Clasificación con Procesos Gaussianos

Jesús Sánchez de Castro

8 Junio 2018

1 Introducción

En este trabajo se emplean los procesos gausianos para construir un clasificador que sea capaz de distinguir entre muestras de tejido cancerígeno de muestras de tejido sano. Para entrenar el modelo se emplean dataset resultantes de la aplicación de LBP uniforme invariante a la rotación, pasando de tener imágenes de 2048x2048 a histogramas como características extraídas de las imágenes.

2 ¿Qué es un proceso gausiano?

Un proceso gaussiano es un proceso estocástico que podemos emplear como método de aprendizaje supervisado para resolver problemas de regresión y clasificación probabilística. Se trata de un método en el que si disponemos de N muestras de un conjunto etiquetadas y se nos da una nueva observación sin etiqueta, tendremos una distribución gaussiana multidimensional de N+1 dimensiones, pudiendo obtener asi la distribución de la muestra desconocida condicionada a las demás. Cuanto mayor sea la covarianza entre las variables de las instancias, mayor información aportarán las muestras etiquetadas y mejor podremos estimar la distribución de la muestra desconocida.

Para un proceso gausiano debemos definir una función de medias, que en este problema ignoramos con media cero y un kernel de covarianza que definirá el proceso gausiano. En este caso se emplean dos kernels, el kernel lineal y el kernel gaussiano. Los GPs al realizar predicciones probabilísticas, podemos calcular intervalos de confianza empíricos y decidir en base a ellos si es necesario volver a entrenar el modelo centrándose en algunas regiones de interés. Por otro lado es versátil gracias a los distintos kernels que pueden utilizarse, existen en las diferentes librerías kernels comunes pero también pueden especificarse kernels personalizados.

Cuando comparamos los procesos gaussianos con otro algoritmo de aprendizaje automático como SVM podemos encontrar una serie de diferencias que se listan a continuación:

- Cuando empleamos GPs, los hiperparámetros de los kernels pueden aprenderse de los datos mientras que los hiperparámetros de SVM no.
- Los resultados de un GP son predicciones puramente probabilísticas, dándonos una estimación de la incertidumbre de la predicción, lo que puede ser útil en muchos casos.

3 Software y experimentación

Para la realización de ésta práctica se ha empleado el lenguaje de programación Python y la librería de procesos gaussianos GPy. Junto a GPy, otras librerías importantes de machine learning o cientificas son sklearn, numpy y pandas. Sklearn proporciona un paquete de preprocesamiento que se ha empleado para normalizar los datos, numpy nos permite utilizar array n-dimensionales con facilidad y un gran conjunto de utilidades que junto a pandas nos permiten manipular y cargar conjuntos de datos. Para la experimentación se han empleado dos kernels, el lineal y el gaussiano, cuya implementaciones se encuentran en GPy. Después, tras ver por encima los valores que tomaban las probabilidades de las variables, se ha ido ajustando el threshold desde 0.5 a 0.25 para ver cual obtenía mejores resultados. En cuanto a parámetros relevantes para el problema, el kernel gaussiano tiene parámetros [variance = 1.0, lenghtscale=1.9], sugeridos por el profesor en la memoria de la práctica.

GPy es un paquete que ofrece una gran cantidad de herramientas para trabajar con procesos gaussianos. Se ha empleando la función 'GPClassification()' para obtener el modelo que nos permite predecir y obtener la probabilidad de cada instancia de test de pertenecer a la clase sano. Por otro lado GPy cuenta con otros paquetes de kernels y funcionalidades de donde hemos obtenido el kernel gaussiano de 'GPy.kern.RBG()' y lineral de 'GPy.kern.Linear()'. Además ofrece una función para calcular la matriz de confusión 'GPy.util.classification.conf_matrix' pasando como parámetro las prabilidades promediadas de los folds y los datos de test para posteriormente calcular el resto de métricas. Por otro lado, se ha utilizado Scikit-learn para obtener la métrica AUC como 'roc_auc_score()' y para la normalización de los datos con 'preprocessing.scale()'. El resto de detalles se pueden encontrar en el código en:
https://github.com/Yussoft/GaussianProcessClassification_Python

4 Análisis de los resultados

Como se comentó anteriormente, los experimentos se han hecho con diferentes thresholds para comprobar que el clasificador iba a clasificar mejor las muestras con un valor de threshold cercano a 0.3 tras haber echado un simple vistazo a los valores de proabilidad que daban los modelos de clasificar como tejido sano. Como se pensó en un principio, se refleja en la medida Accuracy que conforme

más bajo es el threshold mayor número de muestras bien clasificadas obtenemos. Los dos mejores resultados en Accuracy son con un threshold de 0.25 tanto para el kernel gaussiano como para el kernel lineal, siendo el tercer fold en ambos kernels el que alcanza 0.9035 para kernel gaussiano y 0.9112 para el lineal.

Por otro lado, debemos también fijarnos en otras medidas de interés para un problema de salud como el que tratamos. Por un lado, el AUC o área bajo la curva ROC nos permite ver cómo de bien clasifica nuestro modelo sin clasificar como sano a un enfermo. Esto se debe a que en el eje Y tenemos la sensibilidad, que nos permite saber cuantos ejemplos sanos hemos clasificado correctamente como sanos y en el eje X, 1-especificidad, es decir, cuántos ejemplos enfermos hemos clasificado como sanos erróneamente. Cuando realizamos un gráfico de la curva ROC con estos ejes, obtenemos una curva cuya área será mayor cuando mayor sea la sensibilidad, por lo tanto, diciendo que nuestro modelo obtiene mejores resultados cuando mayor sea éste área. Como podemos observar en la tabla, el valor de AUC es bastante alto en general, siendo los valores más pequeños alrededor de 0.8xx y los mayores con 0.97. Esto nos indica que el modelo no está clasificando como sanos a los enfermos por lo general. En este problema es muy relevante observar la sensitivity, debido a que refleja la proporción de casos con cancer bien detectados frente a los casos de cancer bien detectados y aquellos que se estando enfermos se toman como sanos. Como podemos ver en la mayoría de los casos son bastante altos sus valores.

A continuación se puede ver la tabla de resultados.

Table 1: Tabla de resultados

Kernel	Threshold	Fold	Accuracy	Specificity	Sensitivity	Precision	F Score	AUC
RBF	0.5	1	0.6109	0.0741	0.7537	0.7537	0.7537	0.9262
		2	0.5461	0.2639	0.6429	0.7181	0.6784	0.8407
		3	**0.8069**	0.2264	0.9563	0.8277	0.8874	0.9709
		4	0.7317	0.1200	0.8878	0.7982	0.8406	0.9563
		5	0.7425	0.1304	0.9548	0.7600	0.8463	0.9768
	0.4	1	0.7198	0.1852	0.8621	0.7991	0.8294	0.9262
		2	0.6702	0.2778	0.8048	0.7647	0.7842	0.8407
		3	**0.8340**	0.2453	0.9854	0.8354	0.9042	0.9709
		4	0.8171	0.3000	0.9490	0.8416	0.8921	0.9563
		5	0.7649	0.1884	0.9648	0.7742	0.8591	0.9768
	0.3	1	**0.8366**	0.4074	0.9507	0.8578	0.9019	0.9262
		2	0.7660	0.3056	0.9238	0.7951	0.8546	0.8407
		3	0.8610	0.3396	0.9951	0.8542	0.9193	0.9709
		4	0.8537	0.4200	0.9643	0.8670	0.9130	0.9563
		5	0.7836	0.2464	0.9698	0.7878	0.8694	0.9768
	0.25	1	0.8833	0.5185	0.9803	0.8844	0.9299	0.9262
		2	0.8050	0.3194	0.9714	0.8063	0.8812	0.8407
		3	**0.9035**	0.5283	1.0000	0.8918	0.9428	0.9709
		4	0.8740	0.4800	0.9745	0.8802	0.9249	0.9563
		5	0.7873	0.2464	0.9749	0.7886	0.8719	0.9768
Linear	0.5	1	0.6459	0.0185	0.8128	0.7569	0.7838	0.9771
		2	0.3901	0.0000	0.5238	0.6044	0.5612	0.9260
		3	**0.7490**	0.2830	0.8689	0.8249	0.8463	0.9341
		4	0.7358	0.1600	0.8827	0.8047	0.8418	0.9288
		5	0.7090	0.1884	0.8894	0.7597	0.8194	0.9285
	0.4	1	0.7588	0.0370	0.9507	0.7878	0.8616	0.9771
		2	0.4858	0.0556	0.6333	0.6617	0.6472	0.9260
		3	**0.8610**	0.3774	0.9854	0.8602	0.9186	0.9341
		4	0.7967	0.2400	0.9388	0.8288	0.8804	0.9288
		5	0.7575	0.1884	0.9548	0.7724	0.8539	0.9285
	0.3	1	0.8405	0.3148	0.9803	0.8432	0.9066	0.9771
		2	0.5709	0.1389	0.7190	0.7089	0.7139	0.9260
		3	**0.9112**	0.5849	0.9951	0.9031	0.9469	0.9341
		4	0.8293	0.3200	0.9592	0.8468	0.8995	0.9288
		5	0.7799	0.2174	0.9749	0.7823	0.8680	0.9285
	0.25	1	0.8716	0.4444	0.9852	0.8696	0.9238	0.9771
		2	0.6170	0.1806	0.7667	0.7318	0.7488	0.9260
		3	**0.9112**	0.5849	0.9951	0.9031	0.9469	0.9341
		4	0.8577	0.4400	0.9643	0.8710	0.9153	0.9288
		5	0.8097	0.3188	0.9799	0.8058	0.8844	0.9284