Un estudio experimental del uso de combinaciones de redes de funciones de base radial en tareas de regresión

Carlos Pardo Aguilar, César García-Osorio, Juan J. Rodríguez, and José-Francisco Díez-Pastor

Lenguajes y Sistemas Informáticos. Universidad de Burgos

Resumen El presente trabajo es un estudio experimental de combinación de métodos para regresión usando redes de funciones de base radial (RBFs) como método base y 61 conjuntos de datos. Se han comparado *Bagging*, *Iterated Bagging*, *Random Subspaces*, *AdaBoost* y *Rotation Forest* usando 3 configuraciones de RBFs. El mejor resultado ha sido para *Iterated Bagging*, con la configuración que más RBFs usaba.

1. Introducción

Dada la importancia que hay en la bibliografía sobre el uso de algoritmos de combinaciones de métodos [9] y la experiencia de que funcionan mejor que los métodos por separado, se ha decidido analizar el comportamiento de las combinaciones de métodos más habituales [10]: *Bagging*, *Iterated Bagging*, *Random Subspaces*, *AdaBoost* y *Rotation Forest*; usando como método base de regresión redes de funciones de base radial.

Las redes de funciones de base radial son un caso de redes neuronales en el que en cada neurona aprende una función de base radial. [1] Las funciones de base radial son un caso de funciones lineales monótonas crecientes (o decrecientes) con la distancia de la entrada a un valor de referencia (centro). Para cada neurona se fija el centro, la escala en la que se calcula la distancia y el tipo de función. Ejemplos típicos de funciones de base radial monótonas crecientes son el valor absoluto de la distancia, o el cuadrado de la distancia, el ejemplo típico de función de base radial monótona decreciente con la distancia es la Gausiana de centro en $\mathbf{c}$:

$$h(\mathbf{x}) = e^{-\left(\frac{d(\mathbf{x},\mathbf{c})}{r}\right)^2}$$

En *Bagging* [2] cada miembro es entrenado con una muestra de los datos de entrenamiento. Para entrenar cada miembro, se van eligiendo los elementos de uno en uno siempre desde el conjunto de datos de entrenamiento original (*with replacement*), hasta tener tantos elementos como el tamaño del conjunto de datos de entrenamiento original, esto hace que algunos elementos de entrenamiento se puedan repetir, mientras que otros no aparecerán. La predicción de la combinación será el promedio de las predicciones de sus miembros.

Iterated Bagging [3] es un método para regresión basado en *Bagging*, en el que se combina el resultado de varios *Bagging* que se generan de forma consecutiva. Primero se construye un *Bagging* de forma normal. En función de las predicciones del *Bagging* anterior, se alterarán los valores de las variables a predecir. El siguiente *Bagging* se entrena para predecir esos valores alterados. Estos valores son los *residuos*: la diferencia entre el valor real y el predicho. Pero para estas predicciones no se utilizan las predicciones de todos los miembros del modelo *Bagging* obtenido. Se descartan los modelos que han usado un ejemplo durante el entrenamiento. La predicción de una combinación *Iterated Bagging* es la suma de las predicciones de cada *Bagging*. Según [14], *Iterated Bagging* es el método más efectivo en la mayor parte de las ocasiones. Tal como se verá en la parte experimental, este resultado parece confirmarse también cuando los regresores base son redes de funciones de base radial.

En *Random Subspaces* [8] cada miembro se entrena con todos los ejemplos de entrenamiento, pero con un subconjunto de sus atributos. La dimensión de los subespacios es un parámetro del método. También en este caso, la predicción de la combinación es el promedio de las predicciones de sus miembros.

AdaBoost [7] fue un método usado inicialmente para clasificación, pero hay variantes para regresión como *AdaBoost.R2* [6]. En estos métodos, cada ejemplo de entrenamiento tiene asociado un peso. Inicialmente todos los ejemplos empiezan con el mismo peso. Para construir los miembros de la combinación tiene en cuenta el valor de los pesos. Una vez construido un modelo, se recalculan los pesos de los ejemplos. El objetivo es que el peso asociado a cada instancia se incremente en función del error obtenido en la iteración anterior para esa instancia. Por lo que los ejemplos con mayor error son más importantes en la construcción del siguiente modelo. Por otro lado, en función del error de cada modelo, se le asocia un peso para la predicción final de la combinación. La predicción de una combinación *AdaBoost.R2*, es el promedio ponderado de las predicciones de sus miembros. Según [16], este método es el que obtiene uno de los mejores resultados promedio dentro de los métodos comparados.

Rotation Forest [13] fue un método usado inicialmente para clasificación, pero que ya se ha estudiado en otras ocasiones en regresión [16] [12]. Se basa entrenar cada método con un conjunto de entrenamiento al que se ha aplicado una rotación a cada partición aleatoria del conjunto de características, en las opciones por defecto, se le aplica una rotación de componentes principales a cada partición.

El resto del artículo describirá el diseño del experimento realizado, para pasar a mostrar y comentar los resultados y acabará con un apartado de conclusiones.

2. Diseño experimental

2.1. Conjuntos de datos

Para que el resultado del estudio sea suficientemente significativo se ha utilizado una amplia selección de conjuntos de datos, en concreto los 61 conjuntos

que se muestran en la tabla 1. Todos estos conjuntos de datos están disponibles para ser usados en el formato de weka[1], algunos han sido preparado por Luís Torgo[2].

La naturaleza de estos conjuntos de datos es bastante amplia, tienen un promedio de cinco mil trescientos ejemplos, hay conjuntos con trece ejemplos y otros con más de cuarenta mil ejemplos, la mediana es 286 ejemplos. Ocupan un promedio de 580 KB, hay conjuntos entre 1 KB y 4 MB, la mediana es 17 KB. Tienen entre 2 y 60 atributos, de los que entre ninguno y 11 son atributos nominales y entre ninguno y 60 son atributos numéricos.

tam.	ejem.	num.	nom.	conjunto
3 598	13	13	0	detroit
1 387	16	6	0	longley
1 182	27	4	0	gascons
1 622	38	4	1	schlvote
4 534	40	7	0	bolts
1 454	43	2	0	diabetes-numeric
1 155	52	3	0	vineyard
1 056	55	1	1	elusage
6 413	60	15	0	pollution
1 009	61	1	1	mbagrade
3 826	62	7	0	sleep
15 958	74	27	0	pyrimidines
16 811	93	16	6	auto93
2 968	96	4	0	baskball
4 550	108	4	2	cloud
8 722	125	2	2	fruitfly
10 532	130	6	3	echo-months
3 473	137	3	4	veteran
9 050	158	5	2	fishcatch
13 309	159	15	0	auto-price
5 393	167	0	4	servo
81 983	186	60	0	triazines
9 038	189	2	7	lowbwt
49 242	194	32	0	wisconsin
13 447	195	1	10	pharynx
6 266	200	10	0	pw-linear
13 617	205	17	8	auto-horse
6 038	209	6	0	machine-cpu
10 182	209	6	1	cpu
24 433	252	14	0	bodyfat
16 955	286	1	8	breast-tumor
21 224	294	6	7	hungarian
21 923	303	6	7	cleveland
22 358	303	6	7	cholesterol
29 997	398	4	3	auto-mpg
32 916	418	10	8	pbc
37 128	506	12	1	housing
72 762	528	19	2	meta
18 879	576	0	11	sensory
18 628	625	5	1	strike
57 008	950	9	0	stock
44 441	2 178	3	0	quake
197 741	4 177	7	1	abalone
309 015	7 129	5	0	delta-ailerons
580 622	8 192	12	0	cpu-small
690 435	8 192	8	0	bank-8FM
701 353	8 192	8	0	puma8NH
1 MB	8 192	21	0	cpu-act
1 MB	8 192	8	0	kin8nm
3 MB	8 192	32	0	bank-32nh
3 MB	8 192	32	0	puma32H
362 585	9 517	6	0	delta-elevators
4 MB	13 750	40	0	ailerons
2 MB	15 000	48	0	pole
2 MB	16 599	18	0	elevators
2 MB	20 640	8	0	cal-housing
2 MB	22 784	8	0	house-8L
4 MB	22 784	16	0	house-16H
1 MB	40 768	10	0	2d-planes
3 MB	40 768	10	0	friedman
4 MB	40 768	7	3	mv

Tabla 1. Los conjuntos de datos usados en los experimentos, bytes ocupados, número de ejemplos, número de atributos numéricos y nominales.

[1] `http://www.cs.waikato.ac.nz/ml/weka/index_datasets.html`
[2] `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html`

2.2. Algoritmos a comparar

Se han comparado diferentes algoritmos de combinación de métodos y se ha decidido usar redes de funciones de base radial (RBF) como método base. De entre los regresores que usan redes de funciones de base radial se ha elegido el algoritmo RBFNetwork que viene implementado en la distribución de Weka.

Se han utilizado tres configuraciones de este método base con distinto número de funciones de base radial utilizadas:

- con 2 funciones de base radial (es la opción por defecto en Weka).
- con tantas funciones de base radial como la raíz cuadrada de la mitad del número de ejemplos de entrenamiento, número que se ha elegido apoyándonos en la aproximación recomendada por Kanti Mardia [15] para determinar el número de cluster de un conjunto de datos.
- con tantas funciones de base radial como la raíz cuadrada del número de datos de entrenamiento.

Y para cada uno de estos tres métodos base se han probado configuraciones todas ellas con 50 modelos de las siguientes combinaciones:

- *Bagging*. Se han utilizado dos tamaños de conjuntos para entrenar cada modelo el 100 % y el 50 % del tamaño del conjunto de datos de entrenamiento.
- *Iterated Bagging*. Se ha utilizado una configuración de 10 iteraciones, cada una con 5 modelos miembro para tener el todal de 50 modelos.
- *Random Subspaces*. Se han utilizado subespacios con el 50 % y el 75 % del tamaño del espacio original.
- *AdaBoost.R2*. Se han utilizado 6 configuraciones, por un lado se ha utilizado la estrategia de cambiar los pesos[3] de cada ejemplo, y por otro lado se ha utilizado un sistema de selección de los ejemplos,[4] estas dos estrategias se denotan con los sufijos «-W» y «-S» y se han combinado con tres funciones de perdida: lineal, cuadrática y exponencial, que se denotan con los sufijos «-L», «-Q» «-E».
- *Rotation Forest*. Se han utilizado las opciones por defecto.

En el experimento se han añadido además las tres configuraciones del método base directamente, para comparar la mejora del uso de combinaciones para cada uno de ellas.

El resultado del experimento se obtuvo usando *validación cruzada* [5] 5×2 particiones. Los datos se dividen en dos mitades, una para comprobar la predicción obtenida por la combinación entrenada con la otra mitad. Y luego se cambian los papeles de esas mitades. Este proceso se repite cinco veces. Por lo que el resultado del informe representa la media de esas diez ejecuciones.

[3] *reweighting*
[4] *resampling*

3. Resultados

3.1. Resultados con el mismo método base

En la tabla 2 podemos comparar el promedio de la posición para cada conjunto de datos en que queda clasificado cada algoritmo [4], cuando todos usan el mismo método base.

Se puede observar que *Iterated Bagging* queda el mejor cuando se le compara con el resto de los algoritmos en los tres casos, tanto cuando se usan sólo 2 funciones de base radial como cuando se usan $\sqrt{N_T/2}$ funciones de base radial, donde N_T es el número de datos del conjunto de datos de entrenamiento, como si se usan más funciones de base radial. La versión de *Bagging* que utiliza conjuntos del mismo tamaño que el conjunto de datos de entrenamiento es siempre la segunda mejor configuración.

También se puede observar que los *Random Subspaces*, sobre todo su versión con tamaño del subespacios del 75 %, mejoran sus posiciones con respecto a los demás algoritmos al aumentar el número de funciones de base radial del método base, y la versión de *Bagging* con tamaño 50 % del tamaño del conjunto de datos de entrenamiento sigue el la tendencia opuesta.

Llama la atención que la configuración de *AdaBoost.R2* con repesado y función de perdida exponencial quede siempre como la peor configuración, incluso peor que el método base que utiliza.

3.2. Resultados globales

El ranking con los resultados globales de todas las configuraciones, que se pueden consultar en la tabla 3, sitúan como mejor algoritmo a la configuración de *Iterated Bagging* que usa $\sqrt{N_T}$ funciones de base radial, donde N_T en el número de datos del conjunto de datos de entrenamiento.

También se puede observar que, como norma general, aumentar el número de funciones de base radial es más importante en la reducción del error que el algoritmo de combinación elegido, dado que doce de las trece configuraciones de los algoritmos comparados que sólo usan dos funciones de base radial han quedado las posiciones del último tercio de la tabla y ninguna en el primer tercio; mientras que de las configuraciones que usan $\sqrt{N_T/2}$ funciones de base radial cinco quedan en el primer tercio de la tabla de posiciones, siete en el tercio central y solo una en el último tercio; y de las trece configuraciones que usan más funciones de base radial, ocho quedan en el primer tercio de la tabla y cinco en el tercio central, no quedando ninguna tercio final.

También se puede defender el aumento del número de funciones de base radial mirando el promedio de posiciones de todos las combinaciones que usan la misma configuración de método base (promedio de la columna en la tabla 3). Los algoritmos con más funciones de base radial, tiene mejor promedio que el resto.

También se observa que algunos algoritmos han cambiado ligeramente su posición relativa, con respecto al resto de los que utilizan su mismo método

2 funciones de base radial

# posición		algoritmo
1	4,10	*Iterated Bagging*
2	4,57	*Bagging* 100 %
3	4,69	*Bagging* 50 %
4	6,08	*Random Subspaces* 50 %
5	6,18	*AdaBoost.R2-S*-L
6	6,76	*Random Subspaces* 75 %
7	7,18	*AdaBoost.R2-S*-Q
8	7,31	*Rotation Forest*
9	8,22	Red RBF
10	8,34	*AdaBoost.R2-S*-E
11	7,92	*AdaBoost.R2-W*-Q
12	8,43	*AdaBoost.R2-W*-L
13	11,21	*AdaBoost.R2-W*-E

$\sqrt{N_T/2}$ funciones de base radial

# posición		algoritmo
1	3,26	*Iterated Bagging*
2	4,59	*Bagging* 100 %
3	5,56	*Bagging* 50 %
4	5,64	*Random Subspaces* 50 %
5	5,84	*Random Subspaces* 75 %
6	6,64	*AdaBoost.R2-S*-L
7	6,72	*Rotation Forest*
8	7,46	*AdaBoost.R2-S*-E
9	7,62	*AdaBoost.R2-S*-Q
10	8,79	*AdaBoost.R2-W*-L
11	9,11	*AdaBoost.R2-W*-Q
12	9,14	Red RBF
13	10,62	*AdaBoost.R2-W*-E

$\sqrt{N_T}$ funciones de base radial

# posición		algoritmo
1	4,36	*Iterated Bagging*
2	5,03	*Bagging* 100 %
3	5,29	*Random Subspaces* 75 %
4	5,97	*Random Subspaces* 50 %
5	6,28	*Bagging* 50 %
6	6,46	*Rotation Forest*
7	6,64	*AdaBoost.R2-S*-E
8	6,74	*AdaBoost.R2-S*-Q
9	7,03	*AdaBoost.R2-S*-L
10	8,70	*AdaBoost.R2-W*-L
11	8,95	*AdaBoost.R2-W*-Q
12	9,01	Red RBF
13	10,54	*AdaBoost.R2-W*-E

Tabla 2. Ranking promedio de los algoritmos utilizando el mismo tipo de modelo base.

base, si lo comparamos con la tabla 2, como por ejemplo el método base que estaba el 9 en la tabla anterior ha sido superado por dos de las configuraciones de *AdaBoost.R2*.

Se ha comparado el número de veces que los métodos ganaban para cada conjunto de datos al método base por defecto (con dos funciones de base radial), usando *t-test estadístico corregido remuestreado* [11] (con nivel de significación: 0,05).

En lo que se refiere a victorias significativas, *Bagging* 100 % es el que gana más veces al método base por defecto (con dos funciones de base radial), un total de 30. Pero llama la atención que este método base, pese a quedar en la posición 37, la antepenúltima de la tabla, es capaz de ganar significativamente hasta en 103 ocasiones a 21 configuraciones de diferentes combinaciones de métodos.

Las configuraciones del algoritmo *AdaBoost.R2* tiene un comportamiento muy variable, el caso más extremo es su configuración *AdaBoost.R2-S*-Q con $\sqrt{N_T}$ funciones de base radial que pese a ser capaz de ganar en 13 de los 61 conjunto de datos a todos los demás algoritmos, ganando de forma significativa 19 veces al algoritmo base por defecto, en el ranking cae al puesto 13 y pierde de forma significativa 6 vez con ese método base.

La configuración del algoritmo *Rotation Forest* con $\sqrt{N_T}$ funciones de base radial es la segunda que más veces es capaz de ganar a todos los demás algoritmos en un mismo conjunto de datos, en 10, pese a ello queda en el décimo puesto de la tabla, aunque en este caso no pierde significativamente con el algoritmo base en ningún conjunto de datos y lo bate en 27 de ellos.

4. Conclusiones

Se ha analizado el comportamiento de diferentes combinaciones de métodos usando redes de funciones de base radial como método base. Se ha visto que la combinaciones que mejores resultados obtiene es *Iterated Bagging*, confirmando los resultados de [14] para otros métodos base.

Se han analizado los comportamientos de las redes con un número fijo de funciones de base radial (2) y con un número variable en función del tamaño del conjunto de datos de entrenamiento (N_T). El tener redes con más funciones de base radial mejora, en el promedio de los 61 conjunto de datos del experimento, los resultados para todas las combinaciones de métodos.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Economía y Competitividad, proyecto TIN 2011-24046.

Referencias

1. Bishop, C.M.: Pattern recognition and machine learning. Springer, 2nd edn. (2006)

#	posición			algoritmo	derrotas/empates/victorias	# victorias
	n°RBF				vs 2 RBF	conjunto de datos
	2	$\sqrt{N_T/2}$	$\sqrt{N_T}$			
1			8,16	*Iterated Bagging*	0/35/26	6
2		8,20		*Iterated Bagging*	0/34/27	5
3			9,85	*Bagging* 100 %	0/31/30	3
4			10,40	*Random Subspaces* 75 %	0/35/26	3
5		10,95		*Bagging* 100 %	0/31/30	-
6			11,77	*Random Subspaces* 50 %	0/34/27	5
7			12,66	*Bagging* 50 %	0/33/28	1
8		13,55		*Random Subspaces* 75 %	0/36/25	-
9		13,59		*Bagging* 50 %	0/35/26	1
10			13,74	*Rotation Forest*	0/34/27	10
11		13,85		*Random Subspaces* 50 %	0/35/26	1
12			15,43	*AdaBoost.R2-S*-E	7/34/20	4
13			15,77	*AdaBoost.R2-S*-Q	6/36/19	13
14			16,02	*AdaBoost.R2-S*-L	5/39/17	-
15		16,21		*Rotation Forest*	0/37/24	2
16		17,25		*AdaBoost.R2-S*-L	7/33/21	1
17			18,66	*AdaBoost.R2-W*-L	2/39/20	-
18			18,97	*AdaBoost.R2-W*-Q	1/45/15	-
19			19,11	Red RBF	0/36/25	-
20		19,28		*AdaBoost.R2-S*-E	7/33/21	2
21		20,23		*AdaBoost.R2-S*-Q	9/31/21	-
22		20,86		Red RBF	0/42/19	1
23		20,89		*AdaBoost.R2-W*-L	1/44/16	-
24		22,08		*AdaBoost.R2-W*-Q	3/44/14	-
24	22,64			*Iterated Bagging*	0/52/9	-
26			24,08	*AdaBoost.R2-W*-E	4/42/15	-
27	24,28			*Bagging* 100 %	0/54/7	-
28	24,54			*Bagging* 50 %	0/52/9	1
29	26,54			*AdaBoost.R2-S*-L	5/50/6	-
30	26,59			*Random Subspaces* 50 %	2/50/9	-
31		26,75		*AdaBoost.R2-W*-E	5/38/18	-
32	27,42			*Random Subspaces* 75 %	5/46/10	-
33	27,98			*AdaBoost.R2-S*-Q	1/56/4	-
34	28,23			*Rotation Forest*	2/52/7	-
35	29,52			*AdaBoost.R2-W*-Q	2/51/8	1
36	29,51			*AdaBoost.R2-S*-E	10/46/5	1
37	29,55			Red RBF	-/61/-	-
38	29,92			*AdaBoost.R2-W*-L	7/48/6	-
39	34,98			*AdaBoost.R2-W*-E	12/47/2	-
	27,82	17,21	14,97	promedio de la columna		

Tabla 3. Ranking promedio de todas las configuraciones de los algoritmos, mostrando el resultado en columnas separadas en función del número de funciones de base radial del modelo base utilizado. La penúltima columna es el número de veces que la configuración ha perdido significativamente con el método base configurado por defecto (con 2 funciones de base radial) / o se considera empate / o ha ganado significativamente. La última columna es el número de conjuntos de datos en los que el algoritmo ha quedado en la mejor posición.

2. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
3. Breiman, L.: Using iterated bagging to debias regressions. Machine Learning 45(3), 261–277 (Dec 2001), http://dx.doi.org/10.1023/A:1017934522171
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
5. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
6. Drucker, H.: Improving regressors using boosting techniques. In: ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning. pp. 107–115. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1997), http://portal.acm.org/citation.cfm?id=645526.657132
7. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: 13th International Conference on Machine Learning. pp. 148–156. Morgan Kaufmann, San Francisco (1996)
8. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
9. Kuncheva, L.I.: Combining Pattern Classifiers. Wiley, 2nd edition edn. (2014)
10. Mendes-Moreira, J., Soares, C., Jorge, A.M., de Sousa, J.F.: Ensemble approaches for regression: A survey. ACM Computing Surveys 45(1) (Nov 2012), http://dx.doi.org/10.1016/j.amc.2007.05.010
11. Nadeau, C., Bengio, Y.: Inference for the generalization error. Machine Learning 52(3) (2003)
12. Pardo, C., Diez-Pastor, J.F., García-Osorio, C., Rodríguez, J.J.: Rotation forest for regression. Applied Mathematics and Computation 219(19), 9914–9924 (Jun 2013), doi:10.1016/j.amc.2013.03.139
13. Rodríguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(10), 1619–1630 (Oct 2006), http://doi.ieeecomputersociety.org/10.1109/TPAMI.2006.211
14. Suen, Y., Melville, P., Mooney, R.: Combining bias and variance reduction techniques for regression trees. In: Machine Learning: ECML 2005. pp. 741–749. Springer (2005), http://dx.doi.org/10.1007/11564096_76
15. Wikipedia: Determining the number of clusters in a data set — wikipedia, the free encyclopedia (2015), \url{http://en.wikipedia.org/w/index.php?title=Determining_the_number_of_clusters_in_a_data_set&oldid=661006527#Rule_of_thumb}, [Online; accessed 21-May-2015]
16. Zhang, C., Zhang, J., Wang, G.: An empirical study of using rotation forest to improve regressors. Applied Mathematics and Computation 195(2), 618–629 (Feb 2008), http://dx.doi.org/10.1016/j.amc.2007.05.010