Kernel methods, kernel SVM and ridge regression

Le Song

Machine Learning II: Advanced Topics
CSE 8803ML, Spring 2012
Collaborative Filtering
Collaborative Filtering

- R: rating matrix; U: user factor; V: movie factor

- Low rank matrix approximation approach

- Probabilistic matrix factorization

- Bayesian probabilistic matrix factorization

\[
\begin{align*}
\min_{U,V} & \; f(U,V) = \| R - UV^T \|_F^2 \\
\text{s.t.} & \; U \geq 0, V \geq 0, k \ll m, n.
\end{align*}
\]
Parameter Estimation and Prediction

- Bayesian treats the unknown parameters as a random variable:
  \[
P(\theta | D) = \frac{P(D|\theta)P(\theta)}{P(D)} = \frac{P(D|\theta)P(\theta)}{\int P(D|\theta)P(\theta)d\theta}
  \]
  \[
  \text{Posterior mean estimation: } \theta_{\text{bayes}} = \int \theta P(\theta | D)d\theta
  \]

- Maximum likelihood approach
  \[
  \theta_{ML} = \arg\max_{\theta} P(D|\theta), \theta_{MAP} = \arg\max_{\theta} P(\theta | D)
  \]

- Bayesian prediction, take into account all possible value of \(\theta\)
  \[
  P(x_{new}|D) = \int P(x_{new}, \theta | D)d\theta = \int P(x_{new}|\theta)P(\theta | D)d\theta
  \]

- A frequentist prediction: use a “plug-in” estimator
  \[
  P(x_{new}|D) = P(x_{new} | \theta_{ML}) \text{ or } P(x_{new}|D) = P(x_{new} | \theta_{MAP})
  \]
PMF: Parameter Estimation

- Parameter estimation: MAP estimate
  \[ \theta_{MAP} = \arg \max_\theta P(\theta|D, \alpha) = \arg \max_\theta P(\theta, D|\alpha) \]
  \[ = \arg \max_\theta P(D|\theta)P(\theta|\alpha) \]

- In the paper:
  \[ \ln p(U, V|R, \alpha, \alpha_V, \alpha_U) = \ln p(R|U, V, \alpha) + \]
  \[ + \ln p(U|\alpha_U) + \ln p(V|\alpha_V) + C, \]
PMF: Interpret prior as regularization

- Maximize the posterior distribution with respect to parameter $U$ and $V$

$$\ln p(U, V | R, \alpha, \alpha_V, \alpha_U) = \ln p(R | U, V, \alpha) +$$

$$+ \ln p(U | \alpha_U) + \ln p(V | \alpha_V) + C,$$

- Equivalent to minimize the sum-of-squares error function with quadratic regularization term (Plug in Gaussians and take log)

$$E = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} \left( R_{ij} - U_i^T V_j \right)^2$$

$$+ \frac{\lambda_U}{2} \sum_{i=1}^{N} \| U_i \|_{Fro}^2 + \frac{\lambda_V}{2} \sum_{j=1}^{M} \| V_j \|_{Fro}^2,$$
Bayesian PMF: predicting new ratings

- Bayesian prediction, take into account all possible value of $\theta$
  \[ P(x_{new} | D) = \int P(x_{new}, \theta | D) d\theta = \int P(x_{new} | \theta) P(\theta | D) d\theta \]

- In the paper, integrating out all parameters and hyperparameters.

\[
p(R_{ij}^* | R, \Theta_0) = \int \int p(R_{ij}^* | U_i, V_j) p(U, V | R, \Theta_U, \Theta_V) p(\Theta_U, \Theta_V | \Theta_0) d\{U, V\} d\{\Theta_U, \Theta_V\}.
\quad (9)
\]
Bayesian PMF: overall algorithm

Gibbs sampling for Bayesian PMF

1. Initialize model parameters \{U^1, V^1\}

2. For \( t = 1, \ldots, T \)
   - Sample the hyperparameters (Eq. 14):
     \[
     \Theta_U^t \sim p(\Theta_U | U^t, \Theta_0)
     \]
     \[
     \Theta_V^t \sim p(\Theta_V | V^t, \Theta_0)
     \]
   - For each \( i = 1, \ldots, N \) sample user features in parallel (Eq. 11):
     \[
     U_{i}^{t+1} \sim p(U_i | R, V^t, \Theta_U^t)
     \]
   - For each \( i = 1, \ldots, M \) sample movie features in parallel:
     \[
     V_{i}^{t+1} \sim p(V_i | R, U^{t+1}, \Theta_V^t)
     \]
Nonlinear classifier

Nonlinear Decision Boundaries

Linear SVM Decision Boundaries
Nonlinear regression and want to estimate the variance
Need advanced methods such as Gaussian processes and kernel regression
Nonconventional clusters

Need more advanced methods, such as kernel methods or spectral clustering to work
Nonlinear principal component analysis

PCA

Nonlinear PCA
Support Vector Machines (SVM)

\[
\begin{align*}
\text{min } & \frac{1}{2} \mathbf{w}^\top \mathbf{w} + C \sum_j \xi_j \\
\text{s.t. } & (\mathbf{w}^\top \mathbf{x}_j + b) y_j \geq 1 - \xi_j, \; \xi_j \geq 0, \; \forall j
\end{align*}
\]

\(\xi_j\): Slack variables
Solve *nonlinear* problem with *linear* relation in feature space

Nonlinear clustering, principal component analysis, canonical correlation analysis ...

Transform data points

Linear decision boundary in feature space

Non-linear decision boundary
SVM for nonlinear problems

- Some problem needs complicated and even infinite features
  \[ \phi(x) = (x, x^2, x^3, x^4, \ldots)^T \]

- Explicitly computing high dimension features is time consuming, and makes subsequent optimization costly
SVM Lagrangian

- Primal problem:
  \[
  \min_{w, \xi} \frac{1}{2} w^T w + C \sum_j \xi_j \\
  s.t. (w^T x_j + b)y_j \geq 1 - \xi_j, \xi_j \geq 0, \forall j
  \]

- Lagrangian
  \[
  L(w, \xi, \alpha, \beta) = \\
  \frac{1}{2} w^T w + C \sum_j \xi_j + \sum_j \alpha_j (1 - \xi_j - (w^T x_j + b)y_j) - \beta_j \xi_j
  \]

- \[\alpha_i \geq 0, \beta_i \geq 0\]

- Can be infinite dimensional features \(\phi(x_j)\)

- Take derive of \(L(w, \xi, \alpha, \beta)\) with respect to \(w\) and \(\xi\) we have
  \[
  w = \sum_j \alpha_j y_j x_j \\
  b = y_k - w^T x_k \text{ for any } k \text{ such that } 0 < \alpha_k < C
  \]
SVM dual problem

Plug in $w$ and $b$ into the Lagrangian, and the dual problem

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^\top x_j$$

Subject to

$$\sum_i \alpha_i y_i = 0$$

$$0 \leq \alpha_i \leq C$$

It is a quadratic programming; solve for $\alpha$, then we get

$$w = \sum_j \alpha_j y_j x_j$$

$$b = y_k - w^\top x_k$$ for any $k$ such that $0 < \alpha_k < C$

Data points corresponding to nonzeros $\alpha_i$ are called support vectors
Kernel Functions

Denote the inner product as a function $k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.

$$K(\text{ACAAGAT GCCATTG TCCCCCG GCCCTCT GCTGCTG}, \text{ACAAGAT GCCATTG TCCCCCG GCCCTCT GCTGCTG}) = 0.7$$

$$K(\text{GCATGAC GCCATTG GCCCTCT ACCTGCT GGTCCTA}, \text{GCATGAC GCCATTG GCCCTCT ACCTGCT GGTCCTA}) = 0.6$$

$$K(\text{ACAAGAT GCCATTG TCCCCCG GCCCTCT GCTGCTG}, \text{GCATGAC GCCATTG GCCCTCT ACCTGCT GGTCCTA}) = 0.5$$

$$K(\text{GCATGAC GCCATTG GCCCTCT ACCTGCT GGTCCTA}, \text{GCATGAC GCCATTG GCCCTCT ACCTGCT GGTCCTA}) = 0.2$$
Problem of explicitly construct features

- Explicitly construct feature
  \( \phi(x): \mathbb{R}^m \rightarrow F \), feature space can grow really large and really quickly

- Eg. Polynomial feature of degree \( d \)
  - \( x_1^d, x_1 x_2 \ldots x_d, x_1^2 x_2 \ldots x_{d-1} \)
  - Total number of such feature is huge for large \( m \) and \( d \)
  - \( \binom{d + m - 1}{d} = \frac{(d+m-1)!}{d!(m-1)!} \)
  - \( d = 6, m = 100 \), there are 1.6 billion terms
Can we avoid expanding the features?

- Rather than computing the features explicitly, and then compute inner product.

- Can we merge two steps using a clever kernel function $k(x_i, x_j)$
  - Eg. Polynomial kernel $d = 2$

$$
\phi(x)^T \phi(y) = \begin{pmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \\ x_2 x_1 \end{pmatrix}^T \begin{pmatrix} y_1^2 \\ y_1 y_2 \\ y_2^2 \\ y_2 y_1 \end{pmatrix} = x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2
$$

$$
= (x_1 y_1 + x_2 y_2)^2 = (x^T y)^2
$$

- Polynomial kernel $d = 2$, $k(x, y) = (x^T y)^d = \phi(x)^T \phi(y)$
What $k(x, y)$ can be called a kernel function?

- $k(x, y)$ equivalent to first compute feature $\phi(x)$, and then perform inner product $k(x, y) = \phi(x)^T \phi(y)$

- A dataset $D = \{x_1, x_2, x_3 ... x_n\}$

- Compute pairwise kernel function $k(x_i, x_j)$ and form a $n \times n$ kernel matrix (Gram matrix)

  $$K = \begin{pmatrix} k(x_1, x_2) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{pmatrix}$$

- $k(x, y)$ is a kernel function, iff matrix $K$ is positive semidefinite
  - $\forall v \in R^n, v^T K v \geq 0$
Kernel trick

- The dual problem of SVMs, replace inner product by kernel
  \[ \max_{\alpha} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \]
  - s.t. \( \sum_i \alpha_i y_i = 0 \)
  - \( 0 \leq \alpha_i \leq C \)

- It is a quadratic programming; solve for \( \alpha \), then we get
  - \( w = \sum_j \alpha_j y_j \phi(x_j) \)
  - \( b = y_k - w^\top x_k \) for any \( k \) such that \( 0 < \alpha_k < C \)

- Evaluate the decision boundary on a new data point
  \[ f(x) = w^\top \phi(x) = (\sum_j \alpha_j y_j \phi(x_j))^\top \phi(x) = \sum_j \alpha_j y_j k(x_j, x) \]
Typical kernels for vector data

- Polynomial of degree d
  \[ k(x, y) = (x^\top y)^d \]
- Polynomial of degree up to d
  \[ k(x, y) = (x^\top y + c)^d \]
- Exponential kernel (infinite degree polynomials)
  \[ k(x, y) = \exp(s \cdot x^\top y) \]
- Gaussian RBF kernel
  \[ k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \]
- Laplace Kernel
  \[ k(x, y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right) \]
- Exponentiated distance
  \[ k(x, y) = \exp\left(-\frac{d(x,y)^2}{s^2}\right) \]
Shape of some kernels

- Translation invariant kernel $k(x, y) = g(x - y)$

The decision boundary is weighted sum of bumps
- $f(x) = \sum_j \alpha_j y_j k(x_j, x)$
How to construct more complicated kernels

- Know the feature space $\phi(x)$, but find a fast way to compute the inner product $k(x, y)$
  - Eg. string kernels, and graph kernels

- Find a function $k(x, y)$ and prove it is positive semidefinite
  - Make sure the function captures some useful similarity between data points

- Combine existing kernels to get new kernels
  - What combination still results in kernel
Combining kernels I

Positive weighted combination of kernels are kernels
- \( k_1(x, y) \) and \( k_2(x, y) \) are kernels
- \( \alpha, \beta \geq 0 \)
- Then \( k(x, y) = \alpha k_1(x, y) + \beta k_2(x, y) \) is a kernel

Weighted combination kernel is like concatenated feature space
- \( k_1(x, y) = \phi(x)^\top \phi(y), k_2(x, y) = \psi(x)^\top \psi(y) \)
- \( k(x, y) = \begin{pmatrix} \phi(x) \\ \psi(x) \end{pmatrix}^\top \begin{pmatrix} \phi(y) \\ \psi(y) \end{pmatrix} \)

Mapping between spaces give you kernels
- \( k(x, y) \) is a kernel, then \( k(\phi(x), \phi(y)) \) is a kernel
- \( k(x, y) = x^2 y^2 \)
Kernel combination II

- Product of kernels are kernels
  - \( k_1(x, y) \) and \( k_2(x, y) \) are kernels
  - Then \( k(x, y) = k_1(x, y)k_2(x, y) \) is a kernel

- Product kernel is like using tensor product feature space
  - \( k_1(x, y) = \phi(x)^\top \phi(y), k_2(x, y) = \psi(x)^\top \psi(y) \)
  - \( k(x, y) = (\phi(x) \otimes \psi(x))^\top (\phi(y) \otimes \psi(y)) \)

- \( k(x, y) = k_1(x, y)k_2(x, y) \ldots k_d(x, y) \) is a kernel with higher order tensor features
  - \( k(x, y) = (\phi(x) \otimes \psi(x) \otimes \ldots \otimes \xi(x))^\top (\phi(y) \otimes \psi(y) \otimes \ldots \otimes \xi(y)) \)
Nonlinear regression

Linear regression

Nonlinear regression and want to estimate the variance
Need advanced methods such as Gaussian processes and kernel regression
Ridge regression

A dataset $X = (x_1, ..., x_n) \in R^{d \times n}, y = (y_1, ..., y_n)^T \in R^d$

With some regularization parameter $\lambda$, the goal is to find regression function $w$

$$w^* = \arg\min_w \left( \sum_i (y_i - x_i^T w)^2 + \lambda \|w\|^2 \right)$$

$$= \arg\min_w (\|y - X^T w\|^2 + \lambda \|w\|^2)$$

Find $w$: take derivative of the objective and set it to zeros

$$w^* = (XX^T + \lambda I)^{-1} Xy$$
Matrix inversion lemma

- Find $w$: take derivative of the objective and set it to zeros
  \[ w^* = (XX^T + \lambda I_d)^{-1} X y \]

- \[ w^* = (XX^T + \lambda I_d)^{-1} X y = X(X^TX + \lambda I_n)^{-1} y \]

- Evaluate $w^*$ in a new data point, we have
  \[ w^T x = y^T (X^TX + \lambda I_n)^{-1} X^T x \]

- The above expression only depends on inner products!
Kernel ridge regression

\[ w^*^\top x = y^\top (X^\top X + \lambda I_n)^{-1} X^\top x \]
only depends on inner products!

\[ X^\top X = \begin{pmatrix} x_1^\top x_1 & \ldots & x_1^\top x_n \\ \vdots & \ddots & \vdots \\ x_n^\top x_1 & \ldots & x_n^\top x_n \end{pmatrix} \]

\[ X^\top x = \begin{pmatrix} x_1^\top x \\ \vdots \\ x_n^\top x \end{pmatrix} \]

Kernel ridge regression: replace inner product by a kernel function

\[ X^\top X \rightarrow K = \begin{pmatrix} k(x_i, x_j) \end{pmatrix}_{n \times n} \]
\[ X^\top x \rightarrow k_x = \begin{pmatrix} k(x_i, x) \end{pmatrix}_{n \times 1} \]
\[ f(x) = y^\top (K + \lambda I_n)^{-1} k_x \]
Kernel ridge regression

- Use Gaussian rbf kernel
  $$k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

- Use cross-validation to choose parameters

  - large $\sigma$, large $\lambda$
  - small $\sigma$, small $\lambda$
  - small $\sigma$, large $\lambda$