CMSC 726
Lecture 12:Kernel Machines

Lise Getoor

October 12, 2010

ACKNOWLEDGEMENTS: The material in this course is a synthesis of materials from many sources, including: Hal Daume III, Mark Drezde, Carlos Guestrin, Andrew Ng, Ben Taskar, **Eric Xing**, and others. I am very grateful for their generous sharing of insights and materials.

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 α_i can always be found.
  - The solution: $w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$
  - How to predict: $\mathbf{w}^T \mathbf{x}_{\text{new}} + b \lessgtr 0$

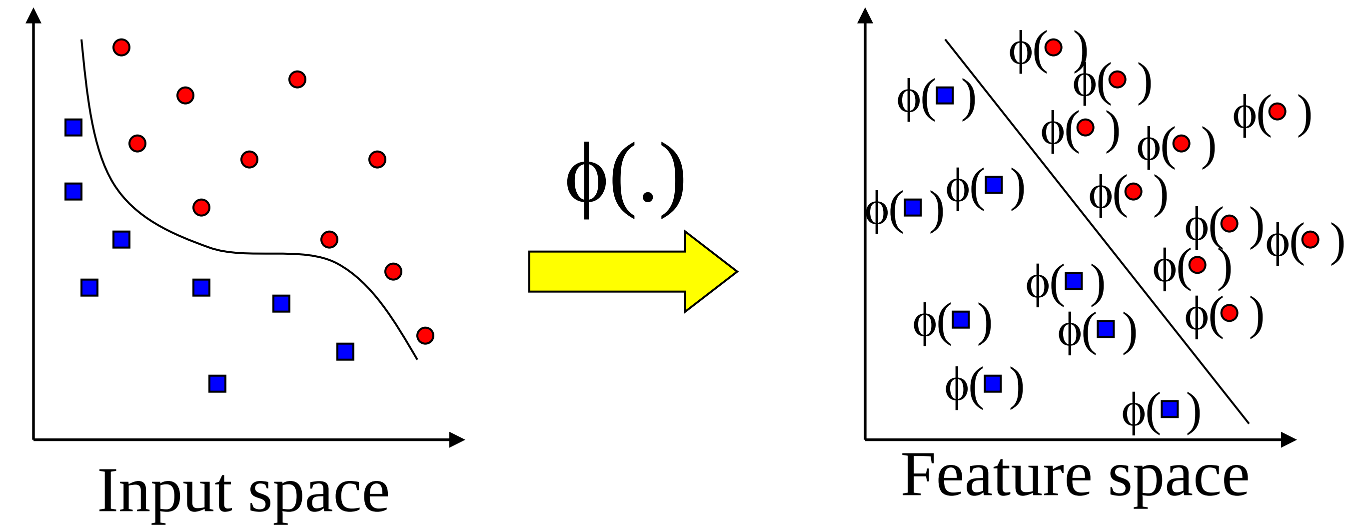
Outline

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

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 $x_1 x_2$ make the problem linearly separable

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 can even be 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

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

- Polynomial kernel

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

where $p = 2, 3, \ldots$ To get the feature vectors we concatenate all pth 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 kernels

- Feature mapping, but “without paying a cost”
  - E.g., polynomial kernel $$K(x, z) = (x^T z + c)^d$$
  - # of dimensions in transformed space?
  - # of 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 reflected in the following “Gaussian” function
$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$
  - Similarly can easily come up with other K() in the same spirit

  - What is necessary for a “legal” kernel?

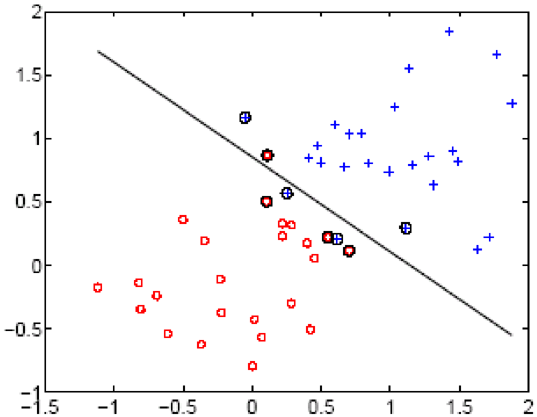
Kernel matrix

- Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ, 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 $y^T K y \geq 0 \quad \forall y$

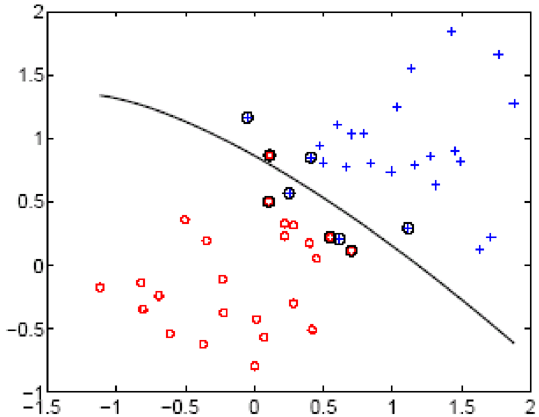
Mercer kernel

Theorem (Mercer): Let $K\colon \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}^n$ 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.

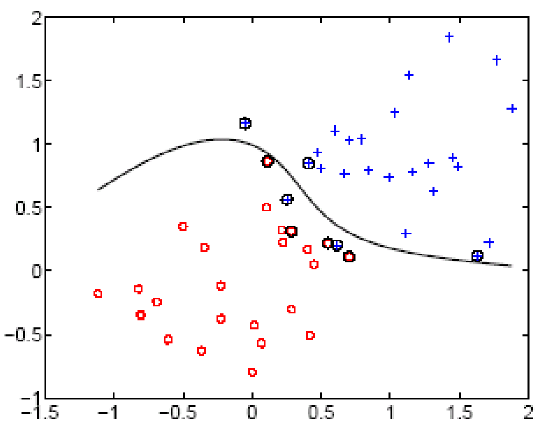
SVM examples



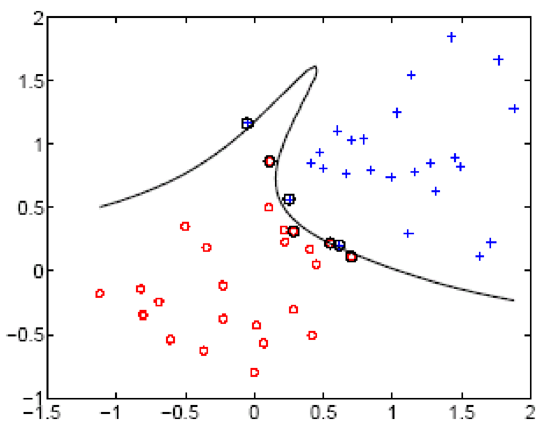
linear



2^{nd} order polynomial

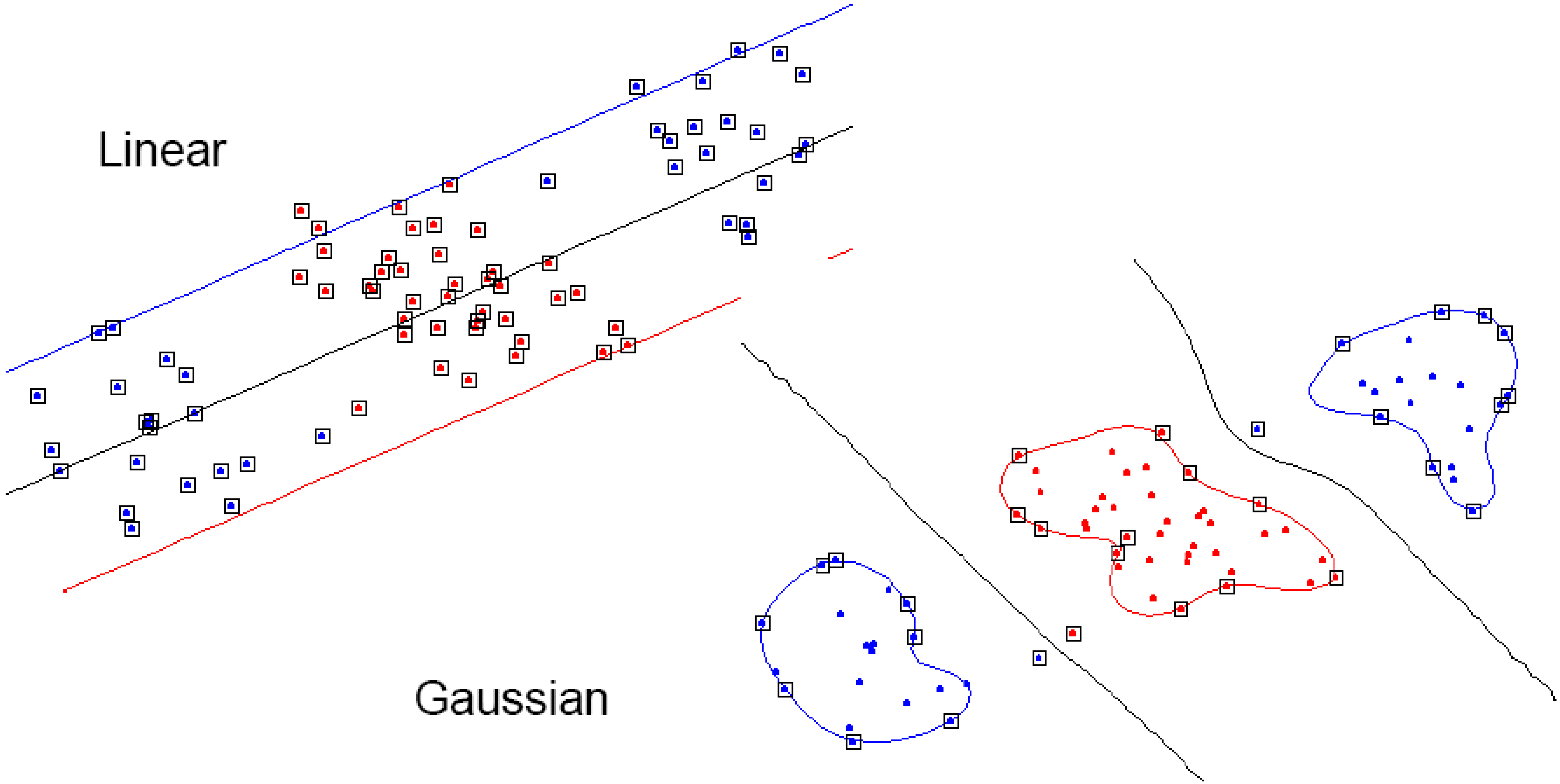


4^{th} order polynomial



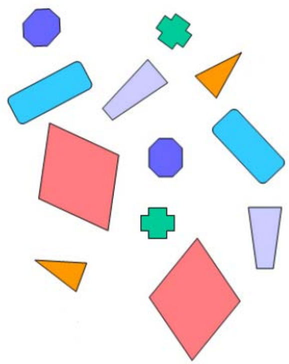
8^{th} order polynomial

Examples for Non Linear SVMs – Gaussian Kernel



Structured Prediction

- Unstructured prediction



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

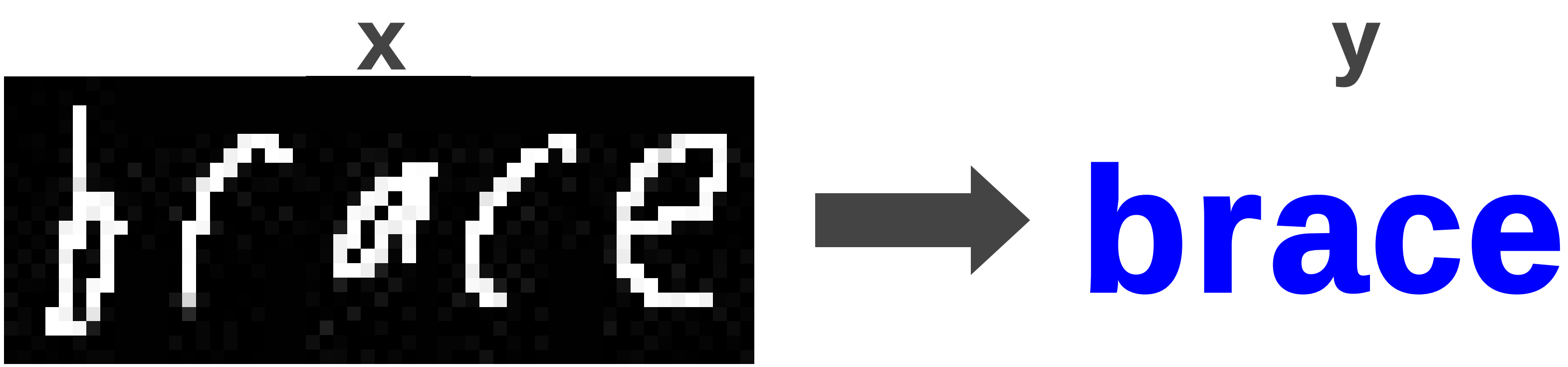
- Structured prediction
  - Part of speech tagging

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

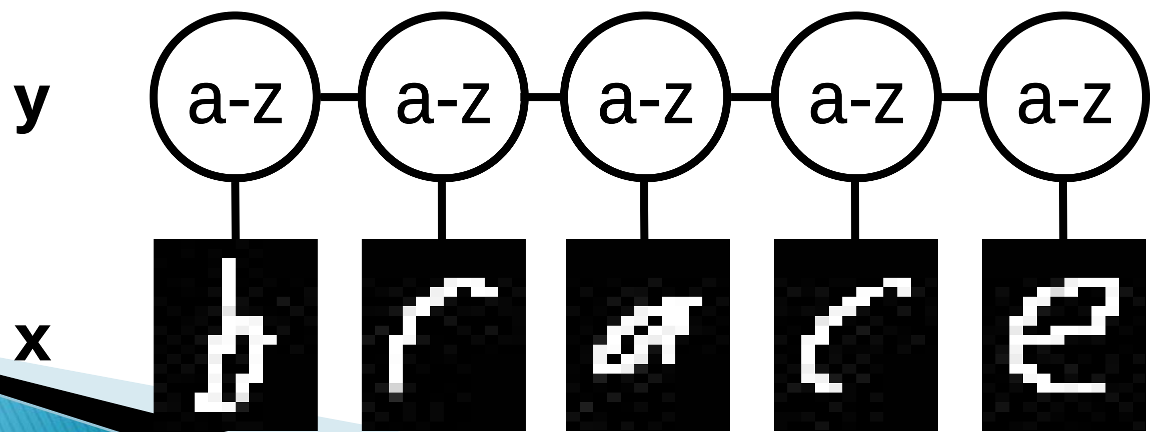
  - Image segmentation

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

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

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

discriminant function

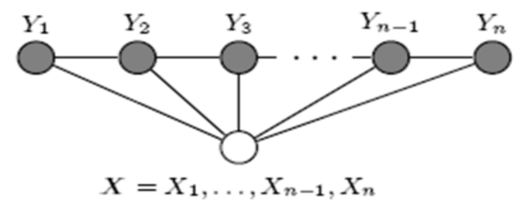
space of feasible outputs

- 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:

$$\mathrm{P0\ (M^3N)}: \qquad \min_{\mathbf{w},\xi} \ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \neq \mathbf{y}_i : \qquad \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \Delta\ell_i(\mathbf{y}) - \xi_i, \ \ \xi_i \geq 0 \ ,$$

- 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:

  $\text{argmax}_{\text{word}}\ \mathbf{w}^T \mathbf{f}($  $, \text{word}) = \text{“brace”}$

- Equivalently:

  $\mathbf{w}^T \mathbf{f}($  $,\text{“brace”}) > \mathbf{w}^T \mathbf{f}($  $,\text{“aaaaa”})$

  $\mathbf{w}^T \mathbf{f}($  $,\text{“brace”}) > \mathbf{w}^T \mathbf{f}($  $,\text{“aaaab”})$

  …

  $\mathbf{w}^T \mathbf{f}($  $,\text{“brace”}) > \mathbf{w}^T \mathbf{f}($  $,\text{“zzzzz”})$

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

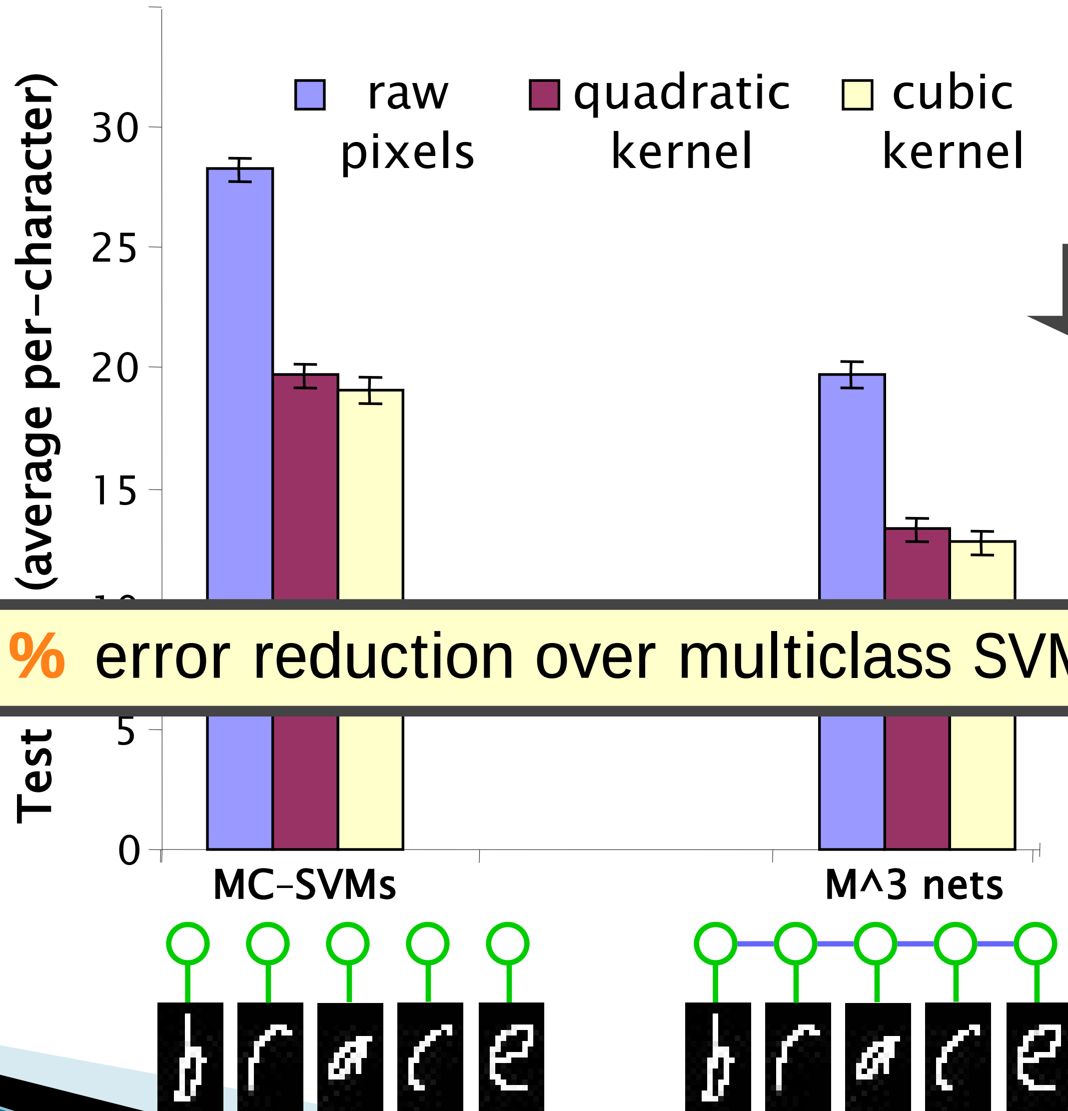
5000/50000 letters

600/6000 words

Models:

Multiclass-SVMs

M^3 nets



33% error reduction over multiclass SVMs

Summary

- Kernel trick
- Structured Prediction