Statistical LDA's Results on Various Datasets

Sihui Wang

1. Methods

1.1 Formulation

$$\mathbb{P} \sim \mathcal{N}(\mu_1, \Sigma_1)$$
$$\mathbb{Q} \sim \mathcal{N}(\mu_2, \Sigma_2)$$

Suppose $\mathbb{P}, \mathbb{Q} \in \mathbb{R}^{n\times 1}$, our aim is to find matrices $\mathbf{A} \in \mathbb{R}^{r\times n}$ which could maximally preserve several *f* divergences between $\mathbb{P}_{\mathrm{A}} = \mathbf{A} \cdot \mathbb{P}$ and $\mathbb{Q}_{\mathrm{A}} = \mathbf{A} \cdot \mathbb{Q}$.

1.2 Computation of Gradients

For brevity, in the following we denote $\mu_2 - \mu_1$ by μ.

KL-divergence $\mathbf{D_{KL}}(\mathbb{Q}_{\mathbf{A}}||\mathbb{P}_{\mathbf{A}})$

$$\mathrm{D_{KL}}(\mathbb{Q}_{\mathrm{A}}||\mathbb{P}_{\mathrm{A}}) = \frac{1}{2}[\log\frac{|A\Sigma_1A^T|}{|A\Sigma_2A^T|} + tr[(A\Sigma_1A^T)^{-1}(A\Sigma_2A^T)] + \mu^TA^T(A\Sigma_1A^T)^{-1}A\mu - r]$$

Its gradient on Euclidean space:

$$\begin{aligned}\nabla_{\mathrm{A}}KL(\mathbb{Q}||\mathbb{P}) = {} & (\mathrm{A}\Sigma_1\mathrm{A}^{\mathrm{T}})^{-1}A\Sigma_1 - (\mathrm{A}\Sigma_2\mathrm{A}^{\mathrm{T}})^{-1}A\Sigma_2 + (A\Sigma_1A^T)^{-1}A\Sigma_2 \\ & - (A\Sigma_1A^T)^{-1}(A\Sigma_2A^T)(A\Sigma_1A^T)^{-1}A\Sigma_1 - (A\Sigma_1A^T)^{-1}A\mu\mu^TA^T(A\Sigma_1A^T)^{-1}A\Sigma_1 \\ & + (A\Sigma_1A^T)^{-1}A\mu\mu^T\end{aligned}$$

KL-divergence $\mathbf{D_{KL}}(\mathbb{P}_{\mathbf{A}}||\mathbb{Q}_{\mathbf{A}})$

$$\mathrm{D_{KL}}(\mathbb{P}_{\mathrm{A}}||\mathbb{Q}_{\mathrm{A}}) = \frac{1}{2}[\log\frac{|A\Sigma_2A^T|}{|A\Sigma_1A^T|} + tr[(A\Sigma_2A^T)^{-1}(A\Sigma_1A^T)] + \mu^TA^T(A\Sigma_2A^T)^{-1}A\mu - r]$$

Its gradient on Euclidean space:

$$\begin{aligned}\nabla_{\mathrm{A}}KL(\mathbb{P}||\mathbb{Q}) = {} & (\mathrm{A}\Sigma_2\mathrm{A}^{\mathrm{T}})^{-1}A\Sigma_2 - (\mathrm{A}\Sigma_1\mathrm{A}^{\mathrm{T}})^{-1}A\Sigma_1 + (A\Sigma_2A^T)^{-1}A\Sigma_1 \\ & - (A\Sigma_2A^T)^{-1}(A\Sigma_1A^T)(A\Sigma_2A^T)^{-1}A\Sigma_2 - (A\Sigma_2A^T)^{-1}A\mu\mu^TA^T(A\Sigma_2A^T)^{-1}A\Sigma_2 \\ & + (A\Sigma_2A^T)^{-1}A\mu\mu^T\end{aligned}$$

Symmetric KL-divergence

$$\begin{aligned}\mathrm{D_{SKL}}(\mathbb{P}_{\mathrm{A}}, \mathbb{Q}_{\mathrm{A}}) = {} & \frac{1}{2}[tr[(A\Sigma_1A^T)^{-1}(A\Sigma_2A^T)] + tr[(A\Sigma_2A^T)^{-1}(A\Sigma_1A^T)] + \mu^TA^T(A\Sigma_1A^T)^{-1}A\mu \\ & + \mu^TA^T(A\Sigma_2A^T)^{-1}A\mu] - r\end{aligned}$$

Its gradient on Euclidean space:

$$\begin{aligned}\nabla_{\mathrm{A}}SKL = {} & (A\Sigma_1A^T)^{-1}A\Sigma_2 - (A\Sigma_1A^T)^{-1}(A\Sigma_2A^T)(A\Sigma_1A^T)^{-1}A\Sigma_1 + (A\Sigma_2A^T)^{-1}A\Sigma_1 \\ & - (A\Sigma_2A^T)^{-1}(A\Sigma_1A^T)(A\Sigma_2A^T)^{-1}A\Sigma_2 - (A\Sigma_1A^T)^{-1}A\mu\mu^TA^T(A\Sigma_1A^T)^{-1}A\Sigma_1 \\ & + (A\Sigma_1A^T)^{-1}A\mu\mu^T - (A\Sigma_2A^T)^{-1}A\mu\mu^TA^T(A\Sigma_2A^T)^{-1}A\Sigma_2 + (A\Sigma_2A^T)^{-1}A\mu\mu^T\end{aligned}$$

Hellinger distance

$$\log\left(1-\mathrm{H}^2(\mathbb{P}_{\mathrm{A}},\mathbb{Q}_{\mathrm{A}})\right)$$
$$= -\frac{1}{2}\log\left|A\frac{\Sigma_1+\Sigma_2}{2}A^T\right| + \frac{1}{4}\log|A\Sigma_1A^T| + \frac{1}{4}\log|A\Sigma_2A^T|$$
$$-\frac{1}{8}\mu^TA^T\left(A\frac{\Sigma_1+\Sigma_2}{2}A^T\right)^{-1}A\mu$$

Its gradient on Euclidean space (For brevity, we denote $\frac{\Sigma_1+\Sigma_2}{2}$ by Σ):

$$\nabla_{\mathrm{A}}(1-H^2) = -(A\Sigma A^T)^{-1}A\Sigma + \frac{1}{2}(A\Sigma_1A^T)^{-1}A\Sigma_1 + \frac{1}{2}(A\Sigma_2A^T)^{-1}A\Sigma_2$$
$$+\frac{1}{4}(A\Sigma A^T)^{-1}A\mu\mu^TA^T(A\Sigma A^T)^{-1}A\Sigma - \frac{1}{4}(A\Sigma A^T)^{-1}A\mu\mu^T$$

1.3 An Overview of Possible Algorithms

Here our problem is to minimize the cost functions under certain constraints. The cost functions are $-\mathrm{D}_{\mathrm{KL}}(\mathbb{Q}_{\mathrm{A}}||\mathbb{P}_{\mathrm{A}})$, $-\mathrm{D}_{\mathrm{KL}}(\mathbb{P}_{\mathrm{A}}||\mathbb{Q}_{\mathrm{A}})$, $-\mathrm{D}_{\mathrm{SKL}}(\mathbb{P}_{\mathrm{A}},\mathbb{Q}_{\mathrm{A}})$ and $\log\left(1-\mathrm{H}^2(\mathbb{P}_{\mathrm{A}},\mathbb{Q}_{\mathrm{A}})\right)$. The constraints could be $\mathrm{AA}^{\mathrm{T}} = I_r$. Since we could verify that all the above-mentioned cost functions are invariant under *A*'s left multiplying arbitrary rank-*r* matrices, we can actually further narrow down the constraints (which will be addressed later in this section).

Virtually all the gradient-based algorithms have to answer two questions:

1. How to deal with the constraints?
2. How to improve the rate of convergence?

1.3.1 Dealing with the constraints

Non-Manifolds based algorithms

One branch of algorithms considers the constraints as an extra *penalty term* in the cost functions. This includes *Augmented Lagrange Method* (ALM).

Manifolds based algorithms

Another branch of algorithms manages to convert the constrained optimization problems on Euclidean spaces to unconstrained optimization problems on *manifolds*. The constraint $\mathrm{AA}^{\mathrm{T}} = I_r$, for example, means a $nr - \frac{1}{2}r(r+1)$ dimensional Stiefel manifold $\mathrm{St}(r,n)$:

$$\mathrm{St}(r,n) = \{A \in \mathbb{R}^{\mathrm{r}\times\mathrm{n}}: \mathrm{AA}^{\mathrm{T}} = I_r\}$$

In our cases, the cost functions are linear invariant, which means that *A* and $\tilde{A}$ can be considered as equivalent if $\mathrm{Span}(A) = \mathrm{Span}(\tilde{A})$ ($\mathrm{Span}(A)$ is the linear space spanned by *r* row vectors of *A*). This equivalent relation defines a *quotient manifold* structure, which is called *Grassmann manifold* $G(r,n)$. $G(r,n)$ is a $r(n-r)$ dimensional manifold consisting of all the *r* dimensional linear subspaces in an *n* dimensional linear space.

First-order manifolds based algorithms

There are first-order and second-order algorithms on the manifolds. For the first-order algorithms, the core idea is to compute the gradient and to impose the constraints at the same time, which is achieved by the technique of '*retraction*'. Intuitively, retraction means to calculate the gradients on Euclidean spaces first, and then to somehow pull the gradient vectors back onto the surface of the manifolds which are immersed in the Euclidean spaces.

Classical first-order algorithms tried to calculate the *geodesics* on the manifolds. Geodesics are curves that never change their directions, which are the generalized form of straight lines on the manifolds. For any point x_k on the manifold *M*, they first compute the gradient direction $\gamma'(0)$ in the tangent space $T_{x_k}M$, then they try to compute the geodesics $\gamma(t)$ which obey the following condition:

$$\mathrm{D}_{\gamma'(t)}\gamma'(t) = 0$$

The geodesics are obtained by exponential maps. We call it exponential maps, partially because we are trying to obtain $\gamma(t)$ given $\gamma(0)$ and $\gamma'(0)$.

Another reason is that, at least in some circumstances, the geodesics are obtained literally by taking the exponential maps of certain matrices (we will have a discussion of this later).

Calculating the exponential maps and geodesics are computationally expensive, so more recent works focus on other kinds of projections. They don't search points strictly along the geodesics, instead they compute the gradients and project them back to the manifolds in a certain manner. These projections/retractions might be not accurate, but they are much more computationally efficient.

Second-order manifolds based algorithms

Just like their counterparts on Euclidean spaces, second-order algorithms on manifolds take the information of the second order derivatives into consideration.

Unlike Euclidean spaces, on manifolds there is a technical problem: how to compare two tangent vectors on the tangent bundle TM?

To solve that technical problem in Riemannian geometry we resort to the concept of *connection*, which can be intuitively interpreted as the directional derivatives of vectors, that is, the second order derivatives since vectors themselves are the first-order derivatives.

We have the following:

$$\mathrm{D}_{\frac{\partial}{\partial x^i}}\frac{\partial}{\partial x^j} = \sum\nolimits_k \Gamma_{ij}^k \frac{\partial}{\partial x^k}$$

Additionally, we have the *Christoffel symbols*, which shows that the coefficients Γ_{ij}^k are determined by the Riemannian metric $g_{ij} =< \frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j} >$. So, each Riemannian manifold can uniquely determine a torsion free affine connection (*Levi-Civita Connection*), which in turn determines the second order derivatives on the manifolds.

The above is how the second derivatives are determined on the manifolds. Another important concept that is determined by *connections* is *parallel translation of vectors.* Intuitively, *parallel translation* is to identify "the same direction" on a rugged surface. For example, once we learned what is the direction of East for every point on the Earth, we established a parallel vector field. By establishing a parallel vector field we build isomorphisms (linear correspondence) among tangent spaces and hence we are able to tell the difference between two tangent vectors on manifolds.

Just like geodesics, parallel translations are hard to compute. So, in recent works researchers turn to a generalized term, *vector transport*, which is important for efficient numerical implementations.

Newton's method on manifolds can guarantee to converge to a *critical* point on the manifold, but it can't tell whether it is a local minimum, a local maximum, or a saddle point. Riemannian Trust Regions Method, on the other hand, can guarantee to converge to a local minimum.

1.3.2 Rate of Convergence (some preliminary thoughts)

Virtually all the first-order and second-order manifolds-based optimization algorithms have their counterparts in Euclidean spaces. Although manifolds-based algorithms are formalized in the language of Riemannian geometry and they have certain challenges in the field of numerical implementation mostly because of their increased complexity, the ideas behind these algorithms are not something new from the viewpoint of optimization. Apart from the problem mentioned in 1.3.1, a much more important and challenging problem is how to improve the rate of convergence.

For rate of convergence, we have known the following:

1) If the cost function is *strongly convex* and has *Lipschitz continuous gradient*, then the first-order gradient descent algorithm has *linear convergence rate* and its convergence order is $\mathcal{O}(\log\frac{1}{\epsilon})$.

2) If the cost function is *convex* and has *Lipschitz continuous gradient*, then the first order gradient descent algorithm has *sublinear convergence rate* and its convergence order is $\mathcal{O}(\frac{1}{\epsilon})$.

3) If the cost function is *non-convex* and has *Lipschitz continuous gradient*, then the first order gradient descent algorithm has *sublinear convergence rate* and its convergence rate is $\mathcal{O}(\frac{1}{\epsilon^2})$.

4) Newton-like algorithms are said to have *superlinear convergence rate* if the approximated Hessian operator is accurate enough.

What I have observed from the experiments are:

Case I: $\mathbb{P}\sim\mathcal{N}(0,I_n)$, $\mathbb{Q}\sim\mathcal{N}(0,\Sigma)$. In this situation, almost surely first-order algorithms have linear convergence rate and second-order algorithms have quadratic convergence rate.

Case II: $\mathbb{P}\sim\mathcal{N}(0,I_n)$, $\mathbb{Q}\sim\mathcal{N}(\mu,\Sigma)$, $\mu\neq 0$. In this situation, usually the algorithms converge as fast as in Case I. However, sometimes the algorithms take more iterations to converge.

Case III: $\mathbb{P}\sim\mathcal{N}(\mu_1,\Sigma_1)$, $\mathbb{Q}\sim\mathcal{N}(\mu_2,\Sigma_2)$, $\mu_1\neq\mu_2$, $\Sigma_1\neq\Sigma_2$. In this situation, the algorithms usually take much more iterations to converge. Sometimes we can observe that second-order algorithms have linear or sublinear convergence rate.

Case IV: The cost function is $-\sum_{1\le i<j\le n} D_{SKL}(\mathbb{P}_i,\mathbb{P}_j)$, $n>2$. In this situation, the algorithms could be very slow to converge. Frequently the second-order algorithms only have sublinear convergence rate.

At this time, I didn't have a thorough investigation of these, my preliminary thought, however, is that the cost functions tend to worsen from Case I to Case IV in their convexity (or maybe smoothness). So, maybe we could try some *convex relaxation* techniques to accelerate the algorithms.

1.3.3 Discussions and Possible Improvements

Manifolds-based optimization algorithms have been considered as common practice by some researchers. For example, in the paper, "Unifying Linear Dimensionality Reduction", John P. Cunningham and Zoubin Ghahramani have proposed that a number of linear dimensionality reduction problems could be formulated as unconstrained optimization problems on manifolds. According to their experiments, manifolds-based optimization algorithms could produce favorable outcomes compared with heuristic algorithms.

For manifolds-based optimization algorithms, it is important to find efficient ways to compute retractions and vector transportations. Meanwhile, according to Jiang Hu, Xin Liu, Zaiwen Wen and Yaxiang Yuan's paper, "A Brief Introduction to Manifold Optimization", many problems that are associated with matrices with orthogonal constraints or rank constraints have been efficiently solved.

We can treat the constraints with either manifolds-based methods or non-manifold based methods. Anyway, how to accelerate the algorithms and guarantee the rate of convergence is a much more important question. A good example of this is *Robust PCA*, which can be efficiently and accurately solved by augmented Lagrange multiplier method after convex relaxations. So, maybe looking for efficient convex relaxations for statistical LDA is a possible direction for future works.

1.4 An Example of First-Order Algorithms

Here I'll describe a first-order gradient descent algorithm that I've come up with.

1.4.1 Motivation

Since in our problems the cost functions are linear invariant, we could find the solution in the following manner:

Obtain an orthonormal basis in $\mathbb{R}^n$;

Rotate these orthonormal vectors to a right position;

Obtain the first *r* vectors to build $\mathbf{A} \in \mathbb{R}^{r\times n}$.

All the rotations are elements in the Lie Group of $\mathrm{GL}(n)$. Their derivatives, are elements in the Lie Algebra of $\mathrm{gl(n)}$. This is why we might need some knowledge about Lie Groups and Lie Algebras.

1.4.2 Preliminaries about Lie Group and Lie Algebra

We take GL(3) and gl(3) as examples.

The Lie Algebra gl(3) is a 3 dimensional algebra. It has the following elements as a basis:

$$r_{12} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, r_{13} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, r_{23} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$

By taking exponential map $\mathrm{e}^{\mathrm{A}} = I + \sum_{i=1}^{\infty} \frac{A^i}{i!}$, we could obtain the following elements in GL(3):

$$\mathrm{e}^{\theta r_{12}} = \begin{pmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\mathrm{e}^{\theta r_{13}} = \begin{pmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{pmatrix}$$

$$\mathrm{e}^{\theta r_{23}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{pmatrix}$$

So, any 3 dimensional rotation Φ can be decomposed into a rotation in the *xy*-plane, a rotation in the *xz*-plane, and a rotation in the *yz*-plane. In the language of Lie Algebra, we say:

$$\phi = \theta_1 r_{12} + \theta_2 r_{13} + \theta_3 r_{23}$$

By taking exponential maps, we obtain:

$\Phi = \mathrm{e}^{\phi} = e^{\theta_1 r_{12} + \theta_2 r_{13} + \theta_3 r_{23}}$

$$= \begin{pmatrix} cos\theta_1 & sin\theta_1 & 0 \\ -sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} cos\theta_2 & 0 & sin\theta_2 \\ 0 & 1 & 0 \\ -sin\theta_2 & 0 & cos\theta_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta_3 & sin\theta_3 \\ 0 & -sin\theta_3 & cos\theta_3 \end{pmatrix}$$

For *n* dimensional linear spaces, we have similar facts that Lie Algebra gl(n) has a basis of $r_{ij} (1 \le i < j \le n)$, which means that all *n* dimensional rotations can be decomposed into rotations in 2 dimensional linear subspaces. The elements in gl(n) could be considered as derivatives and we could obtain the associated elements in GL(n) which actually describe the rotational transformations by taking exponential maps.

1.4.3 An Algorithm: Rotational Search

For any $\mathbf{A} \in \mathbb{R}^{r\times n}$, $AA^T = I_r$, we could expand $\mathbf{A}$ into an orthonormal basis:

$$\widetilde{\mathbf{A}} = \begin{pmatrix} \mathbf{A} \\ \mathbf{A}^{\perp} \end{pmatrix}$$

Now suppose we have a rotation $\Phi = \mathrm{e}^{\theta r_{ij}}$. Our question is: how will the cost functions change if we are applying the transformation $\widetilde{\mathbf{A}}' = \Phi\widetilde{\mathbf{A}}$?

Suppose $\widetilde{\mathbf{A}} = (a_{ij})$, then we have:

$$\widetilde{\mathbf{A}}' = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{i1}cos\theta + a_{j1}sin\theta & \cdots & a_{in}cos\theta + a_{jn}sin\theta \\ \vdots & & \vdots \\ a_{j1}cos\theta - a_{i1}sin\theta & \cdots & a_{jn}cos\theta - a_{in}sin\theta \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

Suppose $\theta \to 0$, then we have:

$$\widetilde{\mathbf{A}}' - \widetilde{\mathbf{A}} = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ a_{j1}\theta & \cdots & a_{jn}\theta \\ \vdots & & \vdots \\ -a_{i1}\theta & \cdots & -a_{in}\theta \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

For brevity we denote $\nabla_{\mathrm{A}} D_f$ by $G = (g_{ij})$. Then we have:

$$\Delta D_f = \theta(\sum_{k=1}^{n} g_{ik}a_{jk} - \sum_{k=1}^{n} g_{jk}a_{ik})$$

This means that by applying the transformation $\widetilde{\mathbf{A}}' = \Phi\widetilde{\mathbf{A}}$ the cost function D_f will change by ΔD_f if θ is sufficiently small.

Reorganize the above formulas in the matrix form and we will have:

$$\nabla_{\Phi} D_f = (\nabla_A D_f) \cdot \widetilde{\mathbf{A}}^T - \widetilde{\mathbf{A}} \cdot (\nabla_A D_f)^T = G\widetilde{\mathbf{A}}^T - \widetilde{\mathbf{A}}G^T$$

$\nabla_{\Phi} D_f$ could be considered as a "*rotational gradient*" for the cost functions D_f. In fact $\nabla_{\Phi} D_f$ is an anti-symmetric matrix:

$$\nabla_{\Phi} D_f = \begin{pmatrix} 0 & L \\ -L & 0 \end{pmatrix}$$

where $L \in \mathbb{R}^{r\times(n-r)}$. So $\nabla_{\Phi} D_f$ can be considered as an element in the Lie Algebra of gl(n) which shows the optimal direction of rotation along which the cost functions will change the fastest. By applying certain exponential map, we could obtain the associated optimal rotational transformation $\mathrm{R} \in \mathrm{GL}(n)$:

$$\mathrm{R} = \mathrm{e}^{\eta \cdot \nabla_{\Phi} D_f}$$

Then we update $\widetilde{\mathbf{A}}$ by this rotational transformation R:

$$\widetilde{\mathbf{A}}_{k+1} = \mathrm{R}\widetilde{\mathbf{A}}_k$$

which completes the search along the direction of the rotational gradient.

For unconstrained optimization problems, all we need to do is to calculate $\nabla_{\mathrm{A}} D_f$ and update

A by $A_{k+1} = A_k + \eta \nabla_A D_f$. Here for our constrained optimization problems, we need to additionally calculate $\nabla_\Phi D_f = (\nabla_A D_f) \cdot \widetilde{\mathbf{A}}^T - \widetilde{\mathbf{A}} \cdot (\nabla_A D_f)^T$ and $R = e^{\eta \cdot \nabla_\Phi D_f}$, and update A by $\widetilde{\mathbf{A}}_{k+1} = R\widetilde{\mathbf{A}}_k$. These additional calculations guarantee that our search obey the constraints of Grassmann manifolds.

1.5 Equivalence of the problems

Suppose we have two problems:

Problem 1:

$$\mathbb{P} \sim \mathcal{N}(\mu_1, \Sigma_1)$$
$$\mathbb{Q} \sim \mathcal{N}(\mu_2, \Sigma_2)$$

Problem 2:

$$\mathbb{P} \sim \mathcal{N}(P\mu_1, P\Sigma_1 P^T)$$
$$\mathbb{Q} \sim \mathcal{N}(P\mu_2, P\Sigma_2 P^T)$$

where $P \in \mathbb{R}^{n \times n}$ is an invertible, symmetric matrix. For brevity, we denote the cost functions of problem 1 and problem 2 by $F(A, \Sigma_1, \Sigma_2, \mu_1, \mu_2)$. Then we can verify the following:

1) A is a critical point for $F(\cdot, \Sigma_1, \Sigma_2, \mu_1, \mu_2) \Leftrightarrow AP^{-1}$ is a critical point for $F(\cdot, P\Sigma_1 P^T, P\Sigma_2 P^T, P\mu_1, P\mu_2)$;

and 2) $F(A, \Sigma_1, \Sigma_2, \mu_1, \mu_2) = F(AP^{-1}, P\Sigma_1 P^T, P\Sigma_2 P^T, P\mu_1, P\mu_2)$.

This means that 1) problem 1 and problem 2 have the same number of critical points, 2) the extremal values of the cost functions for problem 1 and problem 2 are identical, and 3) there is an invertible transformation P^{-1} between the solutions for problem 1 and the solutions for problem 2.

To conclude, problem 1 and problem 2 are equivalent to each other.

Why mentioning this equivalence?

As has been mentioned in Section 1.3.2, usually it is faster and takes fewer iterations to solve the problems of $\mathbb{P} \sim \mathcal{N}(0, I_n)$, $\mathbb{Q} \sim \mathcal{N}(\mu, \Sigma)$ than the problems of $\mathbb{P} \sim \mathcal{N}(\mu_1, \Sigma_1)$, $\mathbb{Q} \sim \mathcal{N}(\mu_2, \Sigma_2)$. So, we can always convert the problem of $\mathbb{P} \sim \mathcal{N}(\mu_1, \Sigma_1)$, $\mathbb{Q} \sim \mathcal{N}(\mu_2, \Sigma_2)$ to the problem of $\mathbb{P} \sim \mathcal{N}(\Sigma_1^{-\frac{1}{2}}\mu_1, I_n)$, $\mathbb{Q} \sim \mathcal{N}(\Sigma_1^{-\frac{1}{2}}\mu_2, \Sigma_1^{-\frac{1}{2}}\Sigma_2\Sigma_1^{-\frac{1}{2}})$, which could practically save the computational time.

Remaining Questions

Discrepancy

Although problem 1 and problem 2 are equivalent in theory, sometimes they just have different numerical results.

For example, in some cases the algorithms constantly produce better results for problem 1. Suppose the algorithms find A as the solution for problem 1. What I have observed is that, the algorithms are able to confirm that AP^{-1} is a solution for problem 2, if we use AP^{-1} as a start point for problem 2. However, if we let the algorithms search from randomized start points, then they will produce sub-optimal results and they sometimes are not able to find and converge to AP^{-1}. For some datasets, these constantly happen, while for other datasets, these seldom happen.

Degeneration: what if the covariance matrices are not of full rank?

One possible solution is to use $\Sigma + \epsilon I_n$ instead of Σ if Σ is degenerate. Preliminary tests have shown that a small perturbation of ϵI_n is not likely to produce dramatic change of the results. Since the algorithms are probably robust against such perturbations, using $\Sigma + \epsilon I_n$ instead of Σ

could be a possible solution to the problem of degeneration.

1.6 Statistical LDA for multiple classes

The cost function that I have tried so far is the following:

$$-\sum_{1\le i<j\le n} D_{SKL}(\mathbb{P}_i, \mathbb{P}_j)$$

Sometimes it takes a very long time for the optimization algorithms to converge. Once I tried to use this cost function to study the MNIST dataset. The Riemannian Trust Regions Method run for more than 2 days with no sign of convergence, and the *f*-divergence it found is more than 10^8. So, for multiple classes, maybe we should keep a balance between accuracy and speed.

2. Experiments

In this section, we compare Fisher's LDA's performance with our methods. For statistical LDA, we use several different cost functions: KL-1, KL-*2* (forwards and backwards KL divergence), SKL (symmetric KL divergence) and H (Hellinger distance).

First, we use the whole dataset as the training set. After we learn the projections, we test how well are the projections able to separate data points from different classes on the same dataset.

Then, we split the dataset into a training set and a testing set. The size of the training set and the testing set is 7:3. We learn the projections from the training set and test their performance on the testing set.

At last, we compare Fisher's LDA with statistical LDA for total number of errors and estimated P_{error}.

2.1 Occurrence of Misclassification

2.1.1 Iris Dataset

In the Iris Dataset, there are 3 classes of data points: *Setosa*, *Versicolor*, and *Virginica*.

Occurrence of Misclassification: *Setosa* VS *Versicolor*

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50
dim=2	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	-	-
dim=3	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	-	-

Occurrence of Misclassification: *Setosa* VS *Virginica*

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50
dim=2	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	-	-
dim=3	0/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50	-	-

Occurrence of Misclassification: *Versicolor* VS *Virginica*

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	4/50	3/50	2/50	0/50	3/50	1/50	2/50	1/50	2/50	1/50
dim=2	2/50	1/50	2/50	1/50	2/50	1/50	2/50	1/50	-	-
dim=3	2/50	1/50	2/50	1/50	2/50	1/50	2/50	1/50	-	-

Occurrence of Misclassification: Multi-SLDA VS LDA

	SKL-3 (Multi-SLDA)			LDA		
	Type I	Type II	Type III	Type I	Type II	Type III
dim=1	0/50	2/50	2/50	0/50	2/50	1/50
dim=2	0/50	3/50	0/50	0/50	2/50	1/50
dim=3	0/50	2/50	1/50	-	-	-

2.1.2 Banknote Authentication Dataset

Occurrence of Misclassification

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	38/762	16/610	20/762	0/610	20/762	0/610	23/762	1/610	32/762	0/610
dim=2	23/762	0/610	20/762	0/610	23/762	0/610	23/762	0/610	-	-
dim=3	23/762	0/610	23/762	0/610	23/762	0/610	23/762	0/610	-	-

2.1.3 Climate Model Simulation Crashes Authentication Dataset

Occurrence of Misclassification

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	2/46	82/494	2/46	64/494	2/46	79/494	2/46	65/494	18/46	1/494
dim=2	2/46	59/494	1/46	59/494	2/46	54/494	1/46	51/494	-	-
dim=3	2/46	57/494	1/46	47/494	1/46	37/494	1/46	48/494	-	-

2.1.4 Seeds Dataset

In this dataset, there are 3 classes of different seeds.

Occurrence of Misclassification: Class 1 VS Class 2

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	1/70	7/70	2/70	4/70	2/70	4/70	1/70	0/70	2/70	6/70
dim=2	2/70	4/70	2/70	4/70	2/70	4/70	1/70	0/70	-	-
dim=3	2/70	1/70	2/70	2/70	2/70	2/70	1/70	1/70	-	-

Occurrence of Misclassification: Class 1 VS Class 3

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	5/70	4/70	10/70	1/70	10/70	1/70	2/70	2/70	5/70	5/70
dim=2	7/70	3/70	3/70	1/70	7/70	3/70	3/70	1/70	-	-
dim=3	3/70	1/70	4/70	1/70	4/70	1/70	3/70	1/70	-	-

Occurrence of Misclassification: Class 2 VS Class 3

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	0/70	0/70	0/70	0/70	0/70	0/70	0/70	0/70	0/70	0/70
dim=2	0/70	0/70	0/70	0/70	0/70	0/70	0/70	0/70	-	-
dim=3	0/70	0/70	0/70	0/70	0/70	0/70	0/70	0/70	-	-

Occurrence of Misclassification: Multi-SLDA VS LDA

	SKL-3 (Multi-SLDA)			LDA		
	Type I	Type II	Type III	Type I	Type II	Type III
dim=1	17/70	9/70	1/70	4/70	6/70	3/70
dim=2	16/70	7/70	4/70	4/70	6/70	3/70
dim=3	7/70	1/70	2/70	-	-	-

2.1.5 MAGIC Gamma Telescope Dataset

Occurrence of Misclassification

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	510/ 12332	4338/ 6688	509/ 12332	4330/ 6688	511/ 12332	4336/ 6688	510/ 12332	4330/ 6688	1163/ 12332	2963/ 6688
dim=2	599/ 12332	3777/ 6688	585/ 12332	3632/ 6688	599/ 12332	3760/ 6688	585/ 12332	3659/ 6688	-	-
dim=3	610/ 12332	3892/ 6688	912/ 12332	2922/ 6688	607/ 12332	3877/ 6688	888/ 12332	2877/ 6688	-	-

2.1.6 HTRU2 Dataset

Occurrence of Misclassification

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	232/ 16259	327/ 1639	1118/ 16259	161/ 1639	232/ 16259	327/ 1639	427/ 16259	154/ 1639	72/ 16259	381/ 1639
dim=2	297/ 16259	218/ 1639	1128/ 16259	142/ 1639	282/ 16259	216/ 1639	686/ 16259	138/ 1639	-	-
dim=3	319/ 16259	205/ 1639	424/ 16259	237/ 1639	335/ 16259	201/ 1639	345/ 16259	199/ 1639	-	-

2.1.7 Abalone Dataset

In this dataset, there are 3 classes of data points: 1-Male, 2-Female, and 3-Infant.

Occurrence of Misclassification: Male VS Female

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	1180/ 1528	213/ 1307	1176/ 1528	212/ 1307	1176/ 1528	212/ 1307	1178/ 1528	213/ 1307	407/ 1528	833/ 1307
dim=2	288/ 1528	1000/ 1307	288/ 1528	1001/ 1307	288/ 1528	1001/ 1307	289/ 1528	1000/ 1307	-	-
dim=3	320/ 1528	927/ 1307	343/ 1528	941/ 1307	320/ 1528	925/ 1307	320/ 1528	924/ 1307	-	-

Occurrence of Misclassification: Male VS Infant

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	210/ 1528	547/ 1342	626/ 1528	104/ 1342	622/ 1528	104/ 1342	567/ 1528	109/ 1342	363/ 1528	262/ 1342
dim=2	511/ 1528	152/ 1342	579/ 1528	125/ 1342	580/ 1528	127/ 1342	504/ 1528	147/ 1342	-	-
dim=3	502/ 1528	168/ 1342	549/ 1528	147/ 1342	502/ 1528	169/ 1342	496/ 1528	169/ 1342	-	-

Occurrence of Misclassification: Female VS Infant

	KL-1		KL-2		SKL		H		LDA	
	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II	Type I	Type II
dim=1	132/ 1307	567/ 1342	458/ 1307	104/ 1342	448/ 1307	104/ 1342	377/ 1307	128/ 1342	297/ 1307	206/ 1342
dim=2	325/ 1307	187/ 1342	476/ 1307	109/ 1342	332/ 1307	193/ 1342	312/ 1307	187/ 1342	-	-
dim=3	383/ 1307	145/ 1342	383/ 1307	145/ 1342	383/ 1307	144/ 1342	382/ 1307	149/ 1342	-	-

Occurrence of Misclassification: Multi-SLDA VS LDA

	SKL-3 (Multi-SLDA)			LDA		
	Type I	Type II	Type III	Type I	Type II	Type III
dim=1	1457/1528	501/1307	104/1342	716/1528	868/1307	319/1342
dim=2	1030/1528	877/1307	128/1342	716/1528	868/1307	319/1342
dim=3	1423/1528	422/1307	191/1342	-	-	-

2.2 Performance on the Testing Sets

2.2.1 Iris Dataset

(yet to come)

2.2.2 Banknote Authentication Dataset

Misclassification Percentage on Testing Sets (Number of Tests:100)

	KL-1	KL-2	SKL	H	LDA
dim=1	3.83%	1.38%	1.42%	1.77%	2.35%
dim=2	1.59%	1.36%	1.67%	1.58%	-
dim=3	1.62%	1.60%	1.59%	1.56%	-

2.2.3 Climate Model Simulation Crashes Authentication Dataset

Misclassification Percentage on Testing Sets (Number of Tests:100)

	KL-1	KL-2	SKL	H	LDA
dim=1	19.48%	13.13%	18.03%	12.64%	4.68%
dim=2	13.98%	11.25%	12.10%	11.60%	-
dim=3	11.15%	11.01%	10.67%	10.02%	-

2.2.4 Seeds Dataset

Misclassification Percentage on Testing Sets (Number of Tests:100)

	SKL-3	LDA
dim=2	12.35%	8.59%

2.2.5 MAGIC Gamma Telescope Dataset

Misclassification Percentage on Testing Sets (Number of Tests:100)

	KL-1	KL-2	SKL	H	LDA
dim=1	25.4%	25.4%	25.45%	25.42%	21.73%
dim=2	22.86%	22.12%	22.93%	22.27%	-
dim=3	23.6%	20.38%	22.37%	20.00%	-

2.2.6 HTRU2 Dataset

(yet to come)

2.2.7 Abalone Dataset

Misclassification Percentage on Testing Sets (Number of Tests:100)

	SKL-3	LDA
dim=2	48.74%	45.99%

2.3 Total number of Errors and Estimated P_{error}

2.3.1 Iris Dataset

(The results of *Setosa* VS *Versicolor* and *Setosa* VS *Virginica* are omitted because all algorithms are error-free in those situations.)

Total Number of Errors and Estimated P_{error}: *Versicolor* VS *Virginica*

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	7/100	0.14	2/100	0.04	4/100	0.08	3/100	0.06	3/100	0.06
dim=2	3/100	0.06	3/100	0.06	3/100	0.06	3/100	0.06	-	-
dim=3	3/100	0.06	3/100	0.06	3/100	0.06	3/100	0.06	-	-

Total Number of Errors and Estimated P_{error}: Multi-SLDA VS LDA

	SKL-3 (Multi-SLDA)		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	4/150	0.08	3/150	0.06

dim=2	3/150	0.06	3/150	0.06
dim=3	3/150	0.06	-	-

2.3.2 Banknote Authentication Dataset

Total Number of Errors and Estimated P_{error}

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	54/ 1372	0.076	20/ 1372	0.026	20/ 1372	0.026	24/ 1372	0.032	32/ 1372	0.042
dim=2	23/ 1372	0.030	20/ 1372	0.026	23/ 1372	0.030	23/ 1372	0.030	-	-
dim=3	23/ 1372	0.030	23/ 1372	0.030	23/ 1372	0.030	23/ 1372	0.030	-	-

2.3.3 Climate Model Simulation Crashes Authentication Dataset

Total Number of Errors and Estimated P_{error}

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	84/540	0.209	66/540	0.173	81/540	0.203	67/540	0.175	19/540	0.393
dim=2	61/540	0.163	60/540	0.141	56/540	0.153	52/540	0.125	-	-
dim=3	59/540	0.159	48/540	0.117	38/540	0.097	49/540	0.119	-	-

2.3.4 Seeds Dataset

(The results of Class 2 VS Class 3 are omitted because all algorithms are error-free in that situation.)

Total Number of Errors and Estimated P_{error}: Class 1 VS Class 2

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	8/140	0.114	6/140	0.086	6/140	0.086	1/140	0.014	8/140	0.114
dim=2	6/140	0.086	6/140	0.086	6/140	0.086	1/140	0.014	-	-
dim=3	3/140	0.043	4/140	0.057	4/140	0.057	2/140	0.029	-	-

Total Number of Errors and Estimated P_{error}: Class 1 VS Class 3

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	9/140	0.129	11/140	0.157	11/140	0.157	4/140	0.057	10/140	0.143
dim=2	10/140	0.143	4/140	0.057	10/140	0.143	4/140	0.057	-	-
dim=3	4/140	0.057	5/140	0.071	5/140	0.071	4/140	0.057	-	-

Total Number of Errors and Estimated P_{error}: Multi-SLDA VS LDA

	SKL-3 (Multi-SLDA)		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	27/210	0.386	13/210	0.186

dim=2	27/210	0.386	13/210	0.186
dim=3	10/210	0.143	-	-

2.3.5 MAGIC Gamma Telescope Dataset

Total Number of Errors and Estimated P_{error}

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	4848/19020	0.690	4839/19020	0.689	4847/19020	0.690	4840/19020	0.689	4126/19020	0.537
dim=2	4376/19020	0.613	4217/19020	0.591	4359/19020	0.611	4244/19020	0.595	-	-
dim=3	4502/19020	0.631	3834/19020	0.511	4484/19020	0.629	3765/19020	0.502	-	-

2.3.6 HTRU2 Dataset

Total Number of Errors and Estimated P_{error}

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	559/17898	0.214	1279/17898	0.167	559/17898	0.214	581/17898	0.120	453/17898	0.237
dim=2	515/17898	0.151	1270/17898	0.156	498/17898	0.149	824/17898	0.126	-	-
dim=3	524/17898	0.145	661/17898	0.171	536/17898	0.143	544/17898	0.143	-	-

2.3.7 Abalone Dataset

Total Number of Errors and Estimated P_{error}: Male VS Female

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	1393/2835	0.935	1388/2835	0.932	1388/2835	0.932	1391/2835	0.934	1240/2835	0.904
dim=2	1288/2835	0.954	1289/2835	0.954	1289/2835	0.954	1289/2835	0.954	-	-
dim=3	1247/2835	0.919	1284/2835	0.944	1245/2835	0.917	1244/2835	0.916	-	-

Total Number of Errors and Estimated P_{error}: Male VS Infant

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	757/2870	0.545	730/2870	0.487	726/2870	0.485	676/2870	0.457	625/2870	0.433
dim=2	663/2870	0.448	704/2870	0.472	707/2870	0.474	651/2870	0.439	-	-

dim=3	670/ 2870	0.454	696/ 2870	0.469	671/ 2870	0.454	665/ 2870	0.451	-	-

Total Number of Errors and Estimated P_{error}: Female VS Infant

	KL-1		KL-2		SKL		H		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	699/ 2649	0.523	562/ 2649	0.428	552/ 2649	0.420	505/ 2649	0.384	503/ 2649	0.381
dim=2	512/ 2649	0.388	585/ 2649	0.445	525/ 2649	0.398	499/ 2649	0.378	-	-
dim=3	528/ 2649	0.401	528/ 2649	0.401	527/ 2649	0.400	531/ 2649	0.403	-	-

Total Number of Errors and Estimated P_{error}: Multi-SLDA VS LDA

	SKL-3 (Multi-SLDA)		LDA	
	total err.	*est.* P_{err}	total err.	*est.* P_{err}
dim=1	2062/4177	1.414	1903/4177	1.370
dim=2	2035/4177	1.440	1903/4177	1.370
dim=3	2036/4177	1.398	-	-

3. Discussions

3.1 Decision Regions and Dimension

Fisher's LDA's decision regions are linear.

For statistical LDA, if we formulate the classification problem as the problem of hypothesis testing between the projected lower dimensional distributions, then its decision regions are usually not linear.

For example, LDA can't separate two distributions if $\mu_1 = \mu_2$. Statistical LDA, on the other hand, is able to separate two distributions with the same means $\mu_1 = \mu_2$, and the decision regions under that circumstances are likely to be circles, spheres, or their high dimensional counterparts.

For Fisher's LDA, the projected data are at most (k-1) dimensional (k is the number of classes).

For Statistical LDA, the dimension of projected data could be $1 \leq d \leq n$.

3.2 Issues with Unbalanced Datasets

We have learned that maximizing total variation is equivalent to minimizing P_{error}. Since TV is bounded by several *f*-divergences, we could expect that some versions of SLDA would be able to approximately maximize TV and approximately minimize P_{error}.

To some degree, to minimize P_{error} means to minimize the total number of errors when the prior distribution is $\mathrm{P}(\mathbb{P}) = \mathrm{P}(\mathbb{Q}) = 0.5$. From section 2.3.1 and section 2.3.4 we could see that at least some versions of SLDA could produce results that are better than LDA's results when the datasets are balanced.

On the other hand, from section 2.3.3 and section 2.3.6 we could see that SLDA produces better results in P_{error} while LDA produces better results in total number of errors when the datasets are not balanced. This might be counterintuitive yet reasonable.

At this time, I can't tell which method is better. Perhaps they both have their own suitable scenarios. For one thing, the information of prior distribution is valuable and we should make use of that in some way. For another, we can't totally rely on total number of errors to judge which method is better (for example, suppose there are 1000 "normal" data points and 1 "abnormal" data point, to minimize the total number of errors, some machine learning algorithms might simply predict that all data points are "normal", which is not a desirable outcome). Currently, my preliminary thought is that, SLDA might be able to produce "unbiased" results and alleviate the problem of overfitting to the datasets, especially when the training datasets are unbalanced. Meanwhile, it seems that SLDA doesn't consider much about the influence of prior distributions.

3.3 Numerical Implementation of SLDA

The results from experiments might potentially show some problems for numerical implementation of statistical LDA.

1) In theory, as the projected dimension *r* grows, the performance of SLDA should be improved (which means fewer number of errors). However, experiments show that sometimes the number of errors actually increase as *r* grows. This is a sign indicating that the true optimal solution for SLDA might be hard to compute and sometimes SLDA might have a high probability of converging to a suboptimal local minimum. This problem might become prominent when *r* is large.

2) Sometimes we could see that the results are "swinging" as the projected dimension *r* grows. For example, when r=2 the obtained solution shows large Type I error and small Type II error, while when r=3 the obtained solution shows small Type I error and large Type II error. I think this might be caused by algorithms' converging to different suboptimal local minimum points.