The reason Riemannian geometry based classifier works well in ECoG/EEG classification problems:

Common classifiers such as k-nearest neighbor uses arithmetic distance and mean. Riemannian classifier implements the same algorithms but uses Riemannian distance and mean. Riemannian distance and mean shares many properties with Log-Euclidean distance, which has many benefits over arithmetic distance.

Benefits of Log Euclidean distance:

$$d_G(a,b) = |\log a - \log b| = \left|\log \tfrac{a}{b}\right|$$

1. Log Euclidean distance is scale invariant. D(a,b) = D(xa, xb). When this generalize to matrices

   $$\delta_G\left(\boldsymbol{X}\boldsymbol{C}_1\boldsymbol{X}^T, \boldsymbol{X}\boldsymbol{C}_2\boldsymbol{X}^T\right) = \delta_G\left(\boldsymbol{C}_1, \boldsymbol{C}_2\right)$$

   In terms of Sampling in ECoG or EEG, it means it does not matter if the electrodes are placed in slightly different positions, the distance between two measured covariance matrices will be the same. (X is the mixing matrix what is determined by the positions of electrodes.)
   This give Log Euclidean based classifier good ability of transfer learning across session and across subjects.

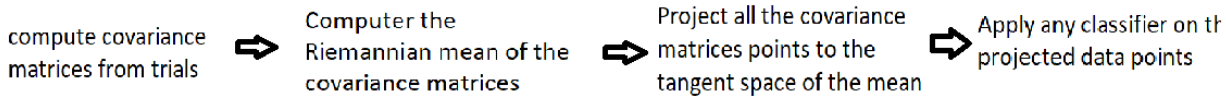
2. The arithmetic mean is only a good indicator of central tendency if the distribution is symmetrical (such as Gaussian), but not so good when the distribution is not symmetrical (such of Chi-squared). In this regard, Log Euclidean mean is more resistant to outlier in both Symmetrical or non-symmetrical distributions.

According to [1], Riemannian distance inherit both property from Log Euclidean distance. It is defined as

$$\delta_G\left(\boldsymbol{C}_1, \boldsymbol{C}_2\right) = \left\|\mathrm{Log}\left(\boldsymbol{C}_1^{-½}\boldsymbol{C}_2\boldsymbol{C}_1^{-½}\right)\right\|_F = \sqrt{\sum_{n=1}^{\mathrm{N}} \log^2 \lambda_n}\,,$$

Tangent Space classifier goes a step further. After finding the Riemannian mean of the covariance matrix points, it projects the data points onto the tangent space of the mean. According to [1], “Tangent space mapping is a local projection that maps the elements of the manifold into a Euclidean space, keeping their distance relationship intact.” This allows us to use regular classifier such as SVM, logistic regression, etc which are defined in arithmetic distance.

The workflow of the Tangent space classifier is:



By default, the tangent space classifier uses logistic regression on the last step. Part of the novelty in my project is applying SVM and ensemble classifiers to replace the logistic classifier.

My Novelty in this project:

1. Replace default logistic regression with SVM. Since overfitting is observed during training SVM. A bagging classifier with 21 fine tuned SVM base classifiers was created.

2. Two-stage classifier: Cross validation with 50 iterations was performed, and "bad" epochs that were repeatedly classified wrong were picked out. Another classifier was trained to distinguish "good" epochs and "bad" epochs. This classifier is placed before the tangent space classifier. If the epoch is classified to be "bad", it will be thrown away and not be passed to further classification.

Results and further explanation

Novelty 1: Bagging with SVM as base classifier

SVM with radial basis kernel was used instead of logistic regression. The parameters were fine-tuned with grid search (gamma= 0.03, C=100).

SVM displays overfitting during cross validation. To reduce overfitting, bagging was used to even out the bias (21 base SVM classifier, each learns a random subset of 2/3 of the whole data)

	Tangent Space + logistic regression (default)	Tangent Space + SVM	Bagging with SVM as base classifier
	71.7%	**78.5%**	**79.5%**

Novelty 2: Two-stage classifier

The dataset was divided into three splits: training, testing, validation. After 50 iterations of cross validation on training and testing set, epochs that were repeatedly classified wrong were picked out. Another classifier was built to classify "good" epochs and "bad" epochs. Tangent space classifier with logistic regression was used for this classifier. The reason is

that good epochs are a lot more than bad ones. SVM is more susceptible to imbalanced dataset than logistic regression, thus logistic classifier was used.

The motivation for this two-stage design is that many BCI applications do not care if part of the data is discarded. If a person is controlling a wheelchair with EEG, he/she could simply try again if the first epoch is classified as invalid. This is more useful than misclassify the intention of the user. In fact, when the sampling rate is high, the user might not realize multiple classifications have been made.

Intuitively, if a Riemannian-geometry based classifiers can differentiate covariance matrices that represent different user intentions, they should also be capable of detecting outliers and invalid covariance matrices against the good ones. A threshold on predict_proba can be adjusted to increase accuracy at the cost of decreasing proportion of valid epochs. The two stag classifier was tested using the validation set (never used in training and testing).

Threshold	1.0	0.85	0.8	0.7	0.6	0.4
Valid epochs	817/817	207/817	169/817	119/817	90/817	56/817
Accuracy	73.9%	**80.7%**	**84.02%**	**88.2%**	**91.1%**	**91.1%**

As shown in the data, if we are allowed to retain only 25% of the data, the accuracy can be boosted to 80.7%. If we are allowed to retain 15% of the data, the accuracy can be boosted to 88.2%.