CS281B/Stat241B: Advanced Topics in Learning & Decision Making

Regularizing KCCA, Cholesky decomposition

Lecturer: Michael I. Jordan *Scribe: Guillaume Obozinski*

1 KCCA

1.1 Overfitting issue

We obtained in the previous lecture a formulation of KCCA as the following generalized eigenvector problem (GEP):

$$\begin{array}{rl} \max\limits_{\alpha_a,\alpha_b} & \alpha_a \mathbf{K}_a \mathbf{K}_b \alpha_b \\ s.t. & \alpha_a^T {\mathbf{K}_a}^2 \alpha_a = 1 \\ & \alpha_b^T {\mathbf{K}_a}^2 \alpha_b = 1 \end{array}$$

Unfortunately because of the high dimensionality of the feature space the solution of this GEP overfits the data as soon as $\mathbf{K}_a$ is full rank as we show hereafter.

Writing the Lagrangian and taking the derivative with respect to α_a and α_b we get the pair of equations:

$$\begin{array}{rcl} \mathbf{K}_a \mathbf{K}_b \alpha_b & = & \lambda_a {\mathbf{K}_a}^2 \alpha_a \\ \mathbf{K}_b \mathbf{K}_a \alpha_a & = & \lambda_b {\mathbf{K}_b}^2 \alpha_b \end{array}$$

but pre-multiplying the equations respectively by α_a and α_b we notice that $\lambda_a = \lambda_b \doteq \lambda$ as a consequence of the constraints and that λ is the correlation between α_a and α_b. If we assume $\mathbf{K}_a$ to be full rank, which for example will always be the case with a gaussian kernel, then $\alpha_a = \lambda^{-1} \mathbf{K}_a^{-1} \mathbf{K}_b \alpha_b$ and therefore the equations are solved if we have $\mathbf{K}_b^2 \alpha_b = \lambda^2 \mathbf{K}_b^2 \alpha_b$ which holds for all α_b with $\lambda = 1$. So for every α_b there exists α_a such that they are correlated perfectly. To prevent overfitting, we add a constraint to regularize.

1.2 Regularized version of KCCA

We add a constraint on the norm of $\mathbf{w}_a$ and $\mathbf{w}_b$ which corresponds to

$$\begin{array}{rc} \max\limits_{\alpha_a,\alpha_b} & \alpha_a \mathbf{K}_a \mathbf{K}_b \alpha_b \\ s.t. & (1-\tau)\ \alpha_a^T {\mathbf{K}_a}^2 \alpha_a + \tau\ \alpha_a^T \mathbf{K}_a \alpha_a = 1 \\ & (1-\tau)\ \alpha_b^T {\mathbf{K}_b}^2 \alpha_b + \tau\ \alpha_b^T \mathbf{K}_b \alpha_b = 1 \end{array}$$

Taking the derivative of the Lagrangian we get:

$$\begin{pmatrix} 0 & \mathbf{K}_a \mathbf{K}_b \\ \mathbf{K}_b \mathbf{K}_a & 0 \end{pmatrix} \begin{pmatrix} \alpha_a \\ \alpha_b \end{pmatrix} = \lambda \begin{pmatrix} (1-\tau)\ {\mathbf{K}_a}^2 + \tau\ \mathbf{K}_a & 0 \\ 0 & (1-\tau)\ {\mathbf{K}_b}^2 + \tau\ \mathbf{K}_b \end{pmatrix} \begin{pmatrix} \alpha_a \\ \alpha_b \end{pmatrix}$$

Solving the regularized version of KCCA is therefore also equivalent to solving a generalized eigenvalue problem.

2 Cholesky decomposition

Working with kernel matrices that grow with the size of the data set allows us to work in a non-parametric setting. However, it doesn't only imply to have to store n^2 entries, but also to face the complexity of matrix eigenvalue problems and that of solving linear regressions which in general have a complexity in n^3. For a positive definite matrix A, we can use an incomplete version of the Cholesky decomposition $A = LL^T$, where L is a lower triangular matrix determined uniquely by this equation, to reduce the cost of subsequent operations on the matrix. The Cholesky decomposition is a variant of Gaussian elimination that has a complexity reduced by a factor 2 compared to standard Gaussian elimination.

2.1 Left-looking Cholesky

The relation between the components of A and L are as follows:

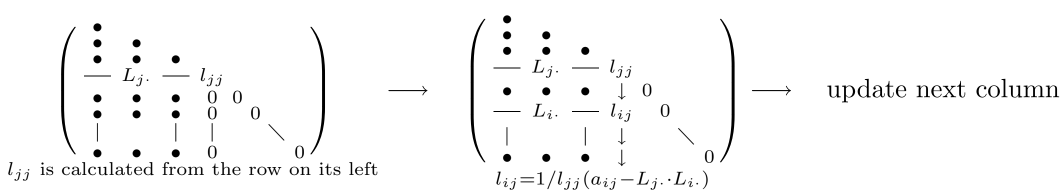
$$a_{ij} = \sum_{k=1}^{j} l_{ik} l_{jk} = \sum_{k=1}^{j-1} l_{ik} l_{jk} + l_{ij} l_{jj} \ , \quad j < i \quad \text{w.l.o.g}$$

So that:

$$l_{jj}^2 = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \qquad (1)$$

$$l_{ij} = \frac{1}{l_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right) \qquad (2)$$

where we assumed A is full rank so that $l_{jj} \neq 0$.
These equations indicate that the l_{ij} can be calculated recursively from left to right column by column, hence the name left-looking. Moreover is could be calculated directly in the matrix A without requiring additional memory.



2.2 Right-looking Cholesky

We give here an algorithm that proceeds in the opposite order. Let's identify:

$$A = \begin{pmatrix} \lambda_{11} & 0 \\ l_1 & \tilde{L} \end{pmatrix} \begin{pmatrix} \lambda_{11} & 0 \\ l_1 & \tilde{L} \end{pmatrix}^T = \begin{pmatrix} \alpha_{11} & a_1^T \\ a_1 & \tilde{A} \end{pmatrix}$$

which yields

$$
\begin{aligned}
\alpha_{11} = \lambda_{11}^2 \quad &\Leftrightarrow \quad \lambda_{11} = \sqrt{\alpha_{11}} \\
a_1 = \lambda_{11} l_1 \quad &\Leftrightarrow \quad l_1 = \frac{a_1}{\sqrt{\alpha_{11}}} \\
\tilde{A} = l_1 l_1^T + \tilde{L}\tilde{L}^T \quad &\Leftrightarrow \quad \tilde{L}\tilde{L}^T = \tilde{A} - \frac{a_1 a_1^T}{\alpha_{11}} \quad \text{(i.e. the Schur complement of } \alpha_{11}\text{)}
\end{aligned}
$$

The algorithm suggested by these equations also goes from left to right but instead of using the previously calculated columns of L to build the new ones, we modify the matrix $\tilde{A}$ to the right of the current column before we go on and calculate the next columns only using the updated $\tilde{A}$.

2.3 Incomplete Cholesky decomposition

To reduce the complexity we look for a low rank approximation for L: we try to find $\tilde{L}$ so that $A \approx \tilde{L}\tilde{L}^T$ by which we mean that $||A - \tilde{L}\tilde{L}^T||_F$ is small (where $|| \cdot ||_F$ is the Frobenius norm). The principle of the algorithm is to skip a column if its diagonal element l_{jj} is small (for a kernel l_{jj} is equal to the residual of $\phi(x_j)$ after projection on the subspace spanned by the previously selected vectors $\phi(x_k)$). In order to do this the diagonal coefficients l_{jj} are not just updated when the whole column is selected and updated: all the diagonal elements the the right of the column being updated are updated when that column is selected. To select a column to update, we find the largest diagonal element, and pivot that element up to the current diagonal (by exchanging corresponding rows and columns). Then the column is updated. The algorithm halts when all of the remaining diagonal elements are belows some threshold.

$$
\begin{pmatrix}
\vdots & \vdots & \vdots & & & \\
- & L_{j\cdot} & - & l_{jj} & & \\
\bullet & \bullet & \bullet & 0 & \ddots & \\
- & L_{i\cdot} & - & 0 & l_{ii} & \\
| & & | & | & & \ddots \\
- & L_{n\cdot} & - & 0 & & l_{nn}
\end{pmatrix}
\longrightarrow
\begin{pmatrix}
\vdots & \vdots & \vdots & & & \\
- & L_{i^*\cdot} & - & l_{i^*i^*} & & \\
\bullet & \bullet & \bullet & 0 & \ddots & \\
- & L_{j\cdot} & - & 0 & l_{jj} & \\
| & & | & | & & \ddots \\
- & L_{n\cdot} & - & 0 & & l_{nn}
\end{pmatrix}
\longrightarrow
\begin{pmatrix}
\vdots & \vdots & \vdots & & & \\
- & L_{i^*\cdot} & - & l_{i^*i^*} & & \\
\bullet & \bullet & \bullet & \downarrow & \ddots & \\
- & L_{j\cdot} & - & l_{ji^*} & l_{jj} & \\
| & & | & \downarrow & & \ddots \\
\bullet & \bullet & \bullet & \downarrow & & l_{nn}
\end{pmatrix}
$$

$\forall i \geq j$, l_{ii} is updated — row & column j & $i^* = \underset{i}{argmax}\, l_{ii}$ are swapped — the column is calculated as previously

2.4 Connection between Cholesky and QR decomposition

The QR decomposition decomposes a matrix A into $A = QR$ where

- Q is orthonormal
- R is upper triangular

The QR decomposition is obtained from the Gram-Schmidt orthonormalization: Q is the orthormal basis obtained from orthonormalizing the columns of A; the columns of R are the coordinates of the columns of A in the orthonormal basis formed by the columns of Q [1]. So given a design matrix

$$X = \begin{pmatrix} -\phi(x_1)- \\ \vdots \\ -\phi(x_n)- \end{pmatrix}$$

[1] the Gram-Schmidt algorithm is actually known to be somewhat unstable but Householder provided a numerically stable algorithm to calculate the QR decomposition

$X^T = QR$ corresponds to an orthonormalization of the feature vectors and we have $K = XX^T = R^T R$ which shows that the Cholesky decomposition of a kernel gives the coordinates of the data in an orthonormal basis of feature space.

This means that Incomplete Cholesky corresponds to a Partial Gram-Schmidt orthogonalization in which we skip every vector that is too close to the current subspace, in the sense that its orthogonal component is too small. Indeed, the norm of that orthogonal component is exactly the diagonal element of the current column.

2.5 Uses of the incomplete Cholesky decomposition

Many kernel matrices have low rank, and incomplete Cholesky decomposition often yields a decomposition involving many fewer columns than the original matrix. If k columns are returned, the storage requirements are $O(kn)$ instead of $O(n^2)$, and the running time of many algorithms reduces to $O(nk^2)$ instead of $O(n^3)$.