Bayes Classifier

Yusen Lin *116765664*

Implement details

Problem: When calculating the det of the covariance, the det always is either 0 or inf because of the dimension size of covariance is very large. Therefore, to solve this problem, I assume the log(det(covariance)) is the same in each class.

Code for calculating the posterior and predicting the class

```
def predict(self, x):
    log_posterior_list = []
    for _class in self.__classes:
        params = self.__param_list[_class]
        log_probability_w_i = params['log_probability_w_i']

        x = x.reshape((-1, 1))
        inv_cov = params['inv_cov']
        mean_inv_cov = params['mean_inv_cov']
        mean_inv_cov_mean_05 = params['mean_inv_cov_mean_05']

        # since the det of cov is either 0 or inf, I assume the "log(det)" is the same in each class
        det = 0
        log_likelihood = -0.5 * mat(x.transpose(), inv_cov, x) + \
                         np.matmul(mean_inv_cov, x) - \
                         mean_inv_cov_mean_05

        log_posterior = log_likelihood + log_probability_w_i

        log_posterior_list.append([_class, log_posterior])

    log_posterior_list.sort(key=lambda x: -x[1])
    prediction_class = log_posterior_list[0][0]

    return prediction_class
```

Code for getting the parameter of each class

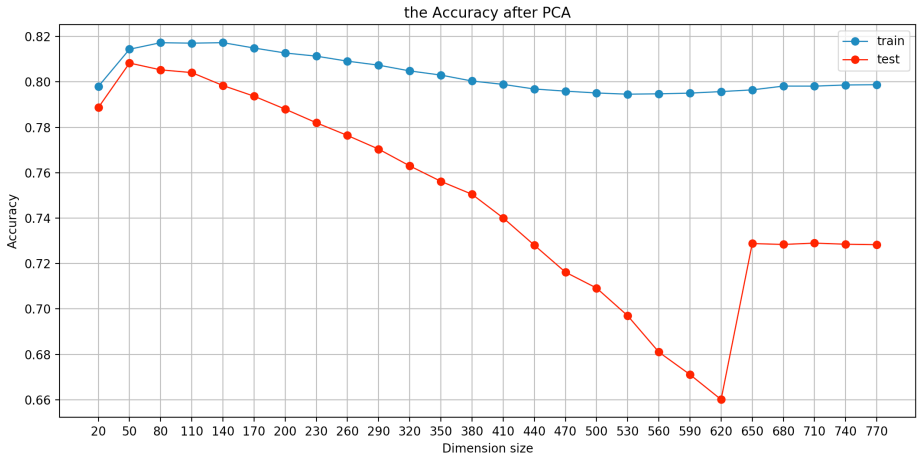
```
def __evaluate_params(self):
    epision = 0.001

    for _class in self.__classes:
        data = self.__x_train[self.__y_train == _class]

        prob_w = float(data.shape[0]) / self.__x_train.shape[0]

        mean = np.mean(data, axis=0)
        cov = np.cov(data.transpose())

        # regularization
        cov = cov + epision * np.identity(cov.shape[0], dtype=np.float32)
        det = np.linalg.det(cov)
        inv_cov = np.linalg.inv(cov)

        # pre calculate some variables
        m = mean.reshape((-1, 1))
        mean_inv_cov = np.matmul(m.transpose(), inv_cov)
        mean_inv_cov_mean_05 = np.matmul(mean_inv_cov, m) * 0.5

        self.__param_list.append({
            'class': _class,
            'log_probability_w_i': np.log(prob_w),
            'det': det,
            'inv_cov': inv_cov,
            'mean_inv_cov': mean_inv_cov,
            'mean_inv_cov_mean_05': mean_inv_cov_mean_05,
        })
```

Data

number of training data: 60000
number of test data: 10000

Experiments and performance

Original data														
Dataset	Dim	accuracy	Global f1	F1 of class 0	F1 of class 1	F1 of class 2	F1 of class 3	F1 of class 4	F1 of class 5	F1 of class 6	F1 of class 7	F1 of class 8	F1 of class 9	Use time (seconds)
Train	784	0.7987	0.7987	0.855102	0.875457	0.682124	0.797809	0.725338	0.822585	0.636013	0.770833	0.92094	0.924621	344.000000
Test	784	0.7277	0.7277	0.76335	0.843262	0.517066	0.744643	0.600985	0.793305	0.511371	0.749077	0.866219	0.914672	53.000000
After PCA														
Dataset	Dim	accuracy	Global f1	F1 of class 0	F1 of class 1	F1 of class 2	F1 of class 3	F1 of class 4	F1 of class 5	F1 of class 6	F1 of class 7	F1 of class 8	F1 of class 9	Use time (seconds)
Train	20	0.798	0.798	0.779022	0.943496	0.74023	0.817555	0.717426	0.827101	0.574996	0.731636	0.923918	0.911215	8.000000
Test	20	0.7887	0.7887	0.769539	0.93617	0.727001	0.813166	0.70632	0.825784	0.534795	0.74923	0.911005	0.908119	1.000000
Train	50	0.814383	0.814383	0.803636	0.953261	0.780529	0.839926	0.751135	0.812197	0.619346	0.711205	0.929506	0.92191	9.000000
Test	50	0.8083	0.8083	0.795785	0.95288	0.775388	0.829151	0.741573	0.81193	0.59932	0.717172	0.91627	0.925926	1.00000
Train	80	0.817267	0.817267	0.80844	0.948392	0.797215	0.841148	0.762262	0.801932	0.644343	0.694382	0.928809	0.92433	10.000000
Test	80	0.8053	0.8053	0.800202	0.944034	0.772567	0.825714	0.731707	0.806688	0.603629	0.703775	0.921178	0.925795	1.00000
Train	110	0.81705	0.81705	0.814821	0.94422	0.8008	0.841047	0.760325	0.795553	0.65873	0.684536	0.929711	0.923259	22.000000
Test	110	0.8041	0.8041	0.796087	0.937931	0.777261	0.822465	0.732143	0.803086	0.620783	0.700965	0.913105	0.927595	3.000000
Train	140	0.817283	0.817283	0.824251	0.941706	0.798998	0.843419	0.76085	0.792037	0.666469	0.674179	0.932574	0.924092	27.000000
Test	140	0.7984	0.7984	0.788035	0.93617	0.773504	0.814885	0.735441	0.794389	0.612084	0.686684	0.914634	0.923233	4.000000
Train	170	0.814917	0.814917	0.825047	0.937959	0.796718	0.841513	0.758259	0.787255	0.667967	0.66718	0.930335	0.925023	32.000000
Test	170	0.7937	0.7937	0.790625	0.929336	0.760594	0.809636	0.735376	0.786963	0.613095	0.68152	0.908836	0.922051	5.000000
Train	200	0.812733	0.812733	0.830868	0.932764	0.7892	0.838298	0.757386	0.782689	0.671761	0.657871	0.931135	0.924204	37.000000
Test	200	0.788	0.788	0.793067	0.921833	0.747029	0.805256	0.727375	0.777953	0.612211	0.663564	0.906977	0.926002	6.000000
Train	230	0.811333	0.811333	0.834187	0.92771	0.786768	0.835296	0.755527	0.780934	0.671615	0.656819	0.933892	0.922853	44.000000
Test	230	0.782	0.782	0.791337	0.914813	0.737629	0.802607	0.715922	0.774471	0.602082	0.650538	0.909515	0.924025	7.000000
Train	260	0.809133	0.809133	0.839424	0.924346	0.783149	0.832432	0.753893	0.775489	0.67283	0.648703	0.934526	0.920586	51.000000
Test	260	0.7764	0.7764	0.786589	0.911268	0.728555	0.799258	0.710195	0.767198	0.596863	0.634942	0.90943	0.923314	8.000000
Train	290	0.807367	0.807367	0.84043	0.918919	0.780417	0.830537	0.752987	0.773125	0.671949	0.642383	0.936167	0.921192	60.000000
Test	290	0.7704	0.7704	0.788009	0.903509	0.710495	0.792801	0.704729	0.762273	0.590701	0.626968	0.909771	0.921569	9.000000
Train	320	0.804833	0.804833	0.843354	0.914027	0.77306	0.825604	0.749625	0.770572	0.672	0.636559	0.93687	0.921639	68.000000
Test	320	0.763	0.763	0.787325	0.896856	0.695546	0.791724	0.695449	0.751622	0.585768	0.59467	0.909515	0.920455	11.000000
Train	350	0.803	0.803	0.84549	0.908098	0.769307	0.822909	0.748333	0.769091	0.668926	0.633152	0.939819	0.921876	78.000000
Test	350	0.7562	0.7562	0.784441	0.888271	0.68483	0.784799	0.684272	0.744045	0.581046	0.571021	0.913696	0.921488	13.000000
Train	380	0.800383	0.800383	0.846283	0.902205	0.759461	0.818736	0.746398	0.767943	0.667181	0.63146	0.939978	0.921994	88.000000
Test	380	0.7505	0.7505	0.787223	0.882057	0.662864	0.782332	0.679623	0.73882	0.57555	0.555155	0.912494	0.919338	15.000000
Train	410	0.7989	0.7989	0.848517	0.896856	0.753804	0.815332	0.74343	0.768481	0.664587	0.635541	0.940605	0.922178	104.000000
Test	410	0.7401	0.7401	0.781522	0.871968	0.649936	0.774399	0.667063	0.724926	0.567901	0.50597	0.914634	0.920684	16.000000
Train	440	0.796867	0.796867	0.850426	0.891557	0.745521	0.813105	0.741682	0.767902	0.662129	0.633009	0.940428	0.923169	108.000000
Test	440	0.7281	0.7281	0.77942	0.863636	0.64244	0.76638	0.655875	0.703677	0.565066	0.424862	0.907649	0.921162	17.000000
Train	470	0.795933	0.795933	0.850262	0.886621	0.738215	0.81066	0.741893	0.770494	0.659586	0.63918	0.940334	0.92457	124.000000
Test	470	0.7162	0.7162	0.781421	0.850575	0.626649	0.756076	0.655502	0.685413	0.556552	0.334443	0.90689	0.921558	20.00000
Train	500	0.795117	0.795117	0.852701	0.881432	0.729984	0.808075	0.740663	0.77334	0.657035	0.647191	0.940129	0.924979	144.000000
Test	500	0.7092	0.7092	0.782369	0.841251	0.603239	0.750223	0.645858	0.681104	0.554086	0.305085	0.903676	0.920124	23.000000
Train	530	0.794583	0.794583	0.853281	0.876089	0.723151	0.805169	0.738377	0.77769	0.654172	0.661205	0.939849	0.925178	155.000000
Test	530	0.697	0.697	0.779249	0.826291	0.594154	0.741135	0.638118	0.66507	0.545516	0.206278	0.900232	0.921721	25.000000
Train	560	0.79475	0.79475	0.854652	0.87271	0.716405	0.802493	0.736772	0.783586	0.651237	0.677666	0.938665	0.925509	164.000000
Test	560	0.681	0.681	0.778889	0.789346	0.57538	0.72285	0.62871	0.649916	0.538716	0.095238	0.896488	0.919087	27.000000
Train	590	0.79505	0.79505	0.85457	0.872074	0.707503	0.801475	0.735084	0.790579	0.647244	0.693917	0.937371	0.926511	186.000000
Test	590	0.6711	0.6711	0.775056	0.748436	0.568436	0.704129	0.62363	0.64639	0.530933	0.076923	0.892775	0.918357	30.000000
Train	620	0.795733	0.795733	0.854745	0.873451	0.698664	0.800904	0.733772	0.796264	0.644994	0.709753	0.9375	0.925983	201.000000
Test	620	0.6601	0.6601	0.771125	0.704663	0.552799	0.6875	0.616322	0.640293	0.526794	0.043053	0.887466	0.917231	33.000000
Train	650	0.796483	0.796483	0.854007	0.874085	0.692184	0.799819	0.732005	0.803338	0.640884	0.72794	0.93554	0.926854	235.000000
Test	650	0.7288	0.7288	0.768886	0.842593	0.545325	0.743705	0.613317	0.775709	0.522838	0.704676	0.884826	0.917058	35.000000
Train	680	0.79815	0.79815	0.855	0.874719	0.686548	0.798823	0.731605	0.811829	0.639952	0.747885	0.932082	0.926708	248.000000
Test	680	0.7284	0.7284	0.764474	0.843262	0.533141	0.743268	0.611145	0.781506	0.518877	0.721767	0.881202	0.91658	44.00000
Train	710	0.798133	0.798133	0.854159	0.874719	0.680417	0.798345	0.728331	0.817896	0.636784	0.761167	0.927634	0.925929	197.000000
Test	710	0.729	0.729	0.762817	0.843262	0.522496	0.744852	0.609069	0.789517	0.515152	0.742255	0.875226	0.916667	27.000000
Train	740	0.798633	0.798633	0.855053	0.875351	0.679704	0.798586	0.728539	0.820693	0.636298	0.768045	0.92345	0.924807	188.000000
Test	740	0.7285	0.7285	0.762549	0.843262	0.518519	0.744415	0.604537	0.792469	0.513269	0.748308	0.871403	0.915237	29.000000
Train	770	0.798767	0.798767	0.854591	0.875457	0.683054	0.797657	0.725757	0.822163	0.636566	0.77046	0.921371	0.924858	204.000000
Test	770	0.7283	0.7283	0.762655	0.843262	0.5235	0.744082	0.600123	0.793305	0.513338	0.749077	0.866726	0.914672	32.000000
After LDA														
Dataset	Dim	accuracy	Global f1	F1 of class 0	F1 of class 1	F1 of class 2	F1 of class 3	F1 of class 4	F1 of class 5	F1 of class 6	F1 of class 7	F1 of class 8	F1 of class 9	Use time (seconds)
Train	9	0.81745	0.81745	0.791766	0.966306	0.695855	0.840801	0.728093	0.878941	0.594133	0.85049	0.923867	0.918888	7.000000
Test	9	0.8046	0.8046	0.780639	0.966976	0.675189	0.823586	0.70463	0.872933	0.563929	0.861673	0.906667	0.907206	1.000000



Conclusion

1. The PCA can greatly reduce the consumed time without hurting the performance. The performance after PCA is basically the same with the performance before PCA. When the dimension size reaches to 50, the model could reach the best performance.

2. Not only can LDA reduce the consumed time, but also can improve the performance. Because the LDA could do better in capturing the features.

3. From the f1 score of each classes, the model performs poorly on class 0, 2, 4, 6. The performance of those four classes need to further improve.

4. When the dimension size decreases, the model's ability of generalization increases. The simplest the model is, the stronger the model's generalization ability will be.

5. When the dimension size of the data reaches to 50 using PCA for dimension reduction, the model have the best performance with test set accuracy of 0.8083.

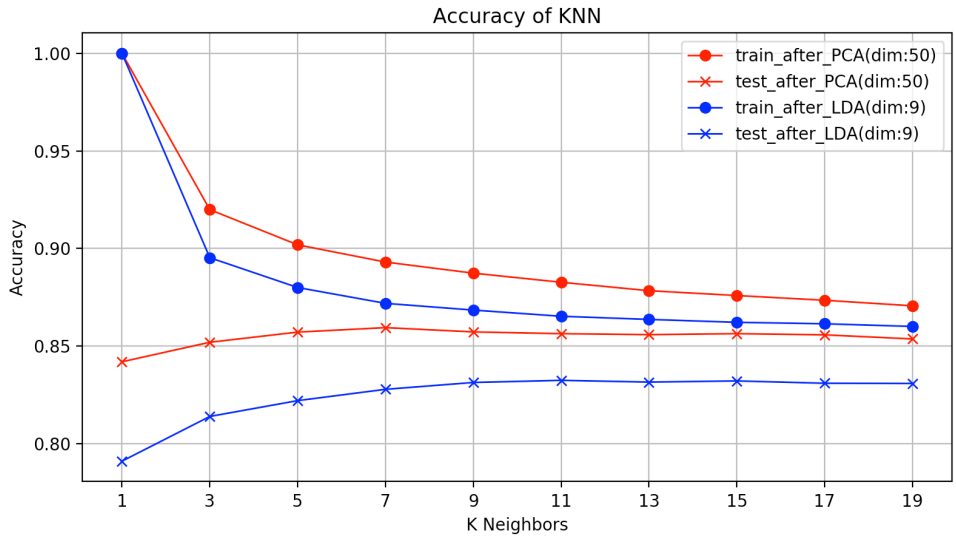
6. PCA and LDA could capture the most significant features and reduce the dimension. They can even filter some noise so that the model could have better performance.

K Nearest Neighbors

Experiments and performance

Original data															
Dataset	K	Dim	accuracy	Global f1	F1 of class 0	F1 of class 1	F1 of class 2	F1 of class 3	F1 of class 4	F1 of class 5	F1 of class 6	F1 of class 7	F1 of class 8	F1 of class 9	use time(seconds)
Train	5	784	0.899767	0.899767	0.87019	0.987412	0.835072	0.919105	0.844922	0.936302	0.738215	0.938264	0.973538	0.948556	5531
Test	5	784	0.8554	0.8554	0.810811	0.977284	0.77155	0.881148	0.780414	0.899344	0.613661	0.917861	0.9636	0.930769	886
After PCA															
Dataset	K	Dim	accuracy	Global f1	F1 of class 0	F1 of class 1	F1 of class 2	F1 of class 3	F1 of class 4	F1 of class 5	F1 of class 6	F1 of class 7	F1 of class 8	F1 of class 9	use time(seconds)
Train	1	50	1	1	1	1	1	1	1	1	1	1	1	1	2
Test	1	50	0.842	0.842	0.773453	0.976346	0.736313	0.871611	0.725549	0.934997	0.602713	0.913532	0.968844	0.926781	7
Train	3	50	0.919933	0.919933	0.886026	0.988875	0.858598	0.931514	0.869946	0.965751	0.796327	0.955486	0.981578	0.961931	47
Test	3	50	0.852	0.852	0.788799	0.974307	0.740203	0.88764	0.759288	0.931877	0.618428	0.919078	0.966349	0.932485	8
Train	5	50	0.901917	0.901917	0.870258	0.985894	0.837524	0.918347	0.840682	0.952814	0.742218	0.941709	0.974157	0.9491	54
Test	5	50	0.8572	0.8572	0.800962	0.974255	0.762327	0.879433	0.770059	0.938118	0.62018	0.923152	0.96233	0.935879	10
Train	7	50	0.893083	0.893083	0.861607	0.983753	0.823558	0.911846	0.826654	0.947014	0.717697	0.93638	0.970704	0.94434	66

Train	9	50	0.8874	0.8874	0.856869	0.981527	0.814097	0.908198	0.819581	0.942266	0.701974	0.932412	0.969217	0.940364	65
Test	9	50	0.8573	0.8573	0.800776	0.976673	0.766895	0.885974	0.769231	0.931161	0.617492	0.921951	0.960159	0.936709	10
Train	11	50	0.882733	0.882733	0.852717	0.979998	0.807578	0.903307	0.810806	0.938113	0.69208	0.928607	0.968038	0.938063	63
Test	11	50	0.8564	0.8564	0.809547	0.97561	0.765748	0.881937	0.765832	0.92891	0.621522	0.916342	0.957341	0.935673	11
Train	13	50	0.878417	0.878417	0.846674	0.979395	0.803564	0.900493	0.808099	0.934111	0.679782	0.924239	0.96527	0.934607	65
Test	13	50	0.8559	0.8559	0.814672	0.975585	0.76514	0.881699	0.769457	0.926572	0.616525	0.91606	0.955357	0.930799	11
Train	15	50	0.875933	0.875933	0.844292	0.978356	0.797817	0.899633	0.806082	0.931998	0.675292	0.922003	0.963612	0.932406	65
Test	15	50	0.8564	0.8564	0.812772	0.975635	0.767937	0.882056	0.772189	0.924788	0.622642	0.916342	0.953373	0.930165	11
Train	17	50	0.873517	0.873517	0.842038	0.977767	0.792537	0.898447	0.799708	0.930857	0.668921	0.92213	0.962963	0.932255	68
Test	17	50	0.8558	0.8558	0.814922	0.97561	0.768245	0.8833	0.766125	0.925515	0.6196	0.915651	0.9529	0.929195	12
Train	19	50	0.870667	0.870667	0.838905	0.977413	0.789408	0.89533	0.796404	0.929745	0.658888	0.919757	0.962074	0.930762	72
Test	19	50	0.8537	0.8537	0.813886	0.97561	0.7636	0.88262	0.762654	0.92324	0.616359	0.914146	0.949876	0.927915	12
After LDA															
Dataset	K	Dim	accuracy	Global f1	F1 of class 0	F1 of class 1	F1 of class 2	F1 of class 3	F1 of class 4	F1 of class 5	F1 of class 6	F1 of class 7	F1 of class 8	F1 of class 9	use time(seconds)
Train	1	9	1	1	1	1	1	1	1	1	1	1	1	1	1
Test	1	9	0.7911	0.7911	0.743337	0.964447	0.66537	0.786551	0.652196	0.882322	0.530902	0.86103	0.923462	0.901339	1
Train	3	9	0.895317	0.895317	0.856874	0.979922	0.823379	0.903204	0.839057	0.940287	0.751371	0.931702	0.966409	0.953529	8
Test	3	9	0.814	0.814	0.75477	0.967742	0.697761	0.817452	0.679421	0.908444	0.556541	0.88845	0.939638	0.914968	1
Train	5	9	0.88005	0.88005	0.848784	0.978335	0.805441	0.888815	0.807651	0.929523	0.709582	0.917976	0.960696	0.943026	7
Test	5	9	0.8221	0.8221	0.782112	0.970157	0.711485	0.827586	0.699244	0.90425	0.559091	0.892683	0.934935	0.918622	1
Train	7	9	0.871933	0.871933	0.83856	0.978164	0.790059	0.881532	0.794935	0.9257	0.68906	0.914351	0.955426	0.940141	8
Test	7	9	0.8279	0.8279	0.78492	0.969697	0.719962	0.839154	0.718154	0.906186	0.567148	0.896282	0.938592	0.921326	1
Train	9	9	0.868483	0.868483	0.834055	0.978374	0.784915	0.877069	0.794084	0.921825	0.677107	0.911471	0.954105	0.939308	8
Test	9	9	0.8314	0.8314	0.793204	0.97166	0.729948	0.842209	0.731514	0.908058	0.568245	0.89561	0.936571	0.918114	1
Train	11	9	0.865317	0.865317	0.831117	0.977692	0.78019	0.875155	0.79097	0.919765	0.666297	0.908364	0.952008	0.937832	9
Test	11	9	0.8325	0.8325	0.794574	0.972124	0.73222	0.845472	0.728981	0.906363	0.57429	0.894839	0.936508	0.921549	1
Train	13	9	0.8637	0.8637	0.830767	0.976931	0.777995	0.874434	0.787156	0.918597	0.661924	0.906795	0.951153	0.937204	9
Test	13	9	0.8316	0.8316	0.794374	0.972152	0.728324	0.847374	0.731278	0.905426	0.568736	0.895479	0.936571	0.918489	1
Train	15	9	0.862217	0.862217	0.830367	0.976916	0.773761	0.872103	0.783664	0.918983	0.657466	0.907433	0.950591	0.936237	10
Test	15	9	0.8322	0.8322	0.799611	0.970096	0.721689	0.846802	0.734079	0.906653	0.576063	0.895813	0.93617	0.916957	1
Train	17	9	0.861467	0.861467	0.829789	0.976834	0.772556	0.871066	0.784377	0.91737	0.655224	0.906232	0.95049	0.935505	10
Test	17	9	0.831	0.831	0.794773	0.970558	0.721452	0.844973	0.72878	0.90881	0.574803	0.894839	0.935116	0.917413	1
Train	19	9	0.8601	0.8601	0.827935	0.976248	0.771496	0.871238	0.783289	0.915985	0.65146	0.903357	0.950561	0.934165	10
Test	19	9	0.8309	0.8309	0.794231	0.971051	0.723261	0.842885	0.731612	0.904491	0.573696	0.893969	0.935707	0.917413	1



Conclusion

1. The PCA and LDA could greatly reduce the dimension size and consumed time with only slightly or even no hurting performance.
2. When the K neighbors is between 7 to 11, the model could reach the best performance (for test set). After that, the model's performance (for test set) would stay steady even though the K keeps increasing.

3. When K = 1, the train set accuracy should be ignore. Because this result is meaningless, the output will always be the same with the class of the input.
4. When K increases from 1, the train set accuracy will keep decreasing while the test set accuracy will keep increasing. Until K reach 7 to 11, both the train set and the test set accuracy will stay steady.
5. When the dimension size of the data is large, the consumed time will become incredibly long.