Máster en Big Data Analytics

Curso 2017-2018

Introducción al Aprendizaje Automático

Práctica 3

Máxima Verosimilitud con distribuciones normales o de Gauss

Jon Ander Gómez Adrián
jon@dsic.upv.es
Departament de Sistemes Informàtics i Computació
Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Índice

1. Motivación **2**

2. Objetivos **2**
2.1. Conceptuales 2
2.2. En el manejo de utilidades 2

3. Tareas **2**
3.1. Tarea 1 3
3.2. Tarea 2 3
3.3. Tarea 3 4
3.4. Tarea 3 4

4. Conclusiones **4**

5. Trabajo a evaluar **5**

6. Bibliografía **6**

1. Motivación

Tras ver en prácticas anteriores la precisión conseguida con clasificadores *Naive Bayes* en base a distribuciones normales o de Gauss, vamos a ver ahora cómo se comporta un clasificador basado en distribuciones de Gauss pero sin asumir que las variables de entrada son independientes. Es decir, se considera que las muestras están formadas por vectores de características (variables de entrada) que pueden estar correlacionadas.

2. Objetivos

2.1. Conceptuales

- Observar las diferencias en los resultados de un clasificador normal basado en distribuciones normales con respecto al del tipo *Naive Bayes* y ver qué tipo de clasificador es más sensible a transformaciones sobre los datos.

- Analizar si los resultados de un clasificador *Naive Bayes* y un clasificador normal basado en Gaussianas son muy distintos o muy similares dependiendo del *dataset* sobre el que se apliquen.

2.2. En el manejo de utilidades

- Saber cómo implementar un clasificador mediante una clase en Python con las funciones `fit()` y `predict()`.

- Aprender a modificar el código para trabajar con la matriz de varianzas-covarianzas diagonal.

- Aprender a modificar el código para trabajar con una matriz de varianzas-covarianzas global para todas las clases.

- Dominar el proceso de escoger un *dataset* nuevo, descargárselo y realizar experimentos sobre él.

3. Tareas

Para seguir esta práctica se utilizará el código Python `MyGaussian.py` disponible en *PoliformaT*.

3.1. Tarea 1

Utilizar la versión de código Python de la práctica anterior y modificarlo para que se utilice un objeto de la clase `MyGaussian` en lugar de un objeto de la clase `GaussianNB`.

En el código `MyGaussian.py` se dispone de una implementación de un clasificador basado en Gaussianas, es decir, cada clase se modela mediante una distribución normal o de Gauss, y los parámetros de cada Gaussiana (μ y Σ) se estiman por máxima verosimilitud.

Revisad bien el código de `MyGaussian.py` para entender cómo se realiza la estimación por máxima verosimilitud.

Ejecutad repetidas veces el clasificador sobre el *dataset* MNIST variando el preproceso a los datos para estudiar el efecto que sobre este nuevo clasificador tienen las transformaciones a las muestras. ¿Qué preproceso es el más idóneo para este clasificador? ¿Coincide en algunos casos con el comportamiento del clasificador anterior?

3.2. Tarea 2

Se propone trabajar con normalización de los datos de entrada, es decir, que cada componente de los vectores de características o muestras esté normalizado de manera que siga una distribución normal o de Gauss con media cero y varianza 1: $\mathcal{N}(0, 1)$.

Para ello debe importarse la classe `StandardScaler` de *Scikit-Learn*:

```
from sklearn.preprocessing import StandardScaler
```

y se utiliza como sigue para transformar un conjunto de muestras:

```
norm = StandardScaler()
new_X = norm.fit_transform( X )
```

Se propone también probar distintas variantes del preproceso. Llegados a este punto, se aconseja prepararse una hoja de cálculo e ir apuntando los resultados obtenidos por cada variante de clasificador, de preproceso, etc. Será de mucha utilidad para hacer el estudio o exploración de combinaciones de variantes para responder a las preguntas que figuran al final, cuyas respuestas servirán como trabajo para evaluar una parte de esta asignatura.

Otra transformación sugerida, que podría combinarse con otras es *Principal Component Analysis*:

```
from sklearn.decomposition import PCA
```

y se utiliza como sigue para transformar un conjunto de muestras:

```
pca = PCA(n_components=30)
new_X = pca.fit_transform( X )
```

3.3. Tarea 3

Utilizad el código de `MyGaussian.py` para trabajar con la matriz de varianzas-covarianzas diagonal.

El código `MyGaussian.py` ya está preparado para este caso. Puede verse que en el constructor de la clase se puede especificar tipo de matriz de varianzas-covarianzas. Ejemplos de cómo invocar al constructor:

```
classificador = MyGaussian( covar_type='full' )

classificador = MyGaussian( covar_type='diag' )
```

Como en la tarea anterior, los resultados obtenidos aquí deben servir para responder algunas de las preguntas del trabajo a evaluar.

3.4. Tarea 3

Utilizad el código de `MyGaussian.py` para trabajar con la matriz de varianzas-covarianzas diagonal o completa pero compartida entre todas las clases. En este caso la media se sigue estimando por cada clase, pero la matriz de varianzas-covarianzas se calcula a partir de la media global de todas las muestras.

```
classificador = MyGaussian( covar_type='tied' )

classificador = MyGaussian( covar_type='tied_diag' )
```

Los resultados obtenidos aquí también deben servir para responder algunas de las preguntas del trabajo a evaluar.

4. Conclusiones

Con esta práctica se pretende culminar una parte de la asignatura dedicada a los clasificadores, aplicándolos sobre varios *datasets* para poder compararlos.

También es importante que en esta práctica se entienda cómo se realiza la estimación por máxima verosimilitud para la modalidad de aprendizaje supervisado.

Y otro aspecto importante es el ver cómo modelos más complejos que no asumen premisas sobre independencia de las variables obtienen mejores resultados. Aunque para algunos *datasets* las diferencias no sean tan relevantes como en otros.

Por último, en cuanto a programación, esta práctica debe servir para que el alumno aprenda a implementar un clasificador según lo necesite.

5. Trabajo a evaluar

A partir de lo estudiado en esta práctica se proponen una serie de cuestiones que el alumno debe responder. Para ello debe reprogramar el código facilitado.

¿Qué ocurriría si se trabaja con una matriz de varianzas-covarianzas compartida para todas las clases tanto en su versión completa como en la diagonal? ¿Mejoran o empeoran los resultados?

En resumen podemos trabajar con cuatro posibles configuraciones de matriz de varianzas-covarianzas:

1. Matriz de varianzas-covarianzas diagonal compartida por todas las clases. **Caso más simple**. (`covar_type='tied_diag'`)

2. Matriz de varianzas-covarianzas diagonal pero se estima una diferente para cada clase. (`covar_type='diag'`)

3. Matriz de varianzas-covarianzas completa compartida por todas las clases. (`covar_type='tied'`)

4. Matriz de varianzas-covarianzas completa estimándose una diferente para cada clase. **Caso más complejo**. (`covar_type='full'`)

¿Es posible que dependiendo del *dataset* con el que se trabaje convenga una modalidad diferente de las cuatro posibles?

¿Cómo afectan las transformaciones sobre las muestras de entrada según la modalidad?

Para responder esta respuesta se deberá trabajar con otros *datasets*.

6. Bibliografía

[1] S. authors. Deep Learning. `http://www.deeplearning.net`, 2015. [Online: accessed June 2015].

[2] Y. Bengio, I. J. Goodfellow, and A. Courville. Deep learning. Book in preparation for MIT Press, 2015.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Addison-Wesley, second edition, 2000.

[6] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):2–42, 2006.

[7] G. Hinton. A practical guide to training restricted boltzmann machines, 2010.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, JMLR*, 12:2825–2830, 2011.

[9] Wikipedia. Machine Learning. `http://en.wikipedia.org/wiki/Machine_learning`, 2015. [Online: accessed April 2015].