Lecture Mon 21.11.

CLASSIFICATION & SUPPORT VECTOR MACHINES

LECTURE MATERIAL COURTESY OF
GUNNAR RÄTSCH GUNNAR.RAETSCH(AT)TUEBINGEN.MPG.DE
SÖREN SONNENBURG SOEREN.SONNENBURG(AT)TOMTOM.COM

Classification Problems in bioinformatics

Sequence classification:

Given: DNA sequence

Predict: Does sequence belong to an CpG island or not

Diagnostic models:

Given: Expression levels of genes from a sample

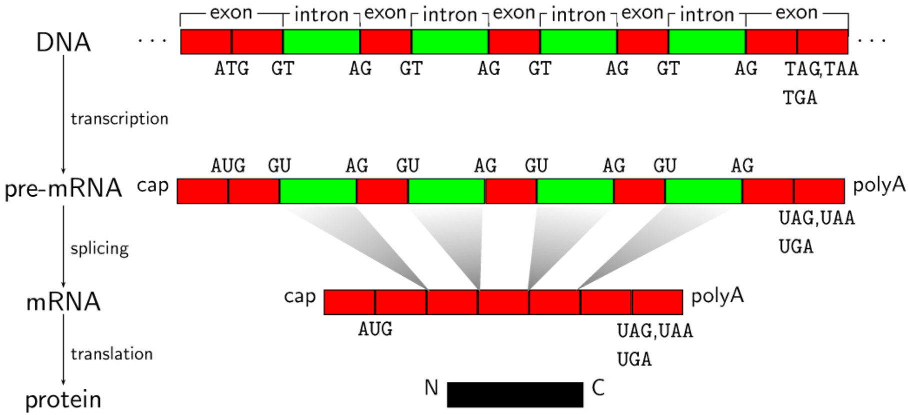
Predict: Diseased or healthy

Functional genomics:

Given: Sequence of a gene

Predict: The biological function of the gene (e.g. Gene Ontology category)

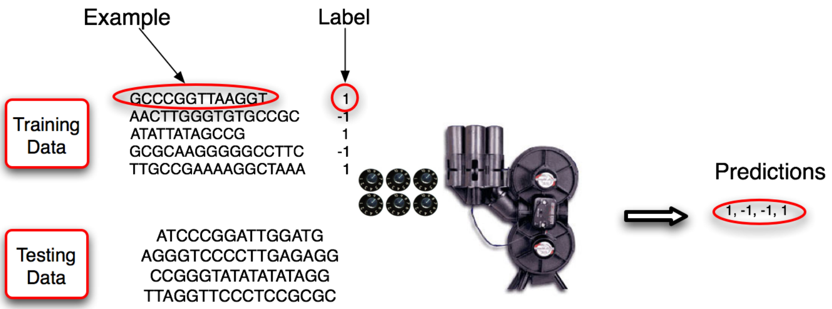
Running example: Splice site recognition



- Almost all *donor splice sites* exhibit `GU`
- Almost all *acceptor splice site* exhibit `AG`
- Not all `GU`s and `AG`s are used as splice site ⇒ Classification task

Classification setup

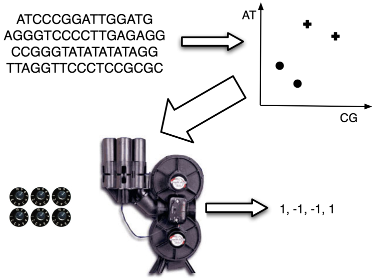
- Training data from real (label +1) and decoy (label -1) acceptor sites
  - All training data contain 'AG' dinucleotide
- Discrminative model is learned from training data
- Testing on an independent test set



Classification learning with feature vectors

We will concentrate on methods that rely on numerical feature representation for the data

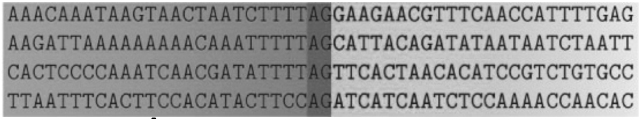
- Each example is a vector of values (features).
- If the example is not a vector, a feature representation needs to be first computed
- What are good features to extract? Requires background knowledge on the application domain.



Classification learning with feature vectors

Possible features in the acceptor site recognition problem

- GC content in a window before 'AG'
- GC content in a window after 'AG'
- Occurrence of specific subsequences 'TTTAG'



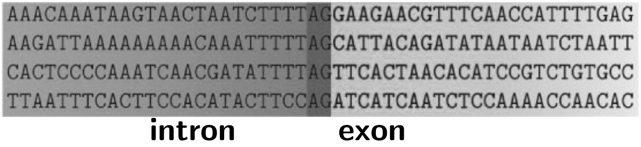
```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$	$\mathbf{x}_7$	$\mathbf{x}_8$	$\dots$
GC before	0.6	0.2	0.4	0.3	0.2	0.4	0.5	0.5	$\dots$
GC after	0.7	0.7	0.3	0.6	0.3	0.4	0.7	0.6	$\dots$
AG**AG**AAG	0	0	0	1	1	0	0	1	$\dots$
TTT**AG**	1	1	1	0	0	1	0	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
Label	$+1$	$+1$	$+1$	-1	-1	$+1$	-1	-1	$\dots$

Recognition of Splice Sites

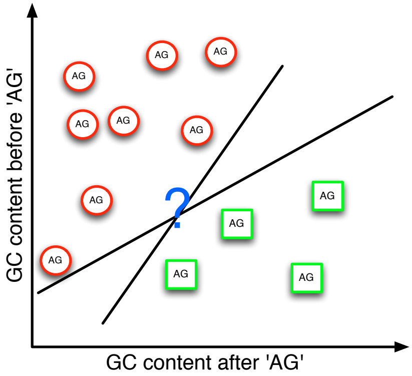
- Given: Potential acceptor splice sites



```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron
exon

- Goal: Rule that distinguishes true from false ones



exploit that exons have higher GC content

or

that certain motifs are located nearby

Classification learning with linear models

We will concentrate on models that take a linear form:

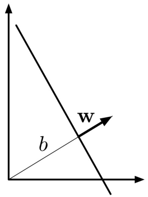
$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j + b$$

- x_j is the value of the j'th feature for example $\mathbf{x}$, e.g. 'GC content before'
- w_j is the weight of the j'th feature, to be learned from the data
- b is an offset term, to be learned from the data

Classification learning with linear models

We will concentrate on models that take a linear form:

$$f(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j + b = \langle \mathbf{w}, \mathbf{x} \rangle + b$$



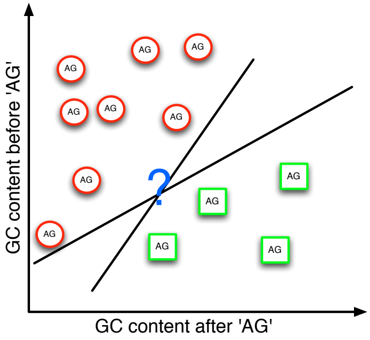
- $\langle \mathbf{w}, \mathbf{x} \rangle$ denotes the inner product (also known as dot product and scalar poduct), between the weight vector and the feature vector
- Geometrically $f(\mathbf{x})$ is a hyperplane ($d-1$-dimensional plane) dividing the feature space into two half-spaces
- w is the normal vector of the hyperplane, orthogonal to the hyperplane
- Values of $f(\mathbf{x})$ increase in the direction of w

Classification learning with linear models

The model $f(\mathbf{x})$ is turned into a classifier by thresholding at 0:

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } f(\mathbf{x}) > 0 \\ -1 & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

- The goal of learning the parameters $(\mathbf{w}, b)$ is to put the hyperplane in between the two classes
  - $h(x) = -1 \Leftrightarrow f(\mathbf{x}) < 0$ for the negative class
  - $h(x) = +1 \Leftrightarrow f(\mathbf{x}) > 0$ for the positive class

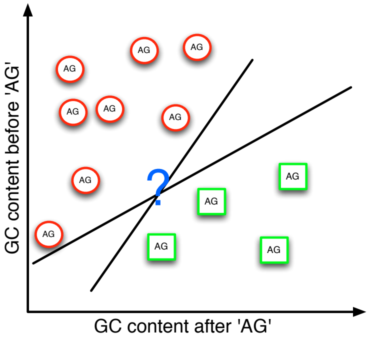


Measuring classification success: loss function

In binary classification ($\mathcal{Y} = \{-1, +1\}$), we one may use the 0/1-*loss function*:

$$\ell(f(\mathbf{x}_i), y_i) = \begin{cases} 0 & \text{if } h(x_i) = y_i \\ 1 & \text{if } h(x_i) \neq y_i \end{cases}$$

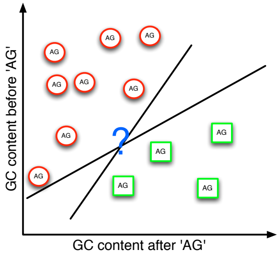
- If all training data are on the correct side of the hyperplane we have $\sum_{i=1}^{n} \ell(f(x_i), y_i) = 0$
- However, there might be several hyperplanes that achieve zero loss
- Does it matter which one we choose?

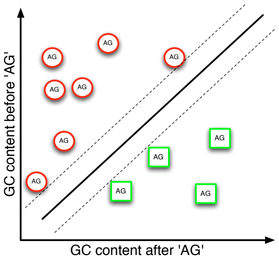


Maximum margin hyperplane

One good solution is to choose the hyperplane that lies furthest away from the training data:

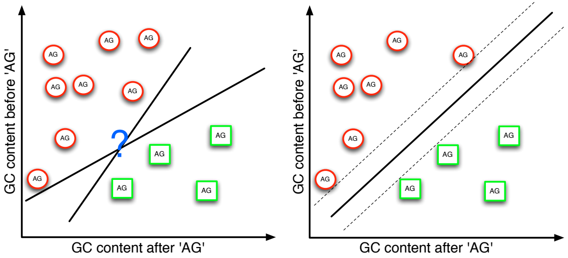
- Robustness: small change in the training data will not change the classifications too much
- Theoretically can be shown to lead to good performance — a large margin, distance between the hyperplane, is tied to low error on unseen data
- Support vector machines (SVM) are based on this principle





How to Maximize the Margin?

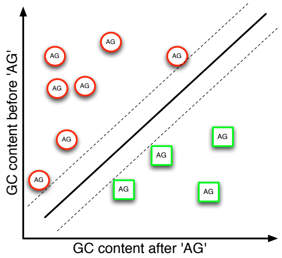
- For positive class the margin is given by $\gamma(x_i) = y_i f(x_i) = y(< \mathbf{w}, \mathbf{x} > +b)$
- By multiplying the weights with a arbitrary $c > 1$, one can increase the margin without limit $y(< c\mathbf{w}, \mathbf{x} > +b) = c \cdot y(\mathbf{w}, \mathbf{x} > +b) = c \cdot \gamma(x_i)$
- Any separating hyperplane can be made to have as large margin as we wish, cannot choose between them by taking the maximum!



How to Maximize the Margin?

Margin maximization becomes sensible if we add a constraint that the length of the weight vector should not change: $||\mathbf{w}|| = \sqrt{\sum_{i=1}^{m} w_j^2} = 1$.
Our optimization problem becomes:

$$\begin{array}{rc} \textit{Maximize} & \gamma \\ \textit{Subject to} & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geqslant \gamma \\ & \text{for all } i = 1, \ldots, n, \\ & ||\mathbf{w}|| = 1 \end{array}$$

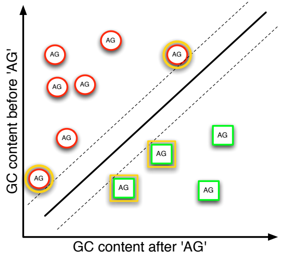


- First constraint says that all examples have at least margin of γ
- Second constraint fixes the norm of the weight vector — intuitively, gives fixed measurement scale
- Now there will be a maximum margin hyperplane

How to Maximize the Margin?

It turns out that we can equivalently fix the margin $\gamma = 1$ and seek for shortest weight vector that achieves the margin

$$\begin{array}{rl} \textit{Minimize} & \frac{1}{2}||\mathbf{w}||^2 \\ \textit{Subject to} & y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geqslant 1 \\ & \text{for all } i = 1, \ldots, n. \end{array}$$

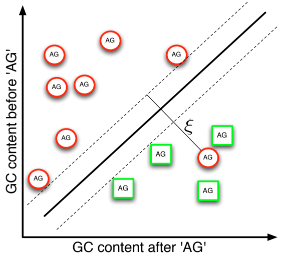


- All points have at least margin of $= 1$
- The points that have margin $\gamma(x) = 1$ are called *support vectors*
- The set of support vectors uniquely identifies the hyperplane

How to Maximize the Margin? Non-separable data

In practise data rarely separates cleanly into two halfspaces by a hyperplane, for multitude of reasons:

- Measurement errors
- Insufficient features
- Annotation errors

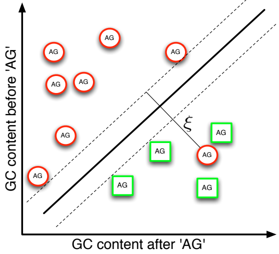


- For any hyperplane, there will be an example with a negative margin
- Our optimization problem has no feasible solution

Soft-Margin SVM (Cortes & Vapnik, 1995)

The soft-margin SVM allows some of the training points to have smaller margin than $\gamma(x) = 1$, subject to a penalty:

$$\begin{aligned} &\textit{Minimize} && \tfrac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \\ &\textit{Subject to} && y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \\ & && \text{for all } i = 1, \ldots, n. \\ & && \xi_i \geq 0 \end{aligned}$$



- ξ_i is called the slack variable, when positive the margin $\gamma(x_i) < 1$
- The sum of slacks is to be minimized so the objective still favours hyperplanes that separates the classes well
- The coefficient $C > 0$ controls the balance between maximizing the margin and the amount of slack needed

An important detail (I)

$$\begin{aligned} \underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \text{ for all } i = 1, \ldots, n. \\ & \xi_i \geqslant 0 \text{ for all } i = 1, \ldots, n \end{aligned}$$

Theorem: The optimal **w** can be written as a linear combination of the examples (for appropriate α's):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

An important detail (II)

$$\underset{\mathbf{w},b,\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geqslant 1 - \xi_i \text{ for all } i = 1, \ldots, n.$$
$$\xi_i \geqslant 0 \text{ for all } i = 1, \ldots, n$$

Theorem: The optimal **w** can be written as a linear combination of the examples (for appropriate α's):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \qquad \Rightarrow \text{Plug in!}$$

An important detail (III)

$$\underset{\boldsymbol{\alpha},b,\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\left\|\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\right\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i\left(\textstyle\sum_{j=1}^{N}\alpha_j y_j \langle \mathbf{x}_j, \mathbf{x}_i\rangle + b\right) \geqslant 1 - \xi_i \text{ for all } i = 1,\ldots,n.$$

$$\xi_i \geqslant 0 \text{ for all } i = 1,\ldots,n$$

Theorem: The optimal **w** can be written as a linear combination of the examples (for appropriate α's):

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \qquad \Rightarrow \text{Plug in!}$$

Now optimize for the variables $\boldsymbol{\alpha}$, b, and $\boldsymbol{\xi}$!

An important detail (IV)

$$\underset{\boldsymbol{\alpha},b,\boldsymbol{\xi}}{\text{minimize}} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\langle \mathbf{x}_i,\mathbf{x}_j\rangle + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to} \quad y_i\left(\sum_{j=1}^{N}\alpha_j y_j\langle \mathbf{x}_j,\mathbf{x}_i\rangle + b\right) \geqslant 1-\xi_i \text{ for all } i=1,\ldots,n.$$
$$\xi_i \geqslant 0 \text{ for all } i=1,\ldots,n$$

Theorem: The optimal $\mathbf{w}$ can be written as a linear combination of the examples (for appropriate α's):

$$\mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \qquad \Rightarrow \text{Plug in!}$$

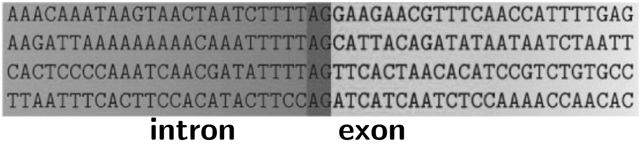
Now optimize for the variables $\boldsymbol{\alpha}$, b, and $\boldsymbol{\xi}$!

Corollary: Optimization problem only depends on the inner products of the examples

$$\langle \mathbf{x},\hat{\mathbf{x}}\rangle = \sum_{d=1}^{D} x_d \hat{x}_d$$

Recognition of Splice Sites

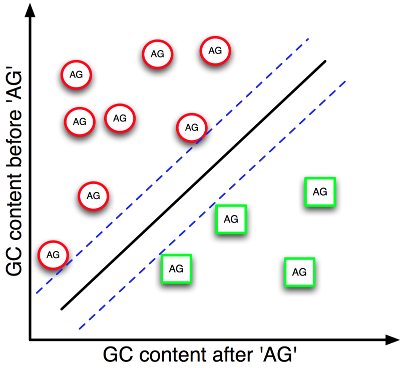
- Given: Potential acceptor splice sites



```
AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC
```

intron **exon**

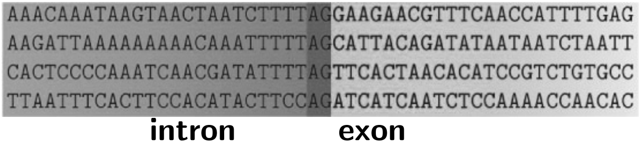
- Goal: Rule that distinguishes true from false ones



Linear Classifiers
with large **margin**

Recognition of Splice Sites

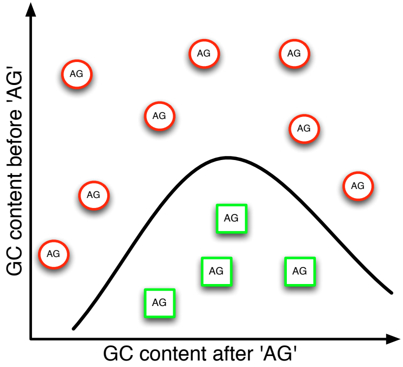
- Given: Potential acceptor splice sites



AAACAAATAAGTAACTAATCTTTTAGGAAGAACGTTTCAACCATTTTGAG
AAGATTAAAAAAAAACAAATTTTTAGCATTACAGATATAATAATCTAATT
CACTCCCCAAATCAACGATATTTTAGTTCACTAACACATCCGTCTGTGCC
TTAATTTCACTTCCACATACTTCCAGATCATCAATCTCCAAAACCAACAC

intron
exon

- Goal: Rule that distinguishes true from false ones



More realistic problem?

- Not linearly separable!
- **Need nonlinear separation?**
- Need more features?

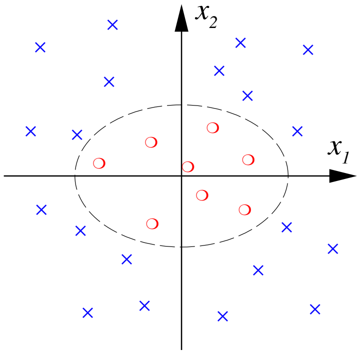
Nonlinear Separations

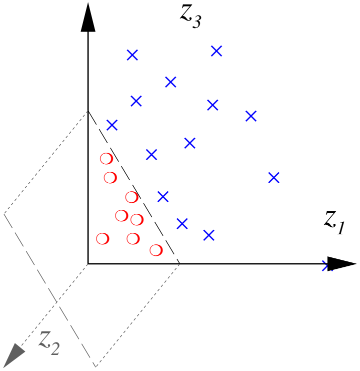
Linear separation might not be sufficient!

$\Rightarrow$ Map into a higher dimensional feature space

Example: all pairwise products of features

$$\begin{aligned}\Phi : \mathbb{R}^2 &\to \mathbb{R}^3 \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)\end{aligned}$$





Kernel "Trick"

Example: $\mathbf{x} \in \mathbb{R}^2$ and $\Phi(\mathbf{x}) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$

$$\begin{aligned}
\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle &= \Big\langle (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2), (\hat{x}_1^2, \sqrt{2}\, \hat{x}_1 \hat{x}_2, \hat{x}_2^2) \Big\rangle \\
&= \langle (x_1, x_2), (\hat{x}_1, \hat{x}_2) \rangle^2 \\
&= \langle \mathbf{x}, \hat{\mathbf{x}} \rangle^2 \\
&=: \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}})
\end{aligned}$$

- Inner product in feature space (here $\mathbb{R}^3$) can be computed in input space (here $\mathbb{R}^2$)!
- Also works for higher orders and dimensions
  $\Rightarrow$ relatively low-dimensional input spaces
  $\Rightarrow$ very high-dimensional feature spaces

Putting Things Together . . .

- Use a non-linear map $\Phi(\mathbf{x})$ instead of original features $\mathbf{x}$
- Use linear classifier on the $\Phi(\mathbf{x})$’s
- From theorem: $\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i)$.
- Non-linear :

$$\begin{aligned} f(\mathbf{x}) &= \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle + b \\ &= \sum_{i=1}^{n} \alpha_i y_i \underbrace{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle}_{\mathrm{k}(\mathbf{x}_i, \mathbf{x})} + b \end{aligned}$$

- Trick: $\mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) = \langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle$, i.e. **do not use Φ, but k!**

Kernel $\approx$ Similarity Measure

Distance:

$$\|\Phi(\mathbf{x}) - \Phi(\hat{\mathbf{x}})\|^2 = \|\Phi(\mathbf{x})\|^2 - 2\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}})\rangle + \|\Phi(\hat{\mathbf{x}})\|^2$$

Inner product: $\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}})\rangle$

- If $\|\Phi(\mathbf{x})\|^2 = \|\Phi(\hat{\mathbf{x}})\|^2 = 1$, then

$$\text{inner product} = 2 - \text{distance}$$

- Angle between vectors, i.e.,

$$\frac{\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}})\rangle}{\|\Phi(\mathbf{x})\| \; \|\Phi(\hat{\mathbf{x}})\|} = \cos(\Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}))$$

How to Construct a Kernel

At least two ways to get to a kernel:

1. Construct Φ and think about efficient ways to compute the inner product $\langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle$
   - If $\mathbf{x}$ is very high-dimensional, computing the inner product element by element is slow, we don't want to do that
2. Construct similarity measure and show that it qualifies as a kernel (Mercer condition)
   - Show that for any set of examples the matrix $K = (k(x_i, x_j))_{i,j=1}^{n}$ is positive semi-definite.
   - In that case, there always is an underlying feature representation, for which the kernel represents the inner product

Common Kernels

$$\begin{array}{rrcl}
\text{Polynomial} & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) & = & (\langle \mathbf{x}, \hat{\mathbf{x}} \rangle + c)^d \\
\text{Sigmoid} & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) & = & \tanh(\kappa \langle \mathbf{x}, \hat{\mathbf{x}} \rangle) + \theta) \\
\text{RBF} & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) & = & \exp\left(-\|\mathbf{x} - \hat{\mathbf{x}}\|^2 / (2\,\sigma^2)\right) \\
\text{Convex combinations} & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) & = & \beta_1 \mathrm{k}_1(\mathbf{x}, \hat{\mathbf{x}}) + \beta_2 \mathrm{k}_2(\mathbf{x}, \hat{\mathbf{x}}) \\
\text{Normalization} & \mathrm{k}(\mathbf{x}, \hat{\mathbf{x}}) & = & \dfrac{\mathrm{k}'(\mathbf{x}, \hat{\mathbf{x}})}{\sqrt{\mathrm{k}'(\mathbf{x}, \mathbf{x}) \mathrm{k}'(\hat{\mathbf{x}}, \hat{\mathbf{x}})}}
\end{array}$$

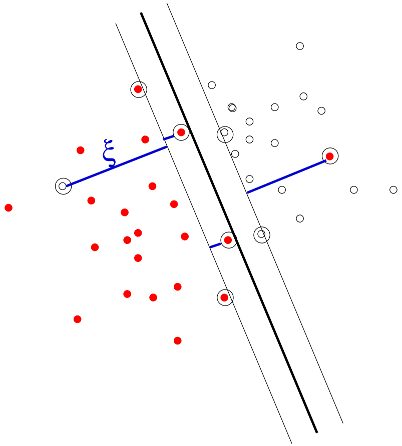
Notes:

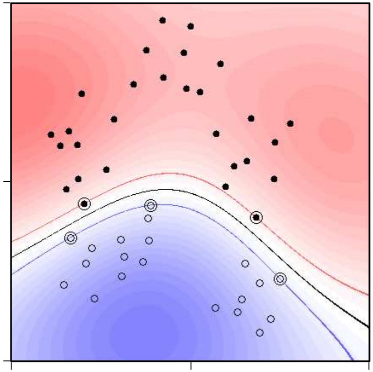
- Kernels may be combined in case of heterogeneous data
- These kernels are good for real-valued examples
- Sequences need special care (coming soon!)

Toy Examples

Linear kernel
$k(\mathbf{x}, \hat{\mathbf{x}}) = \langle \mathbf{x}, \hat{\mathbf{x}} \rangle$

RBF kernel
$k(\mathbf{x}, \hat{\mathbf{x}}) = \exp(-\|\mathbf{x} - \hat{\mathbf{x}}\|^2/2\sigma)$





Kernel Summary

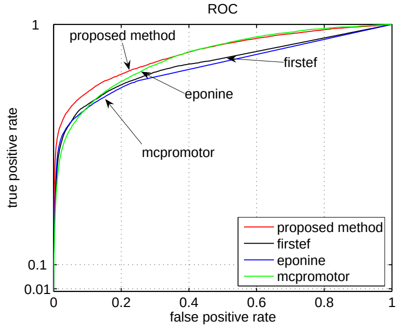
- Nonlinear separation $\Leftrightarrow$ linear separation of nonlinearly mapped examples
- Mapping Φ defines a kernel by

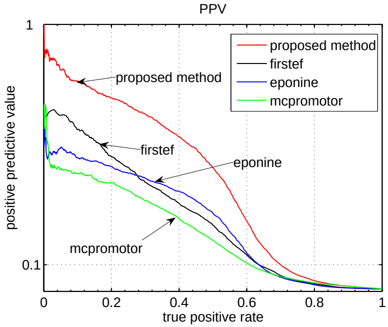
$$k(\mathbf{x}, \hat{\mathbf{x}}) := \langle \Phi(\mathbf{x}), \Phi(\hat{\mathbf{x}}) \rangle$$

- (Mercer) Kernel defines a mapping Φ (nontrivial)
- Choice of kernel has to match the data at hand
- RBF kernel often works pretty well

(More) Evaluation Measures for Classification

[left] Receiver Operating Characteristic (ROC) Curve **[right]** Precision Recall Curve



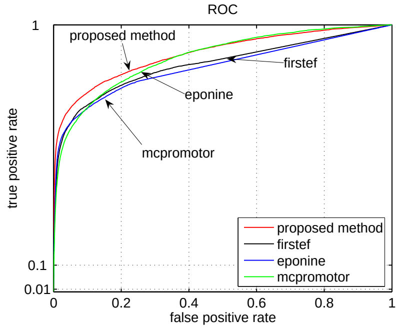


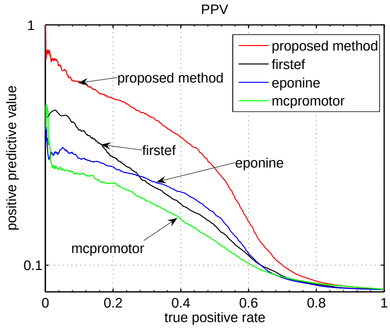
- Obtained by varying a threshold $-\infty < \tau < \infty$ and using that instead of 0: $f(\mathbf{x}) > \tau \implies +1, f(\mathbf{x}) < \tau \implies -1$
- Record TPR/FPR or Precision(PPV)/Recall(TPR) for all τ
- In practise: sort training data in the order of $f(\mathbf{x})$, and sweep over the sorted sequence once, connect the dots

(More) Evaluation Measures for Classification

[left] Receiver Operating Characteristic (ROC) Curve

[right] Precision Recall Curve





Two summarize the classifier performance over the whole curve, one typically computes:

- Area under ROC Curve (auROC)
- Area under Precision Recall Curve (auPRC)

Running example: GC-Content-based Splice Site Recognition

Kernel	auROC
Linear	88.2%
Polynomial $d = 3$	91.4%
Polynomial $d = 7$	90.4%
Gaussian $\sigma = 100$	87.9%
Gaussian $\sigma = 1$	88.6%
Gaussian $\sigma = 0.01$	77.3%

SVM accuracy of acceptor site recognition using polynomial and Gaussian kernels with different degrees d and widths σ. Accuracy is measured using the area under the ROC curve (auROC) and is computed using five-fold cross-validation