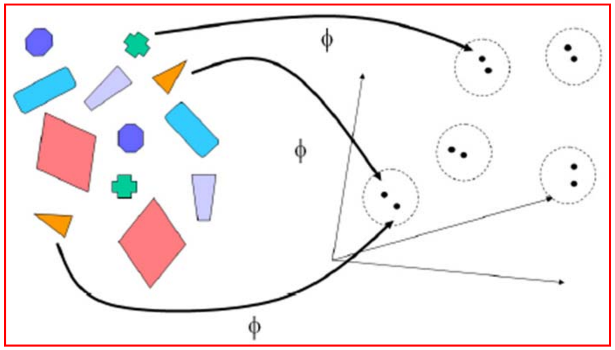
# Advanced Introduction to Machine Learning

**10715, Fall 2014**

## The Kernel Trick, Reproducing Kernel Hilbert Space, and the Representer Theorem





**Eric Xing**


**Lecture 6, September 24, 2014**

**Reading:**

# Recap: the SVM problem

- We solve the following constrained opt problem:

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is a **quadratic programming** problem.
  - A global maximum of $\alpha_i$ can always be found.
  - The solution: $w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$
  - How to predict: $\mathbf{w}^T \mathbf{x}_{\text{new}} + b \lessgtr 0$

$$\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\mathbf{w}^T \mathbf{x}_{\text{new}} + b \lessgtr 0$$

- Kernel
- Point rule or average rule
- Can we predict vec(y)?

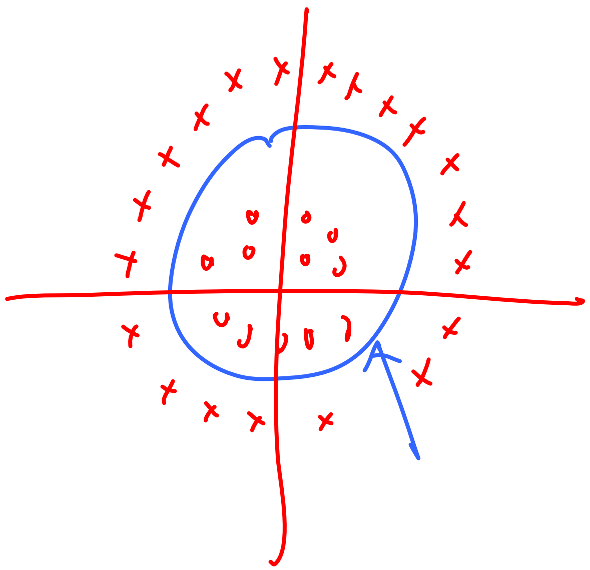
# Outline

- The Kernel trick
- Maximum entropy discrimination
- Structured SVM, aka, Maximum Margin Markov Networks

# (1) Non-linear Decision Boundary

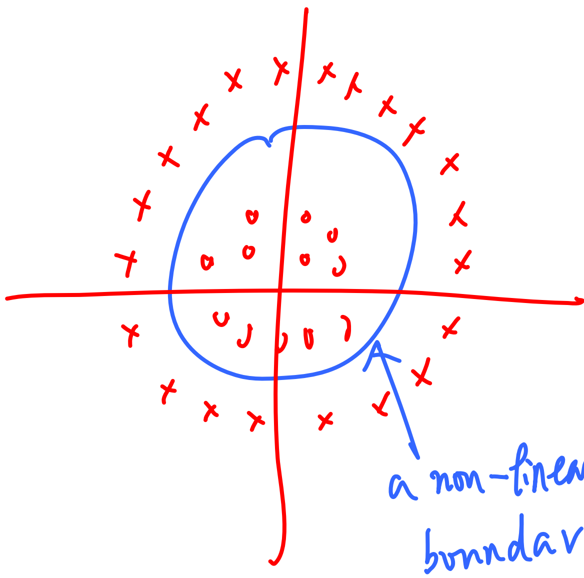
- So far, we have only considered large-margin classifier with a linear decision boundary
- How to generalize it to become nonlinear?
- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - Classification can become easier with a proper transformation. In the XOR problem, for example, adding a new feature of $x_1x_2$ make the problem linearly separable (homework)

# Non-linear Decision Boundary



# Non-linear Decision Boundary

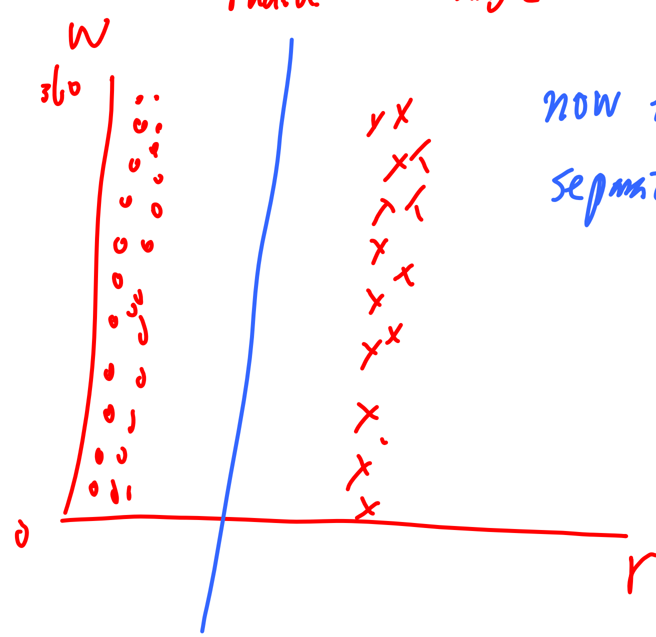
This data set is not linearly separable!




transformation ⟹

$(x_1, x_2) \rightarrow (r(x_1, x_2),\ w(x_1, x_2))$

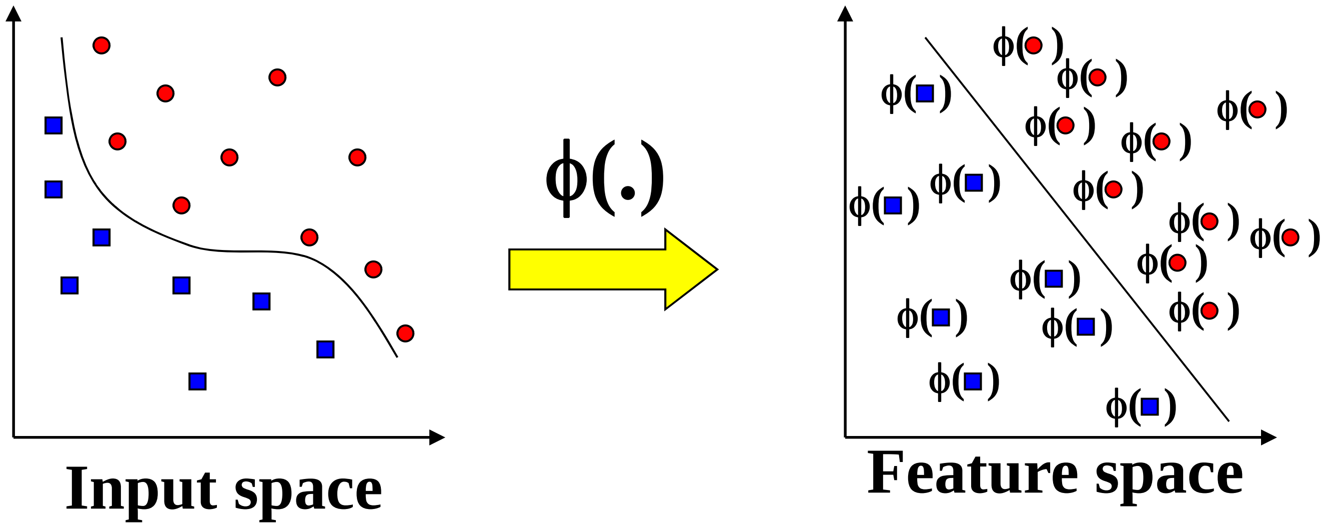
↑ radius ↑ angle




now linearly separable!

How to find a useful and inexpensive transformation?

# Transforming the Data




Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

# The Kernel Trick

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

# An Example for feature mapping and kernels

- Consider an input $\mathbf{x}=[x_1,x_2]$
- Suppose $\phi$(.) is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

- So, if we define the **kernel function** as follows, there is no need to carry out $\phi$(.) explicitly

$$K(\mathbf{x},\mathbf{x}') = \left(1 + \mathbf{x}^T\mathbf{x}'\right)^2$$

# More examples of kernel functions

- Linear kernel (we've seen it)

$$K(\mathbf{x},\mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x},\mathbf{x}') = \left(1 + \mathbf{x}^T \mathbf{x}'\right)^p$$

where $p$ = 2, 3, … To get the feature vectors we concatenate all $p$th order polynomial terms of the components of x (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\left\|\mathbf{x}-\mathbf{x}'\right\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.

# The essence of kernel

- Feature mapping, but "without paying a cost"
  - E.g., polynomial kernel

$$K(x,z) = (x^T z + c)^d$$

  - How many dimensions we've got in the new space?
  - How many operations it takes to compute K()?

- Kernel design, any principle?
  - K(x,z) can be thought of as a similarity function between x and z
  - This intuition can be well reflected in the following "Gaussian" function (Similarly one can easily come up with other K() in the same spirit)

$$K(x,z) = \exp\left(-\frac{||x-z||^2}{2\sigma^2}\right)$$

  - Is this necessarily lead to a "legal" kernel? (in the above particular case, K() is a legal one, do you know how many dimension $\phi$(x) is?

# Kernel matrix

- Suppose for now that $K$ is indeed a valid kernel corresponding to some feature mapping $\phi$, then for $x_1, \ldots, x_m$, we can compute an $m \times m$ matrix $K = \{K_{i,j}\}$, where $K_{i,j} = \phi(x_i)^T \phi(x_j)$
- This is called a kernel matrix!
- Now, if a kernel function is indeed a valid kernel, and its elements are dot-product in the transformed feature space, it must satisfy:
  - Symmetry $K=K^T$

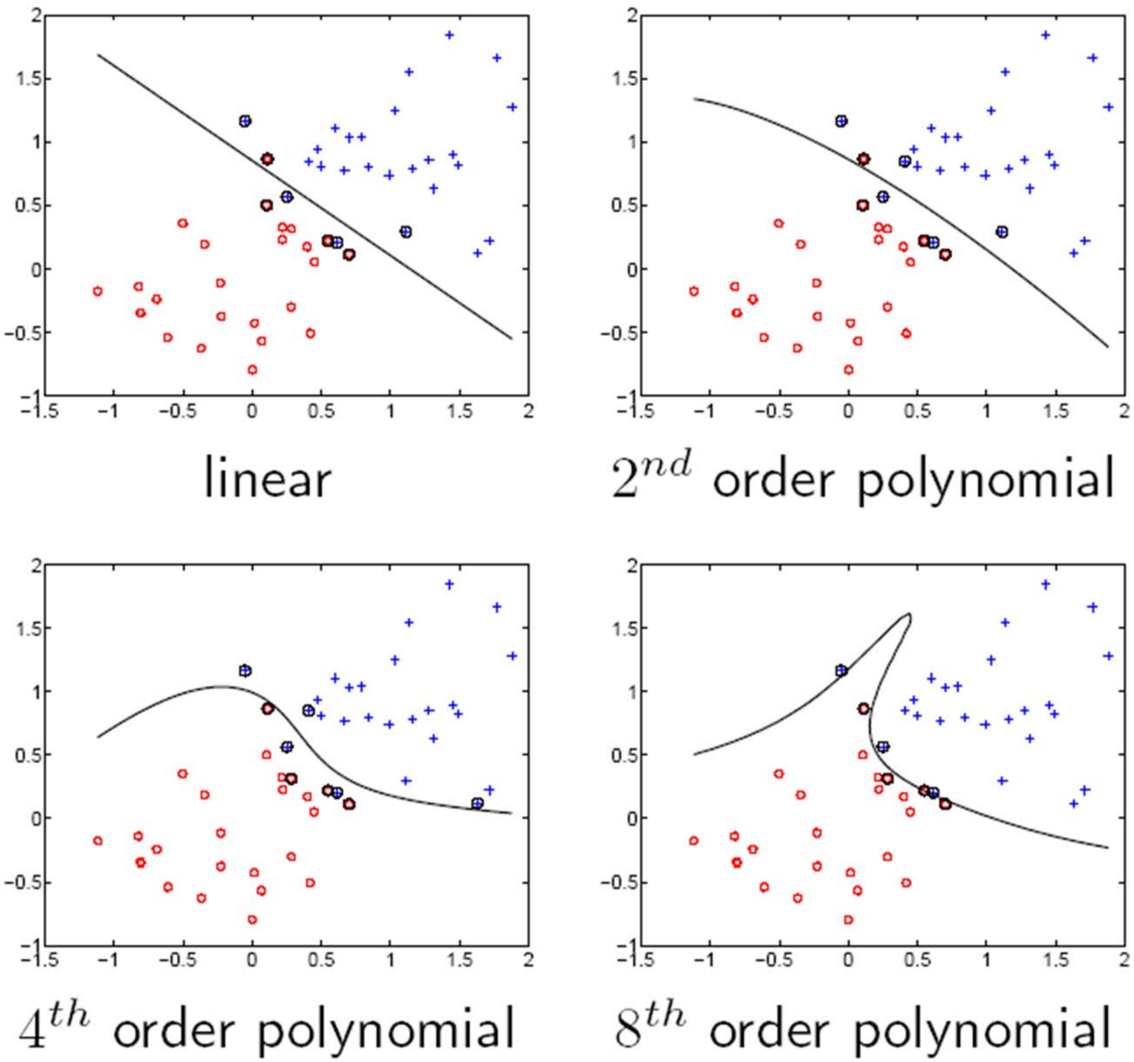
    proof $K_{i,j} = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = K_{j,i}$
  - Positive –semidefinite $y^T K y \geq 0 \quad \forall y$

    proof?

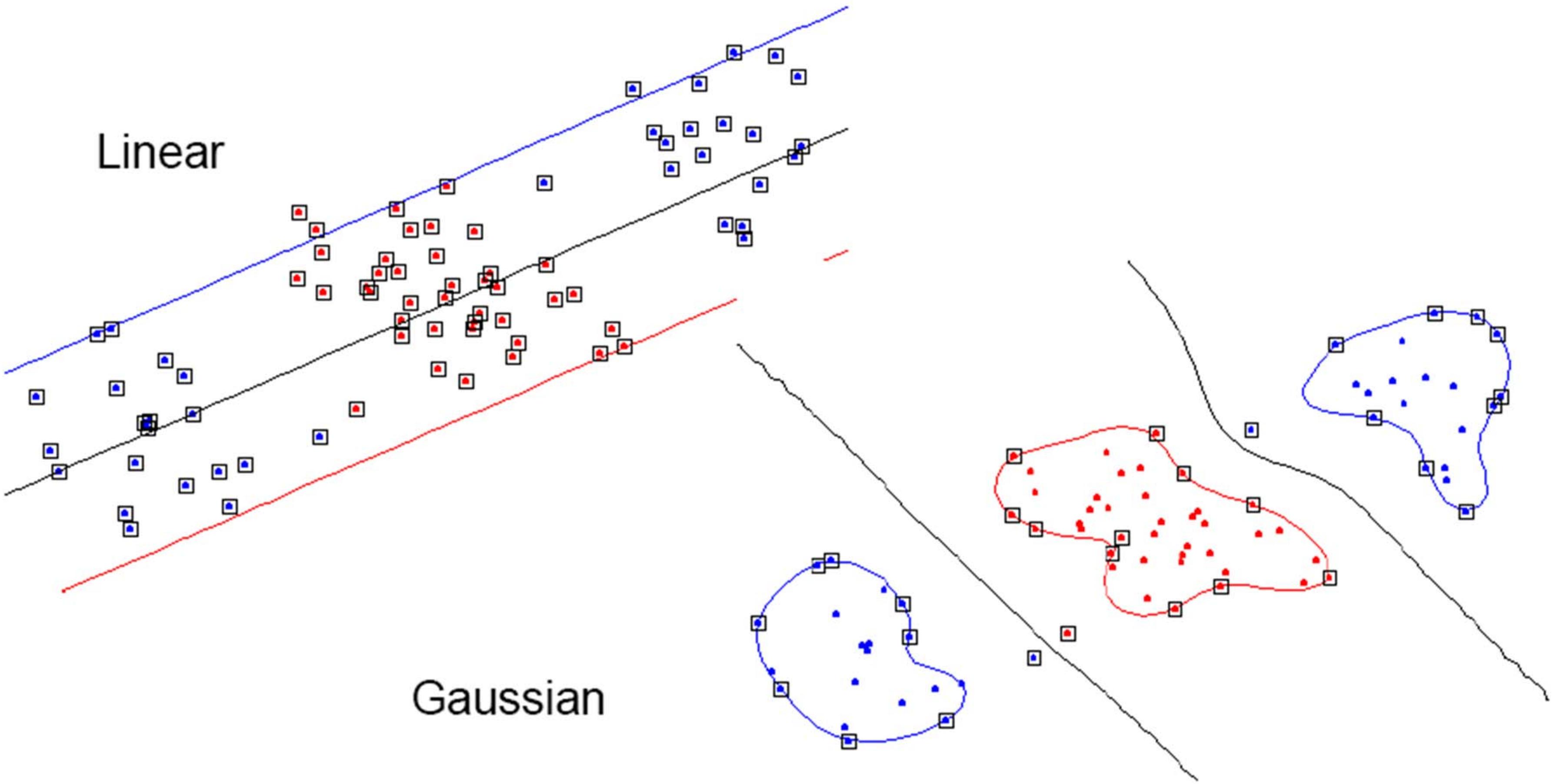
# Mercer kernel

**Theorem (Mercer)**: Let $K$: $\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be given. Then for $K$ to be a valid (Mercer) kernel, it is necessary and sufficient that for any $\{x_i, \ldots, x_m\}$, $(m < \infty)$, the corresponding kernel matrix is symmetric positive semi-denite.

# SVM examples



# Examples for Non Linear SVMs – Gaussian Kernel



# Remember the Kernel Trick!!!

**Primal Formulation:**

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_j \xi$$

$$(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}_j) + b)y_j \geqslant 1 - \xi_j \quad \forall j$$

$$\xi_j \geqslant 0 \quad \forall j$$

**Infinite, cannot be directly computed**

**But the dot product is easy to compute ☺**

**Dual Formulation:**

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{\phi}(\boldsymbol{x}_i)^\top \boldsymbol{\phi}(\boldsymbol{x}_i)$$

$$\sum_i \alpha_i y_i = 0$$

$$0 \leqslant \alpha_i \leqslant C \quad \forall i$$

# Overview of Hilbert Space Embedding

- Create an infinite dimensional statistic for a distribution.

- Two Requirements:
  - Map from distributions to statistics is one-to-one
  - Although statistic is infinite, it is cleverly constructed such that the kernel trick can be applied.

- Perform Belief Propagation as if these statistics are the conditional probability tables.

- We will now make this construction more formal by introducing the concept of Hilbert Spaces

# Vector Space

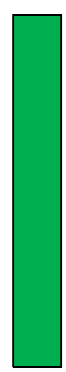
- A set of objects closed under linear combinations (e.g., addition and scalar multiplication):

$$\boldsymbol{v}, \boldsymbol{w} \in \mathcal{V} \implies \alpha \boldsymbol{v} + \beta \boldsymbol{w} \in \mathcal{V}$$

- Obeys distributive and associative laws,

- Normally, you think of these "objects" as finite dimensional vectors. However, in general the objects can be functions.
  - **Nonrigorous Intuition**: A function is like an infinite dimensional vector.

$$\boldsymbol{f} =$$



# Hilbert Space

- A Hilbert Space is a complete vector space equipped with an inner product.

- The inner product $\langle \boldsymbol{f}, \boldsymbol{g} \rangle$ has the following properties:
  - Symmetry $\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \langle \boldsymbol{g}, \boldsymbol{f} \rangle$
  - Linearity $\langle \alpha \boldsymbol{f}_1 + \beta \boldsymbol{f}_2, \boldsymbol{g} \rangle = \alpha \langle \boldsymbol{f}_1, \boldsymbol{g} \rangle + \beta \langle \boldsymbol{f}_2, \boldsymbol{g} \rangle$
  - Nonnegativity $\langle \boldsymbol{f}, \boldsymbol{f} \rangle \geqslant 0$
  - Zero $\langle \boldsymbol{f}, \boldsymbol{f} \rangle = 0 \implies \boldsymbol{f} = 0$

- Basically a “nice” infinite dimensional vector space, where lots of things behave like the finite case
  - e.g. using inner product we can define “norm” or “orthogonality”
  - e.g. a norm can be defined, allows one to define notions of convergence

# Hilbert Space Inner Product

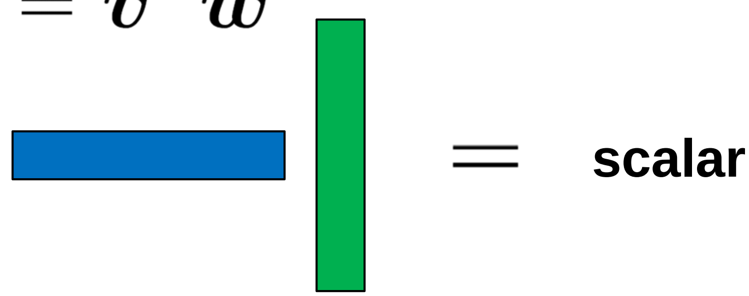
- Example of an inner product (just an example, inner product not required to be an integral)

$$\langle \boldsymbol{f}, \boldsymbol{g} \rangle = \int \boldsymbol{f}(x)\boldsymbol{g}(x)\, dx$$

**Inner product of two functions is a number**

- Traditional finite vector space inner product

$$\langle \boldsymbol{v}, \boldsymbol{w} \rangle = \boldsymbol{v}^\top \boldsymbol{w}$$

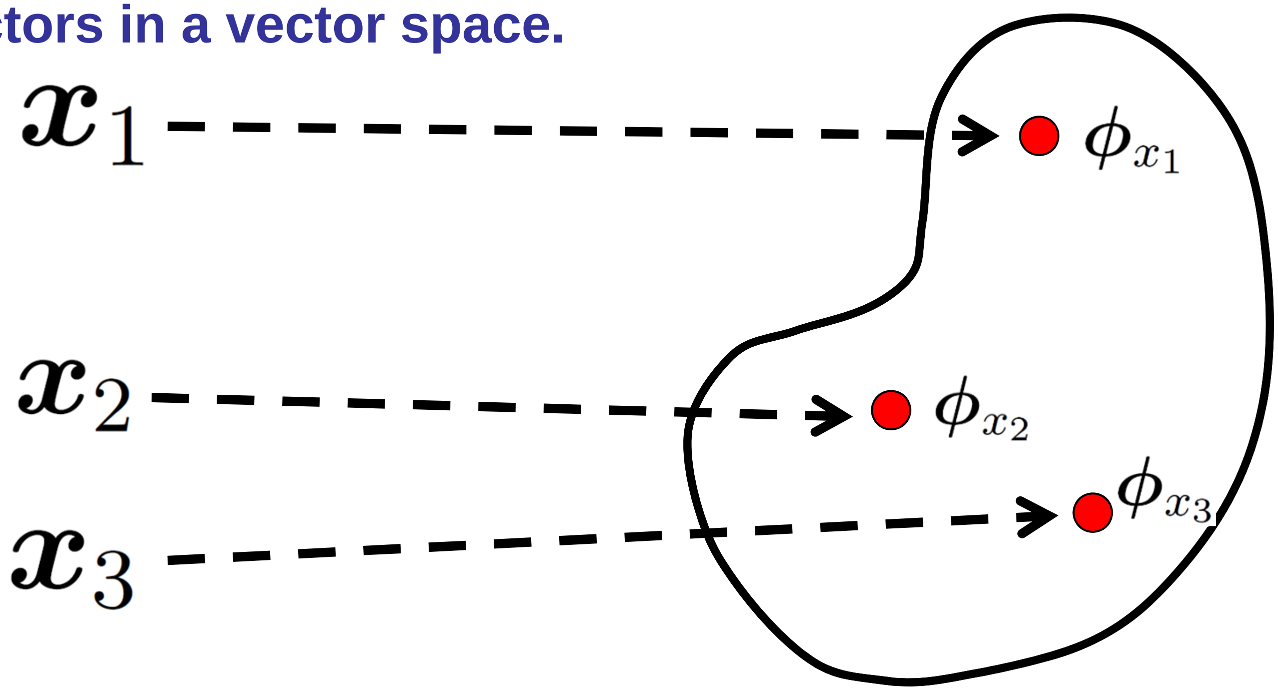


# Recall the SVM kernel Intuition

$$\min_{\boldsymbol{w},b} \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C\sum_j \xi$$

$$(\boldsymbol{w}^\top \phi(\boldsymbol{x}_j) + b)y_j \geqslant 1 - \xi_j \quad \forall j \qquad \xi_j \geqslant 0 \quad \forall j$$

**Maps data points to Feature Functions, which corresponds to some vectors in a vector space.**



# The Feature Function

- Consider holding one element of the kernel fixed. We get a function of one variable which we call the feature function. The collection of feature functions is called the **feature map**.

$$\phi_x := \boldsymbol{K}(x, \cdot)$$

- For a Gaussian Kernel the feature functions are unnormalized Gaussians:

$$\phi_1(y) = \exp\left(\frac{\|1 - y\|_2^2}{\sigma^2}\right)$$

$$\phi_{1.5}(y) = \exp\left(\frac{\|1.5 - y\|_2^2}{\sigma^2}\right)$$

# Reproducing Kernel Hilbert Space

- Given a kernel $k(x,x')$, we now construct a Hilbert space such that $k$ defines an inner product in that space

  - We begin with a kernel map:

$$\Phi \ : \ x \rightarrow k(\cdot, x)$$

  - We now construct a vector space containing all linear combinations of the functions $k( ,x)$:

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$$

  - We now **define** an inner product. Let $g(\cdot) = \sum_{j=1}^{m'} \beta_j k(\cdot, x'_j)$

    we have

$$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j k(x_i, x'_j)$$

please verify this in fact is an inner product: satisfying symmetry, linearity, and zero-norm law : $\langle f, f \rangle = 0 \ \Rightarrow f = 0$

(here we need "reproducing property", and Cauchy-Schwartz inequality

# Reproducing Kernel Hilbert Space

- The $k(\ ,x)$ is a **reproducing** kernel map:

$$\langle k(\cdot,x), f\rangle = \sum_{i=1}^{m} \alpha_i k(x,x_i) = f(x)$$

  - This shows that the kernel is a *representer of evaluation (or, evaluation function)*
  - This is analogous to the Dirac delta function.
  - If we plug in the kernel in for f: $\langle k(\cdot,x), k(\cdot,x')\rangle = k(x,x')$

$\int F(x)\, \delta(x, \hat{x})\, dx = f(\hat{x})$

- With such a definition of inner product, we have constructed a subspace of the Hilbert space --- a **reproducing kernel Hilbert space** (RKHS)

# Mercer's theorem and RKHS

- Recall the following condition for Mercer's theorem for K

$$\int\int \boldsymbol{K}(x,y)\boldsymbol{f}(x)\boldsymbol{f}(y)\,dx\,dy > 0 \quad \forall \boldsymbol{f}$$

- We can also "construct" our Reproducing Kernel Hilbert Space with a **Mercer Kernel**, as a linear combination of its eigen-functions:

$$\int k(x,x')\phi_i(x') = \sum_{j=1}^{\infty} \lambda\phi_j(x)$$

which can be shown to entail reproducing property (homework?)

# Summary: RKHS

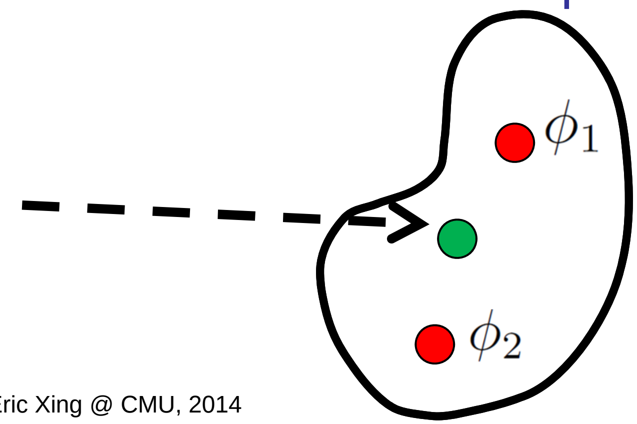
- Consider the set of functions that can be formed with linear combinations of these feature functions:

$$\mathcal{F}_0 = \left\{ f(z) : \sum_{j=1}^{k} \alpha_j \phi_{x_j}(z), \forall k \in \mathbb{N}_+ \text{ and } x_j \in \mathcal{X} \right\}$$

- We define the Reproducing Kernel Hilbert Space $\mathcal{F}$ to the completion of $\mathcal{F}_0$ (like $\mathcal{F}_0$ with the "holes" filled in)

- Intuitively, the feature functions are like an over-complete basis for the RKHS

$$\boldsymbol{f}(z) = \alpha_1 \phi_1(z) + \alpha_2 \phi_2(z)$$

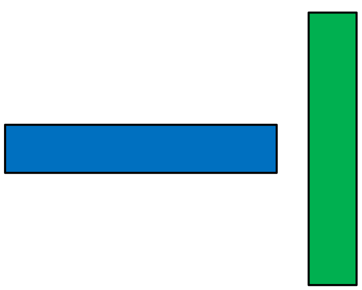


# Summary: Reproducing Property

- It can now be derived that the inner product of a function **f** with $\boldsymbol{\phi}_X$, evaluates a function at point ***x***:

$$
\begin{aligned}
\langle \boldsymbol{f}, \boldsymbol{\phi}_x \rangle &= \left\langle \sum_j \alpha_j \boldsymbol{\phi}_{x_j}, \boldsymbol{\phi}_x \right\rangle \\
&= \sum_j \alpha_j \langle \boldsymbol{\phi}_{x_j}, \boldsymbol{\phi}_x \rangle \qquad \text{Linearity of inner product} \\
&= \sum_j \alpha_j \boldsymbol{K}(x_j, x) \qquad \text{Definition of kernel} \\
&= \boldsymbol{f}(x)
\end{aligned}
$$

**Remember that**
$\boldsymbol{K}(x_j, x) := \boldsymbol{\phi}_{x_j}(x)$



$=$ **scalar**

# Summary: Evaluation Function

- A Reproducing Kernel Hilbert Space is an Hilbert Space where for any ***X***, the evaluation functional indexed by ***X*** takes the following form:

$$\text{Eval}_X(\cdot) = \langle \phi_X , \rangle$$

**Evaluation Function, must be a function in the RKHS**

**Same evaluation function for different functions (but same point)**

$$\boldsymbol{f}(X_1) = \langle \phi_{X_1}, \boldsymbol{f} \rangle$$
$$\boldsymbol{g}(X_1) = \langle \phi_{X_1}, \boldsymbol{g} \rangle$$

**Different points are associated with different evaluation functions**

$$\boldsymbol{f}(X_2) = \langle \phi_{X_2}, \boldsymbol{f} \rangle$$
$$\boldsymbol{g}(X_2) = \langle \phi_{X_2}, \boldsymbol{g} \rangle$$

- Equivalent (More Technical) Definition: An RKHS is a Hilbert Space where the evaluation functionals are bounded. (The previous definition then follows from Riesz Representation Theorem)

# RKHS or Not?

- Is the vector space of 3 dimensional real valued vectors an RKHS?

**Yes!!!**

$$\mathrm{Eval}_i(\cdot) = \langle \boldsymbol{e}_i, \cdot \rangle$$

**Homework !**

# RKHS or Not?

- Is the space of functions such that

$$\int |\boldsymbol{f}(z)|^2 \, dz < \infty$$

an RKHS?

**No!!!!**

**Homework !**

But, can't the evaluation functional be an inner product with the delta function?

$$\mathrm{Eval}_X(\cdot) = \langle \delta_X, \cdot \rangle$$

$$\boldsymbol{f}(X) = \int \boldsymbol{f}(z)\delta_X(z) \, dz$$

**The problem is that the delta function is not in my space!**

# The Kernel

$k(x_i, x_j)$

- I can evaluate my evaluation function with another evaluation function!

$$k(X_1, X_2) := \phi_{X_1}(X_2) = \phi_{X_2}(X_1) = \langle \phi_{X_1}, \phi_{X_2} \rangle = \int \phi_{X_1}(z)\phi_{X_2}(z)\, dz$$

- Doing this for all pairs in my dataset gives me the Kernel Matrix **K**:

$$\boldsymbol{K} = \begin{pmatrix} k(X_1, X_1) & k(X_1, X_2) & k(X_1, X_3) \\ k(X_1, X_2) & k(X_1, X_2) & k(X_1, X_3) \\ k(X_1, X_1) & k(X_1, X_2) & k(X_1, X_3) \end{pmatrix}$$

- There may be infinitely many evaluation functions, but I only have a finite number of training points, so the kernel matrix is finite!!!!

# Correspondence between Kernels and RKHS

- A kernel is positive semi-definite if the kernel matrix is positive semidefinite for any choice of finite set of observations.

- **Theorem (Moore-Aronszajn):** Every positive semi-definite kernel corresponds to a unique RKHS, and every RKHS is associated with a unique positive semi-definite kernel.

- Note that the kernel does not uniquely define the feature map (but we don't really care since we never directly evaluate the feature map anyway).

# RKHS norm and SVM

- Recall that in SVM:

$$f(\cdot) = \langle w, x \rangle = \sum_{i=1}^{m} \alpha_i y_i k(\cdot, x_i)$$

Therefore $f(\cdot) \in \mathcal{H}$

Moreover: $$\|f(\cdot)\|_{\mathcal{H}}^2 = \langle \sum_{i=1}^{m} \alpha_i y_i k(\cdot, x_i), \sum_{j=1}^{m} \alpha_j y_j k(\cdot, x_j) \rangle$$

$$=$$

# Primal and dual SVM objective

- In our primal problem, we minimize $w^T w$ subject to constraints. This is equivalent to:

$$\begin{aligned} \|w\|^2 &= w^T w = \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \Phi(x_i)\Phi(x_j)\rangle \\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ &= \|f\|_{\mathcal{H}}^2 \end{aligned}$$

which is equivalent to minimizing the Hilbert norm of f subject to constraints

# The Representer Theorem

- In the general case, for a primal problem P of the form:

$$\min_{f \in \mathcal{H}} \{ C(f, \{x_i, y_i\}) + \Omega(\|f\|_{\mathcal{H}}) \}$$

  where $\{x_i, y_i\})_{i=1}^{m}$ are the training data.

  If the following conditions are satisfied:
  - The loss function C is point-wise, i.e., $C(f, \{x_i, y_i\}) = C(\{x_i, y_i, f(x_i)\})$
  - $\Omega(\cdot)$ is monotonically increasing
- The representer theorem (Kimeldorf and Wahba, 1971):

  every minimizer of P admits a representation of the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, x_i)$$

i.e., a linear combination of (a finite set of) function given by the data

# Proof of Representer Theorem

$\Omega(f) \Rightarrow \Omega(\|f\|_H^2)$

$$f = f_{\parallel} + f_{\perp}$$

$$= \sum_{i=1}^{m} \alpha_i k(\cdot, x_i) + f_{\perp}(\cdot)$$

$x_i, \; i = 1 \dots m$

$k(\cdot, x_i)$

$H \to H_D, \; H_{D^c}$

$$f(x_j) = \langle f(\cdot), k(\cdot, x_j) \rangle$$

$$= \langle \sum \alpha_i k(\cdot, x_i) + f_{\perp}(\cdot), k(\cdot, x_j) \rangle$$

$$= \sum \alpha_i k(x_i, x_j) + 0$$

$$= \sum \alpha_i k(x_i, x_j)$$

$(C(f(x_i)) + \Omega)$

$$f^* = \sum \alpha_i k(\cdot, x_i)$$

$$\Omega(\|f\|_H^2) = \Omega\left(\left\|\sum_{i=1}^{m} \alpha_i k(\cdot, x_i)\right\|_H^2 + \|f_{\perp}\|_H^2\right)$$

$\|f_{\perp}\|_H^2 \geq 0$

$\min_{f \in H} \Omega, \quad f_{\parallel} \in H_D, \; f_{\perp} \in H_{D^c}$

# Another view of SVM

- Q: why SVM is “dual-sparse”, i.e., having a few support vectors (most of the $\alpha$’s are zero).
  - The SVM loss $w^T w$ does not seem to imply that
  - And the representer theorem does not either!

$f = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$

# Another view of SVM: $L_1$ regularization

- The basis-pursuit denoising cost function (chen & Donoho):

$$J(\alpha) = \frac{1}{2}\|f(\cdot) - \sum_{i=1}^{N} \alpha_i \phi_i()\|_{L_2}^2 + \lambda \|\alpha\|_{L_1}$$

- Instead we consider the following modified cost:

$$J(\alpha) = \frac{1}{2}\sum \|f(\cdot) - \sum_{i=1}^{N} \alpha_i K(\cdot, x_i)\|_{\mathcal{H}}^2 + \lambda \|\alpha\|_{L_1}$$

# RKHS norm interpretation of SVM

$$J(\alpha) = \frac{1}{2}\sum \|f(\cdot) - \sum_{i=1}^{N} \alpha_i K(\cdot, x_i)\|_{\mathcal{H}}^2 + \lambda \|\alpha\|_{L_1}$$

- The RKHS norm of the first term can now be computed exactly!

# RKHS norm interpretation of SVM

- Now we have the following optimization problem:

$$\min_{\alpha} \left\{ -\sum_i \alpha_i y_i + \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) + \sum_i \lambda |\alpha_i| \right\}$$

This is exactly the dual problem of SVM!

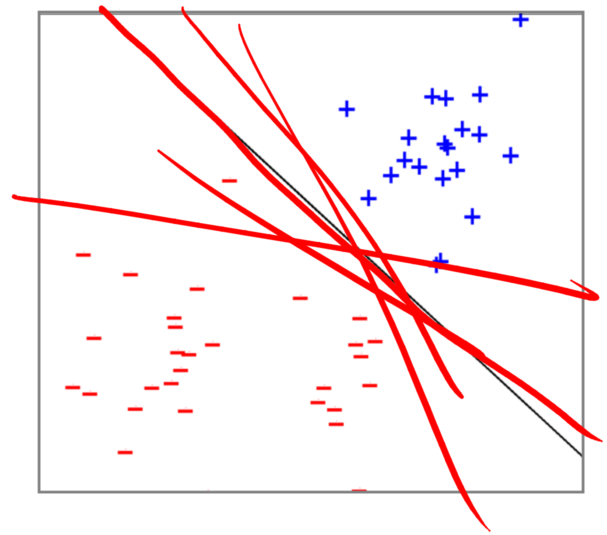
s.t $0 \le \alpha \le C$

# Take home message

- Kernel is a (nonlinear) feature map into a Hilbert space
- Mercer kernels are "legal"
- RKHS is a Hilbert space equipped with an "inner product" operator defined by mercer kernel
- Reproducing property make kernel works like an evaluation function
- Representer theorem ensures optimal solution to a general class of loss function to be in the Hilbert space
- SVM can be recast as an L1-regularized minimization problem in the RKHS

# (2) Model averaging

- Inputs $x$, class $y = +1, -1$
- data $D = \{ (x_1,y_1), \ldots (x_m,y_m) \}$



- Point Rule:
  - learn $f^{opt}(x)$ discriminant function from $F = \{f\}$ family of discriminants
  - classify $y = \text{sign}\, f^{opt}(x)$

$w^*$ $P(w)$

$q(w) = P(w)\,P(X|w)$

- E.g., SVM

$$f^{\text{opt}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}_{\text{new}} + b$$

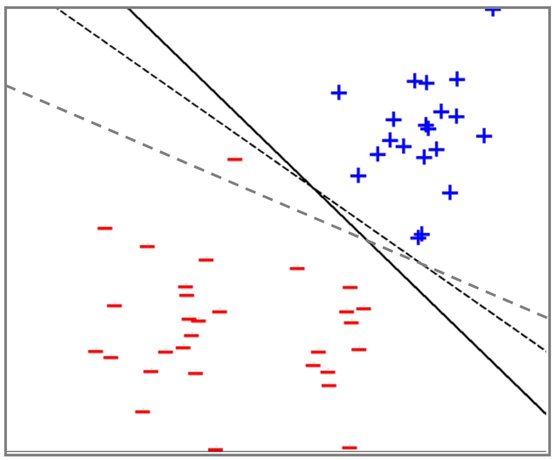
# Model averaging

- There exist many $f$ with near optimal performance

- Instead of choosing $f^{opt}$, average over all $f$ in $F$

  $Q(f)$ = weight of $f$

$$y(x) = \text{sign} \int_F Q(f) f(x) df$$
$$= \text{sign}\langle f(x)\rangle_Q$$



- How to specify:
  $F = \{ f \}$ family of discriminant functions?

- How to learn $Q(f)$ distribution over $F$?

# Recall Bayesian Inference

- Bayesian learning:


$$\text{Bayes Thrm}: p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathbf{w})p(\mathcal{D}|\mathbf{w})}{p(\mathcal{D})}$$

- Bayes Predictor (model averaging):

$$h_1(\mathbf{x}; p(\mathbf{w})) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} \int p(\mathbf{w}) f(\mathbf{x}, \mathbf{y}; \mathbf{w}) d\mathbf{w}$$

**Recall in SVM:** $h_0(\mathbf{x}; \mathbf{w}) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} F(\mathbf{x}, \mathbf{y}; \mathbf{w})$

- What $p_0$?

# How to score distributions?

- Entropy
  - Entropy *H(X)* of a random variable *X*

$$H(X) = -\sum_{i=1}^{N} P(x=i)\log_2 P(x=i)$$

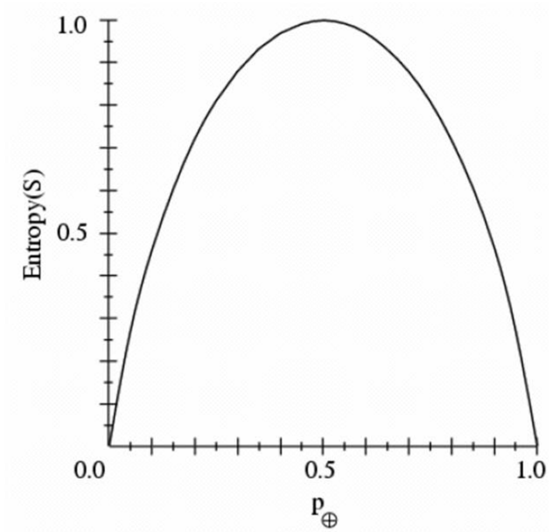
  - *H(X)* is the expected number of bits needed to encode a randomly drawn value of *X* (under most efficient code)
  - Why?

    Information theory:

    Most efficient code assigns $-\log_2 P(X=i)$ bits to encode the message *X=I*, So, expected number of bits to code one random *X* is:

$$-\sum_{i=1}^{N} P(x=i)\log_2 P(x=i)$$

# Sample Entropy




- $S$ is a sample of training examples
- $p_+$ is the proportion of positive examples in $S$
- $p_-$ is the proportion of negative examples in $S$
- Entropy measure the impurity of $S$

$$H(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

# More definitions on entropy

- Conditional Entropy
  - Specific conditional entropy *H(X|Y=v)* of *X* given *Y=v* :

$$H(X|y=j) = -\sum_{i=1}^{N} P(x=i|y=j) \log_2 P(x=i|y=j)$$

  - Conditional entropy *H(X|Y)* of *X* given *Y* :

$$H(X|Y) = -\sum_{j \in Val(y)} P(y=j) \log_2 H(X|y=j)$$

  - Mututal information (aka information gain) of *X* and *Y* :

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) = H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned}$$

# Relative Entropy

- How to measure similarity between two distributions?

$$D(q,p) = \sum_x Q(X=x) \log \frac{Q(X=x)}{P(X=x)}$$

This is also known as the Kullback–Leibler divergence

- How does KL relate to MI?

# Maximum Entropy Discrimination

- Given data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, find

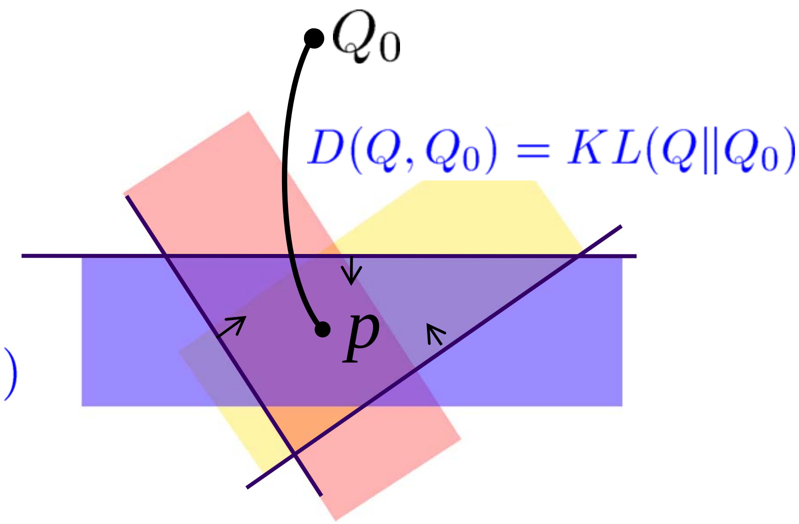
$$\begin{aligned} Q_{\mathrm{ME}} &= \arg\max \ \mathrm{H}(Q) \\ \text{s.t.} \quad & y^i \langle f(\mathbf{x}^i) \rangle_{Q_{\mathrm{ME}}} \geq \xi_i, \quad \forall i \\ & \xi_i \geq 0 \quad \forall i \end{aligned}$$

- solution $Q_{\mathrm{ME}}$ correctly classifies $\mathcal{D}$
- among all admissible $Q$, $Q_{\mathrm{ME}}$ has max entropy
- max entropy → "minimum assumption" about $f$

# Introducing Priors

- Prior $Q_0(f)$

- Minimum Relative Entropy Discrimination




$$Q_{\mathrm{MRE}} = \arg\min \quad \mathrm{KL}(Q\|Q_0) + U(\xi)$$
$$\text{s.t.} \quad y^i \langle f(\mathbf{x}^i) \rangle_{Q_{\mathrm{ME}}} \geq \xi_i, \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

- Convex problem: $Q_{\mathrm{MRE}}$ unique solution
- MER ⇒ "minimum *additional* assumption" over $Q_0$ about $f$

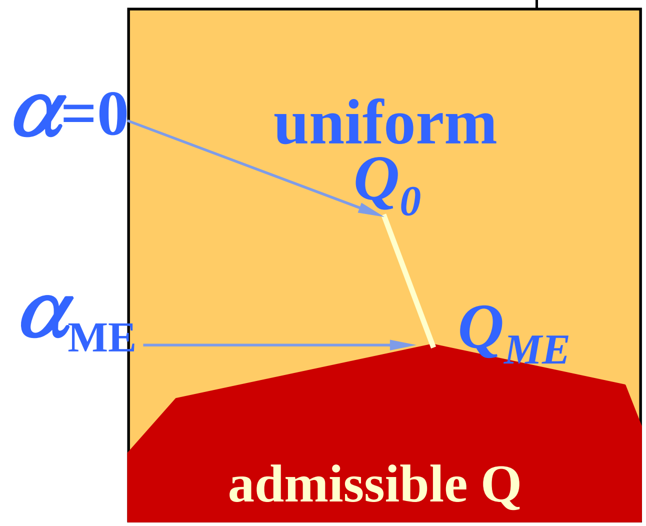
# Solution: $Q_{ME}$ as a projection

- Convex problem: $Q_{ME}$ unique

- Theorem:

$$Q_{\mathrm{MRE}} \propto \exp\{\sum_{i=1}^{N} \alpha_i y_i f(x_i; w)\} Q_0(w)$$

$\alpha_i \geq 0$ Lagrange multipliers




$W = \alpha y_i f( x_i )$

- finding $Q_M$ : start with $\alpha_i = 0$ and follow gradient of unsatisfied constraints

# Solution to MED

- Theorem (Solution to MED):
  - Posterior Distribution:

$$Q(\mathbf{w}) = \frac{1}{Z(\alpha)} Q_0(\mathbf{w}) \exp\{\sum_i \alpha_i y_i [f(\mathbf{x}_i; \mathbf{w})]\}$$

  - Dual Optimization Problem:

$$\begin{aligned} \mathrm{D1}: \quad & \max_{\alpha} \; -\log Z(\alpha) - U^\star(\alpha) \\ & \text{s.t.} \;\; \alpha_i(\mathbf{y}) \geq 0, \; \forall i, \end{aligned}$$

$U^\star(\cdot)$ is the conjugate of the $U(\cdot)$, i.e., $U^\star(\alpha) = \sup_\xi \left( \sum_{i,\mathrm{y}} \alpha_i(\mathrm{y}) \xi_i - U(\xi) \right)$

- Algorithm: to computer $\alpha_t$ , $t = 1,\ldots\mathrm{T}$
  - start with $\alpha_t = 0$ (uniform distribution)
  - iterative ascent on $J(\alpha)$ until convergence

# Examples: SVMs

- **Theorem**

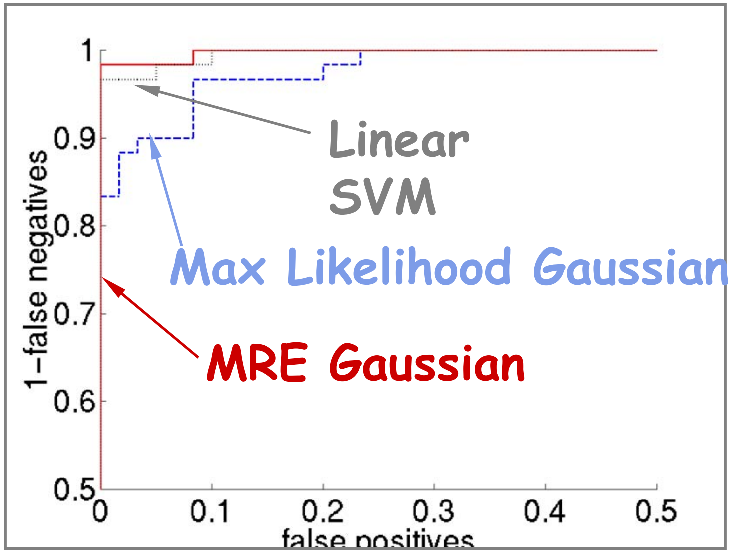
For $f(x) = w^T x + b$, $Q_0(w) = \text{Normal}(0, I)$, $Q_0(b)$ = non-informative prior, the Lagrange multipliers $\alpha$ are obtained by maximizing $J(\alpha)$ subject to $0 \le \alpha_t \le C$ and $\sum_t \alpha_t y_t = 0$, where

$$J(\alpha) = \sum_t \left[\alpha_t + \log(1 - \alpha_t / C)\right] - \frac{1}{2} \sum_{s,t} \alpha_s \alpha_t y_s y_t x_s^T x_t$$

- Separable $D$ ➡ SVM recovered exactly
- Inseparable $D$ ➡ SVM recovered with different misclassification penalty

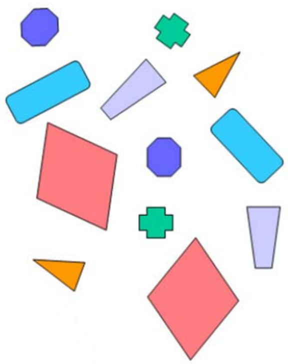
# SVM extensions

- Example: Leptograpsus Crabs (5 inputs, $T_{train}$=80, $T_{test}$=120)



# (3) Structured Prediction

- Unstructured prediction



$$\mathbf{x} = \begin{pmatrix} x_{11} & x_{12} & \cdots \\ x_{21} & x_{22} & \cdots \\ \vdots & \vdots & \cdots \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix}$$

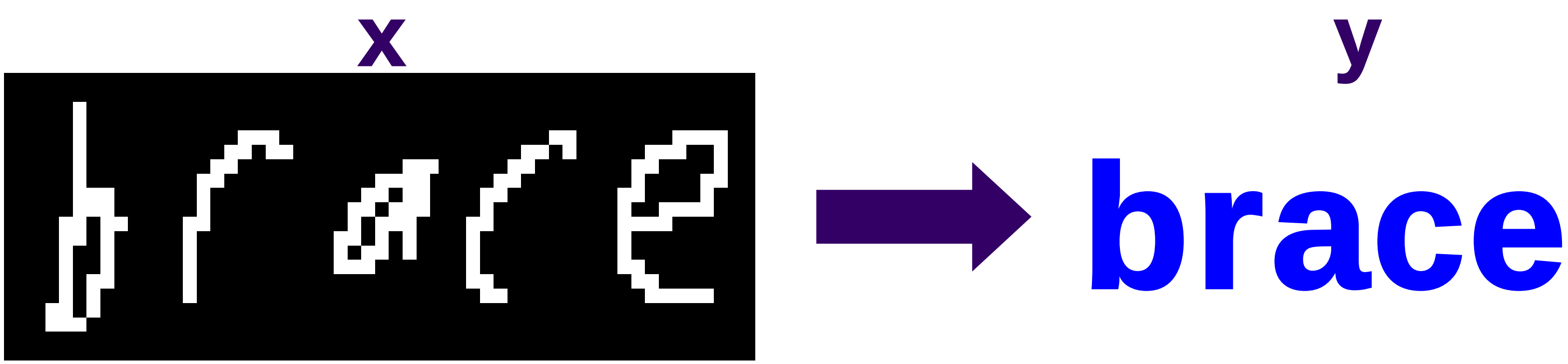
- Structured prediction
  - Part of speech tagging

$\mathbf{x} =$ **"Do you want sugar in it?"** $\Rightarrow$ $\mathbf{y} =$ **<verb pron verb noun prep pron>**

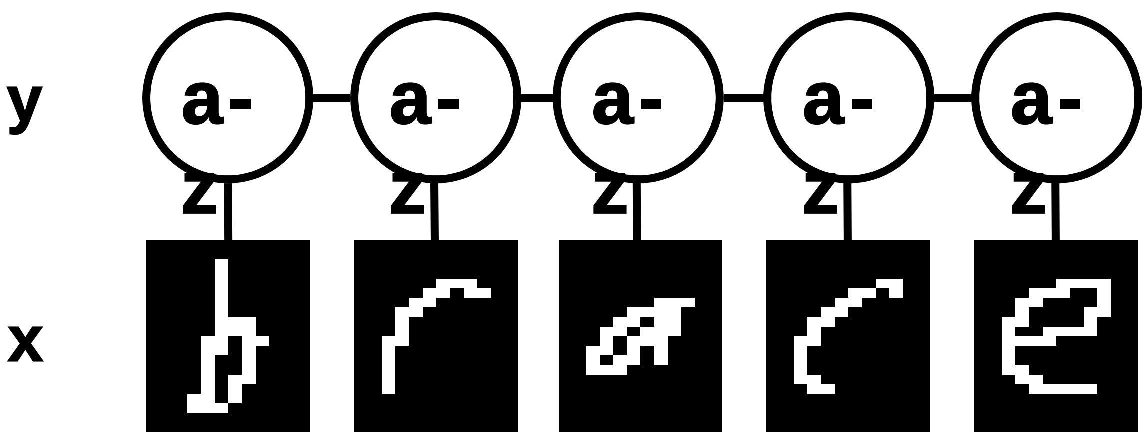
  - Image segmentation

$$\mathbf{x} = \begin{pmatrix} x_{11} & x_{12} & \cdots \\ x_{21} & x_{22} & \cdots \\ \vdots & \vdots & \cdots \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y_{11} & y_{12} & \cdots \\ y_{21} & y_{22} & \cdots \\ \vdots & \vdots & \cdots \end{pmatrix}$$

# OCR example




**Sequential structure**



# Classical Classification Models

- Inputs:
  - a set of training samples $\mathcal{D} = \{(\mathrm{x}_i, y_i)\}_{i=1}^N$ , where $\mathrm{x}_i = [x_i^1, x_i^2, \cdots, x_i^d]^\top$ and $y_i \in C \triangleq \{c_1, c_2, \cdots, c_L\}$
- Outputs:
  - a predictive function $h(\mathrm{x})$: $$y^\star = h(\mathrm{x}) \triangleq \arg\max_y F(\mathrm{x}, y)$$ $$F(\mathrm{x}, y) = \mathbf{w}^\top \mathbf{f}(\mathrm{x}, y)$$
- Examples:
  - SVM: $$\max_{\mathbf{w},\xi} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^N \xi_i; \quad \text{s.t.} \quad \mathbf{w}^\top \Delta \mathbf{f}_i(y) \geq 1 - \xi_i, \ \forall i, \forall y.$$
  - Logistic Regression: $$\max_{\mathbf{w}} \mathcal{L}(\mathcal{D}; \mathbf{w}) \triangleq \sum_{i=1}^N \log p(y_i|\mathrm{x}_i)$$

    where $$p(y|\mathrm{x}) = \frac{\exp\{\mathbf{w}^\top \mathbf{f}(\mathrm{x}, y)\}}{\sum_{y'} \exp\{\mathbf{w}^\top \mathbf{f}(\mathrm{x}, y')\}}$$

# Structured Models

$$h(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} F(\mathbf{x},\mathbf{y})$$

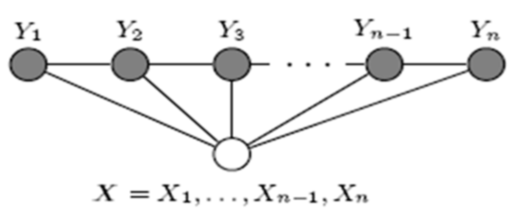
**space of feasible outputs**

**discriminant function**

- Assumptions:

$$F(\mathbf{x},\mathbf{y}) = \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}) = \sum_p \mathbf{w}^\top \mathbf{f}(\mathbf{x}_p,\mathbf{y}_p)$$

  - Linear combination of features
  - Sum of partial scores: index *p* represents a part in the structure
  - Random fields or Markov network features:



# Discriminative Learning Strategies

- Max Conditional Likelihood
  - We predict based on:

$$\mathbf{y}^* \mid \mathbf{x} = \arg\max_{\mathbf{y}} \; p_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{w},\mathbf{x})} \exp\left\{ \sum_c w_c f_c(\mathbf{x}, \mathbf{y}_c) \right\}$$

  - And we learn based on:

$$\mathbf{w}^* \mid \{\mathbf{y}_i, \mathbf{x}_i\} = \arg\max_{\mathbf{w}} \prod_i p_{\mathbf{w}}(\mathbf{y}_i \mid \mathbf{x}_i) = \prod_i \frac{1}{Z(\mathbf{w},\mathbf{x}_i)} \exp\left\{ \sum_c w_c f_c(\mathbf{x}_i, \mathbf{y}_i) \right\}$$

- Max Margin:
  - We predict based on:

$$\mathbf{y}^* \mid \mathbf{x} = \arg\max_{\mathbf{y}} \sum_c w_c f_c(\mathbf{x}, \mathbf{y}_c) = \arg\max_{y} \mathbf{w}^T f(\mathbf{x}, \mathbf{y})$$

  - And we learn based on:

$$\mathbf{w}^* \mid \{\mathbf{y}_i, \mathbf{x}_i\} = \arg\max_{\mathbf{w}} \left( \min_{\mathbf{y} \neq \mathbf{y}^i, \forall i} \mathbf{w}^T \left( f(\mathbf{y}_i, \mathbf{x}_i) - f(\mathbf{y}, \mathbf{x}_i) \right) \right)$$

# E.g. Max-Margin Markov Networks

- Convex Optimization Problem:

$$\text{P0 (M}^3\text{N)}: \quad \min_{\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{s.t. } \forall i, \forall \mathbf{y} \neq \mathbf{y}_i : \quad \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i, \ \xi_i \geq 0 \ ,$$

- Feasible subspace of weights:

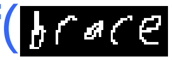
$$\mathcal{F}_0 = \{\mathbf{w} : \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}) \geq \Delta \ell_i(\mathbf{y}) - \xi_i; \ \forall i, \forall \mathbf{y} \neq \mathbf{y}_i\}$$

- Predictive Function:

$$h_0(\mathbf{x}; \mathbf{w}) = \arg \max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} F(\mathbf{x}, \mathbf{y}; \mathbf{w})$$

# OCR Example

- We want:

  argmax$_{word}$ $w^T$ f( , word) = "brace"

- Equivalently:

  $w^T$ f(  ,"brace") > $w^T$ f( ,"aaaaa")

  $w^T$ f(  ,"brace") > $w^T$ f(  ,"aaaab")

  …

  $w^T$ f(  ,"brace") > $w^T$ f(  ,"zzzzz")

**a lot!**

# Min-max Formulation

- Brute force enumeration of constraints:

$$\min \quad \frac{1}{2}||\mathbf{w}||^2$$
$$\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}^*) \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}) + \ell(\mathbf{y}^*, \mathbf{y}), \quad \forall \mathbf{y}$$

  - The constraints are exponential in the size of the structure
- Alternative: min-max formulation
  - add only the most violated constraint

$$\mathbf{y}' = \arg\max_{\mathbf{y} \neq \mathbf{y}*} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}) + \ell(\mathbf{y}_i, \mathbf{y})]$$
$$\text{add to QP}: \; \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}') + \ell(\mathbf{y}_i, \mathbf{y}')$$

  - Handles more general loss functions
  - Only polynomial # of constraints needed
  - Several algorithms exist …

# Results: Handwriting Recognition

Length: ~8 chars

Letter: 16x8 pixels

10-fold Train/Test

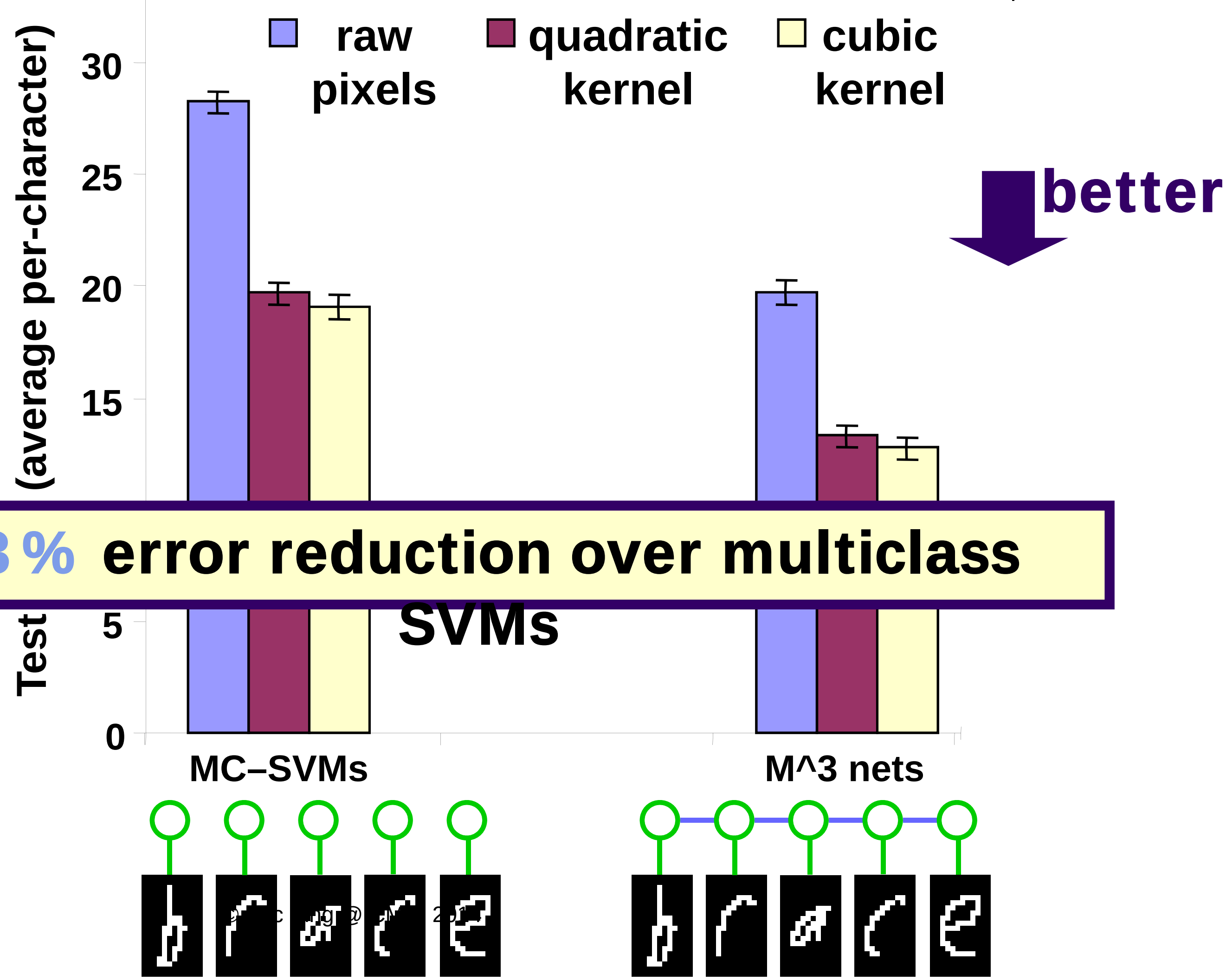
5000/50000 letters

600/6000 words

Models:

Multiclass-SVMs

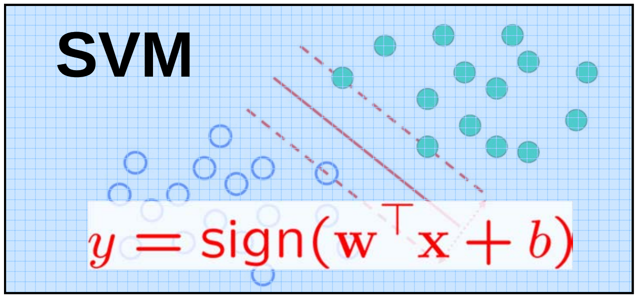
$M^3$ nets



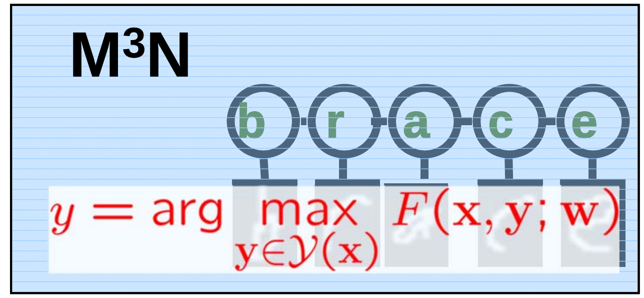

**33 % error reduction over multiclass SVMs**

Crammer & Singer 01

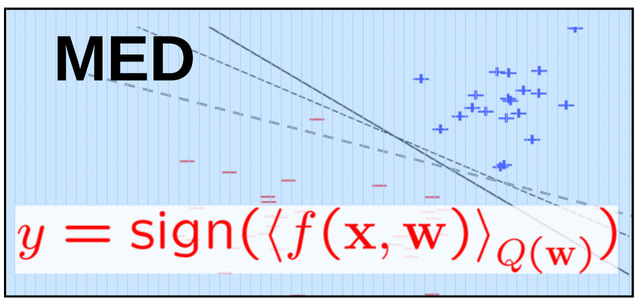
# Discriminative Learning Paradigms




$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i$$
$$y^i(\mathbf{w}^\top \mathbf{x}^i + b) \geq 1 - \xi_i, \quad \forall i$$




$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i$$
$$\mathbf{w}^\top[\mathbf{f}(\mathbf{x}^i, \mathbf{y}^i) - \mathbf{f}(\mathbf{x}^i, \mathbf{y})] \geq \ell(\mathbf{y}^i, \mathbf{y}) - \xi_i, \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}^i$$




$$\min_{Q} \quad \text{KL}(Q||Q_0)$$
$$y^i\langle f(\mathbf{x}^i)\rangle_Q \geq \xi_i, \quad \forall i$$

**MED-MN**

**= SMED + Bayesian M³N**

**See [Zhu and Xing, 2008]**

# Summary

- Maximum margin nonlinear separator
  - Kernel trick
  - Project into linearly separatable space (possibly high or infinite dimensional)
  - No need to know the explicit projection function
- Max-entropy discrimination
  - Average rule for prediction,
  - Average taken over a posterior distribution of w who defines the separation hyperplane
  - P(w) is obtained by max-entropy or min-KL principle, subject to expected marginal constraints on the training examples
- Max-margin Markov network
  - Multi-variate, rather than uni-variate output Y
  - Variable in the outputs are not independent of each other (structured input/output)
  - Margin constraint over every possible configuration of Y (exponentially many!)