Machine Learning

10-701, Fall 2016

Advanced topics in Max-Margin Learning

Eric Xing

Lecture 7, September 28, 2016
Reading: class handouts

© Eric Xing @ CMU, 2006-2016
Recap: the SVM problem

- We solve the following constrained opt problem:

\[
\max_\alpha \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]

s.t. \( \alpha_i \geq 0, \quad i = 1, \ldots, m \)

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

- This is a \textbf{quadratic programming} problem.

  - A global maximum of \( \alpha_i \) can always be found.

- The solution:

\[
w = \sum_{i=1}^{m} \alpha_i y_i x_i
\]

- How to predict:

\[
w^T x_{\text{new}} + b \leq 0
\]
The SMO algorithm

- Consider solving the unconstrained opt problem:
  \[
  \alpha^* = \arg\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_m)
  \]

- We've already seen three opt algorithms!
  - Coordinate ascent
  - Gradient ascent
  - Newton-Raphson

- Coordinate ascent:
  \[
  \begin{align*}
  \frac{\partial W}{\partial \alpha_i} &= \frac{dW}{dx_i} \\
  \frac{\partial^2 W}{\partial \alpha_i^2} &= \frac{d^2W}{dx_i dx_i} = \left( \frac{d^2W}{dx_1 dx_1}, \frac{d^2W}{dx_2 dx_2}, \ldots \right) \\
  \alpha_i &= \frac{1}{\frac{dW}{dx_i} \frac{d^2W}{dx_i dx_i}}
  \end{align*}
  \]
Coordinate ascend
Sequential minimal optimization

- Constrained optimization:

\[
\max_{\alpha} \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]

\[
\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m
\]

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

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

\[
\Delta \alpha_i = \frac{\partial J}{\partial \alpha_i} = \sum_{i \neq j} \alpha_j y_j (x_i^T x_j)
\]

\[
\sum_{i \neq j} \alpha_i y_i = 0
\]
The SMO algorithm

Repeat till convergence

1. Select some pair $\alpha_i$ and $\alpha_j$ 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 $J(\alpha)$ with respect to $\alpha_i$ and $\alpha_j$, while holding all the other $\alpha_k$'s ($k \neq i; j$) fixed.

Will this procedure converge?
Convergence of SMO

\[ \max_\alpha J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \]

KKT:

\[ \begin{align*}
0 & \leq \alpha_i \leq C, \quad i = 1, \ldots, k \\
\sum_{i=1}^{m} \alpha_i y_i & = 0.
\end{align*} \]

- Let's hold \( \alpha_3, \ldots, \alpha_m \) fixed and reopt \( J \) w.r.t. \( \alpha_1 \) and \( \alpha_2 \)
Convergence of SMO

- The constraints:
  \[ \alpha_1 y_1 + \alpha_2 y_2 = \xi \]
  \[ 0 \leq \alpha_1 \leq C \]
  \[ 0 < \alpha_2 < C \]

- The objective:
  \[ J(\alpha_1, \alpha_2, \ldots, \alpha_m) = J((\xi - \alpha_2 y_2) y_1, \alpha_2, \ldots, \alpha_m) \]

- Constrained opt:
Advanced topics in Max-Margin Learning

\[ \max_\alpha J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \]

\[ w^T x_{\text{new}} + b \leq 0 \]

- 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 \( x_i \) to a higher dimensional space to “make life easier”
  - Input space: the space the point \( x_i \) are located
  - Feature space: the space of \( \phi(x_i) \) 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

\[
\max_\alpha \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]

s.t. \(0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m\)

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

- 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

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]

© Eric Xing @ CMU, 2006-2016
An Example for feature mapping and kernels

- Consider an input \( x = [x_1, x_2] \)
- Suppose \( \phi(.) \) is given as follows
  \[
  \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2} x_1, \sqrt{2} x_2, x_1^2, x_2^2, \sqrt{2} x_1 x_2
  \]
- An inner product in the feature space is
  \[
  \langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right)\rangle = (1 + x_1 x_1' + \sqrt{2} x_1 x_2 + x_1^2 + x_1^2 + 2 x_1 \sqrt{2} x_1 x_2 + x_1'^2 + 2 x_1 x_1' x_2' + 2 x_1 x_1' x_2' + x_2'^2 + 2 x_2 x_1 x_2' + x_2^2 + 2 x_2 x_1' x_2')
  \]
  \[
  = (1 + x_1 x_1')^2
  \]
  kernel function:

- So, if we define the kernel function as follows, there is no need to carry out \( \phi(.) \) explicitly
  \[
  K(x, x') = (1 + x^T x')^2
  \]
More examples of kernel functions

- Linear kernel (we've seen it)

\[ K(x, x') = x^T x' \]

- Polynomial kernel (we just saw an example)

\[ K(x, x') = (1 + x^T x')^p \]

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

- Radial basis kernel

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

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

Theorem (Mercer): Let \( K : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R} \) 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-definite.
SVM examples

linear

2nd order polynomial

4th order polynomial

8th order polynomial
Examples for Non Linear SVMs – Gaussian Kernel

Linear

Gaussian
(2) Model averaging

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

Point Rule: $\mathbf{w}^T y$

- learn $f_{\text{opt}}(x)$ discriminant function from $\mathcal{F} = \{ f \}$ family of discriminants
- classify $y = \text{sign } f_{\text{opt}}(x)$

- 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 $f^{\text{opt}}$, average over all $f$ in $F$

  $$Q(f) = \text{weight of } f$$

  $$y(x) = \text{sign} \int_{F} Q(f) f(x) df$$

  $$= \text{sign} \langle f(x) \rangle_{Q}$$

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

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

- Bayesian learning:
  \[ p_0(w) \]
  \[ D = \{(x_i, y_i)\}_{i=1}^N \]
  Bayes Thrm: \[ p(w|D) = \frac{p(w)p(D|w)}{p(D)} \]

- Bayes Predictor (model averaging):
  \[ h_1(x; p(w)) = \arg \max_{y \in \mathcal{Y}(x)} \int p(w)f(x, y; w)dw \]

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

- 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:

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

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

\[
Q_{ME} = \arg \max \quad H(Q) \\
\text{s.t.} \\
y^i (f(x^i))_{Q_{ME}} \geq \xi_i, \quad \forall i \\
\xi_i \geq 0, \quad \forall i
\]

- 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

\[
Q_{MRE} = \arg \min Q \left( \sum_i y_i \langle f(x_i) \rangle_{Q_{ME}} - \xi_i \right) \quad \text{s.t.} \quad \xi_i \geq 0 \quad \forall i
\]

- Convex problem: \( Q_{MRE} \) unique solution

- MER → "minimum additional assumption" over \( Q_0 \) about \( f \)
Solution: $Q_{ME}$ as a projection

- Convex problem: $Q_{ME}$ unique
- Theorem: $f(w) \geq \text{proj}_Q(w)$

\[ Q_{MRE} \propto \exp\left\{ \sum_{i=1}^{N} \alpha_i y_i f(x_i; w) \right\} Q_0(w) \]

$\alpha_i \geq 0$ 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:
    \[ Q(w) = \frac{1}{Z(\alpha)} Q_0(w) \exp \left\{ \sum_i \alpha_i y_i [f(x_i; w)] \right\} \]
  - Dual Optimization Problem:
    \[ \text{D1} : \max_{\alpha} - \log Z(\alpha) - U^*(\alpha) \]
    \[ \text{s.t.} \quad \alpha_i(y) \geq 0, \quad \forall i, \]
    \[ U^*(\cdot) \text{ is the conjugate of the } U(\cdot), \text{ i.e., } U^*(\alpha) = \sup_\xi \left( \sum_{i,y} \alpha_i(y) \xi_i - U(\xi) \right) \]

- Algorithm: to compute \( \alpha_t, \ t = 1, \ldots, T \)
  - start with \( \alpha_t = 0 \) (uniform distribution)
  - iterative ascent on \( J(\alpha) \) until convergence
Examples: SVMs

- **Theorem**

For \( f(x) = w^T x + b \) where \( Q_0(w) = \text{Normal}(0, I) \) and \( Q_0(b) = \text{non-informative prior} \), the Lagrange multipliers \( \alpha \) 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** \( D \rightarrow \text{SVM recovered exactly} \)
- **Inseparable** \( D \rightarrow \text{SVM recovered with different misclassification penalty} \)
SVM extensions

- Example: Leptograpsus Crabs (5 inputs, $T_{\text{train}}=80$, $T_{\text{test}}=120$)
(3) Structured Prediction

- Unstructured prediction

\[ x = \begin{pmatrix} x_{11} & x_{12} & \cdots \\ x_{21} & x_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \end{pmatrix} \]

- Structured prediction
  
  - Part of speech tagging
    
    \[ x = \text{“Do you want sugar in it?”} \quad \Rightarrow \quad y = \text{verb pron verb noun prep pron} > \]

  - Image segmentation

\[ x = \begin{pmatrix} x_{11} & x_{12} & \cdots \\ x_{21} & x_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad y = \begin{pmatrix} y_{11} & y_{12} & \cdots \\ y_{21} & y_{22} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \]
OCR example

Sequential structure

© Eric Xing @ CMU, 2006-2016
Classical Classification Models

- **Inputs:**
  - a set of training samples \( \mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \), where
  \[
x_i = [x_i^1, x_i^2, \ldots, x_i^d]^\top \quad \text{and} \quad y_i \in C \triangleq \{c_1, c_2, \ldots, c_L\}
\]

- **Outputs:**
  - a predictive function \( h(x) \):
    \[
y^* = h(x) \triangleq \arg \max_y F(x, y)
    \]
    \[
    F(x, y) = w^\top f(x, y)
    \]

- **Examples:**
  - SVM:
    \[
    \max_{w, \xi} \frac{1}{2} w^\top w + C \sum_{i=1}^N \xi_i \quad \text{s.t.} \quad w^\top f_i(y) \geq 1 - \xi_i, \quad \forall i, \forall y.
    \]
  - Logistic Regression:
    \[
    \max_w \mathcal{L}(\mathcal{D}; w) \triangleq \sum_{i=1}^N \log p(y_i|x_i)
    \]
    where
    \[
p(y|x) = \frac{\exp\{w^\top f(x, y)\}}{\sum_{y'} \exp\{w^\top f(x, y')\}}
    \]
Structured Models

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

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

\[ F(x, y) = w^\top f(x, y) = \sum_p w^\top f(x_p, y_p) \]
Discriminative Learning Strategies

- **Max Conditional Likelihood**
  - We predict based on:
    \[
    y^* \mid x = \arg \max_y p_w(y \mid x) = \frac{1}{Z(w, x)} \exp \left\{ \sum_c w_c f_c(x, y_c) \right\}
    \]
  - And we learn based on:
    \[
    w^* \mid \{y_i, x_i\} = \arg \max_w \prod_i p_w(y_i \mid x_i) = \prod_i \frac{1}{Z(w, x_i)} \exp \left\{ \sum_c w_c f_c(x_i, y_i) \right\}
    \]

- **Max Margin**:
  - We predict based on:
    \[
    y^* \mid x = \arg \max_y \sum_c w_c f_c(x, y_c) = \arg \max_y w^T f(x, y)
    \]
  - And we learn based on:
    \[
    w^* \mid \{y_i, x_i\} = \arg \max_w \left( \min_{y \neq y', \forall i} w^T (f(y_i, x_i) - f(y, x_i)) \right)
    \]
E.g. Max-Margin Markov Networks

- Convex Optimization Problem:

\[
P_0 (\mathcal{M}^3 \mathcal{N}) : \min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i
\]

s.t. \( \forall i, \forall y \neq y_i : \quad w^\top \Delta f_i(x, y) \geq \Delta l_i(y) - \xi_i, \; \xi_i \geq 0 \),

- Feasible subspace of weights:

\[
\mathcal{F}_0 = \{ w : w^\top \Delta f_i(x, y) \geq \Delta l_i(y) - \xi_i; \; \forall i, \forall y \neq y_i \}
\]

- Predictive Function:

\[
h_0(x; w) = \arg \max_{y \in \mathcal{Y}(x)} F(x, y; w)
\]
OCR Example

- We want:
  \[
  \arg\max_{\text{word}} w^T f(\text{brace} , \text{word}) = \text{“brace”}
  \]

- Equivalently:
  \[
  w^T f(\text{brace} , \text{“brace”}) > w^T f(\text{brace} , \text{“aaaaa”})
  
  w^T f(\text{brace} , \text{“brace”}) > w^T f(\text{brace} , \text{“aaaab”})
  
  \ldots
  
  w^T f(\text{brace} , \text{“brace”}) > w^T f(\text{brace} , \text{“zzzzz”})
  \]
Brute force enumeration of constraints:
\[ \min \frac{1}{2} \|w\|^2 \]
\[ w^T f(x, y^*) \geq w^T f(x, y) + \ell(y^*, y), \quad \forall y \]
- The constraints are exponential in the size of the structure

Alternative: min-max formulation
- add only the most violated constraint

\[ y' = \arg \max_{y \neq y^*} [w^T f(x_i, y) + \ell(y_i, y)] \]

add to QP: \[ w^T f(x_i, y_i) \geq w^T f(x_i, y') + \ell(y_i, y') \]
- 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

© Eric Xing @ CMU, 2006-2016
Discriminative Learning Paradigms

**SVM**

\[ y = \text{sign}(w^T x + b) \]

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \\
y^i(w^T x^i + b) & \geq 1 - \xi_i, \quad \forall i
\end{align*}
\]

**M^3N**

\[ y = \arg \max_{y \in \mathcal{Y}(x)} F(x, y; w) \]

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \\
w^T [f(x^i) - f(x^i, y)] & \geq \ell(y^i, y) - \xi_i, \quad \forall i, \forall y \neq y^i
\end{align*}
\]

**MED**

\[ y = \text{sign}(\langle f(x, w) \rangle_{Q(w)}) \]

\[
\begin{align*}
\min_{Q} & \quad \text{KL}(Q \| Q_0) \\
y^i(f(x^i))_Q & \geq \xi_i, \quad \forall i
\end{align*}
\]

**MED-MN**

\[ = \text{SMED} + \text{Bayesian M}^3\text{N} \]

See [Zhu and Xing, 2008]
Summary

- Maximum margin nonlinear separator
  - Kernel trick
  - Project into linearly separable 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!)