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

Eric Xing
Lecture 6, September 24, 2014
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. \quad 0 \leq \alpha_i \leq C, \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: \(w^T x_{\text{new}} + b \leq 0\)
\[
\max_{\alpha} \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)
\]

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

- 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
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
Non-linear Decision Boundary

This dataset is not linearly separable!

transformation

\[(x_1, x_2) \rightarrow (r(x_1, x_2) \omega (\theta(x_1, x_2)))\]

radin angle

How to find a useful and inexpensive transformation?

now linearly separable!
Transforming the Data

- 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

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

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

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

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

$$K(x, x') = \left(1 + x^T x'\right)^2$$
More examples of kernel functions

- Linear kernel (we've seen it)
  \[ K(x, x') = x^T x' \]

- Polynomial kernel (we just saw an example)
  \[ K(x, x') = \left(1 + x^T x'\right)^p \]
  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 $K = K^T$
    - Proof $K_{i,j} = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = K_{j,i}$
  - Positive –semidefinite $y^T K y \geq 0 \quad \forall y$
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-definite.
SVM examples

linear

$2^{nd}$ order polynomial

$4^{th}$ order polynomial

$8^{th}$ order polynomial
Examples for Non Linear SVMs – Gaussian Kernel

Linear

Gaussian
Remember the Kernel Trick!!!

**Primal Formulation:**

\[
\begin{align*}
\min_{w, b} & \quad \frac{1}{2} w^T w + C \sum_j \xi_j \\
\text{subject to} & \quad (w^T \phi(x_j) + b) y_j \geq 1 - \xi_j \quad \forall j \\
& \quad \xi_j \geq 0 \quad \forall j
\end{align*}
\]

Infinite, cannot be directly computed

**Dual Formulation:**

\[
\begin{align*}
\max_{\alpha} & \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \\
\text{subject to} & \quad \sum_i \alpha_i y_i = 0 \\
& \quad \sum_i \alpha_i = C \\
& \quad 0 \leq \alpha_i \leq C \quad \forall i
\end{align*}
\]

But the dot product is easy to compute 😊
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):
  \[ \nu, w \in \mathcal{V} \implies \alpha \nu + \beta w \in \mathcal{V} \]

- 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.
Hilbert Space

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

- The inner product $\langle f, g \rangle$ has the following properties:
  - Symmetry $\langle f, g \rangle = \langle g, f \rangle$
  - Linearity $\langle \alpha f_1 + \beta f_2, g \rangle = \alpha \langle f_1, g \rangle + \beta \langle f_2, g \rangle$
  - Nonnegativity $\langle f, f \rangle \geq 0$
  - Zero $\langle f, f \rangle = 0 \implies f = 0$

- 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)

\[ \langle f, g \rangle = \int f(x)g(x) \, dx \]

Inner product of two functions is a number

- Traditional finite vector space inner product

\[ \langle v, w \rangle = v^\top w \]

\[ \text{scalar} \]
Recall the SVM kernel Intuition

\[
\min_{\mathbf{w}, b} \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_j \xi_j \\
(\mathbf{w}^\top \phi(\mathbf{x}_j) + b)y_j \geq 1 - \xi_j \quad \forall j \quad \xi_j \geq 0 \quad \forall j
\]

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

© Eric Xing @ CMU, 2014
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.

\[ \phi_x := K(x, \cdot) \]

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

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

\[ \phi_{1.5}(y) = \exp \left( \frac{\|1.5 - y\|^2}{\sigma^2} \right) \]
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 $k(\cdot, x)$:
    $$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i)$$

  - 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 \( k(\cdot, x) \) 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 \( f \):

\[
\langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x')
\]

\[
\int F(x) \delta(x-x') \ dx = f(x)
\]

- 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 $K$

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

- 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:

\[ F_0 = \left\{ f(z) : \sum_{j=1}^{k} \alpha_j \phi_{x_j}(z), \forall k \in \mathbb{N}_+ \text{ and } x_j \in \mathcal{X} \right\} \]

- 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

\[ f(z) = \alpha_1 \phi_1(z) + \alpha_2 \phi_2(z) \]
Summary: Reproducing Property

- It can now be derived that the inner product of a function $f$ with $\phi_x$, evaluates a function at point $x$:

$$\langle f, \phi_x \rangle = \left\langle \sum_j \alpha_j \phi_{x_j}, \phi_x \right\rangle$$

$$= \sum_j \alpha_j \langle \phi_{x_j}, \phi_x \rangle$$

Linearity of inner product

$$= \sum_j \alpha_j K(x_j, x)$$

Definition of kernel

$$= f(x)$$

Remember that

$$K(x_j, x) := \phi_{x_j}(x)$$

$\Rightarrow$ scalar
A Reproducing Kernel Hilbert Space is an Hilbert Space where for any \( X \), the evaluation functional indexed by \( X \) takes the following form:

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

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

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

\[
\begin{align*}
f (X_1) &= \langle \phi_{X_1}, f \rangle \\
g (X_1) &= \langle \phi_{X_1}, g \rangle
\end{align*}
\]

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

\[
\begin{align*}
f (X_2) &= \langle \phi_{X_2}, f \rangle \\
g (X_2) &= \langle \phi_{X_2}, g \rangle
\end{align*}
\]

- **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!!!

$$\text{Eval}_i(\cdot) = \langle e_i, \cdot \rangle$$

Homework!
RKHS or Not?

- Is the space of functions such that
  \[ \int |f(z)|^2 \, dz < \infty \]
  an RKHS?

No!!!!

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

\[ \text{Eval}_X (\cdot) = \langle \delta_X, \cdot \rangle \]

\[ f(X) = \int f(z) \delta_X(z) \, dz \]

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

Homework!
The Kernel

- 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 \( K \):

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

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

Note: \( w \cdot x_i y_i \geq 0 \)
In our primal problem, we minimize $w^T w$ subject to constraints. This is equivalent to:

\[ \|w\|^2 = w^T w = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \Phi(x_i) \Phi(x_j) \]

\[ = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \]

\[ = \|f\|^2_H \]

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:

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

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

\[ f = f_U + f_L \]
\[ = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i) + f_L(\cdot) \]

\[ f(x_j) = \langle f_U, k(\cdot, x_j) \rangle \]
\[ = \langle \sum_{i=1}^{n} \alpha_i k(\cdot, x_i) + f_L(\cdot), k(\cdot, x_j) \rangle \]
\[ = \sum_{i=1}^{n} \alpha_i k(x_i, x_j) + 0 \]
\[ = \sum_{i=1}^{n} \alpha_i k(x_i, x_j) \]

\[ \Omega_2(\|F\|_H^2) = \Omega_2 \left( \| \sum_{i=1}^{n} \alpha_i k(\cdot, x_i) \|_H^2 + \| f_L \|_H^2 \right) \]

\[ \min = \Omega_2 \]
\[ f_{U+1}^* \in H_D, \quad f_L^* \in H_0 \]
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 $w^Tw$ does not seem to imply that
  - And the representer theorem does not either!
Another view of SVM: $L_1$ regularization

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

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

- Instead we consider the following modified cost:

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

- 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!
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 \( f^{\text{opt}}(x) \) discriminant function from \( F = \{ f \} \) family of discriminants
  - classify \( y = \text{sign} \ f^{\text{opt}}(x) \)

- **E.g., SVM**

\[
f^{\text{opt}}(x) = w^T x_{\text{new}} + b
\]
Model averaging

- There exist many $f$ with near optimal performance

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

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

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

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

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

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

- Bayesian learning:
  \[ p_0(w) \]

  \[ \mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \]

  Bayes Thrm: \[ p(w | \mathcal{D}) = \frac{p(w)p(\mathcal{D} | w)}{p(\mathcal{D})} \]

- 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:
\[
- \sum_{i=1}^{N} P(x = i) \log_2 P(x = i)
\]
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 $Y$:
    \[
    H(X|Y) = - \sum_{j \in Val(y)} P(y = j) \log_2 H(X|y = j)
    \]
  - Mutual information (aka information gain) of $X$ and $Y$:
    \[
    I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y)
    \]
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 $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, find

  $$Q_{ME} = \arg\max_Q \sum_i y_i \langle f(x_i) \rangle_{Q_{ME}} - \xi_i, \quad \forall i$$

  s.t.

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

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

- Prior $Q_0(f)$

- Minimum Relative Entropy Discrimination

\[
Q_{MRE} = \arg \min_{Q} KL(Q \| Q_0) + U(\xi)
\]

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

\[
\xi_i \geq 0 \quad \forall i
\]

- 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:

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

$\alpha_i \geq 0$ 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:
    \[ D1: \max_{\alpha} - \log Z(\alpha) - U^*(\alpha) \]
    \[ \text{s.t. } \alpha_i(y) \geq 0, \forall i, \]
    
    \( U^*(\cdot) \) is the conjugate of the \( U(\cdot) \), 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 \( f(x) = w^T x + b \), \( Q_0(w) = \text{Normal}(0, I) \), \( Q_0(b) = \) non-informative prior, the Lagrange multipliers \( \alpha \) are obtained by maximizing \( J(\alpha) \) subject to \( 0 \leq \alpha_t \leq C \) and \( \sum_t \alpha_t y_t = 0 \), where

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

- Separable \( D \) ➞ SVM recovered exactly
- Inseparable \( D \) ➞ SVM recovered with different misclassification penalty

© Eric Xing @ CMU, 2014
SVM extensions

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

- **Unstructured prediction**

- **Structured prediction**
  - Part of speech tagging
    
    $x = \text{"Do you want sugar in it?"} \Rightarrow y = \langle \text{verb pron verb noun prep pron} \rangle$
  
- 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, 2014
Classical Classification Models

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

- **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: $\max_{w, \xi} \frac{1}{2} w^\top w + C \sum_{i=1}^N \xi_i; \text{ s.t. } w^\top \Delta f_i(y) \geq 1 - \xi_i, \forall i, \forall y.$
  - Logistic Regression: $\max_w \mathcal{L}(\mathcal{D}; w) \triangleq \sum_{i=1}^N \log p(y_i | x_i)$

  where

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

© Eric Xing @ CMU, 2014
\[ h(x) = \arg \max_{y \in \mathcal{Y}(x)} F(x, y) \]

**Assumptions:**

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

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

- Feasible subspace of weights:

\[
F_0 = \{w : w^\top \Delta f_i(y) \geq \Delta \ell_i(y) - \xi_i; \ \forall i, \forall y \neq y_i\}
\]

- 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:
  \[
  w^T f(\text{brace}, \text{“brace”}) > w^T f(\text{brace}, \text{“aaaaa”})
  \\
  w^T f(\text{brace}, \text{“brace”}) > w^T f(\text{brace}, \text{“aaaab”})
  \\
  \vdots
  \\
  w^T f(\text{brace}, \text{“brace”}) > w^T f(\text{brace}, \text{“zzzzz”})
  \]
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
M³ nets

33% error reduction over multiclass SVMs

Crammer & Singer 01
Discriminative Learning Paradigms

**SVM**

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

\[ \min_{w,\xi} \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \]

\[ y^i(w^T x^i + b) \geq 1 - \xi_i, \quad \forall i \]

**MED**

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

\[ \min_{Q} \quad \text{KL}(Q||Q_0) \]

\[ y^i(f(x^i))_Q \geq \xi_i, \quad \forall i \]

**MED-MN**

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

**M}^3\text{N**}

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

\[ \min_{w,\xi} \frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i \]

\[ w^T [f(x^i) - f(x^i, y)] \geq \ell(y^i, y) - \xi_i, \quad \forall i, \forall y \neq y^i \]

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!)

© Eric Xing @ CMU, 2014