Синергия алгоритмов классификации (SVM Multimodelling)

С. Иванычев, А. Адуенко
sergeyivanychev@gmail.com, aduenko1@gmail.com
Московский физико-технический институт

В данной статье рассматривается проблема агрегирования небольшого количества сильных классификаторов с целью улучшения решений задач классификации и регрессии. В качестве примера подобной системы рассматривается система SVM алгоритмов использующая kernel-trick с различными ядрами. Для комбинации решений и улучшения качества прогнозирования в задачах классификации и регрессии (SVR) авторы предлагают способ формирования новых признаков на основе сгенерированных отступов (*margins*) каждым классификатором, приводят алгоритм обучения на полученных объектах и анализируют отличия множеств опорных объектов для различных ядер. В качестве практической проверки были проведены эксперименты на различных реальных данных из репозитория UCI.

Ключевые слова: *двухклассовая классификация, композиция алгоритмов, SVM, SVR, бэггинг*

Введение

Работа посвящена комбинированию небольшого количества сильных SVM, использующих kernel-trick с различными ядрами и получению агрегированного классификатора для улучшения решений задач классификации и регрессии.

SVM(Support Vector Machine) или *метод опорных векторов*[1, 2] — это один из наиболее распространенных и эффективных методов в машинном обучении, которые используется для задач классификации и регрессии (SVR). Задача математического программирования сводится к двойственной задаче, функционалы в которой не зависят от векторов признаков как таковых, а лишь от их попарных скалярных произведений. Использование особых функций, *ядер*, то есть скалярных произведений в сопряженном пространстве, позволяет получить разделяющие поверхности между классами более сложной формы [3]. Наша цель — скомбинировать SVM с различными примененными ядрами для улучшения решения, а также анализ множеств опорных объектов в случае разных использованных ядер.

Наиболее классическими методами агрегирования алгоритмов являются бэггинг (*bagging*)[4] и бустинг (*boosting*) [5], и их вариации, однако они работают только с большим количеством слабых классификаторов, что делает невозможным использование его использование для указанного множества базовых алгоритмов.

Среди способов агрегации для небольшого количества классификаторов можно выделить, например, выбор большинства классификаторов [6], комбинирование ранжирований (rankings) по классам, сделанных различными классификаторами [7]. В дальнейшем было показано, что все подобные методы есть особые случаи составного классификатора из [8], появляющиеся при особых условиях или способах аппроксимации.

Различные способы агрегации SVM используются во многих задачах анализа данных. [9] использовали совокупность SVM для уменьшения ошибочно негативных классификаций (FP) в задаче фильтрации спама среди электронных писем. Для этого на электронных письмах были введены различные метрики, для каждой из них был приспособлен SVM, а затем результат получался голосованием [8]. [10], решавшие задачу распознавания

написанных рукой символов, делили множество признаков на четыре непересекающихся подмножества, и на каждом из них обучали SVM, увеличив этим самым коэффициент распознавания по сравнению с одним SVM.

В последнее время стал набирать популярность *метод многоядерного обучения* (MKL, multiple kernel learning) [11, 12, 13], который основывается на том, что линейная комбинация ядер также является ядром. Данный метод хорош при объединении данных из нескольких источников и полной автоматизации, так как суперпозиция функций может быть оптимизирована любым методом валидации (например кросс-валидацией).

Мы также предлагаем использовать накопившийся банк ядер, однако не на этапе обучения SVM, а на этапе агрегирования обученных алгоритмов. Известно, что алгоритм b_i для объекта x_j обучающей выборки генерирует *отступ* (margin). По отступу в общем случае можно определить не только предсказанный класс, но и насколько «уверен» в своем решении алгоритм. В случае банка с n ядрами и обучающей выборки с m сэмплами мы получим матрицу отступов $M \in \mathbb{R}^{m\times n}$. Отнормировав ее, мы получим новую матрицу «объект-признак», где вектором признаков каждого объекта будет вектор отнормированных отступов.

В этой работе предложен алгоритм обучения на матрице отступов, проведен анализ опорных объектов, генерируемые различными ядрами, а также проведено тестирование полученного алгоритма на реальных данных репозитория UCI.

Постановка задачи

Пусть $X^l = (\boldsymbol{x}_i, y_i)_{i=1}^l$ — обучающая выборка, $\boldsymbol{x} \in \mathbb{R}^n, y \in \{\pm 1\}$.

Определение 0.1. Под s-й *моделью* будем понимать SVM с ядром K_s где выбрано множество ядер:

$$\mathcal{K} = \{K_i\}_{j=1}^m$$

При обучении каждая модель дает классификатор или регрессор (в зависимости от типа Y). Например, для случая $Y \in \{-1, +1\}$ классификации алгоритм выглядит следующим образом:

$$b_s(\mathbf{x}) = \mathbf{sign} \sum_{i=1}^{l} \lambda_i y_i K_s(\boldsymbol{x}_i, \boldsymbol{x}) - w_0$$

Где $\{\lambda_i\}$ и w_0 находятся из решения задачи математического программирования[3]

$$\begin{cases} \sum_{i=1}^l \lambda_i + \frac{1}{2}\sum_{i=1}^l \sum_{j=1}^l \lambda_i\lambda_j y_i y_j K_s(\boldsymbol{x}_i, \boldsymbol{x}_j) \to \min\limits_{\lambda} \\ 0 \leqslant \lambda_s \leqslant c, \;\; i = 1 \dots l \\ \sum_{i=1}^l \lambda_i y_i = 0 \end{cases}$$

Результатом обучения является то, что модель для обучающей выборки генерирует вектор *отступов* (margins) для каждого объекта. Совокупность отступов порождает матрицу отступов размерности $M \in \mathbb{R}^{l\times m}$, в котором (i, j)-й элемент — это отступ i-го объекта в SVM с j-м ядром. Авторы утверждают, что эта матрица может быть использована как новая обучающая выборка для построения агрегирующего алгоритма.

Рассмотрим M как новую матрицу «объект-признак», $\mathcal{A}$ — множество алгоритмов вида

$$\mathcal{A} = \{a(\boldsymbol{x}) = g(\boldsymbol{x}, \theta) | \theta \in \Theta\} \quad g : \mathbb{R}^m \to Y \tag{1}$$

где Θ — некое семейство параметров.

Определение 0.2. Пару $(g, \mathcal{K})$ назовем *мультимоделью.*

$L(y, y^*)$ — функционал качества. Тогда перед нами стоит Задача выбора алгоритма агрегирования сильных классификаторов супермодели:

$$L(y, g(M(X), \theta)) \to \min_{\Theta} \tag{2}$$

В качестве этой функции мы предлагаем использовать площадь под ROC-кривой (AUC-ROC) как устойчивый к несбалансированностям в выборке.

Близость моделей

Так как агрегация сильных классификаторов должна быть устойчива к похожим моделям в ее составе, она должна каким-то образом определять идентичные SVM. В данной секции попробуем установить связь между корреляцией векторов отступов, генерируемых разными ядрами на одной и той же обучающей выборке и степенью «похожести» ядер. Понятие «похожести» или близости отождествляется с введением метрики на пространстве ядер.

Определение 0.3. *Расстоянием между ядрами* $\rho(K_i, K_j)$ на выборке X^l будем называть функцию:

$$\rho_{X^l}(K_i, K_j) = \frac{\#\left[\mathrm{SV}_i \,\Delta\, \mathrm{SV}_j\right]}{\#\left[\mathrm{SV}_i \cup \mathrm{SV}_j\right]}$$

Где под SV_j понимается множество опорных объектов на j-м ядре, под знаком Δ — симметрическая разность.

Определение 0.4. *Похожестью SVM с ядрами* K_i, K_j будем называть следующую функцию

$$s_{X^l}(K_i, K_j) = 1 - \mathrm{corr}(M_i, M_j)$$

где под M_i понимается вектор отступов соответственного SVM.

В качестве исходных данных возьмем датасеты Wine [14], German credit data[15] и Heart disease[16]. Варьируя коэффициент L2-регуляризации, в качестве базового набора ядер возьмем

— Линейное
— Полиномиальное (степени 3, 4, 5)
— RBF-ядро ($\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1\}$)

Для каждого значения коэффициента регуляризации построим графики опишем полученное множество классификаторов.

German credit dataset

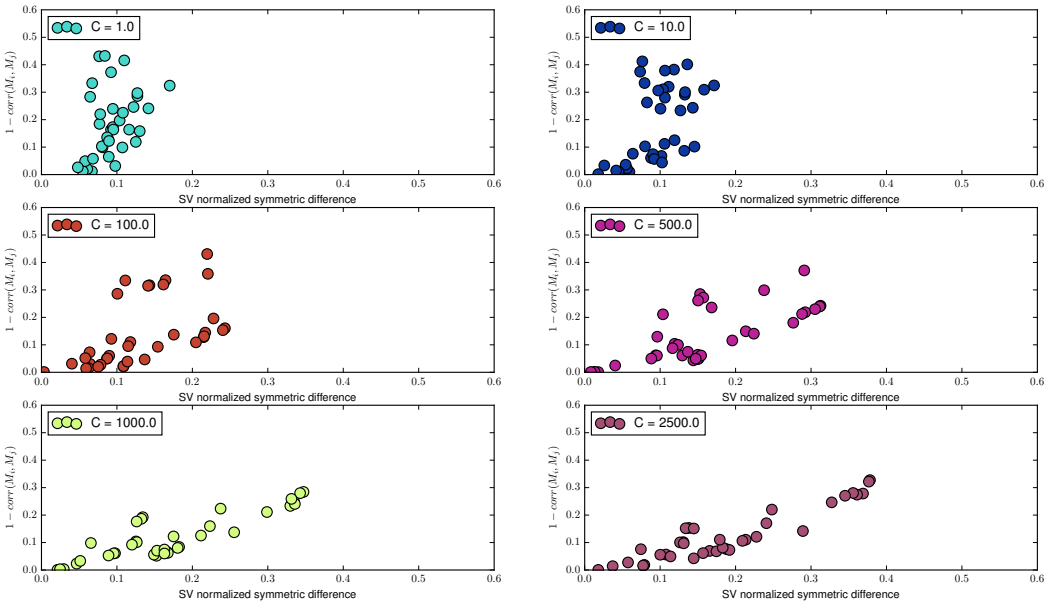

Рис. 1. German credit

Таблица 1. German info

	mean(#SV)	mean$(1 - \text{corr}(M_i, M_j))$	mean$(\rho_{X^l}(K_i, K_j))$	Correlation
$C = 1.0$	603.4	0.184	0.094	0.376
$C = 10.0$	603.6	0.187	0.097	0.537
$C = 100.0$	594.7	0.134	0.131	0.556
$C = 500.0$	584.3	0.133	0.161	0.717
$C = 1000.0$	581.6	0.120	0.172	0.870
$C = 2500.0$	577.9	0.126	0.189	0.918

Обучающая выборка состоит из 1000 семплов по 24 числовых признака.

Wine dataset

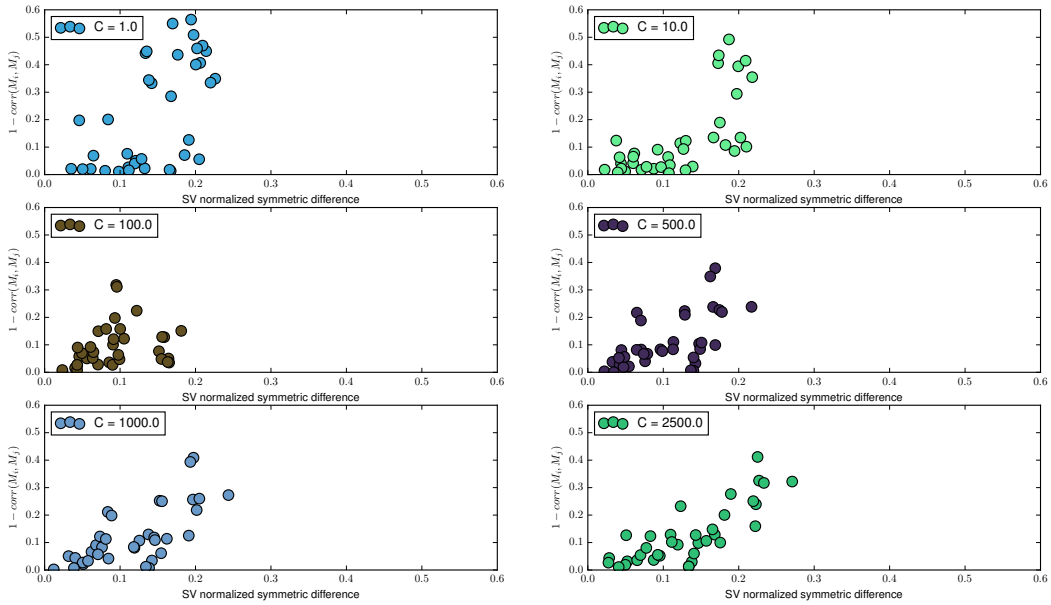

Рис. 2. Wine quality

Таблица 2. Wine info

	mean(#SV)	mean$(1 - \text{corr}(M_i, M_j))$	mean$(\rho_{X^l}(K_i, K_j))$	Correlation
$C = 1.0$	3284.1	0.220	0.144	0.600
$C = 10.0$	3284.9	0.130	0.121	0.687
$C = 100.0$	3275.0	0.091	0.091	0.270
$C = 500.0$	3252.6	0.110	0.105	0.591
$C = 1000.0$	3235.2	0.124	0.118	0.694
$C = 2500.0$	3208.6	0.127	0.133	0.795

Heart dataset

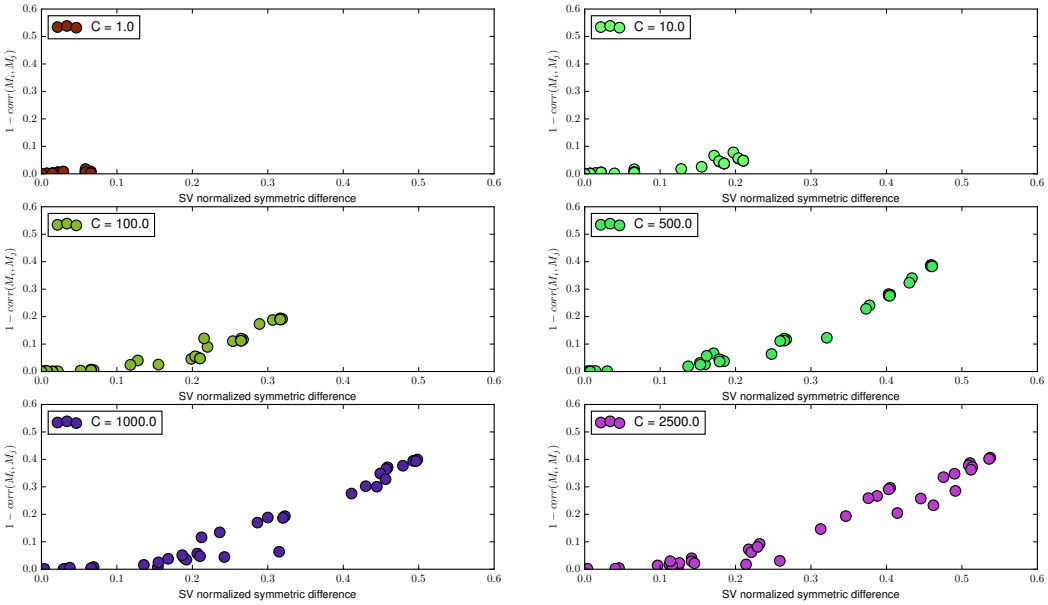

Рис. 3. Heart disease

Таблица 3. Heart info

	mean(#SV)	mean$(1 - \text{corr}(M_i, M_j))$	mean$(\rho_{X^l}(K_i, K_j))$	Correlation
$C = 1.0$	272.0	0.003	0.027	0.608
$C = 10.0$	260.8	0.020	0.088	0.929
$C = 100.0$	249.1	0.063	0.152	0.927
$C = 500.0$	231.9	0.135	0.238	0.940
$C = 1000.0$	223.1	0.157	0.268	0.953
$C = 2500.0$	211.4	0.166	0.297	0.962

Выводы из эксперимента

Уже по этим данным можно сказать, что (будем называть $1-\text{corr}(M_i, M_j)$ *расстоянием между отступами*).

— С ростом константы регуляризации расстояние между ядрами и расстояние между их отступами лучше коррелируют между собой.
— При высоких параметре регуляризации коэффициент корреляции Пирсона достигает более 0.8, то есть расстояния практически линейно зависят друг от друга.
— Вектора средних ядерных и отступных расстояний кореллируют по-разному на различных датасетах (на Wine и Heart корелляции Пирсона 0.85 и 0.99 соответственно, на German — -0.92).

Описание алгоритма

Модели

В качестве множества ядер, участвующих в мультимодели выберем следующий набор

— Линейное
— Полиномиальное (степени 3, 4, 5)
— RBF-ядро ($\gamma \in \{0.0001, 0.001, 0.01, 0.1, 1\}$)
— INK-spline ядро (степени 1, 2)

Данное множество выбрано как наиболее универсальные представители различных классов ядер.

Метрика качества

Так как в работе предполагается построить универсальную модель классификации, метрика качества должна быть устойчивой к различным соотношениям между классами и корректно отображать качество классификации. В качестве метрики в нашей задачи выбрана AUC-ROC — площадь, ограниченная *ROC* кривой.

Мультимодель

В качестве базового алгоритма агрегации классификаторов выбрана логистическая регрессия.

Алгоритм

Алгоритм 1 Обучение мультимодели SVM в случае двухклассовой классификации

Вход: обучающая выборка $X^l = (\mathbf{X}, y) = (\mathbf{x_i}, y_i)_{i=1}^l, x_i \in R^m, y_i \in \{0, 1\}$, множество ядер $\mathcal{K} = \{K_i\}_{i=1}^n$

Выход: весовые параметры логистической регрессии w', вектор обученных классификаторов CLS, нормирующие выборку и отступы коэффициенты $\mathbf{w}$;

отнормировать обучающую выборку

$$Mean \leftarrow \text{mean}(\mathbf{X})) \;\; Std \leftarrow \text{std}(\mathbf{X})$$

$$\mathbf{X} \leftarrow \frac{\mathbf{X} - Mean}{Std}$$

инициализировать массив классификаторов CLS
инициализировать матрицу $Margins \in \mathbb{R}^{l \times n}$
для K in $\mathcal{K}$
Вычисляем матрицу Грамма Γ_K ядра K для объектов $\mathbf{X}$.
Обучаем SVM с ядром K, решаем оптимизационную задачу

$$\sum_{i=1}^{l}(1 - M_i(w, w_0))_+ + \frac{1}{2C}||w||^2 \to \min_{w, w_0}$$

Где $M_i(w, w_0) = y_i(K(w, x_i) - w_0)$
Добавляем обученный классификатор в вектор CLS.
$Margins[:, K] \leftarrow \mathbf{M_K} = (M_1 \dots M_l)'$
Нормируем отступы, чтобы средний модуль отступов был равен единице

$$MeanMargins \leftarrow \frac{Margins}{\text{mean}(\text{abs}(Margins))}$$

$$Margins \leftarrow \frac{Margins}{MeanMargins}$$

Обучаем логистическую регрессию на $X_*^l = (Margins, y)$. Решаем задачу минимизации

$$Q(w) = \sum_{i=1}^{l} \log(1 + \exp(-\langle w, x_i \rangle y_i)) \to \min_w \;\; x_i \in Margins$$

$$w' \leftarrow \arg\min Q(w)$$

return w', $MeanMargins$, $Mean$, Std

Алгоритм 2 Классификация новых объектов обученной мультимоделью SVM в случае двухклассовой классификации

Вход: тестовая выборка $X^l = (\mathbf{X}, y) = (\mathbf{x_i}, y_i)_{i=1}^l, x_i \in R^m, y_i \in \{0, 1\}$, множество ядер $\mathcal{K} = \{K_i\}_{i=1}^n$, вектор весов w', вектор нормировки $MeanMargins$, $Mean$, Std
Выход: вектор спрогнозированных меток классов y_{pred}

отнормировать тестовую выборку

$$\mathbf{X} \leftarrow \frac{\mathbf{X} - Mean}{Std}$$

посчитать отступы на поступивших объектах и отнормировать их

$$Margins \leftarrow CLS(X)/MeanMargins$$

Классифицируем объекты логистической регрессией с вектором весов w'

$$y_{pred} \leftarrow LogisticRegression(Margins, w')$$

return y_{pred}

Литература

[1] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[2] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

[3] Alex J Smola, Bernhard Sch, and B Schölkopf. A Tutorial on Support Vector Regression. *Statistics and Computing*, 14(3):199–222, 2004.

[4] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(421):123–140, 1996.

[5] Y. Freund. Boosting a Weak Learning Algorithm by Majority, 1995.

[6] J. Franke and E. Mandler. A comparison of two approaches for combining the votes of\ncooperating classifiers. *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, pages 1–4, 1992.

[7] Tin Kam T.K. Ho, JJ Hull, Sargur N SN Srihari, and Senior Member. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[8] J. Kittler, M. Hater, and R. P W Duin. Combining classifiers. *Proceedings - International Conference on Pattern Recognition*, 2(3):897–901, 1996.

[9] Manuel Martin-merino and Manuel Mart. Combibing SVM Classifiers for Email Anti-spam Filtering. 4507(February), 2007.

[10] D. Gorgevik and D. Cakmakov. Handwritten Digit Recognition by Combining SVM Classifiers. *EUROCON 2005 - The International Conference on Computer as a Tool*, 2(February):1393–1396, 2005.

[11] Martin Dyrba, Michel Grothe, Thomas Kirste, and Stefan J. Teipel. Multimodal analysis of functional and structural disconnection in Alzheimer's disease using multiple kernel SVM. *Human Brain Mapping*, 36(6):2118–2131, 2015.

[12] S.S. Bucak, R. Jin, and Ak. Jain. Multiple Kernel Learning for Visual Object Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1354–1369, 2014.

[13] Salah Althloothi, Mohammad H. Mahoor, Xiao Zhang, and Richard M. Voyles. Human activity recognition using multi-features and multiple kernel learning. *Pattern Recognition*, 47(5):1800–1812, 2014.

[14] Stefan Aeberhard. Wine data set, 1991.

[15] Dr. Hans Hofmann. Statlog (german credit data) data set, 1994.

[16] Andras Janosi M.D, M.D William Steinbrunn, M.D Matthias Pfisterer, and Ph.D Robert Detrano, M.D. Heart disease data set, 1988.