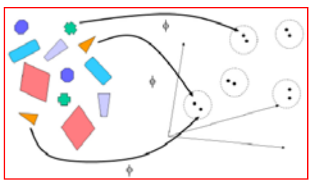
Machine Learning

10-701/15-781, Spring 2010

Advanced topics in Max-Margin Learning



Eric Xing

Lecture 18, March 29, 2008

Reading: class handouts

Recap: the SVM problem

- We solve the following constrained opt problem:

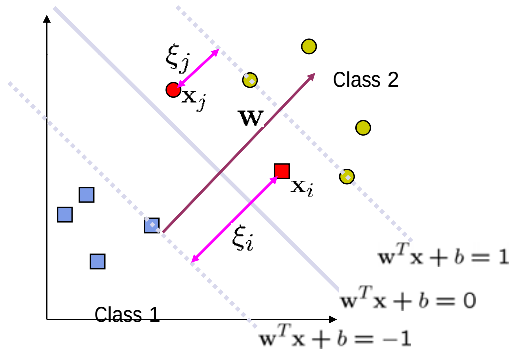
$$\max_\alpha \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

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

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

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

Non-linearly Separable Problems



- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\boldsymbol{w}^T\boldsymbol{x}+b$
- ξ_i approximates the number of misclassified samples

Soft Margin Hyperplane

- Now we have a slightly different opt problem:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t} \quad \begin{aligned} & y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

- ξ_i are "slack variables" in optimization
- Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
- ξ_i is an upper bound of the number of errors
- C : tradeoff parameter between error and margin

The Optimization Problem

- The dual of this new constrained optimization problem is

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

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

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

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now
- Once again, a QP solver can be used to find α_i

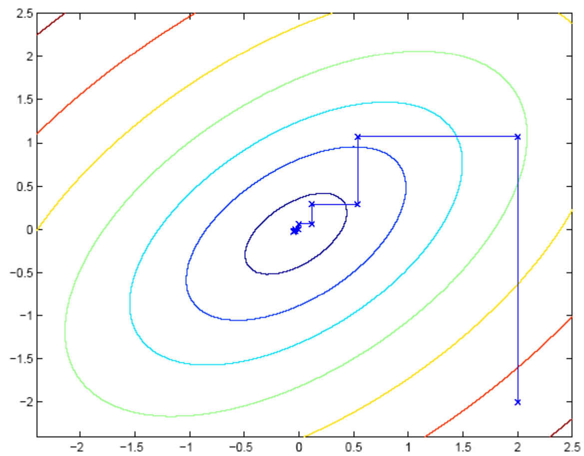
The SMO algorithm

- Consider solving the unconstrained opt problem:

$$\max_\alpha W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

- We've already see three opt algorithms!
  - Coordinate ascent
  - Gradient ascent
  - Newton-Raphson
- Coordinate ascend:

Coordinate ascend



Sequential minimal optimization

- Constrained optimization:

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

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

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

- Question: can we do coordinate along one direction at a time (i.e., hold all $\alpha_{[-i]}$ fixed, and update α_i?)

The SMO algorithm

Repeat till convergence

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).

2. Re-optimize J(α) with respect to α_i and α_j, while holding all the other α_k 's ($k \neq i; j$) fixed.

Will this procedure converge?

Convergence of SMO

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

$$\boxed{\text{KKT:} \quad \begin{aligned} \text{s.t.} \quad & 0 \le \alpha_i \le C, \quad i = 1, \ldots, k \\ & \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned}}$$

- Let's hold α_3 ,…, α_m fixed and reopt J w.r.t. α_1 and α_2

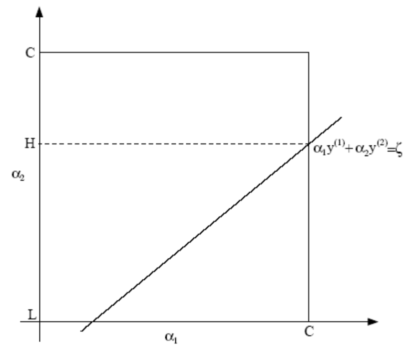
Convergence of SMO

- The constraints:

$$\alpha_1 y_1 + \alpha_2 y_2 = \xi$$

$$0 \leq \alpha_1 \leq C$$

$$0 \leq \alpha_2 \leq C$$



- The objective:

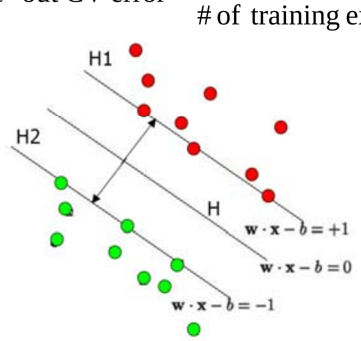
$$\mathcal{J}(\alpha_1, \alpha_2, \ldots, \alpha_m) = \mathcal{J}((\xi - \alpha_2 y_2) y_1, \alpha_2, \ldots, \alpha_m)$$

- Constrained opt:

Cross-validation error of SVM

- The leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the # of support vectors!

$$\text{Leave-one-out CV error} = \frac{\#\text{ support vectors}}{\#\text{ of training examples}}$$



Advanced topics in Max-Margin Learning

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

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

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

Outline

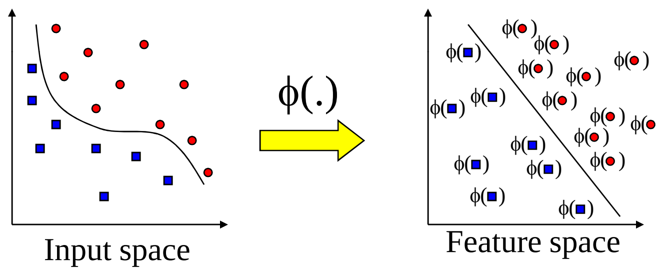
- The Kernel trick
- Maximum entropy discrimination
- Structured SVM, aka, Maximum Margin Markov Networks

(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

Transforming the Data



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

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 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 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 ϕ, then for x_1, …, x_m, we can compute an $m \times m$ matrix $K = \{K_{i,j}\}$, where $K_{i,j} = \phi(x_i)^T \phi(x_j)$

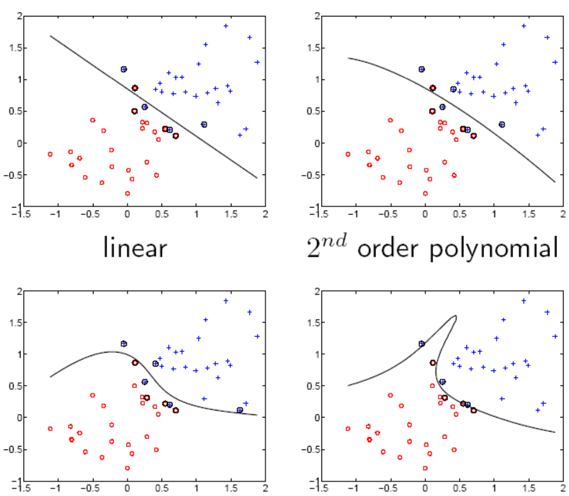
- This is called a kernel matrix!

- 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



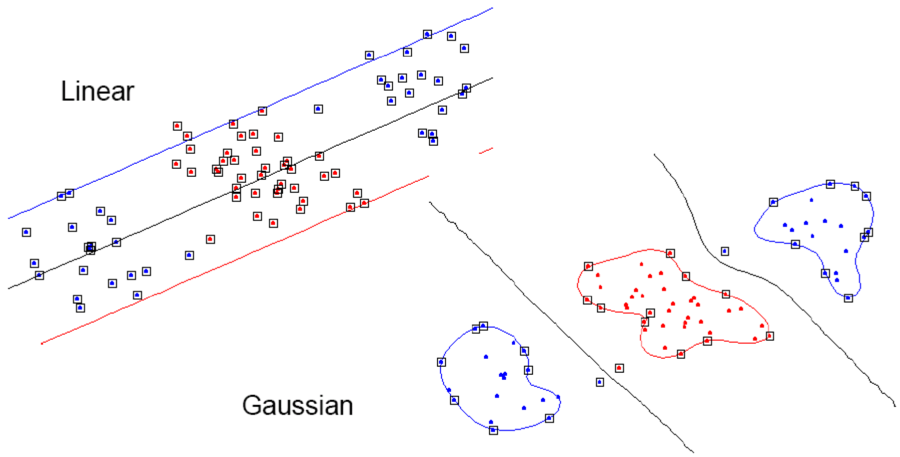
linear

2^{nd} order polynomial

4^{th} order polynomial

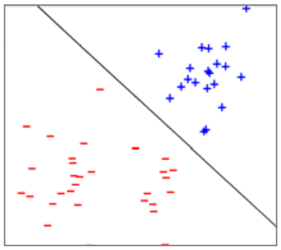
8^{th} order polynomial

Examples for Non Linear SVMs – Gaussian Kernel



(2) Model averaging

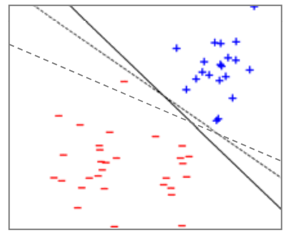
- Inputs x, class $y = +1, -1$
- data $D = \{ (x_1,y_1), \ldots (x_m,y_m) \}$
- Point Rule:
    - learn $f^{opt}(x)$ discriminant function from $F = \{f\}$ family of discriminants
    - classify $y = \text{sign}\, f^{opt}(x)$
- E.g., SVM



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

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:

$p_0(\mathbf{w})$ → Bayes Learner → $p(\mathbf{w}|\mathcal{D})$

$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ → Bayes Learner

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

- Bayes Predictor (model averaging):

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

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

- What p_0?

How to score distributions?

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

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

  - *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:

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

Maximum Entropy Discrimination

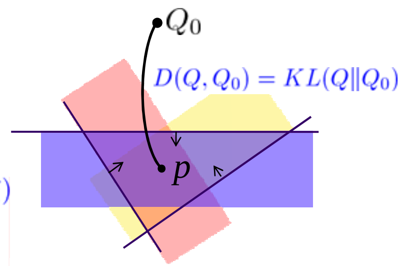
- Given data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, find

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

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

Introducing Priors

- Prior $Q_0(f)$
- Minimum Relative Entropy Discrimination

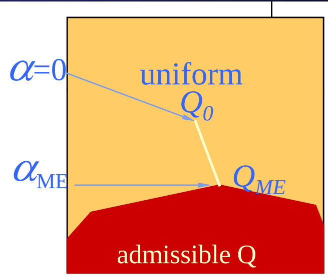


$$
\begin{aligned}
Q_{\mathrm{MRE}} &= \arg\min \;\; \mathrm{KL}(Q\|Q_0) + U(\xi) \\
\text{s.t.} &\quad y^i \langle f(\mathbf{x}^i) \rangle_{Q_{\mathrm{ME}}} \geq \xi_i, \quad \forall i \\
&\quad \xi_i \geq 0 \quad \forall i
\end{aligned}
$$

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

Solution: Q $_{\mathrm{ME}}$ 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

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

Solution to MED

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

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

  - 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., $U^\star(\alpha) = \sup_\xi \left(\sum_{i,y} \alpha_i(y)\xi_i - U(\xi) \right)$

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

Examples: SVMs

- **Theorem**

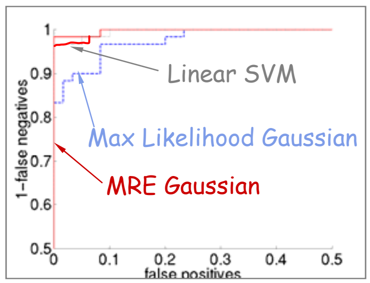
For $f(x) = w^{\mathrm{T}}x + b$, $Q_0(w) = \mathrm{Normal}(0, I)$, $Q_0(b) =$ non-informative prior, the Lagrange multipliers α are obtained by maximizing $J(\alpha)$ subject to $0 \leq \alpha_t \leq C$ and $\sum_t \alpha_t y_t = 0$, where

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

- 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

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

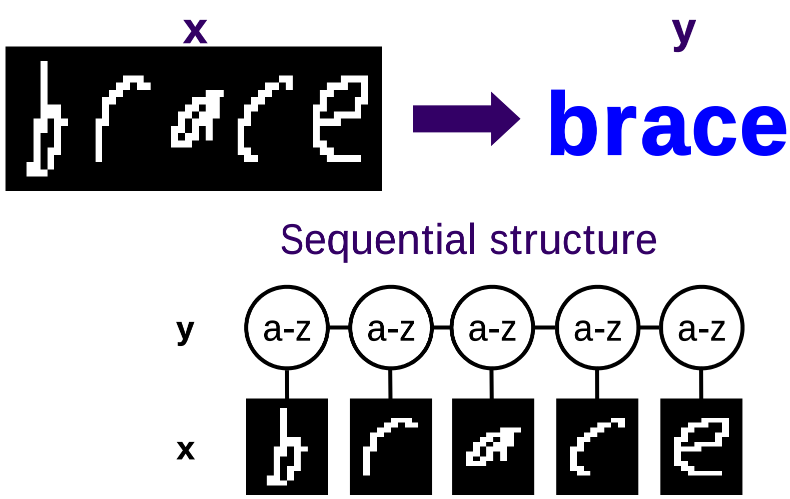
- Structured prediction
  - Part of speech tagging

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

  - Image segmentation

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

OCR example



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

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

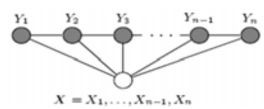
discriminant function

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:

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

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

- Feasible subspace of weights:

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

- 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

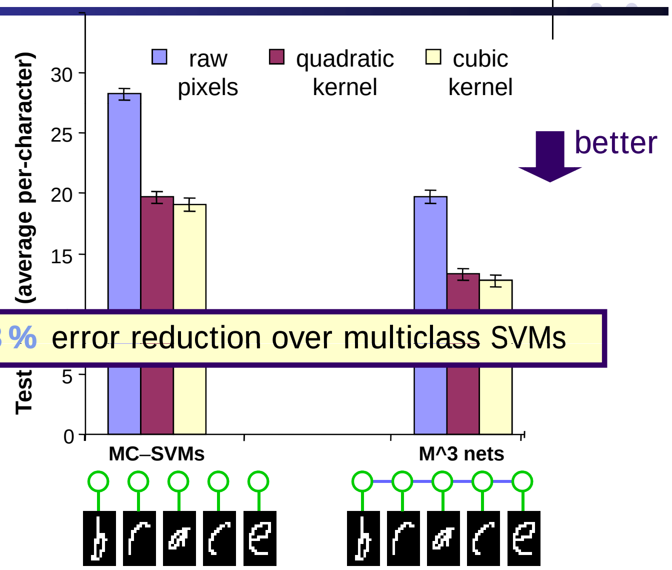
$$\mathbf{y}' = \arg\max_{\mathbf{y} \neq \mathbf{y}*} [\mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}) + \ell(\mathbf{y}_i, \mathbf{y})]$$
$$\text{add to QP}: \ \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}') + \ell(\mathbf{y}_i,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

  - 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

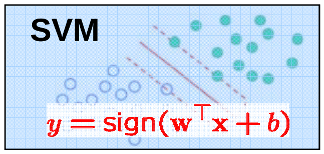
Models:
Multiclass-SV
M^3 nets



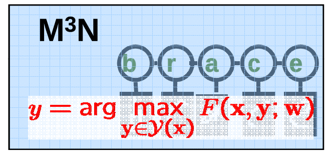
33 % error reduction over multiclass SVMs

*Crammer & Singer 01

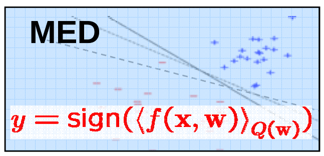
Discriminative Learning Paradigms



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



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



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

MED-MN
= SMED + Bayesian M^{3}N

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!)