymbol{x}$ according to:

$$\text{sign}(\boldsymbol{x}^t \hat{\boldsymbol{w}}) = \text{sign}(\boldsymbol{x}^t \sum_{i=1}^{n} \hat{\alpha}_i y_i \boldsymbol{x}_i) = \text{sign}(\sum_{i \in SV} \hat{\alpha}_i y_i \boldsymbol{x}^t \boldsymbol{x}_i).$$

- Observation: dependency on input vectors only via **dot products.**
- Later we will introduce the **kernel trick** for efficiently computing these dot products in implicitly defined feature spaces.

SVMs: formal derivation

- **Convex optimization problem:** an optimization problem

$$\text{minimize} \quad f(\boldsymbol{x}) \tag{1}$$

$$\text{subject to} \quad g_i(\boldsymbol{x}) \leq 0, \quad i = 1, \ldots, m \tag{2}$$

  is convex if the functions $f, g_1 \ldots g_m : \mathbb{R}^n \to \mathbb{R}$ are convex.

- The **Lagrangian function** for the problem is

$$\mathcal{L}(\boldsymbol{x}, \lambda_0, ..., \lambda_m) = \lambda_0 f(\boldsymbol{x}) + \lambda_1 g_1(\boldsymbol{x}) + ... + \lambda_m g_m(\boldsymbol{x}).$$

- **Karush-Kuhn-Tucker (KKT) conditions:** For each point $\hat{\boldsymbol{x}}$ that minimizes f, there exist real numbers $\lambda_0, \ldots, \lambda_m$, called **Lagrange multipliers**, that simultaneously satisfy:
  1. $\hat{\boldsymbol{x}}$ minimizes $\mathcal{L}(\boldsymbol{x}, \lambda_0, \lambda_1, \ldots, \lambda_m)$,
  2. $\lambda_0 \geq 0, \lambda_1 \geq 0, \ldots, \lambda_m \geq 0$, with at least one $\lambda_k > 0$,
  3. Complementary slackness: $g_i(\hat{\boldsymbol{x}}) < 0 \Rightarrow \lambda_i = 0 \;, 1 \leq i \leq m$.

SVMs: formal derivation

- **Slater's condition:** If there exists a **strictly feasible point** z satisfying $g_1(z) < 0, \dots, g_m(z) < 0$, then one can set $\lambda_0 = 1$.
- Assume that Slater's condition holds. Minimizing the supremum $\mathcal{L}^*(\boldsymbol{x}) = \sup_{\lambda \geq 0} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$, is the **primal problem P:**

$$\hat{\boldsymbol{x}} = \operatorname*{argmin}_{\boldsymbol{x}} \mathcal{L}^*(\boldsymbol{x}).$$

Note that

$$\mathcal{L}^*(\boldsymbol{x}) = \sup_{\lambda \geq 0} \left(f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \right) = \begin{cases} f(\boldsymbol{x}) & \text{, if } g_i(\boldsymbol{x}) \leq 0 \, \forall i \\ \infty & \text{, else.} \end{cases}$$

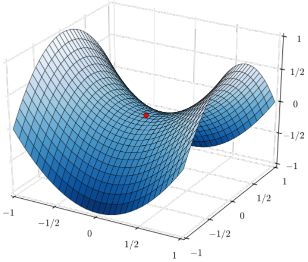
$\rightsquigarrow$ **Minimizing $\mathcal{L}^*(\boldsymbol{x})$ is equivalent to minimizing $f(\boldsymbol{x})$.**

- The maximizer of the **dual problem D** is

$$\hat{\boldsymbol{\lambda}} = \operatorname*{argmax}_{\boldsymbol{\lambda}} \mathcal{L}_*(\boldsymbol{\lambda}), \text{ where } \mathcal{L}_*(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}).$$

SVMs: formal derivation

- The non-negative number min(P) – max(D) is the **duality gap.**
- **Convexity and Slater's condition imply strong duality:**
  1. The optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\lambda}})$ is a saddle point of $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$
  2. The **duality gap is zero.**
- Discussion: For any real function $f(a, b)$
  $\min_a[\max_b f(a,b)] \geq \max_b[\min_a f(a,b)]$.
  Equality $\rightsquigarrow$ saddle value exists.



Kernel functions

- A **kernel function** is a real-valued function of two arguments, $k(\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}$, for $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$.
- Typically the function is **symmetric**, and sometimes non-negative.
- In the latter case, it might be interpreted as a measure of similarity.
- Example: **isotropic Gaussian kernel:**

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

  Here, σ^2 is the bandwidth. This is an example of a **radial basis function (RBF) kernel** (only a function of $\|\boldsymbol{x} - \boldsymbol{x}'\|^2$).

Mercer kernels

- A symmetric kernel is a **Mercer kernel**, iff the Gram matrix

$$K = \begin{pmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ & \vdots & \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \dots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{pmatrix}$$

  is **positive semidefinite** for any set of inputs $\{\boldsymbol{x}_i, \dots, \boldsymbol{x}_n\}$.
- **Mercer's theorem:** Eigenvector decomposition

$$K = V \Lambda V^t = (V \Lambda^{1/2})(V \Lambda^{1/2})^t =: \Phi \Phi^t.$$

  Eigenvectors: columns of V. Eigenvalues: entries of diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$. Note that $\lambda_i \in \mathbb{R}$ and $\lambda_i \geq 0$.
  Define $\phi(\boldsymbol{x}_i)^t = i$-th row of $\Phi = V_{[i\bullet]} \Lambda^{1/2}$
  $\rightsquigarrow k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_{i'})$.
- Entries of K: **inner product of some feature vectors**, implicitly defined by eigenvectors V.

Mercer kernels

- If the kernel is **Mercer**, then there exists $\phi : \boldsymbol{x} \to \mathbb{R}^d$ such that
  $$k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^t \phi(\boldsymbol{x}'),$$
  where ϕ depends on the eigenfunctions of k (d might be infinite).
- Example: **Polynomial kernel**
  $$k(\boldsymbol{x}, \boldsymbol{x}') = (1 + \boldsymbol{x}^t \boldsymbol{x}')^m.$$
  Corresponding feature vector contains terms up to degree m.
  Example: $m = 2,\ x \in \mathbb{R}^2$:
  $$(1 + \boldsymbol{x}^t \boldsymbol{x}')^2 = 1 + 2x_1x_1' + 2x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2 + 2x_1x_1'x_2x_2'.$$
  Thus,
  $$\phi(\boldsymbol{x}) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^t.$$
  Equivalent to working in a 6-dim feature space.
- **Gaussian kernel:** feature map lives in an infinite dimensional space.

Kernels for documents

- In document classification or retrieval, we want to compare two documents, $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$.
- **Bag of words** representation:
  $\boldsymbol{x}_{ij}$ is the number of times word j occurs in document i.
- One possible choice: **Cosine similarity:**

$$k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \frac{\boldsymbol{x}_i^t \boldsymbol{x}_{i'}}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_{i'}\|} =: \phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_{i'}).$$

- Problems:
  - Popular words (like "the" or "and") are not discriminative $\rightsquigarrow$ remove these **stop words**.
  - Bias: once a word is used in a document, it is very likely to be **used again.**
- Solution: Replace word counts with "normalized" representation.

Kernels for documents

- TF-IDF "term frequency inverse document frequency":
  Term frequency is log-transform of the count:
  $$\text{tf}(x_{ij}) = \log(1 + x_{ij})$$
  Inverse document frequency:
  $$\text{idf}(j) = \log \frac{\#(\text{documents})}{\#(\text{documents containing term } j)} = \log \frac{1}{\hat{p}_j}.$$
  $\rightsquigarrow$ Shannon information content:
  idf is a measure of how much information a word provides
- Combine with tf $\rightsquigarrow$ counts weighted by information content:
  $$\text{tf-idf}(\boldsymbol{x}_i) = [\text{tf}(\boldsymbol{x}_{ij}) \cdot \text{idf}(j)]_{j=1}^{V}, \quad \text{where } V = \text{size of vocabulary.}$$
- We then use this inside the **cosine similarity measure.**
  With $\phi(\boldsymbol{x}) = \text{tf-idf}(\boldsymbol{x})$:
  $$k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \frac{\phi(\boldsymbol{x}_i)^t \phi(\boldsymbol{x}_{i'})}{\|\phi(\boldsymbol{x}_i)\| \|\phi(\boldsymbol{x}_{i'})\|}.$$

String kernels

- Real power of kernels arises for **structured input objects.**
- Consider two strings x, and x' of lengths d, d', over alphabet $\mathcal{A}$.
  Idea: define similarity as the **number of common substrings.**
- If s is a substring of $x \rightsquigarrow \phi_s(x) =$ number of times s appears in x.
- **String kernel**

$$k(x, x') = \sum_{s \in \mathcal{A}^*} w_s \phi_s(x) \phi_s(x'),$$

  where $w_s \geq 0$ and $\mathcal{A}^* =$ set of all strings (any length) from $\mathcal{A}$.
- One can show: Mercer kernel, can be computed in $O(|x| + |x'|)$ time using suffix trees (Shawe-Taylor and Cristianini, 2004).
- Special case: $w_s = 0$ for $|s| > 1$: **bag-of-characters kernel:**
  $\phi(x)$ is the number of times each character in $\mathcal{A}$ occurs in x.

The kernel trick

- Idea: modify algorithm so that it **replaces all inner products $\boldsymbol{x}^t\boldsymbol{x}'$** with a call to the **kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$.**
- **Kernelized ridge regression:** $\hat{\boldsymbol{w}} = (X^tX + \lambda I)^{-1}X^t\boldsymbol{y}$.
  Matrix inversion lemma:
  $$(I + UV)^{-1}U = U(I + VU)^{-1}$$
  Define new variables α_i:
  $$\hat{\boldsymbol{w}} = (X^tX + \lambda I)^{-1}X^t\boldsymbol{y}$$
  $$= X^t\underbrace{(XX^t + \lambda I)^{-1}\boldsymbol{y}}_{\hat{\boldsymbol{\alpha}}} = \sum_{i=1}^{n} \hat{\alpha}_i \boldsymbol{x}_i.$$
  $\rightsquigarrow$ **solution is linear sum of the n training vectors.**

The kernel trick

- Use this and the kernel trick to make **predictions for $\boldsymbol{x}$:**

$$\hat{f}(\boldsymbol{x}) = \hat{\boldsymbol{w}}^t \boldsymbol{x} = \sum_{i=1}^{n} \hat{\alpha}_i \boldsymbol{x}_i^t \boldsymbol{x} = \sum_{i=1}^{n} \hat{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x}).$$

- Same for SVMs:

$$\hat{\boldsymbol{w}}^t \boldsymbol{x} = \sum_{i \in SV} \hat{\alpha}_i y_i \boldsymbol{x}_i^t \boldsymbol{x} = \sum_{i \in SV} \hat{\alpha}'_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

- ...and for most other classical algorithms in ML!

Some applications in bioinformatics

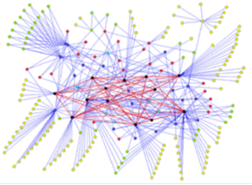
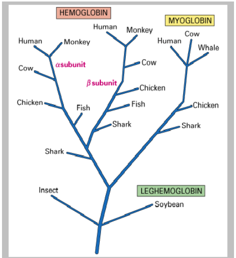
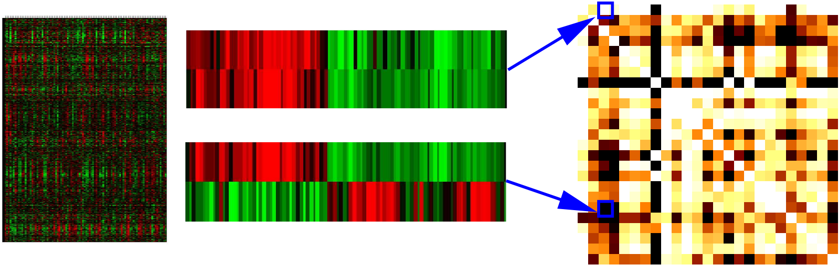
- Bioinformatics: often **non-vectorial** data-types:
  - interaction graphs 
  - phylogenetic trees 

Figure 1: Diagram of Globin Family Tree (Lodish et. al.,2000)
  - strings GSAQVKGHGKKVADALTNAVAHV
- **Data fusion:** convert data of each type into kernel matrix
  $\Rightarrow$ **fuse kernel matrices**
  $\Rightarrow$ **"common language" for heterogeneous data.**

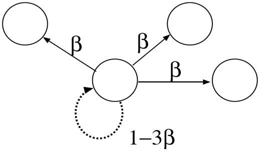
RBF kernels from expression data

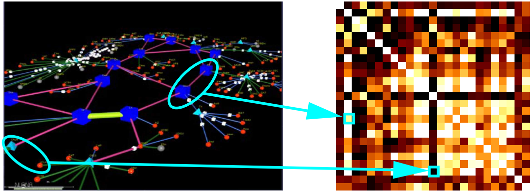
- **Measurements** (for each gene): vector of expression values under different experimental conditions
- "classical" RBF kernel $k(x_1, x_2) = \exp(-\sigma \|x_1 - x_2\|^2)$



Diffusion kernels from interaction-graphs

- A: Adjacency matrix, D: node degrees, $L = D - A$.
- $K := \frac{1}{Z(\beta)} \exp(-\beta L)$ with transition probabilities β.
- **Physical interpretation (random walk):**
  randomly choose next node among neighbors.
- Self-transition occurs with prob. $1 - d_i\beta$





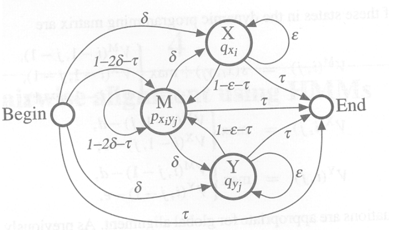
- K_{ij}: prob. for walk from i to j.

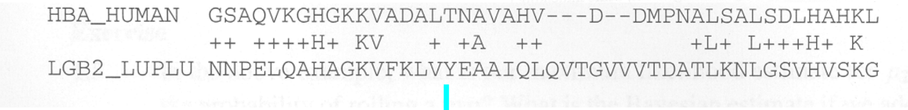
(Kondor and Lafferty, 2002)

Alignment kernels from sequences

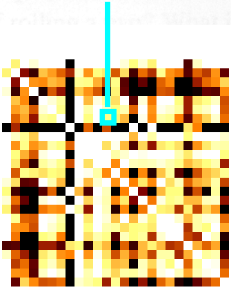
Alignment with **Pair HMMs**
⇝ Mercer kernel (Watkins, 2000).

Image source: Durbin, Eddy, Krogh, Mitchison. Biological Sequence Alignment. Cambridge.





```
HBA_HUMAN    GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
             ++  ++++H+  KV     +  +A   ++           +L+  L+++H+  K
LGB2_LUPLU   NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

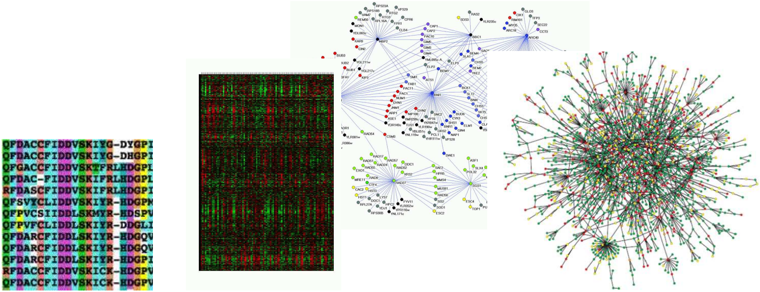


Combination of heterogeneous data

Adding kernels $\Rightarrow$ new kernel:

$$k_1(x,y) = \phi_1(x) \cdot \phi_1(y),$$
$$k_2(x,y) = \phi_2(x) \cdot \phi_2(y) \qquad \Rightarrow k' = k_1 + k_2 = \begin{pmatrix}\phi_1(x)\\ \phi_2(x)\end{pmatrix} \cdot \begin{pmatrix}\phi_1(y)\\ \phi_2(y)\end{pmatrix}$$

Fusion & relevance determination: kernel-combinations



$$\mathbf{K} = c_1\,\mathbf{K_1} + c_2\,\mathbf{K_2} + c_3\,\mathbf{K_3} + c_4\,\mathbf{K_4}$$