Recap: the SVM problem

- We solve the following constrained opt problem:

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

s.t. \( \alpha_i \geq 0, \quad i = 1, \ldots, m \)
\( \sum_{i=1}^{m} \alpha_i y_i = 0. \)

- This is a quadratic programming problem.
  - A global maximum of \( \alpha \) can always be found.
  - The solution:
    \[
    w = \sum_{i=1}^{m} \alpha_i y_i x_i
    \]
  - How to predict:
    \[
    w^T x_{\text{new}} + b \leq 0
    \]
Non-linearly Separable Problems

- We allow “error” $\xi_i$ in classification; it is based on the output of the discriminant function $w^T x + b$
- $\xi_i$ approximates the number of misclassified samples

Soft Margin Hyperplane

- Now we have a slightly different opt problem:

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

s.t. 
$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall i$$
$$\xi_i \geq 0, \quad \forall i$$

- $\xi_i$ are “slack variables” in optimization
- Note that $\xi_i = 0$ if there is no error for $x_i$
- $\xi_i$ 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

\[
\begin{align*}
    \max_{\alpha} & \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j) \\
    \text{s.t.} & \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \\
    & \quad \sum_{i=1}^{m} \alpha_i y_i = 0.
\end{align*}
\]

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

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:

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

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

- Question: can we do coordinate along one direction at a time (i.e., hold all \(\alpha_{[-i]}\) fixed, and update \(\alpha_i\)?)
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. $\alpha_j$ and $\alpha_2$
Convergence of SMO

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

- 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:

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!

Leave-one-out CV error = \[ \frac{\text{# support vectors}}{\text{# of training examples}} \]
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)
Transforming the Data

\[ \phi(.) \]

Input space

Feature space

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

The Kernel Trick

- Recall the SVM optimization problem

\[
\max_{\alpha} \quad J(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle x_i^T x_j \rangle \\
\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, m \\
\sum_{i=1}^{m} \alpha_i y_i = 0.
\]

- 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

\[
K(x_i, x_j) = \phi(x_i)^T \phi(x_j)
\]
An Example for feature mapping and kernels

- Consider an input $x = [x_1, x_2]$
- Suppose $\phi(.)$ is given as follows
  \[
  \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = [1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]
  \]
- An inner product in the feature space is
  \[
  \left< \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) , \phi\left(\begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}\right) \right> =
  \]
- 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 \[ K = K^T \]
    \[ K_{i,j} = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = K_{j,i} \]
  - Positive–semidefinite proof?
    \[ y^T K y > 0 \quad \forall y \]
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
(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
  $$f^{opt}(x) = w^T x_{\text{new}} + b$$
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 $f$
  
  $y(x) = \text{sign} \int_Q Q(f) f(x) df$
  
  $= \text{sign} \langle f(x) \rangle_Q$

- 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):
  
  $h_1(x; p(w)) = \arg \max_{y \in Y(x)} \int p(w)f(x, y; w)dw$

  Recall in SVM: $h_0(x; w) = \arg \max_{y \in Y(x)} \int f(x, y; w)dw$

- 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

  $$Q_{ME} = \arg \max_{Q} H(Q)$$

  such that

  $$y_i \langle f(x_i) \rangle_{Q_{ME}} \geq \xi_i, \quad \forall i$$

  $$\xi_i \geq 0, \quad \forall i$$

  - solution $Q_{ME}$ correctly classifies $\mathcal{D}$
  - among all admissible $Q$, $Q_{ME}$ has max entropy
  - max entropy \textbf{“minimum assumption”} about $f$
Introducing Priors

- Prior $Q_0(f)$

- Minimum Relative Entropy Discrimination

$$Q_{\text{MRE}} = \arg \min Q \ KL(Q||Q_0) + U(\xi)$$

$$\text{s.t.} \quad u^i(f(x^i))_{\text{max}} > \xi_i, \ \forall i$$

$$\xi_i \geq 0, \ \forall i$$

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

Solution: $Q_{\text{ME}}$ as a projection

- Convex problem: $Q_{\text{ME}}$ unique

- Theorem:

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

$\alpha_i \geq 0$ Lagrange multipliers

- finding $Q_{\text{M}}$: 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:
    \[ D1: \max_{\alpha} - \log Z(\alpha) - U^*(\alpha) \]
    \[ \text{s.t. } \alpha_i(y) \geq 0, \forall i, \]
    \[ U^*(\cdot) \text{ is the conjugate of the } U(\cdot), \text{i.e., } U^*(\alpha) = \sup_{\xi} \left\{ \sum_i \alpha_i(y) \xi_i - U(\xi) \right\} \]

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

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

- 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)

![Graph showing SVM extensions](image)

(3) Structured Prediction

- Unstructured prediction

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

- 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

\[
\begin{pmatrix}
x_{11} & x_{12} & \cdots \\
x_{21} & x_{22} & \cdots \\
\vdots & \vdots & \ddots 
\end{pmatrix} \quad \begin{pmatrix}
y_{11} & y_{12} & \cdots \\
y_{21} & y_{22} & \cdots \\
\vdots & \vdots & \ddots 
\end{pmatrix}
\]
**Classical Classification Models**

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

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

- **Examples:**
  - **SVM:**
    $$\max_{w, \xi} \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i : \ s.t. \ w^T \Lambda f_i(y) \geq 1 - \xi_i, \ \forall i, \forall y$$
  - **Logistic Regression:**
    $$\max_{w} \mathcal{L}(\mathcal{D}; w) \triangleq \sum_{i=1}^N \log p(y_i|x_i)$$
    where
    $$p(y|x) = \frac{\exp\{w^T f(x, y)\}}{\sum_{y'} \exp\{w^T f(x, y')\}}$$
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:
    \[ y^* | x = \arg \max_{y} p_{\omega}(y | x) = \frac{1}{Z(w, x)} \exp \left\{ \sum_{p} w_{f_i}(x, y_i) \right\} \]
  - And we learn based on:
    \[ w^* | \{y_j, x_i\} = \arg \max_{w} \prod_{j} p_{\omega}(y_j | x) = \prod_{j} \frac{1}{Z(w, x)} \exp \left\{ \sum_{i} w_{f_i}(x_i, y_i) \right\} \]

- Max Margin:
  - We predict based on:
    \[ y^* | x = \arg \max_{y} \sum_{i} w_{f_i}(x, y_i) = \arg \max_{y} w^\top f(x, y) \]
  - And we learn based on:
    \[ w^* | \{y_j, x_i\} = \arg \max_{w} \min_{\{y_j, x_i\}} \left\{ w^\top \left( f(y_j, x_i) - f(y_i, x_i) \right) \right\} \]
E.g. Max-Margin Markov Networks

- Convex Optimization Problem:
  \[
  P_0 (M^3N) : \min_{w, \xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i
  \]
  s.t. \( \forall i, \forall y \neq y_i : \quad w^\top \Delta f_i(x, y) \geq \Delta f_i(y) - \xi_i, \quad \xi_i \geq 0. \)

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

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

OCR Example

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

- Equivalently:
  \[
  \begin{align*}
  w^\top f(\text{brace}) &> w^\top f(\text{aaaaa}) \quad (\text{brace”) > w^\top f(\text{“aabaab”)}) \\
  ... &> w^\top f(\text{“zzzz”)})
  \end{align*}
  \]
  \[\text{a lot!}\]
Min-max Formulation

- 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

*Crammer & Singer 01
Discriminative Learning Paradigms

**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 \( w \) which 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!)

See [Zhu and Xing, 2008]