Multiple Kernel Learning

Alex Zien

Fraunhofer FIRST.IDA, Berlin, Germany
Friedrich Miescher Laboratory, Tübingen, Germany
(MPI for Biological Cybernetics, Tübingen, Germany)

09. July 2008

Summer School on Neural Networks 2008
Porto, Portugal



Friedrich Miescher Laboratory
of the Max Planck Society

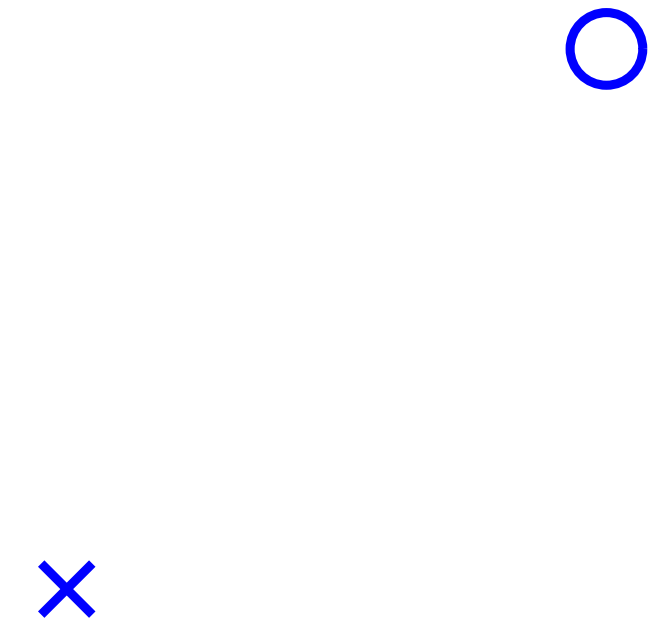


Outline

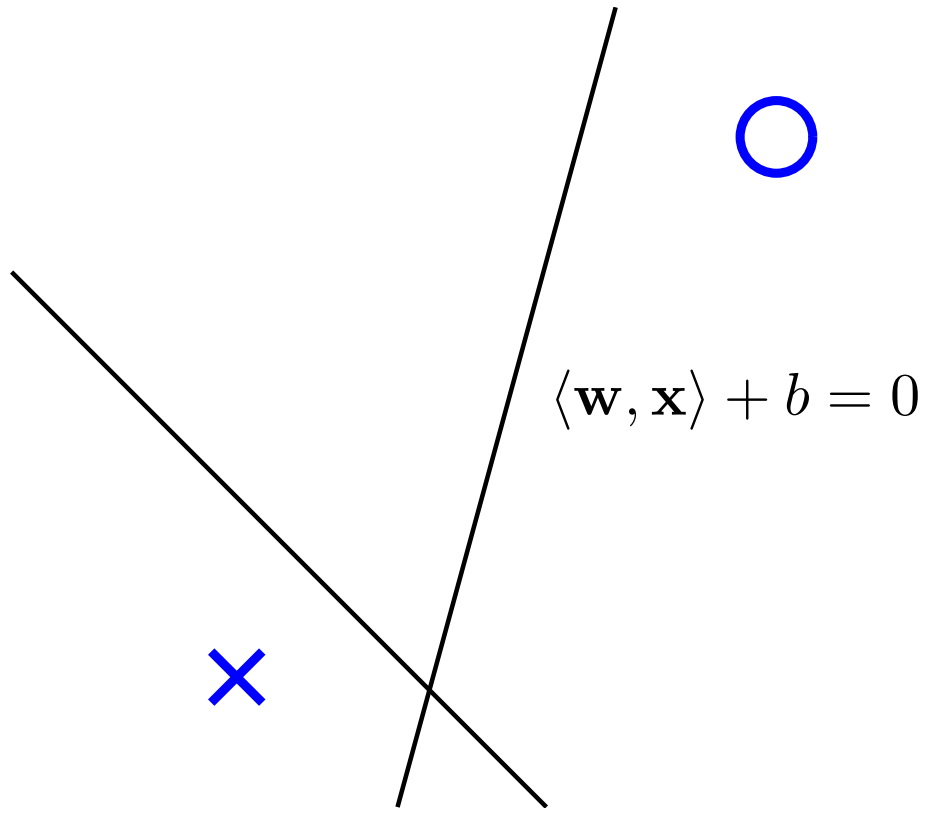
1. Recap: Support Vector Machines (SVMs)
   - SVMs Do Linear Large Margin Separation
   - Non-Linearity via Kernels
   - SVMs are Perceptrons
2. Application: Predicting Protein Subcellular Localization
3. Multiple Kernel Learning (MKL)
   - A Large Margin MKL Model
   - Optimization for MKL
   - Normalization of Kernels Is Important
   - Multiclass Multiple Kernel Learning
4. Back to App: Predicting Protein Subcellular Localization
5. Take Home Messages

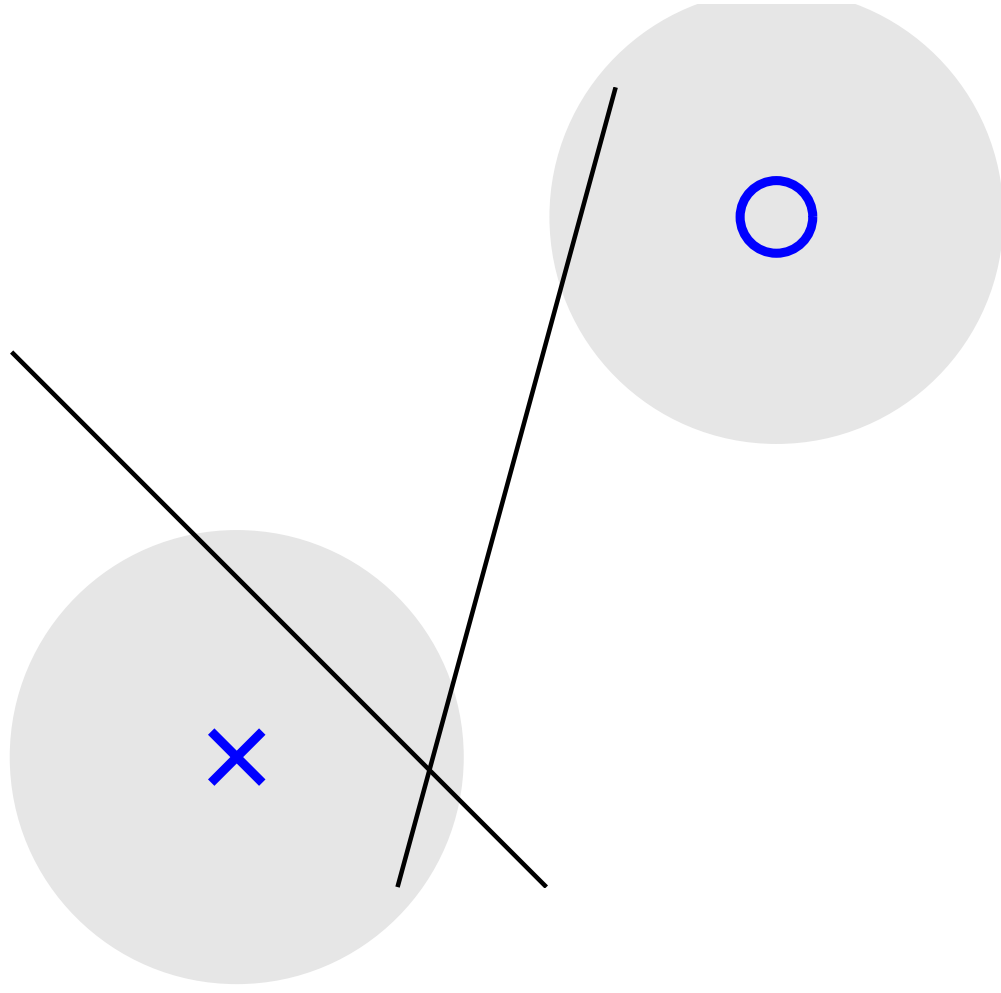
Notation

- labeled training data:
  - input data $\mathbf{x} \in \mathcal{X}$; for simplicity often $\mathcal{X} = \mathbb{R}^D$
  - labels $y \in \mathcal{Y}$; for binary classification always $\mathcal{Y} = \{-1, +1\}$
  - training data: N pairs $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$
- goal of learning:
  - function $f : \mathcal{X} \to \mathcal{Y}$
  - such that $f(\mathbf{x}) = \hat{y} \approx y$
- for linear classification: $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \left\langle \begin{pmatrix} \mathbf{w} \\ b \end{pmatrix}, \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \right\rangle$
  - hyperplane normal $\mathbf{w} \in \mathcal{X}$
  - offset $b \in \mathbb{R}$, aka bias
  - scalar product $\langle \mathbf{w}, \mathbf{x} \rangle$ often written as $\mathbf{w}^\top \mathbf{x}$



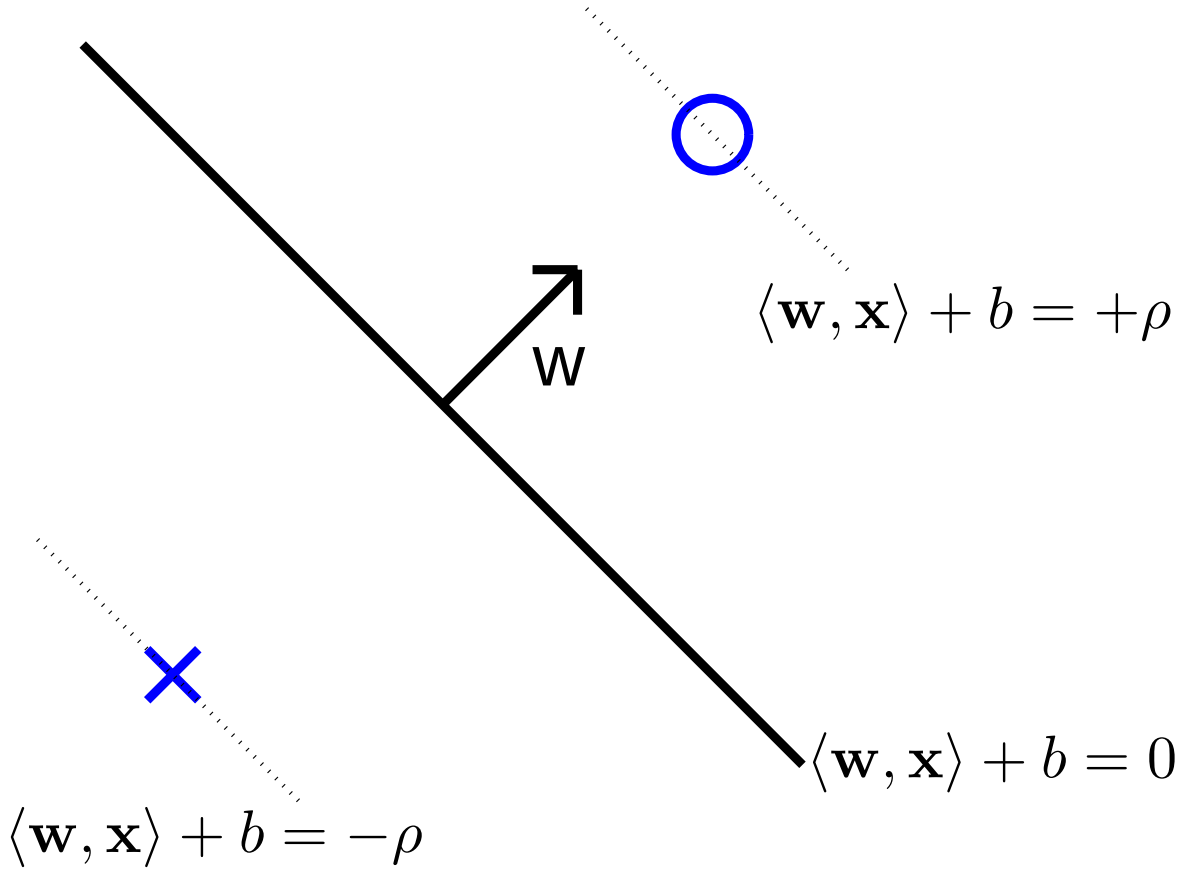
find a linear classification boundary


$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$



not robust wrt input noise!

SVM:
maximum margin
classifier



$$\max_{\mathbf{w},b,\rho} \quad \underbrace{\rho}_{\text{margin}} \quad s.t. \quad \underbrace{y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \rho}_{\text{data fitting}}, \quad \underbrace{\|\mathbf{w}\| = 1}_{\text{normalization}}$$

Equivalent reformulation of the SVM:

$$\max_{\mathbf{w},b,\rho} \quad \rho \qquad s.t. \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \rho, \quad \|\mathbf{w}\| = 1$$

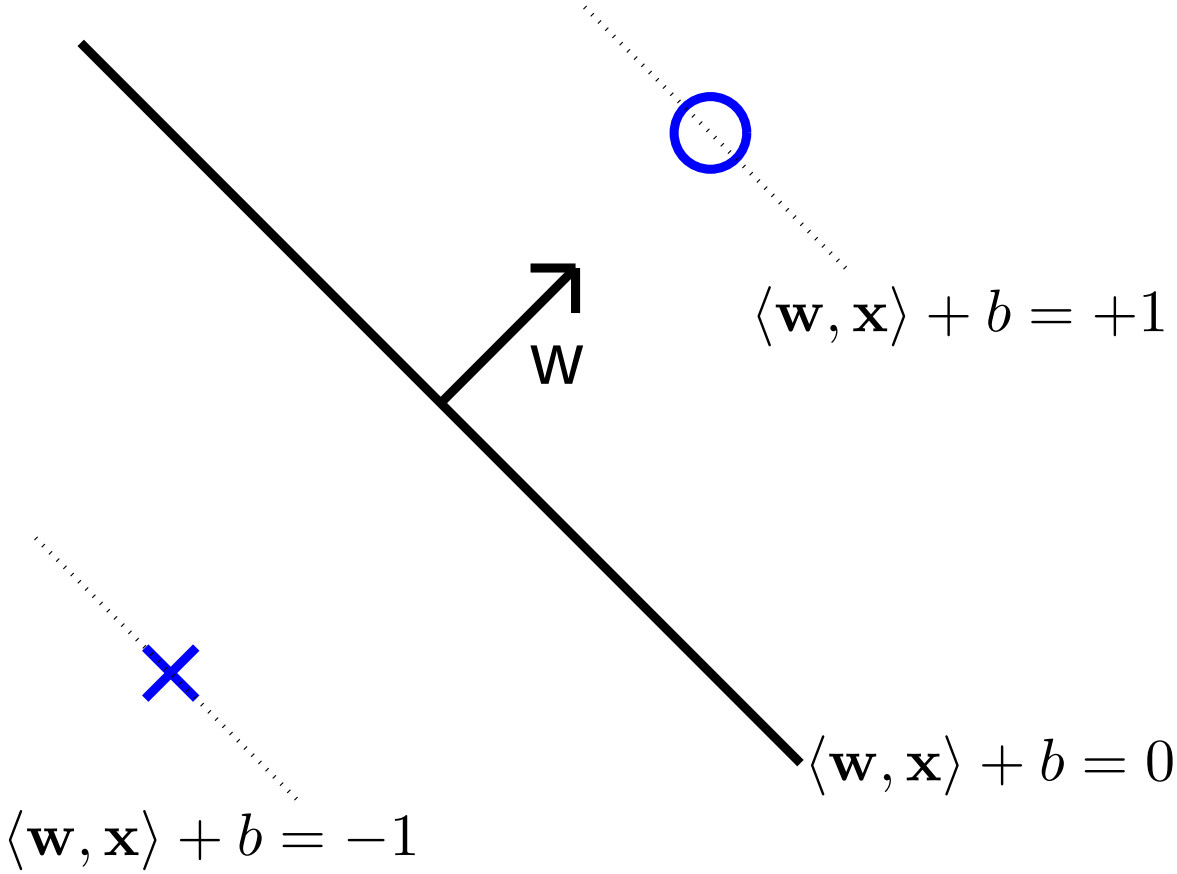
$$\Leftrightarrow \quad \max_{\mathbf{w}',b,\rho} \quad \rho^2 \qquad s.t. \quad y_i \left(\left\langle \frac{\mathbf{w}'}{\|\mathbf{w}'\|}, \mathbf{x}_i \right\rangle + b \right) \geq \rho, \quad \rho \geq 0$$

$$\Leftrightarrow \quad \max_{\mathbf{w}',b,\rho} \quad \rho^2 \qquad s.t. \quad y_i \left(\left\langle \underbrace{\frac{\mathbf{w}'}{\|\mathbf{w}'\| \rho}}_{\mathbf{w}''}, \mathbf{x}_i \right\rangle + \underbrace{\frac{b}{\rho}}_{b''} \right) \geq 1, \quad \rho \geq 0$$

$$\Leftrightarrow \quad \max_{\mathbf{w}'',b''} \quad \frac{1}{\|\mathbf{w}''\|^2} \qquad s.t. \quad y_i \left(\langle \mathbf{w}'', \mathbf{x}_i \rangle + b'' \right) \geq 1,$$

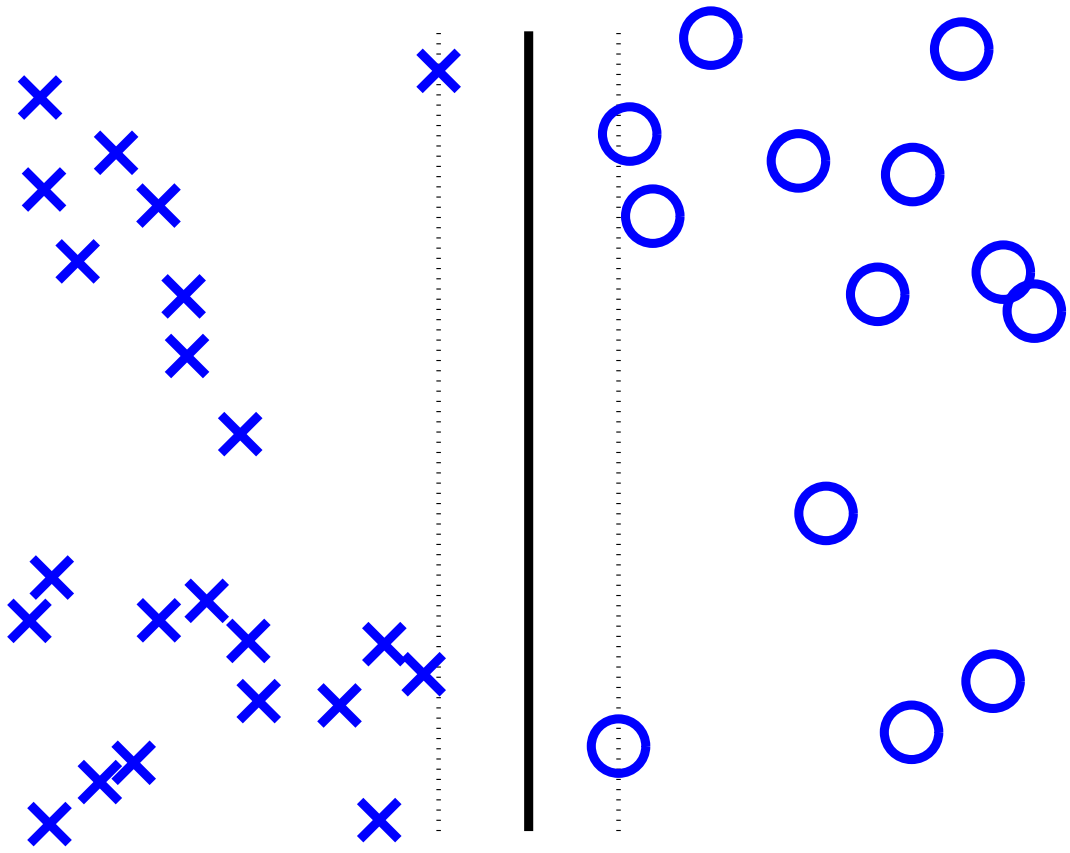
using $\left\|\mathbf{w}''\right\| = \left\| \frac{\mathbf{w}'}{\|\mathbf{w}'\| \rho} \right\| = \left| \frac{1}{\rho} \right| \cdot \left\| \frac{\mathbf{w}'}{\|\mathbf{w}'\|} \right\| = \frac{1}{\rho}$

SVM:
maximum margin
classifier



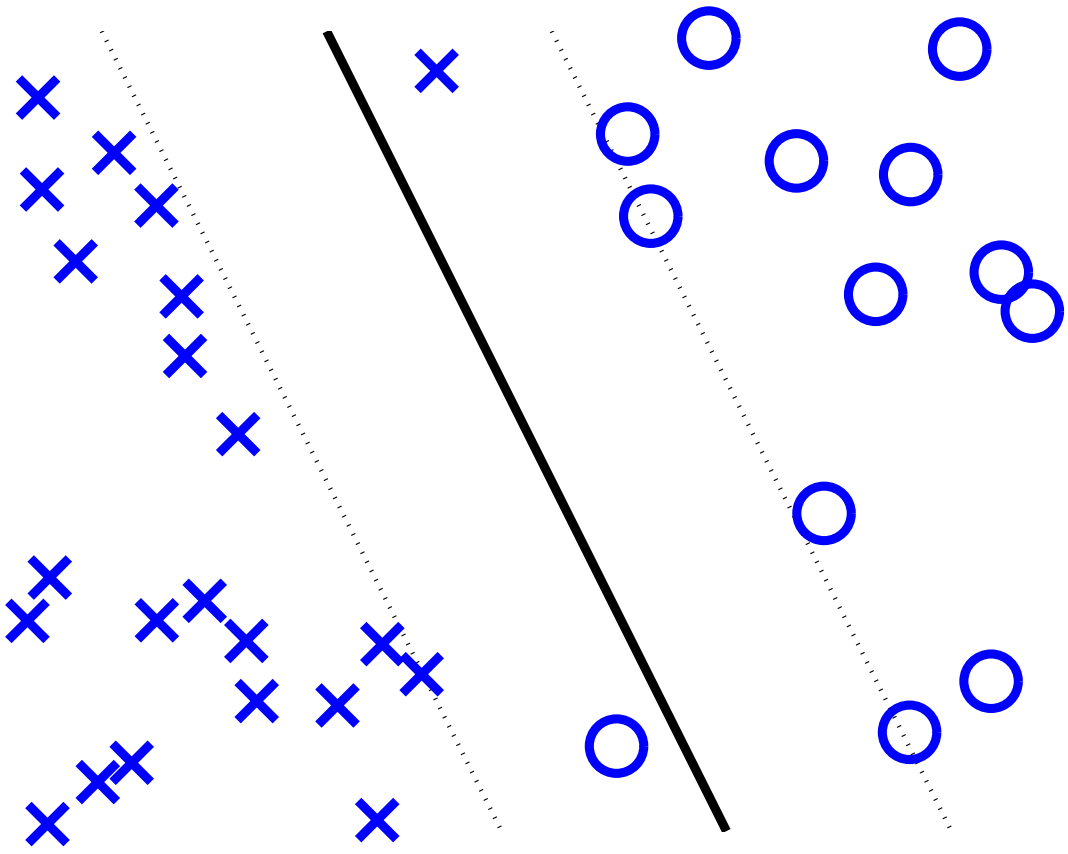
$$\min_{\mathbf{w},b} \quad \underbrace{\frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle}_{\text{regularizer}} \quad s.t. \quad \underbrace{y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1}_{\text{data fitting}}$$

hard margin
SVM



$$\min_{\mathbf{w},b} \quad \underbrace{\frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle}_{\text{regularizer}} \quad s.t. \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

soft margin SVM



$$\min_{\mathbf{w},b,(\xi_k)} \quad \begin{array}{l} \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle \\ +C \sum_i \xi_i \end{array} \quad s.t. \quad \begin{array}{c} \xi_i \geq 0 \\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \end{array}$$

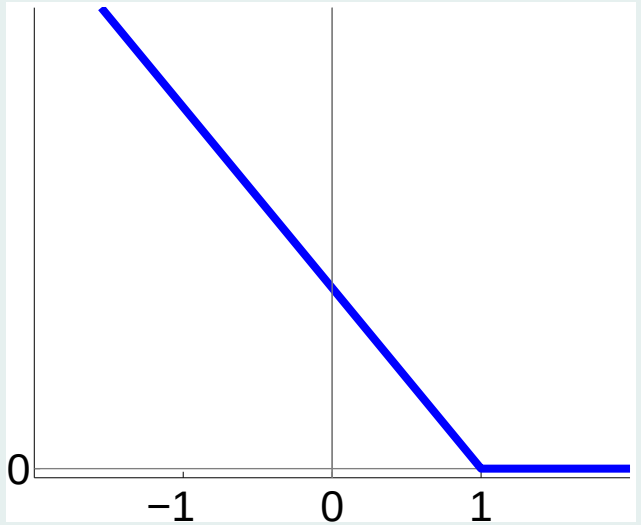
Soft-Margin SVM

$$\min_{\mathbf{w},b,(\xi_i)} \quad \tfrac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_i \xi_i$$

$$s.t. \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

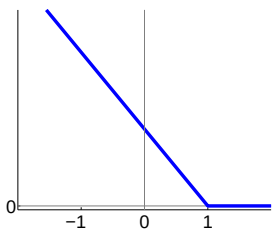
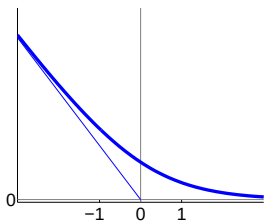
Effective Loss Function

$$\xi_i = \max\{1 - y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b), 0\}$$

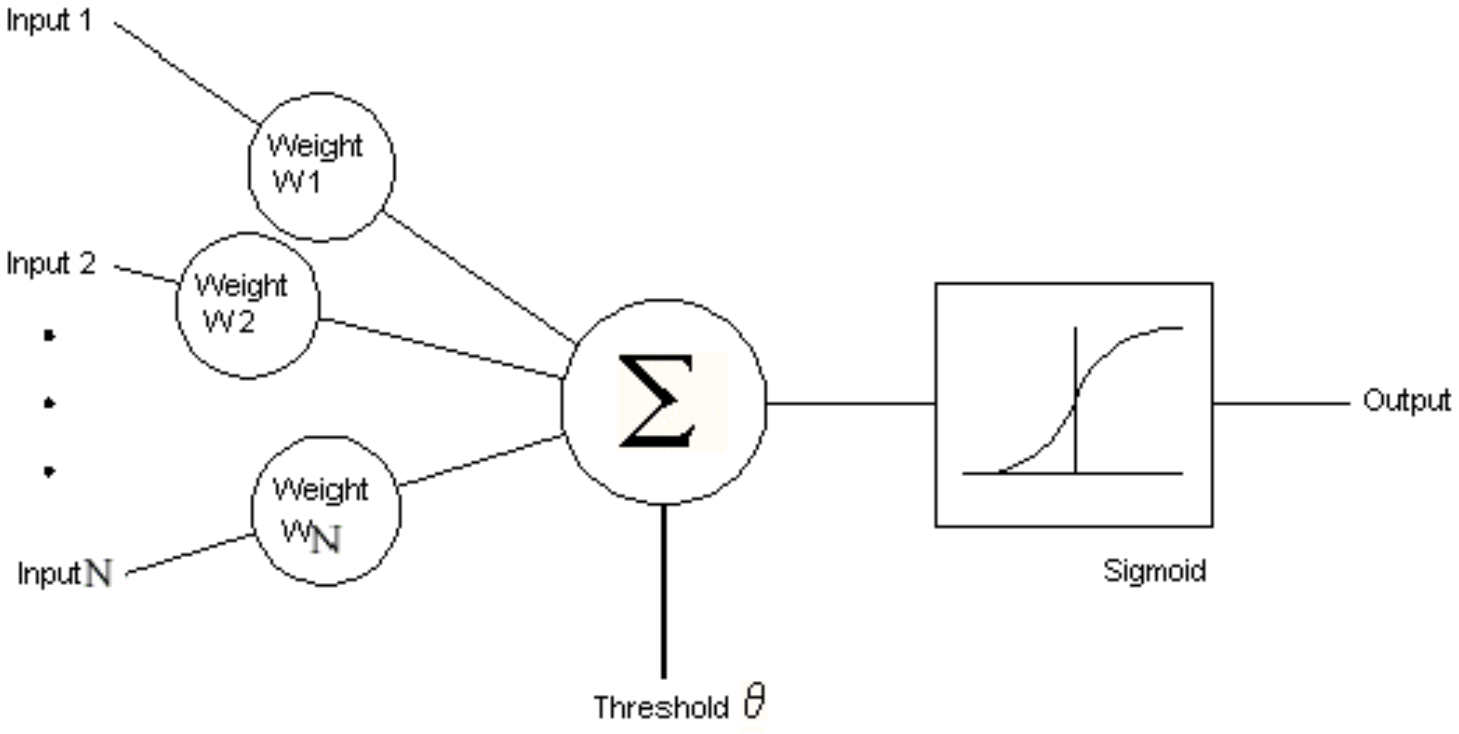


$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b)$$

Support Vector Machine $\approx$ Logistic Regression

method	**SVM**	**Logistic Regression**
training (optimization)	$\min_{\mathbf{w},b} \ \lambda \left\|\mathbf{w}\right\|^2 + \sum_i \ell_{\mathbf{w},b}(\mathbf{x}_i, y_i)$	
	ℓ is hinge loss 	ℓ is logistic loss 
prediction	$\frac{p(y=+1\|\mathbf{x})}{p(y=-1\|\mathbf{x})} > 1 :\Leftrightarrow$ $\mathbf{w}^\top \Phi(\mathbf{x}) + b > 0$	$p(y = +1\|\mathbf{x}) := \frac{1}{1+\exp\left(-(\mathbf{w}^\top \Phi(\mathbf{x})+b)\right)}$

Logistic Regression = Perceptron



$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} = \sum_{d=1}^{D} w_i x_i$$

$$\frac{1}{1 + \exp\left(-f(\mathbf{x})\right)}$$

[image from `http://homepages.gold.ac.uk/nikolaev/311perc.htm`]

Representer Theorem

Objective: $J(\mathbf{w}) = \|\mathbf{w}\|^2 + \sum_i \ell_i\left(\mathbf{w}^\top \mathbf{x}_i\right)$ with $\ell_i(t) := C\ell(t, y_i)$

Representer Theorem:

$\mathbf{w}^\star := \arg\min_{\mathbf{w}} J(\mathbf{w})$ is in the span of the data $\{\mathbf{x}_i\}$, ie

$$\mathbf{w}^\star = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i \quad .$$

Proof: Let $\mathbf{w}^\star = \underbrace{\sum_i \alpha_i \mathbf{x}_i}_{=:\mathbf{w}_\parallel} + \mathbf{w}_\perp$ with $\mathbf{w}_\perp \perp \mathbf{w}_\parallel$. Then

$$J(\mathbf{w}^\star) = \left\|\mathbf{w}_\parallel\right\|^2 + \|\mathbf{w}_\perp\|^2 + \sum_i \ell_i\left(\mathbf{w}_\parallel^\top \mathbf{x}_i + \mathbf{w}_\perp^\top \mathbf{x}_i\right) = J(\mathbf{w}_\parallel) + \|\mathbf{w}_\perp\|^2$$

■

Non-Linearity via Kernels

Kernel Functions

For feature map $\Phi(\mathbf{x})$, **kernel** $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$.

- Intuitively, kernel measures similarity of two objects $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.
- Fct is kernel $\Leftrightarrow$ fct is *positive semi-definite*.

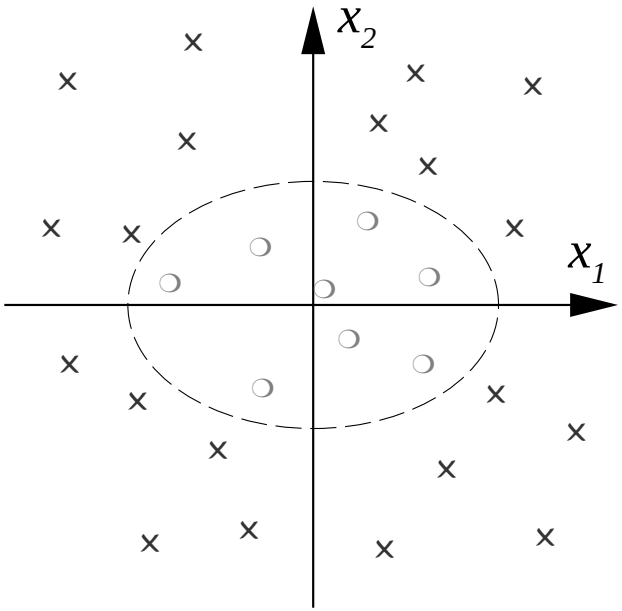
Kernelization: plug in kernel expansion $\mathbf{w}^\star = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i)$

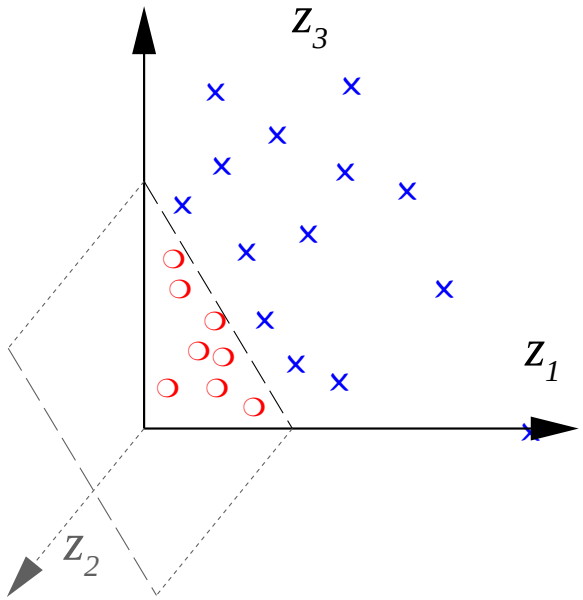
- possible if data access only through dot products
- hence requires 2-norm-regularization: $\|\mathbf{w}\|_2^2 = \langle \mathbf{w}, \mathbf{w} \rangle$
- SVMs, LogReg, LS-Reg, GPs, PCA, LDA, PLS, ...

Non-Linear Mappings

Example: All Degree 2 Monomials for a 2D Input

$$\begin{aligned} \Phi : \mathbb{R}^2 &\rightarrow \mathbb{R}^3 =: \mathcal{H} \quad (\text{“Feature Space”}) \\ (x_1, x_2) &\mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2) \end{aligned}$$





Kernel Trick

Example: All Degree 2 Monomials for a 2D Input

$$
\begin{aligned}
\left\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \right\rangle &= \left\langle (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2), ({x'}_1^2, \sqrt{2}\, x'_1 x'_2, x_2'^2) \right\rangle \\
&= x_1^2 {x'}_1^2 + 2\, x_1 x_2 x'_1 x'_2 + x_2^2 x_2'^2 \\
&= \left(x_1 x'_1 + x_2 x'_2\right)^2 \\
&= \left\langle \mathbf{x}, \mathbf{x}' \right\rangle^2 \\
&=: k(\mathbf{x}, \mathbf{x}')
\end{aligned}
$$

$\Rightarrow$ the dot product in $\mathcal{H}$ can be computed in $\mathbb{R}^2$

Polynomial Kernel

More generally: $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$, $k \in \mathbb{N}$:

$$\begin{aligned}
\left\langle \mathbf{x}, \mathbf{x}' \right\rangle^k &= \left(\sum_{d=1}^{D} x_d \cdot x'_d \right)^k \\
&= \sum_{d_1,\ldots,d_k=1}^{D} x_{d_1} \cdot \ldots \cdot x_{d_k} \cdot x'_{d_1} \cdot \ldots \cdot x'_{d_k} \\
&= \left\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \right\rangle,
\end{aligned}$$

where Φ maps into the space spanned by *all ordered products of k input directions.*

Successful application to DNA [Zien et al.; Bioinformatics, 2000].

Gaussian RBF Kernel

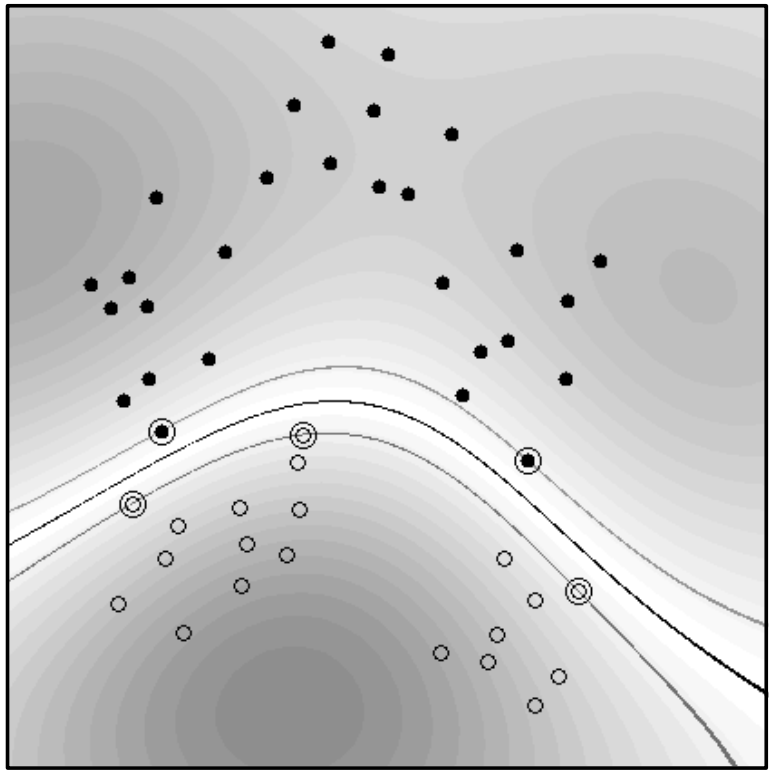
Gaussian RBF kernel: $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{\sigma^2}\left\|\mathbf{x} - \mathbf{x}'\right\|^2\right)$

What is $\Phi(\mathbf{x})$? Ask Ingo Steinwart. [I. Steinwart et al.; IEEE Trans. IT, 2006]

- radial basis fct (RBF):

  $k(\mathbf{x}, \mathbf{x}') = f\left(\left\|\mathbf{x} - \mathbf{x}'\right\|\right)$

- infinite-dimensional feature space
- any smooth discrimination



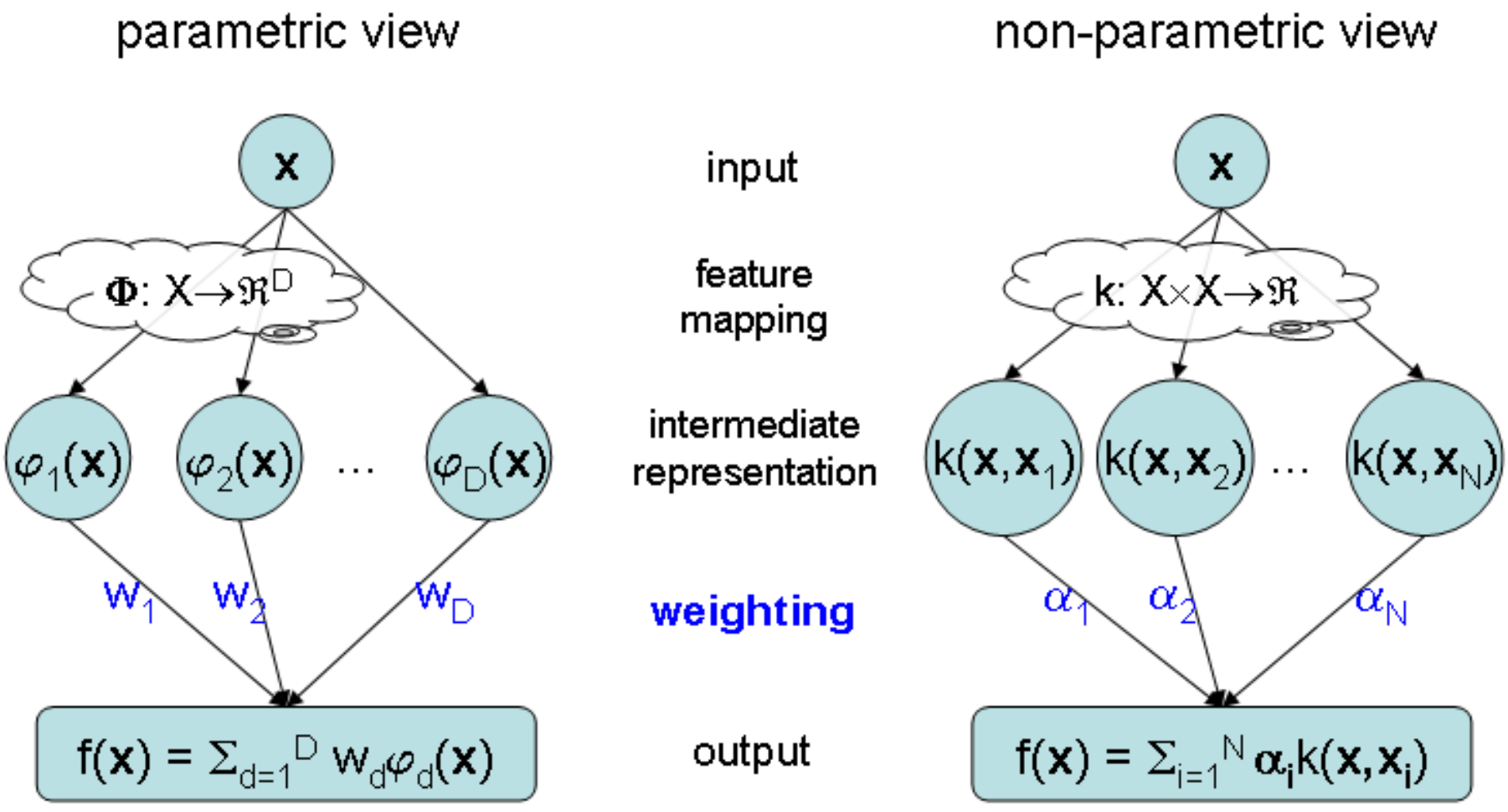
Look for an "SVM applet" on the web.

Parametric vs Non-Parametric

Two **equivalent views** on kernel machines:

- **parametric** method
  - $\Phi(\mathbf{x}) \in \mathbb{R}^D$ computed explicitly
  - optimize $\mathbf{w} \in \mathbb{R}^D$
  - fixed number D of parameters
  - decision function **linear** in $\Phi(\mathbf{x})$
- **non-parametric** method
  - $\Phi(\mathbf{x})$ never computed; always use kernel $k(\mathbf{x}, \cdot)$
  - possibly infinitely many features, $\Phi : \mathcal{X} \to \mathbb{R}^\infty$
  - optimize coefficients $\alpha \in \mathbb{R}^N$ of kernel expansion
  - number of parameters α_i increases with number N of data points
  - decision function **non-linear** in $\mathbf{x}$

Support Vector Machine = Perceptron (1)



SVM = Perceptron (2)

Geoff Hinton's view on SVMs:

- "Vapnik and his co-workers developed **a very clever type of perceptron** called a **Support Vector Machine**."
- "Instead of hand-coding the layer of non-adaptive features, **each training example is used to create a new feature** using a fixed recipe."
- "The feature computes how similar a test example is to that training example."
- "Then a **clever optimization technique is used** to select the best subset of the features and to decide how to weight each feature when classifying a test case."

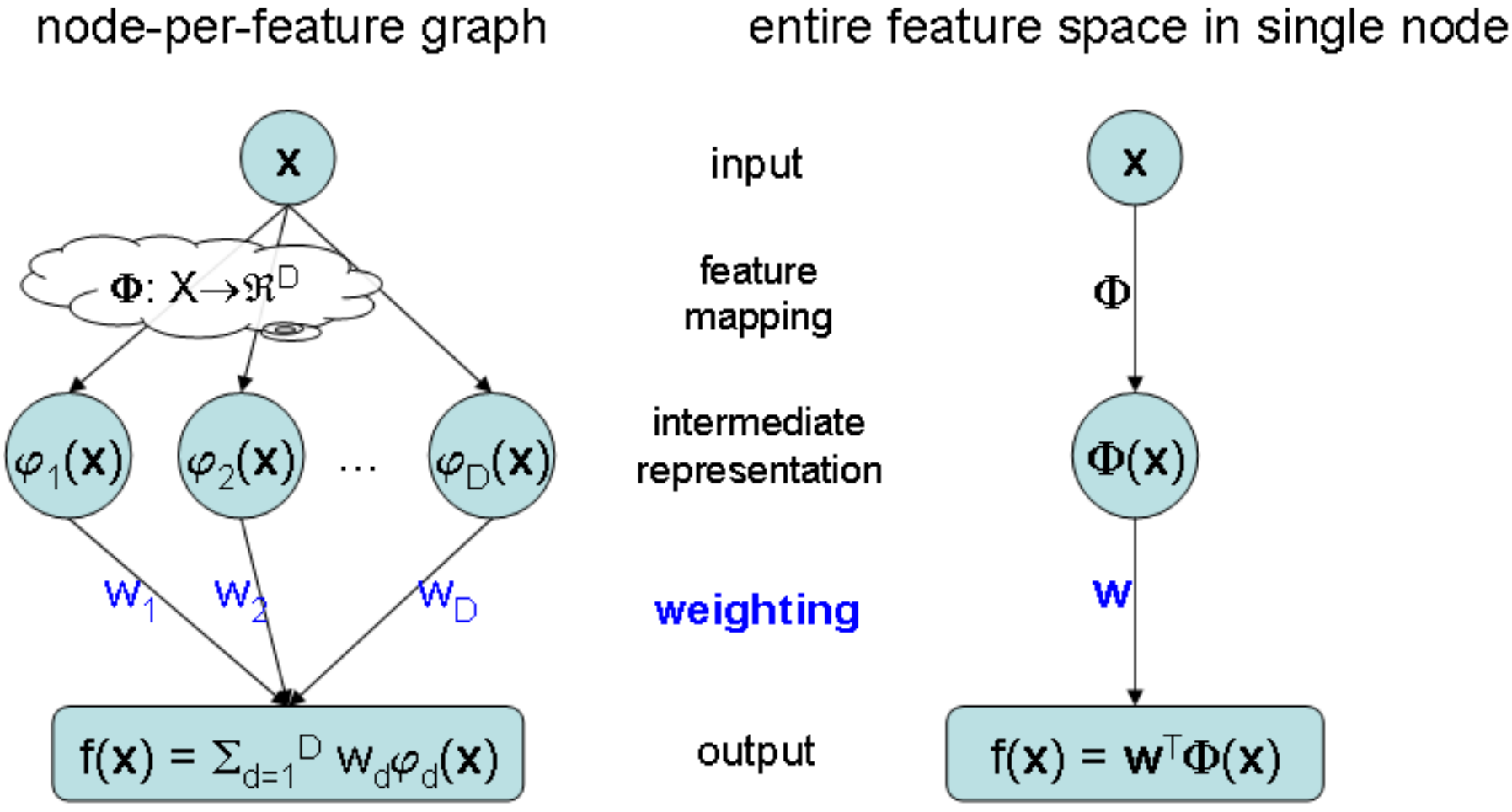
[`http://www.cs.utoronto.ca/~hinton/`, NIPS 2007 tutorial]

So Why Talk About SVMs?

Why not train a perceptron or MLP with backpropagation?

- SVM training = quadratic programming (QP) problem
  - **convex**: no problem with (bad) local minima
  - very **efficient** solvers available
- kernels offer convenient way to use huge sets of features
  - *implicitly* $\Rightarrow$ computational cost independent of dimensionality
  - thus learning with **infinitely many features** possible
- caveat: "flat architectures" may also have disadvantages [Y. Bengio, Y. Le Cun; "Scaling learning algorithms towards AI"; MIT Press, 2007]

SVM Perceptron in Compact Representation



Learning with Kernels: B. Schölkopf, A. Smola; MIT Press, 2002.

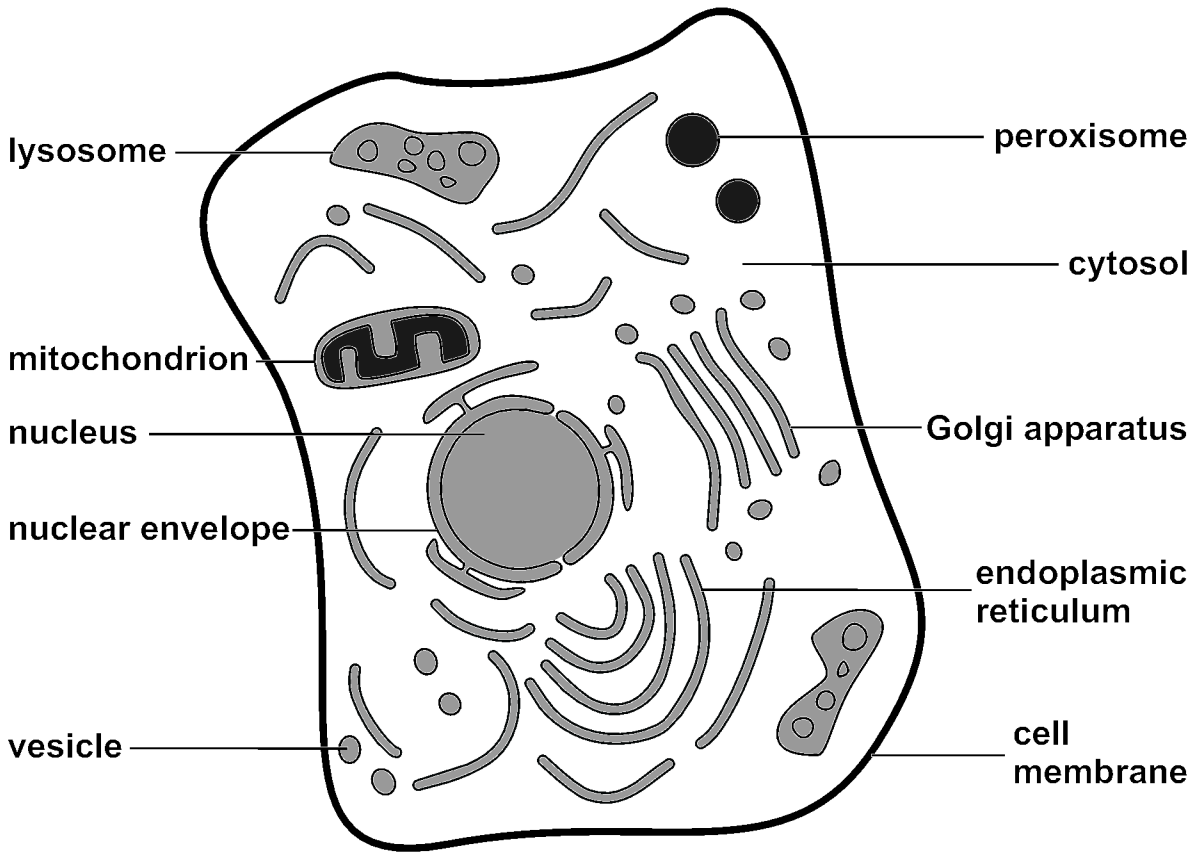
Outline

1. Recap: Support Vector Machines (SVMs)
   - SVMs Do Linear Large Margin Separation
   - Non-Linearity via Kernels
   - SVMs are Perceptrons
2. Application: Predicting Protein Subcellular Localization
3. Multiple Kernel Learning (MKL)
   - A Large Margin MKL Model
   - Optimization for MKL
   - Normalization of Kernels Is Important
   - Multiclass Multiple Kernel Learning
4. Back to App: Predicting Protein Subcellular Localization
5. Take Home Messages

Compartments of a Cell

Input: protein sequence

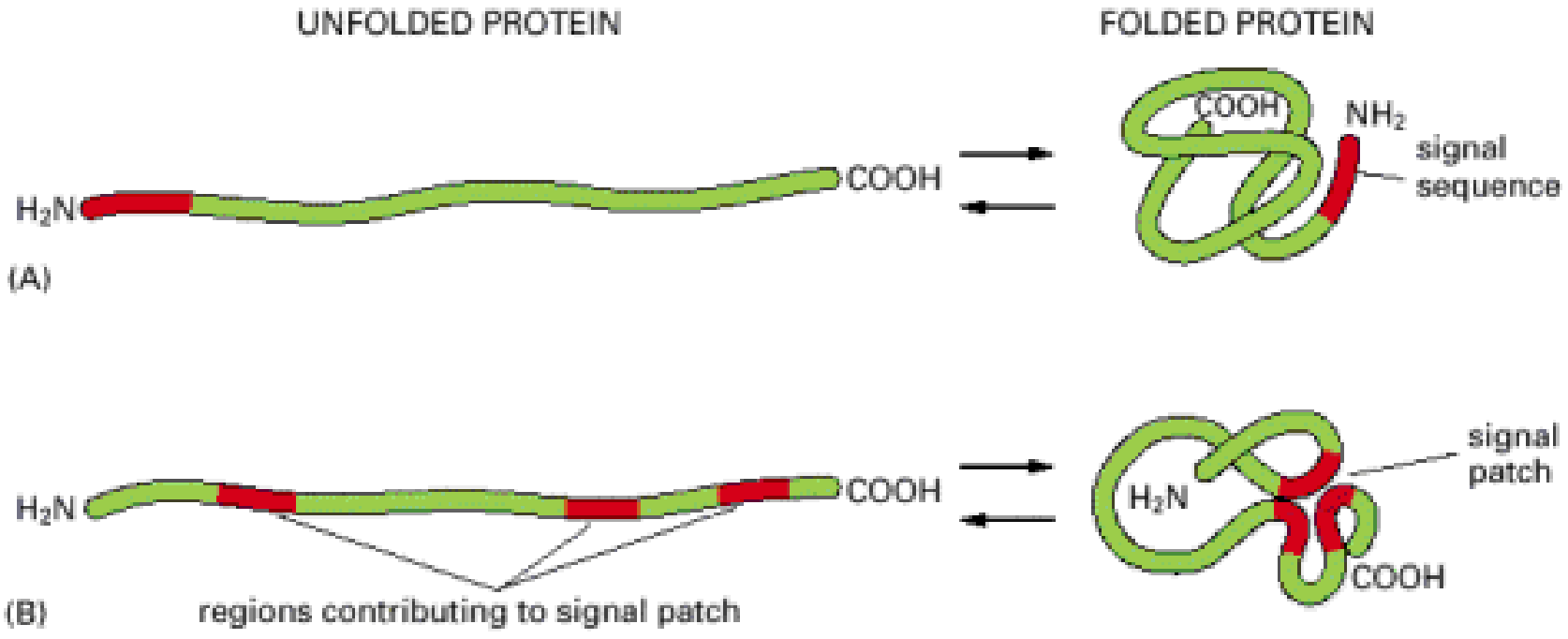
Output: target location of the protein



[image from "A primer on molecular biology" in "Kernel Methods in Computational Biology", MIT Press, 2004]

Signal Peptides

Proteins: chain molecules composed from amino acids (20 types) that fold into intricate 3D shapes



[image from "Molecular Biology of the Cell", 2002; Alberts et al.]

Sequence Features for Predicting Subcellular Localization

- **motif composition**
  - incidence (histogram) of amino acids (letters)
  - incidence of short (possibly non-consecutive) substrings
  - on different subsequences, eg first 60 amino acids
    background: many relevant signal seq's at beginning or end
- **pairwise sequence similarities (BLAST E-values)**
  - alignment of each pair of protein sequences with BLAST
  - E-value: is observed similarity expected by chance?
  - represent protein by alignment E-values to all other proteins
- **phylogenetic profiles**
  - roughly, a binary vector indicating existence of orthologuous protein in each of 89 completely sequenced species
  - taken from PLEX server [Pellegrini et al., 1999]
    `http://apropos.icmb.utexas.edu/plex/`

Motif Patterns

look for motifs by defining r-tuples wrt "patterns"
(instead of just consecutive amino acids)

Examples:

- (•,•,•,•) is a 4-mer on consecutive AAs.
- (•,•,○,○) is a 2-mer on consecutive AAs.
- (•,○,○,•) is a 2-mer with 2 gaps in between.
- (•,•,○,•) is a 3-mer with 1 gap in the third position.

Motif Composition Kernel

Starting from **AA substitution matrix** like BLOSUM62:

1. derive **AA kernel** $k_{AA}(a, b)$ (has to be positive semi-definite)
2. **AA motif kernel**: on r-tuples $s, t \in \{AAs\}^k$ of amino acids

$$k^r_{AA}(s, t) = \sum_{j=1}^{r} k_{AA}(s_j, t_j)$$

3. **motif composition** (wrt given pattern): $p : \{AAs\}^k \to [0, 1]$
   represent sequence by histogram of motif occurences
4. **Jensen-Shannon kernel** [Hein, Bousquet; 2005]
   - compares two histograms p and q of motifs
   - takes into account similarity of motifs s and t

Computational efficiency: exploit sparse support of histograms

List of 69 Kernels

64 = 4*16 Motif kernels

- 4 subsequences (all, last 15, first 15, first 60)
- $16 = 2^{5-1}$ patterns of length 5 (●,○,○,○,○)

3 BLAST similarity kernels

1. linear kernel on E-values
2. Gaussian kernel on E-values, width 1000
3. Gaussian kernel on log E-values, width 1e5

2 phylogenetic kernels

1. linear kernel
2. Gaussian kernel, width 300

[all described in C. S. Ong, A. Zien; WABI 2008]

Traditional Approaches to Use Several Kernels

1. **select best single kernel**
   - eg by cross-validation

2. **engineer a multi-layer prediction system**
   1. train one SVM for each kernel
   2. consider the output of each SVM as meta-feature
   3. combine them into single prediction, eg by another SVM

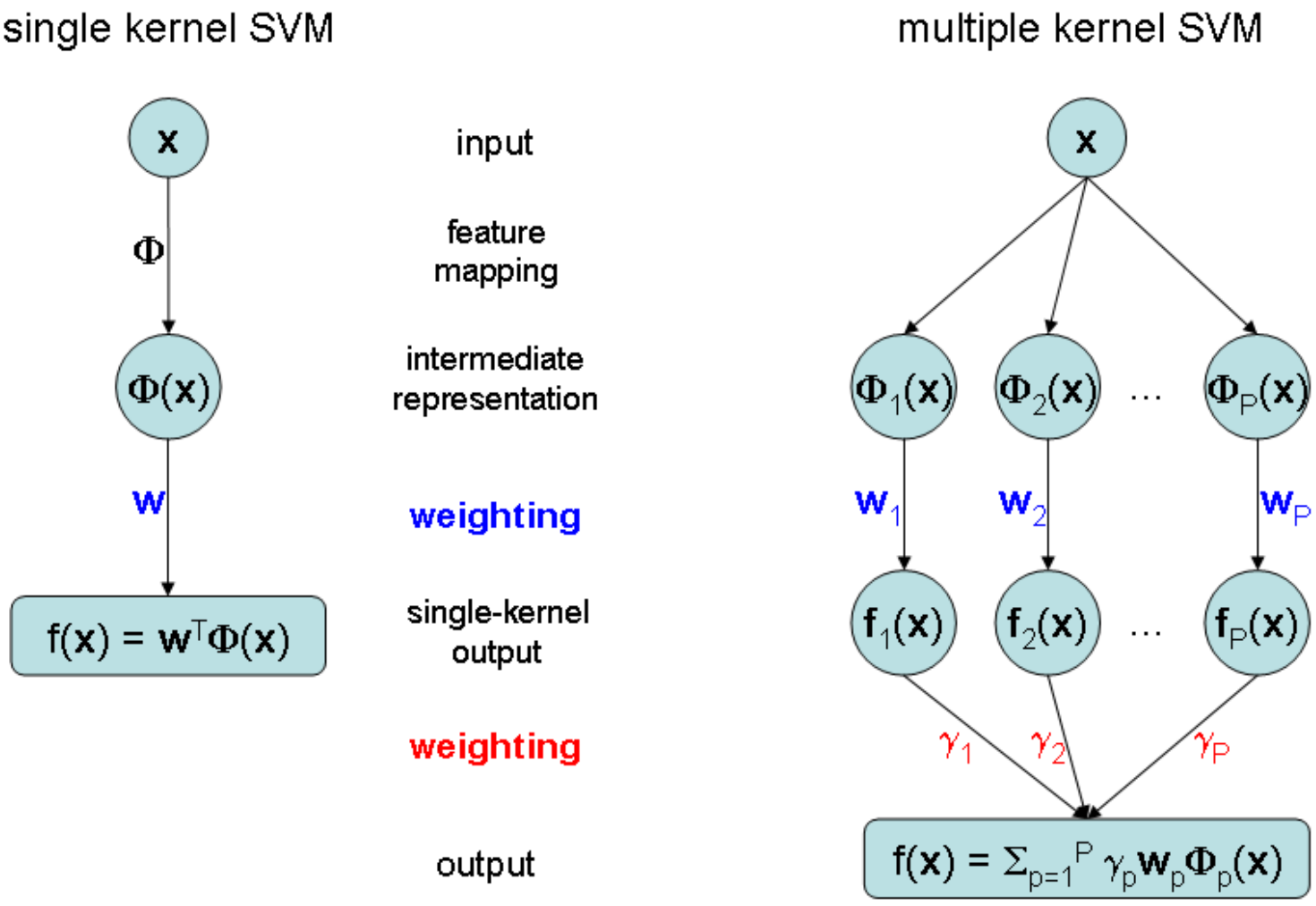
   eg [A. Höglund et al., "MultiLoc", Bioinfomatics, 2006]
   care has to be taken for proper cross-validation

3. **combine all kernels into a single kernel**
   - most popular: add kernels
   - empirically successful [P. Pavlidis et al.; Journal of Computational Biology, 2002]
   - but is the plain (unweighted) sum really optimal?

Outline

1. Recap: Support Vector Machines (SVMs)
   - SVMs Do Linear Large Margin Separation
   - Non-Linearity via Kernels
   - SVMs are Perceptrons
2. Application: Predicting Protein Subcellular Localization
3. Multiple Kernel Learning (MKL)
   - A Large Margin MKL Model
   - Optimization for MKL
   - Normalization of Kernels Is Important
   - Multiclass Multiple Kernel Learning
4. Back to App: Predicting Protein Subcellular Localization
5. Take Home Messages

Perceptron With Multiple Kernels



$$f_p(\mathbf{x}) = \langle \mathbf{w}_p, \Phi_p(\mathbf{x}) \rangle$$

A Multiple Kernel Learning (MKL) Model

MKL Model: weighted linear mixture of P feature spaces

$$
\begin{aligned}
\mathcal{H}_{\boldsymbol{\gamma}} &\leftarrow \gamma_1\mathcal{H}_1 \oplus \gamma_2\mathcal{H}_2 \oplus \ldots \oplus \gamma_P\mathcal{H}_P \\
\Phi_{\boldsymbol{\gamma}}(\mathbf{x}) &\leftarrow \left(\gamma_p\Phi_p(\mathbf{x})^\top\right)_{p=1,\ldots,P}^\top \\
k_{\boldsymbol{\gamma}}(\mathbf{x},\mathbf{x}') &\leftarrow \sum_{p=1}^{P}\left\langle\gamma_p\Phi_p(\mathbf{x}),\gamma_p\Phi_p(\mathbf{x}')\right\rangle = \sum_{p=1}^{P}\gamma_p^2 k_p(\mathbf{x},\mathbf{x}') \\
\mathbf{w}_{\boldsymbol{\gamma}} &\leftarrow \left(\gamma_p\mathbf{w}_p^\top\right)_{p=1,\ldots,P}^\top \\
f_{\boldsymbol{\gamma}}(\mathbf{x}) &\leftarrow \left\langle\mathbf{w}_{\boldsymbol{\gamma}},\Phi_{\boldsymbol{\gamma}}(\mathbf{x})\right\rangle = \sum_{p=1}^{P}\gamma_p^2\left\langle\mathbf{w}_p,\Phi_p(\mathbf{x})\right\rangle
\end{aligned}
$$

Goal: learn mixing coefficients $\boldsymbol{\gamma} = (\gamma_p)_{p=1,\ldots,P}$ along with $\mathbf{w}, \mathbf{b}$

Large Margin MKL

Plugging it into the SVM

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\boldsymbol{\gamma}} \quad & \frac{1}{2}\left\|\mathbf{w}_{\boldsymbol{\gamma}}\right\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \forall i : \xi_i = \ell\left(\left\langle \mathbf{w}_{\boldsymbol{\gamma}}, \Phi_{\boldsymbol{\gamma}}(\mathbf{x})\right\rangle + b, y_i\right) \end{aligned}$$

yields:

$$\begin{aligned} \min_{\mathbf{w},b,\xi,\boldsymbol{\gamma}} \quad & \frac{1}{2}\sum_{p=1}^{P}\gamma_p^2\left\|\mathbf{w}_p\right\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & \forall i : \xi_i = \ell\left(\sum_{p=1}^{P}\gamma_p^2\left\langle \mathbf{w}_p, \Phi_p(\mathbf{x})\right\rangle, y_i\right) \end{aligned}$$

for convenience we substitute $\beta_p := \gamma_p^2$

Extension 1: Non-Negative Weights $\boldsymbol{\beta}$

$$\min_{\mathbf{w},b,\xi,\boldsymbol{\beta}} \quad \frac{1}{2}\sum_{p=1}^{P}\beta_p \left\|\mathbf{w}_p\right\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad \forall i : \xi_i = \ell\left(\sum_{p=1}^{P}\beta_p\left\langle \mathbf{w}_p, \Phi_p(\mathbf{x})\right\rangle, y_i\right)$$

What if $\beta_p < 0$?

- recall that $\beta_p = \gamma_p^2$
  - $\gamma_p \in \mathbb{R}$, supposedly — what would imaginary γ_p mean?
- recall that $k_{\boldsymbol{\gamma}}(\mathbf{x}, \mathbf{x}') = \sum_{p=1}^{P}\gamma_p^2 k_p(\mathbf{x}, \mathbf{x}')$
  - but kernels have to be positive semi-definite!

Solution: add positivity constraints, $\boldsymbol{\beta} \geq \mathbf{0}$

Extension 2: Effective Regularization

$$\min_{\mathbf{w},b,\xi,\boldsymbol{\beta}} \quad \frac{1}{2}\sum_{p=1}^{P}\beta_p \left\|\mathbf{w}_p\right\|^2 + \sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad \forall i : \xi_i = \ell\left(\sum_{p=1}^{P}\beta_p\left\langle \mathbf{w}_p, \Phi_p(\mathbf{x})\right\rangle, y_i\right)$$

$$\forall p : \beta_p \geq 0$$

assume optimal solution $\mathbf{w}^\star$, $\boldsymbol{\beta}^\star$.

what is the objective for $\mathbf{w}' := \mathbf{w}^\star/2$, $\boldsymbol{\beta}' := \boldsymbol{\beta}^\star \cdot 2$?

Two Layers Need Two Regularizers

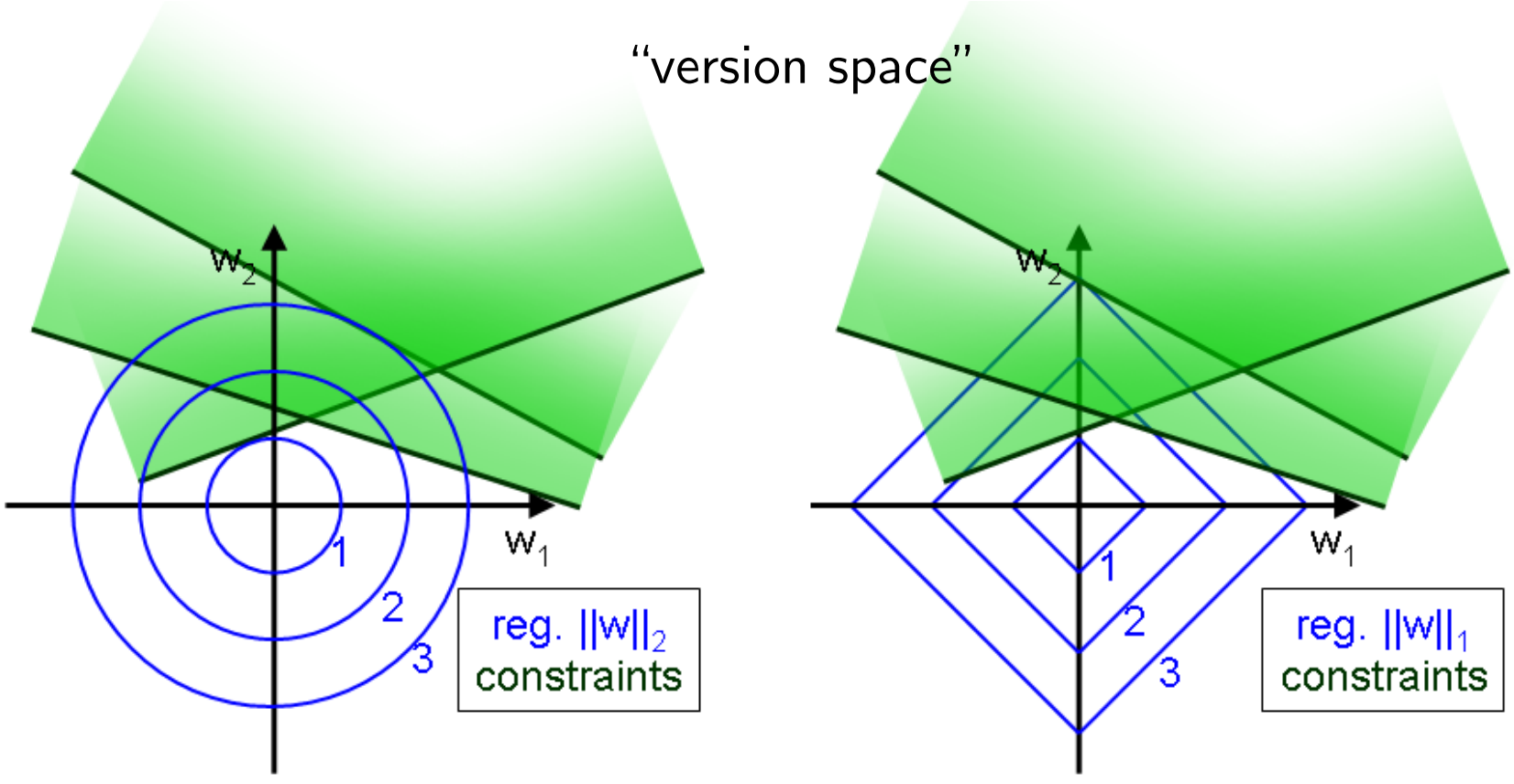
$\Rightarrow$ $\mathbf{w}$ will shrink to zero, $\boldsymbol{\beta}$ will expand to infinity!

$\Rightarrow$ Need regularization on $\boldsymbol{\beta}$ as well!

Two common choices for regularization:

- **standard MKL**: 1-norm-regularization
  - constrain or minimize $\|\boldsymbol{\beta}\|_1 = \sum_p |\beta_p|$
  - promotes sparse solutions: **kernel selection**
  - as $\beta_p \geq 0$, it is enough to require $\sum_p \beta_p \leq 1$
  - why will $\sum_p \beta_p^\star = 1$ hold?
- yet unexplored alternative: 2-norm-regularization
  - constrain or minimize $\|\boldsymbol{\beta}\|_2^2 = \sum_p \beta_p^2$
  - uses all offered kernels

Why Does 1-Norm-Regularization Promote Sparsity?

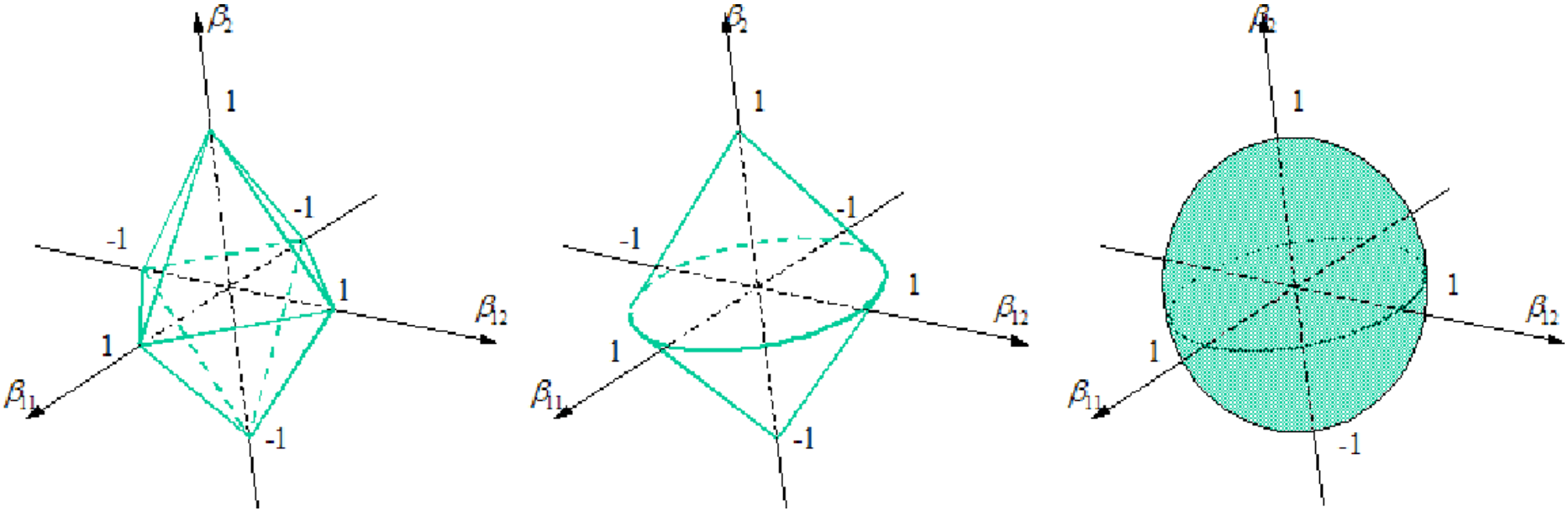


standard (2-norm) SVM

1-norm SVM

feasible region meets regularizer at corners (if any exist)

Standard (1-norm-) MKL: Mixed Regularization



1-norm SVM, lasso:
1-norm-constraints on all individual features

standard MKL:

- 1-norm-constraints *between* groups (ie kernels)
- 2-norm-constraints *within* feature groups

standard SVM, ridge-regression:
2-norm-constraints on all features

[image from M. Yuan, Y. Lin; Journal of the Royal Statistical Society 2006]

Extension 3: Retain Convexity

Problem: products $\beta_p \mathbf{w}_p$ make constraints non-convex

$$\min_{\boldsymbol{\beta},\mathbf{w},\mathbf{b},\xi} \quad \frac{1}{2}\sum_{p=1}^{P} \beta_p \left\|\mathbf{w}_p\right\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad \forall i : \xi_i = \ell\left(\sum_{p=1}^{P} \beta_p \left\langle \mathbf{w}_p, \Phi_p(\mathbf{x}_i)\right\rangle, y_i\right)$$

Solution: change of variables $\mathbf{v}_p := \beta_p \mathbf{w}_p$

$$\min_{\boldsymbol{\beta},\mathbf{v},\mathbf{b},\xi} \quad \frac{1}{2}\sum_{p=1}^{P} \frac{1}{\beta_p} \left\|\mathbf{v}_p\right\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$\text{s.t.} \quad \forall i : \xi_i = \ell\left(\sum_{p=1}^{P} \left\langle \mathbf{v}_p, \Phi_p(\mathbf{x}_i)\right\rangle, y_i\right)$$

Relation to Original MKL Formulation

	shown [Zien & Ong; ICML 2007]	"traditional"
$R(\mathbf{w}, \boldsymbol{\beta})$	$\frac{1}{2}\sum_{p=1}^{P} \beta_p \left\|\mathbf{w}_p\right\|^2$	$\frac{1}{2}\left(\sum_{p=1}^{P} \beta_p \left\|\mathbf{w}_p\right\|\right)^2$
$f(\mathbf{x}, y)$	$\sum_{p=1}^{P} \beta_p \left\langle \mathbf{w}_p, \Phi_p(\mathbf{x}) \right\rangle + b$	$\sum_{p=1}^{P} \beta_p \left\langle \mathbf{w}_p, \Phi_p(\mathbf{x}) \right\rangle + b$
		[Sonnenburg et al.; NIPS 2005]
$R(\mathbf{w}, \boldsymbol{\beta})$	$\frac{1}{2}\sum_{p=1}^{P} \frac{1}{\beta_p} \left\|\mathbf{v}_p\right\|^2$	$\frac{1}{2}\left(\sum_{p=1}^{P} \left\|\mathbf{v}_p\right\|\right)^2$
$f(\mathbf{x}, y)$	$\sum_{p=1}^{P} \left\langle \mathbf{v}_p, \Phi_p(\mathbf{x}) \right\rangle + b$	$\sum_{p=1}^{P} \left\langle \mathbf{v}_p, \Phi_p(\mathbf{x}) \right\rangle + b$
		[Bach et al.; ICML 2004]

Equivalences:

- top row (non-convex) $\Leftrightarrow$ bottom row (convex): transform.
- left column (proposed) $\Leftrightarrow$ right column (existing): same dual (of convex version) plus strong duality

Optimization Approaches

Several possibilities for training/optimization:

- dual is QCQP
  $\Rightarrow$ can use off-the-shelf solver (eg CVXOPT, Mosek, CPLEX)

- transform into semi-infinite linear program (SILP)
  can solve by Column Generation technique
  [Sonnenburg et al., NIPS 2005]

- projected gradient on β [Rakotomamonjy et al., ICML 2007]

- primal gradient-based optimization [work in progress]

MKL Wrapper by Column Generation (1)

1. initialize LP with minimal set of constraints:

$$\sum_p \beta_p = 1, \quad \beta_p \geq 0$$

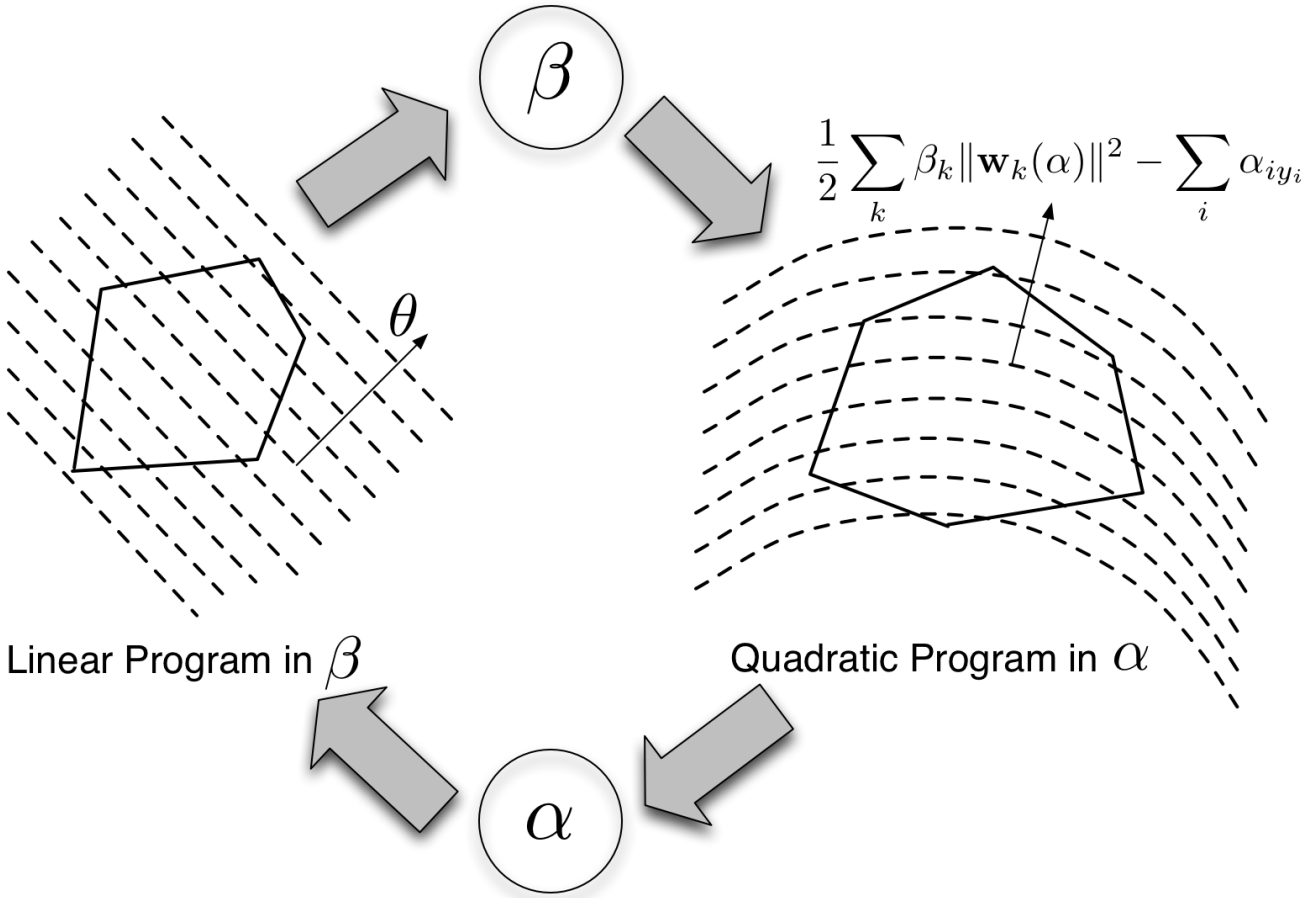
2. initialize $\boldsymbol{\beta}$ to feasible value (eg $\beta_p = 1/P$)
3. iterate:
   - for given $\boldsymbol{\beta}$, find most violated constraint:

     minimize $\frac{1}{2}\sum_p \beta_p \left\|\mathbf{w}_p(\alpha)\right\|^2 - \sum_i \alpha_i$ st $\alpha \in \mathcal{S}$

     $\Rightarrow$ solve single-kernel SVM!
   - add this constraint to LP
   - solve LP to obtain new mixing coefficients $\boldsymbol{\beta}$

$\Rightarrow$ just need wrapper around single-kernel method

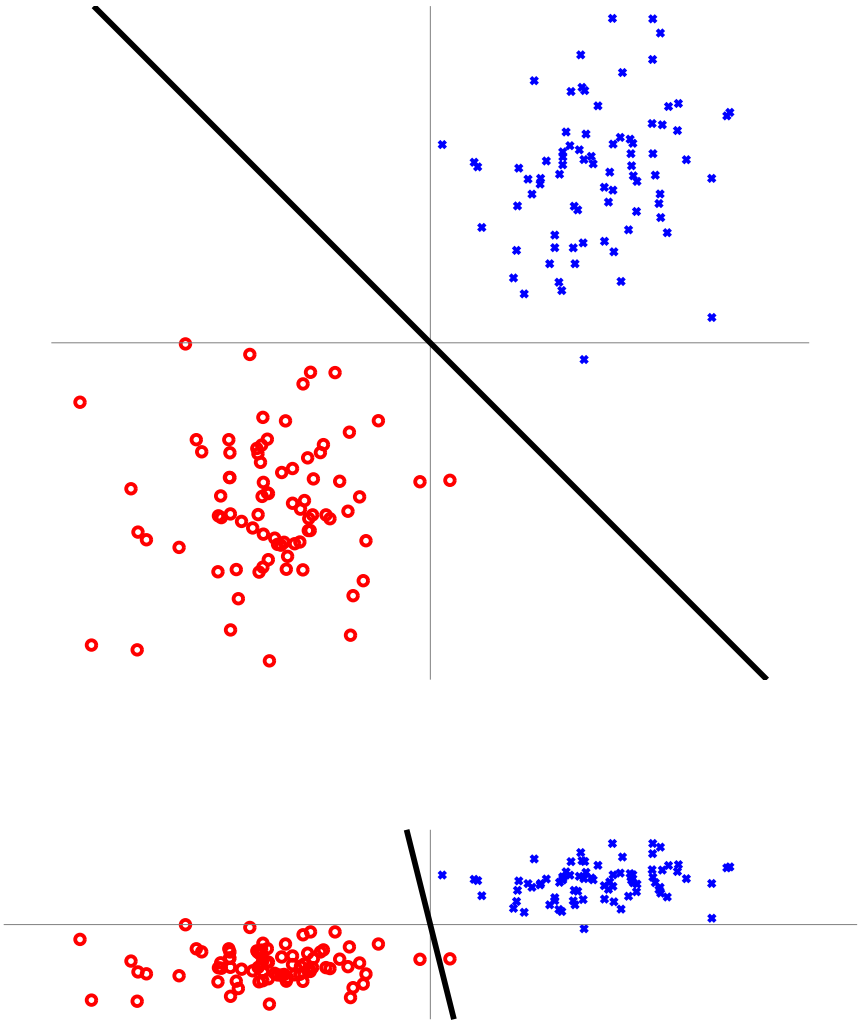
MKL Wrapper by Column Generation (2)



Alternate between solving an LP for β and a QP for α.

Free MKL software (and more) at `http://mloss.org`.

Normalization: Why Does Scaling Matter?



SVM on original data:

$$\min_{w_1,w_2,b} \quad \tfrac{1}{2}(w_1^2 + w_2^2) + C\sum_i \ell\big(y_i, w_1 x_{i,1} + w_2 x_{i,2}\big)$$

SVM on rescaled data:

$$\min_{v_1,v_2,b} \quad \tfrac{1}{2}(v_1^2 + v_2^2) + C\sum_i \ell\big(y_i, v_1 x_{i,1} + v_2 x_{i,2}/s\big)$$

equivalently, with $u_2 := v_2/s$:

$$\min_{u_1,u_2,b} \quad \tfrac{1}{2}(u_1^2 + s^2 u_2^2) + C\sum_i \ell\big(y_i, u_1 x_{i,1} + u_2 x_{i,2}\big)$$

Standardization of Features

Standard solution: **standardization** of features

- scale each feature to unit variance
- $x_{i,d} \rightarrow x_{i,d}/s_d$ where $s_d = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(x_{i,d} - \bar{x}_{\cdot,d}\right)^2}$
- the mean $\bar{x}_{\cdot,d}$ is irrelevant (why?)

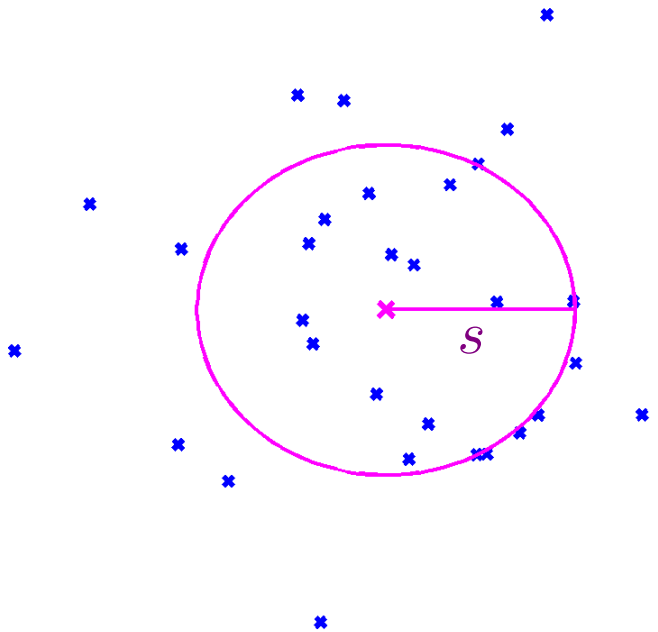
Note: individual features not accessible in kernel machines.

But analoguous problem for MKL with kernel scales!

- "larger" kernels bound to get more weight
- even aggrevated due to 1-norm penalty on $\boldsymbol{\beta}$

Standardization of Kernels

Solution: **standardize** entire kernel



- rescale such that variance s^2 within feature space is const
- variance

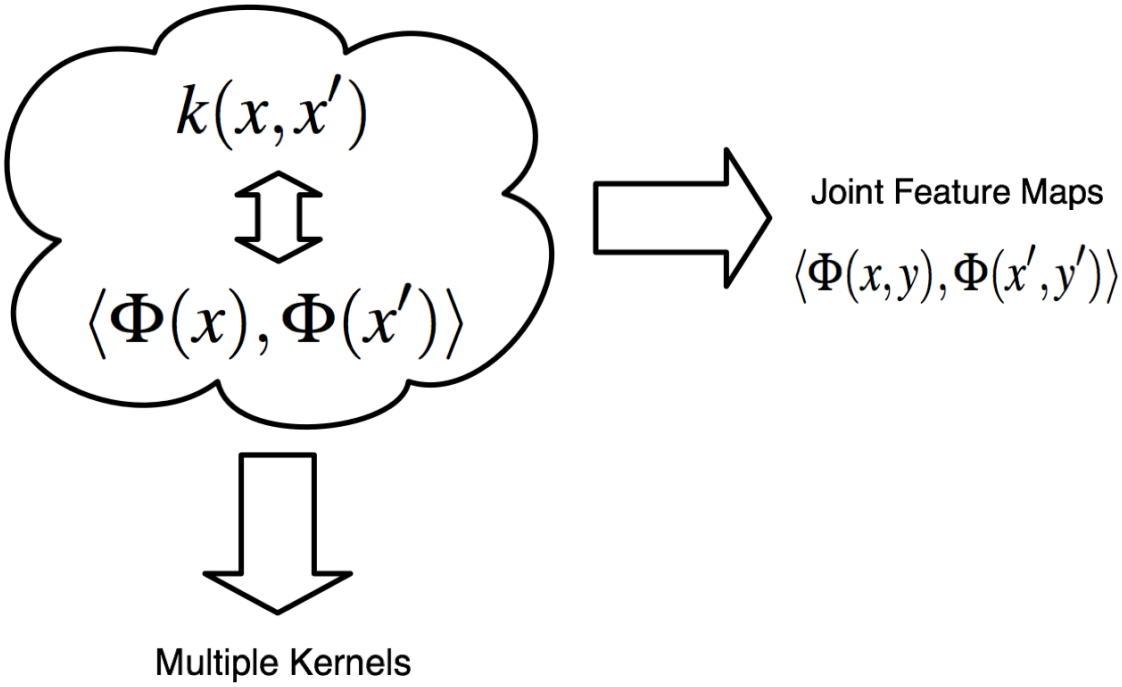
$$s^2 := \frac{1}{N} \sum_{i=1}^{N} \left(\Phi(\mathbf{x}_i) - \overline{\Phi(\mathbf{x})} \right)^2,$$

mean $\overline{\Phi(\mathbf{x})} := \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i)$

- kernel matrix $\mathbf{K} \longrightarrow$

$$\mathbf{K} / \left[\frac{1}{N} \sum_i K_{ii} - \frac{1}{N^2} \sum_{i,j} K_{ij} \right]$$

Two Generalizations of Kernel Methods



- *Joint Feature Maps* to go beyond binary classification [Crammer & Singer; JMLR 2001]
- *Multiple Kernel Learning (MKL)* for selecting from and weighting several sets of features

Multiclass: by Joint Feature Maps (Single Kernel)

- **Joint feature map** $\Phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$

$$k\left((\mathbf{x}, y), (\mathbf{x}', y')\right) \quad = \quad \left\langle \Phi(\mathbf{x}, y), \Phi(\mathbf{x}', y') \right\rangle$$

  - **multiclass**: $k\left((\mathbf{x}, y), (\mathbf{x}', y')\right) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') k_{\mathcal{Y}}(y, y')$
  - **no prior knowledge**: $k_{\mathcal{Y}}(y, y') = \mathbf{1}\{y = y'\}$

- **Prediction**: maximize output function

$$f_{\mathbf{w},\mathbf{b}}(\mathbf{x}, y) = \langle \mathbf{w}, \Phi(\mathbf{x}, y) \rangle + b_y$$

$$\mathbf{x} \mapsto \arg\max_{y \in \mathcal{Y}} f_{\mathbf{w},\mathbf{b}}(\mathbf{x}, y)$$

- **Training**: satisfy $f_{\mathbf{w},\mathbf{b}}(\mathbf{x}_i, y_i) > f_{\mathbf{w},\mathbf{b}}(\mathbf{x}_i, u)$ for all $u \neq y_i$

$$\min_{\mathbf{w},\mathbf{b}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{N} \max_{u \neq y_i} \left\{ \ell\left(f_{\mathbf{w},\mathbf{b}}(\mathbf{x}_i, y_i) - f_{\mathbf{w},\mathbf{b}}(\mathbf{x}_i, u) \right) \right\}$$

Multiclass Multiple Kernel Learning (MCMKL)

MCMKL training objective (omitting biases for simplicity):

$$\min_{\boldsymbol{\beta},\mathbf{w},\mathbf{b},\xi} \quad \frac{1}{2}\sum_{p=1}^{P}\beta_p \left\|\mathbf{w}_p\right\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t.} \quad \forall i : \xi_i = \max_{u\neq y_i} \ell\left(\sum_{p=1}^{P}\beta_p \left\langle \mathbf{w}_p, \Phi_p(\mathbf{x},y) - \Phi_p(\mathbf{x},u)\right\rangle\right)$$

with $\boldsymbol{\beta}$ in the probability simplex

$$\boldsymbol{\beta} \in \Delta^p := \left\{ \boldsymbol{\beta} \,\middle|\, \sum_{p=1}^{P}\beta_p = 1, \forall p : 0 \leq \beta_p \right\}$$

$\Rightarrow$ can use wrapper around M-SVM

True Multiclass or One-vs-Rest Heuristic?

Why **genuine multiclass** MKL instead of 1-vs-rest MKL?

- yields single weighting
  - pro: needs fewer kernels in total
  - con: does not show which kernel helps for which class
- may be used for structured output MKL
- more natural and convenient
- may be used to learn kernel on classes [Alex Smola]

Learning the Kernel on the Classes (1)

$$\sum_{p=1}^{P} k_p\left((\mathbf{x}, y), (\mathbf{x}', y')\right) = \sum_{p=1}^{P} k^{\mathcal{X}}(\mathbf{x}, \mathbf{x}') k_p^{\mathcal{Y}}(y, y')$$

$$= k^{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \sum_{p=1}^{P} k_p^{\mathcal{Y}}(y, y')$$

Problem: No finite basis for the set of positive semi-definite kernels exist.

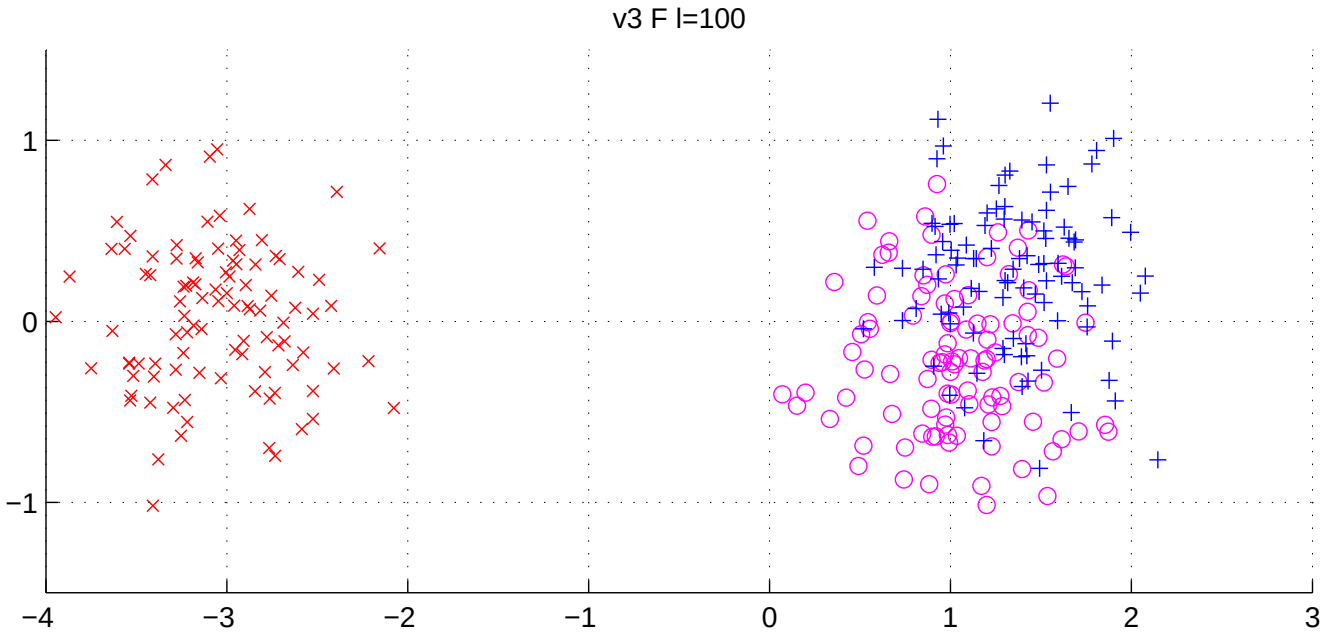
Instead optimize over subspace. Use "extreme" kernels:

	+	o	x
+	+1	0	0
o	0	0	0
x	0	0	0

	+	o	x
+	+1	+1	0
o	+1	+1	0
x	0	0	0

	+	o	x
+	+1	-1	0
o	-1	+1	0
x	0	0	0

Learning the Kernel on the Classes (2)



Toy experiment for learning $k_{\mathcal{Y}}$.

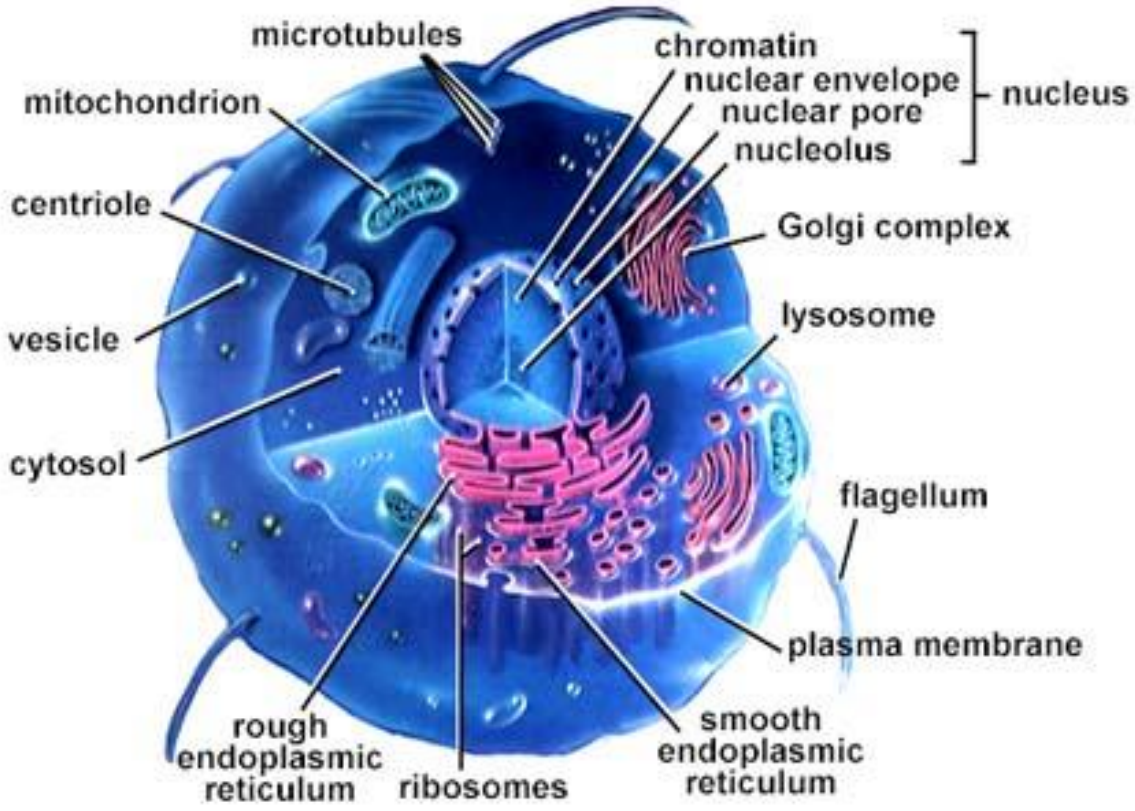
Resulting kernel matrix:

	+	o	x
+	2.0	1.5	-0.4
o	1.5	2.0	0.4
x	-0.4	0.4	2.0

Outline

1. Recap: Support Vector Machines (SVMs)
   - SVMs Do Linear Large Margin Separation
   - Non-Linearity via Kernels
   - SVMs are Perceptrons
2. Application: Predicting Protein Subcellular Localization
3. Multiple Kernel Learning (MKL)
   - A Large Margin MKL Model
   - Optimization for MKL
   - Normalization of Kernels Is Important
   - Multiclass Multiple Kernel Learning
4. Back to App: Predicting Protein Subcellular Localization
5. Take Home Messages

Blue Picture of a Cell



Input: protein sequence

Output: target location of the protein

[image taken from the internet]

List of 69 Kernels

64 = 4*16 Motif kernels

- 4 subsequences (all, last 15, first 15, first 60)
- $16 = 2^{5-1}$ patterns of length 5 (•,○,○,○,○)

3 BLAST similarity kernels

1. linear kernel on E-values
2. Gaussian kernel on E-values, width 1000
3. Gaussian kernel on log E-values, width 1e5

2 phylogenetic kernels

1. linear kernel
2. Gaussian kernel, width 300

Datasets

	TargetP Plant	TargetP Non-Plant	PSORT Gram Pos.	PSORT Gram Neg.
size	940	2732	541	1440
classes	1. chloroplast 2. mitochondria 3. secretory pathway 4. other	1. mitochondria 2. secretory pathway 3. other	1. cytoplasm 2. cytoplasmic membrane 3. cell wall 4. extracellular	1. cytoplasm 2. cytoplasmic membrane 3. periplasm 4. outer membrane 5. extracellular

Performance Measures

per class, count true/false positives/negatives

useful performance measures:

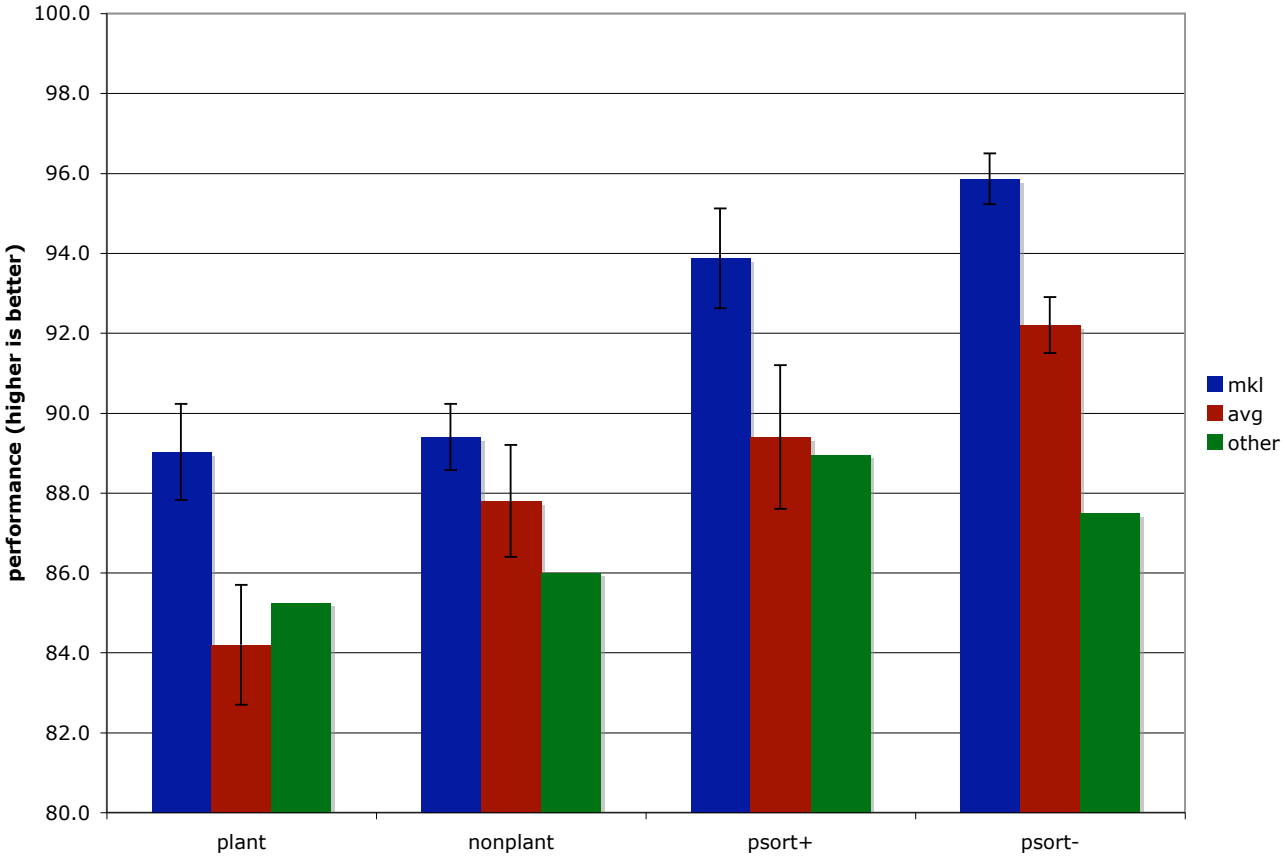
Measure	Formula
Accuracy	$\frac{(TP+TN)}{(TP+TN+FP+FN)}$
Precision	$\frac{TP}{(TP+FP)}$
Recall / Sensitivity	$\frac{TP}{(TP+FN)}$
Specificity	$\frac{TN}{(TN+TP)}$
MCC	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FN)(TP+FP)(TN+FP)(TN+FN)}}$
F1	$\frac{2 * Precision * Recall}{Precision + Recall}$

use weighted averages over classes

Better Than Previous Work

MCC [%] (plant, nonplant)

F1 [%] (psort+, psort-)



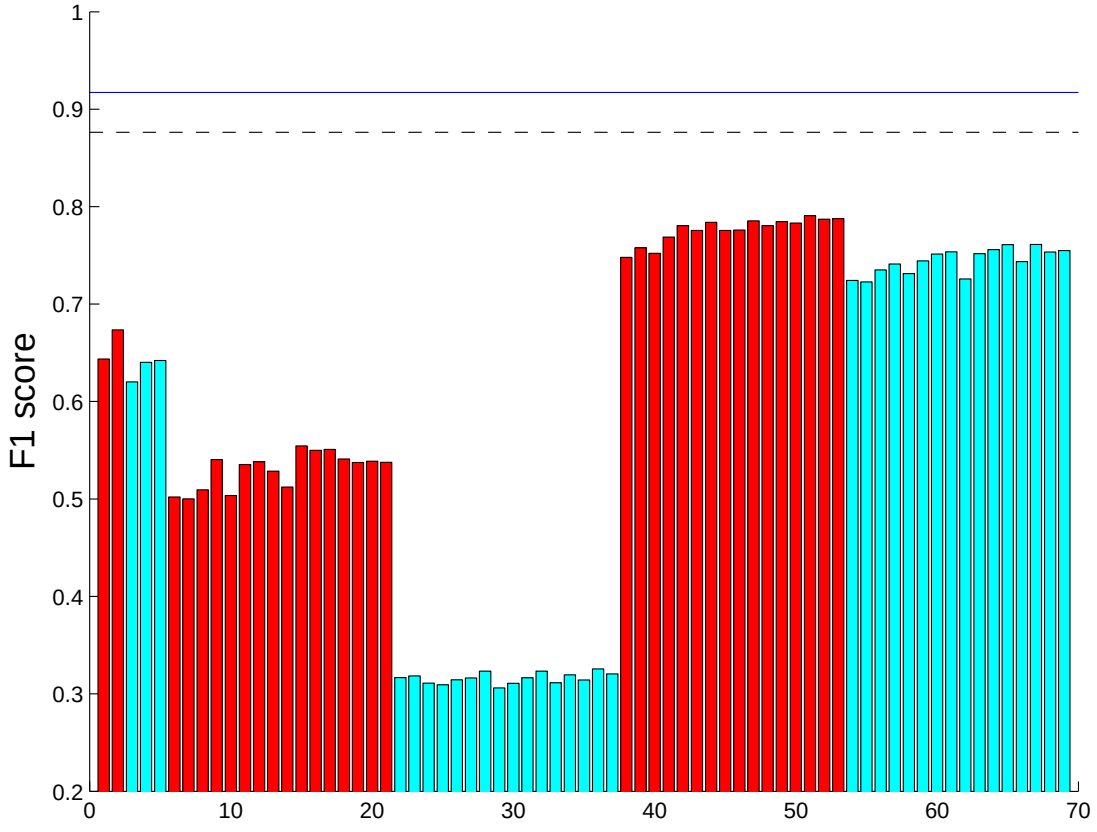
MCMKL unweighted sum of kernels TargetLoc / PSORTb v2.0

Better Than Single Kernels and Than Average Kernel

——, MKL

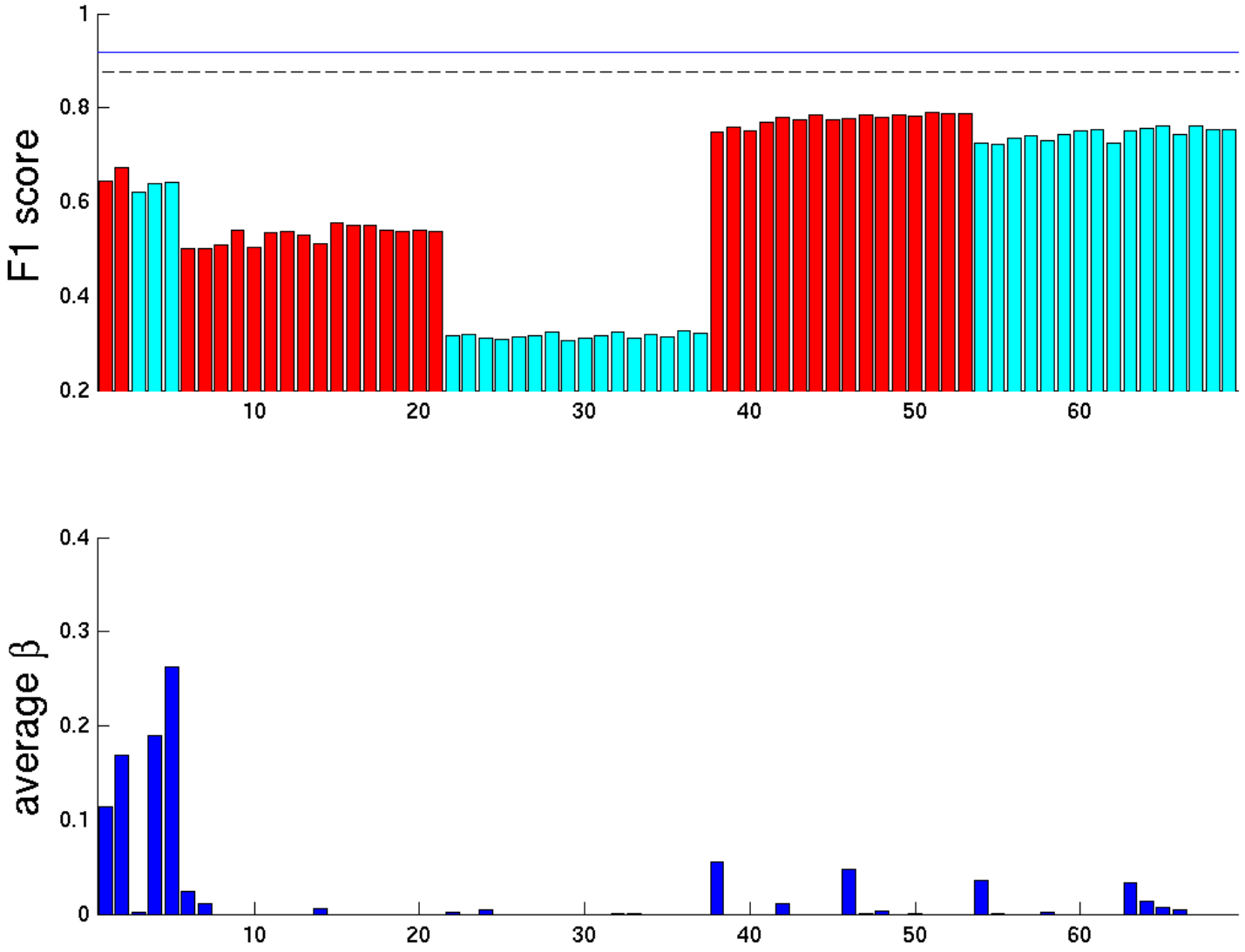
- - - -, sum with uniform weight

bars, single kernel



1. pylogenetic profiles
2. BLAST similarities
3. motifs, complete sequence
4. motifs, last 15 AAs
5. motifs, first 15 AAs
6. motifs, first 60 AAs

Weights $\not\propto$ Single-Kernel Performances



Consistent Sparse Kernel Selection

25 out of 69 kernels selected in 10 repetitions

times selected	mean β_p	kernel
10	26.49%	RBF on log BLAST E-value, $\sigma = 10^5$
10	19.74%	RBF on BLAST E-value, $\sigma = 10^3$
10	16.54%	RBF on inv phyl. profs, $\sigma = 300$
10	11.19%	RBF on lin phyl. profs, $\sigma = 1$
10	5.51%	motif (●,○,○,○,○) on $[1, 15]$
10	4.66%	motif (●,○,○,○,●) on $[1, 15]$
10	3.52%	motif (●,○,○,○,○) on $[1, 60]$
9	3.38%	motif (●,●,○,○,●) on $[1, 60]$
9	2.58%	motif (●,○,○,○,○) on $[1, \infty]$
5	1.32%	motif (●,○,●,○,●) on $[1, 60]$
7	1.06%	motif (●,○,○,●,○) on $[1, 15]$
7	0.93%	motif (●,●,○,○,○) on $[1, \infty]$
5	0.62%	motif (●,○,○,○,●) on $[1, \infty]$
3	0.52%	motif (●,●,●,○,●) on $[1, 60]$
2	0.41%	motif (●,○,○,●,●) on $[1, 60]$
6	0.40%	motif (●,○,●,○,○) on $[-15, \infty]$
7	0.27%	motif (●,○,○,○,○) on $[-15, \infty]$
3	0.26%	motif (●,○,●,○,●) on $[1, 15]$
2	0.18%	motif (●,○,○,●,○) on $[1, 60]$
3	0.12%	linear kernel on BLAST E-value
2	0.12%	motif (●,○,○,●,●) on $[1, 15]$
2	0.10%	motif (●,○,●,○,●) on $[-15, \infty]$
1	0.06%	motif (●,●,●,○,●) on $[-15, \infty]$
1	0.03%	motif (●,●,○,○,○) on $[1, 60]$
1	0.02%	motif (●,●,○,○,●) on $[1, 15]$

Biologically Meaningful Motifs

times selected	mean β_p	kernel (PSORT+)
10	6.23%	motif (•,○,○,○,○) on $[1, \infty]$
10	3.75%	motif (•,○,•,○,•) on $[1, \infty]$
9	2.24%	motif (•,○,•,•,•) on $[1, 60]$
10	1.32%	motif (•,○,○,○,•) on $[1, 15]$
8	0.53%	motif (•,○,○,○,○) on $[1, 15]$

times selected	mean β_p	kernel (plant)
10	5.50%	motif (•,○,○,○,○) on $[1, 15]$
10	4.68%	motif (•,○,○,○,•) on $[1, 15]$
10	3.48%	motif (•,○,○,○,○) on $[1, 60]$
8	3.17%	motif (•,•,○,○,•) on $[1, 60]$
9	2.56%	motif (•,○,○,○,○) on $[1, \infty]$

Outline

1. Recap: Support Vector Machines (SVMs)
   - SVMs Do Linear Large Margin Separation
   - Non-Linearity via Kernels
   - SVMs are Perceptrons
2. Application: Predicting Protein Subcellular Localization
3. Multiple Kernel Learning (MKL)
   - A Large Margin MKL Model
   - Optimization for MKL
   - Normalization of Kernels Is Important
   - Multiclass Multiple Kernel Learning
4. Back to App: Predicting Protein Subcellular Localization
5. Take Home Messages

What You Should Take Home From This Lecture

- SVMs — mere but "clever" perceptrons — can be very good
  - use huge numbers of features with kernels
  - a practical advantage is convexity
- MKL can be seen as two-layer perceptron
  - convexity can be retained
  - sparse solutions can be enforced ($\Rightarrow$ understanding)
  - can be built on existing single-kernel code
  - learned kernel weights $\boldsymbol{\beta}$ hard to beat manually
- be aware of normalization

Questions?

Further Reading

- presented work: `http://www.fml.tuebingen.mpg.de/raetsch/projects/protsubloc`
  - A. Zien and C. S. Ong. Multiclass multiple kernel learning. ICML 2007.
  - C. S. Ong and A. Zien. An Automated Combination of Kernels for Predicting Protein Subcellular Localization. WABI 2008.
- the beginnings of MKL:
  - G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. JMLR, 2004.
  - G. Lanckriet, T. De Bie, N. Cristianini, M. I. Jordan, and W. Stafford Noble. A statistical framework for genomic data fusion. Bioinfomatics, 2004.
- efficient optimization:
  - F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple Kernel Learning, Conic Duality, and the SMO Algorithm. ICML 2004.
  - S. Sonnenburg, G. Rätsch, and C. Schäfer. A General and Efficient Multiple Kernel Learning Algorithm. NIPS, 2006.
  - A. Rakotomamonjy, F. R. Bach, S. Canu, Y. Grandvalet. More Efficiency in Multiple Kernel Learning. ICML 2007.
- Fisher discriminant analysis with multiple kernels:
  - J. Ye, S. Ji, and J. Chen. Learning the kernel matrix in discriminant analysis via quadratically constrained quadratic programming. SIGKDD 2007.
- in statistics literature:
  - Y. Lee, Y. Kim, S. Lee, and J.-Y. Koo. Structured multicategory support vector machines with analysis of variance decomposition. Biometrika, 2006.
  - M. Yuan, Y. Lin. Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society, 2006.
- many more...