عنوان: **Accord.NET #1**
نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۵/۲۲ ۱۹:۳۰
آدرس: www.dotnettips.info
گروه‌ها: OpenCV, Accord.NET, Machine Learning, Artificial Intelligence

Accord.NET کتابخانه‌ای است متن‌باز و بسیار کارآمد که در آن توابع بسیار زیادی در حوزه‌ی تحلیل آماری (statistical analysis)، یادگیری ماشین (machine learning)، پردازش تصویر (Image processing) و بینایی ماشین (computer vision) قرار گرفته‌اند تا در برنامه‌های .NET. ایی مورد استفاده قرار گیرند.

چارچوب Accord.NET توسط آقای سزار سوزا بر پایه کتابخانه‌ی مشهور و محبوب AForge.NET (که توسط آقای اندرو کریلو ایجاد شده بود) بنا شده و البته ابزارهای جدید زیادی به همراه یک محیط کامل برای محاسبات علمی (scientific computing) در .NET. به آن اضافه شده است.

این چارچوب متشکل از چندین کتابخانه است که می‌توان آن را از طریق NuGet دریافت و نصب کرد.

کتابخانه‌های Accord.NET را می‌توان به سه دسته‌ی کلی تقسیم کرد :

1. محاسبات علمی (scientific computing)

1.1. Accord.Math	جهت کار با ماتریس‌ها عددی تجزیه ماتریس‌ها (decomposition matrix) الگوریتم‌های بهینه سازی عددی برای مسائل محدود و نامحدود توابع و ابزارهای خاص جهت استفاده در کاربردهای علمی
1.2. Accord.Statistics	شامل توابعی جهت توزیع‌های احتمال (probability distributions) آزمایش فرضیات (hypothesis testing) مدل‌های آماری (statistical models) و توابعی شامل : رگرسیون خطی، مدل پنهان مارکوف (Hidden Markov Models)، آنالیز اجزای اساسی (Principal Component Analysis) و خیلی از تکنیک‌های مرتبط دیگر.
1.3. Accord.MachineLearning	شامل دسته بندهای معروف از جمله : ماشین برداری پشتیبان - Support Vector Machines درخت تصمیم - Decision Trees مدل نیو بیز - Naive Bayesian models K-means مدل ترکیبی گوسین - Gaussian Mixture models و الگوریتم‌های متداول دیگری مانند : Ransac, Cross-validation و Grid-Search
1.4. Accord.Neuro	شامل الگوریتم‌های معروف در حوزه شبکه‌های عصبی مصنوعی مانند: لونبرگ مارکوارت - Levenberg-Marquardt Parallel Resilient Back-propagation شبکه باور عمیق - Deep Belief Networks ماشین بولتزمن - Restrictured Boltzmann Machines و تعدادی از شبکه‌های عصبی دیگر

2. پردازش تصویر و سیگنال

2.1. Accord.Imaging	شامل آشکارسازهای نقاط از جمله Harris, SURF, FAST و FREAK فیلترهایی برای تصاویر توابعی جهت انطباق (matching) و دوخت (stitching) تصاویر استخراج ویژگی‌های خوبی مانند - هیستوگرام گرادیان‌های شیب‌گرا و یا هاگ (Histograms of Oriented Gradients) و ویژگی‌های توصیفی بافتی هارلیک (Haralick’s textural)
2.2. Accord.Vision	تشخیص و ردیابی بی‌درنگ چهره توابعی برای تشخیص، ردیابی و تبدیل اشیایی که در جریانی(streams) از تصاویر هستند
2.3. Accord.Audio	شامل توابعی جهت پردازش صدا از جمله اسپکتروم آنالایزر

3. سایر کتابخانه‌های پشتیبانی

3.1. Accord.Controls	شامل نمودار هیستوگرام، پلات‌ها و نمایشگرها و نمودارهایی برای داده‌های جدولی جهت کاربردهای علمی.
3.2. Accord.Controls.Imaging	شامل ابزاری برای نمایش سریع تصاویر برای برنامه‌های Windows Forms
3.3. Accord.Controls.Audio	شامل کنترل‌های Windows Forms برای نمایش شکل موج صوت و اطلاعات آن
3.4. Accord.Controls.Vision	شامل اجزاء و کنترل‌های Windows Forms برای ردیابی حرکات سر، صورت، دست و سایر کارهای مرتبط با بینایی ماشین

اگر با مفاهیم یادگیری ماشین و هوش موصنوعی کمتر آشنا هستید و در این قسمت کمی کلمات تخصصی به کار رفته نگران نباشید؛ در مطالب آتی به صورت کاربردی به استفاده‌ی از آنها خواهیم پرداخت.

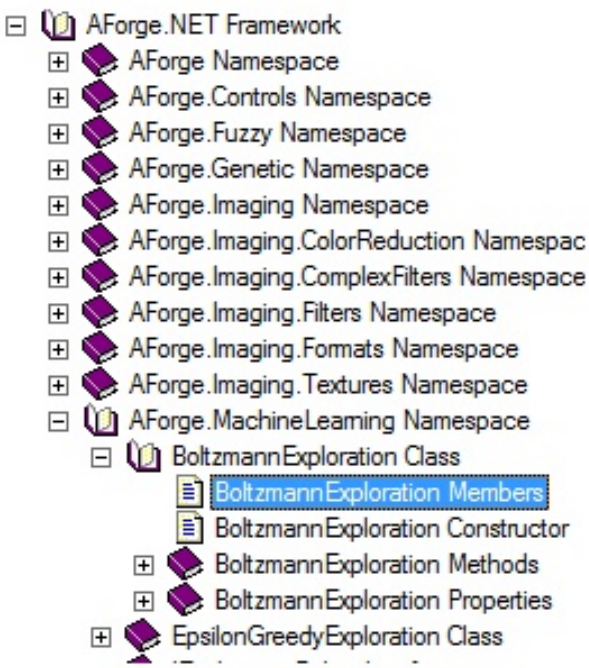
نظرات خوانندگان

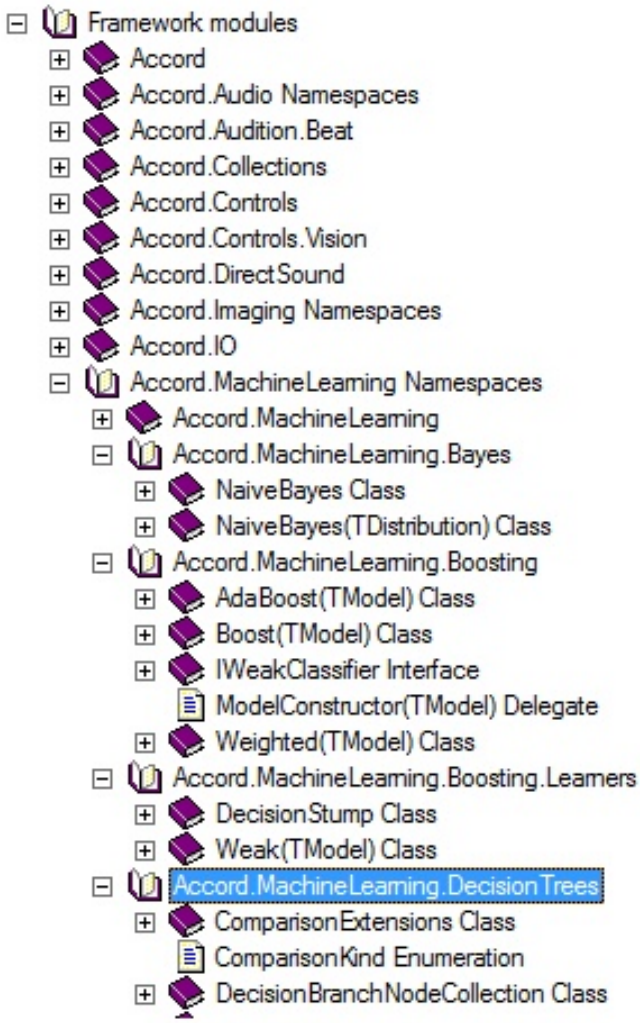
نویسنده: مصطفی عسگری
تاریخ: ۱۳۹۴/۰۵/۲۴ ۹:۵۰

مزایای این framework نسبت به AForge.NET چیست؟

نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۵/۲۴ ۱۷:۱۹

Accord.NET در حقیقت یک توسعه ای برای AForge.NET است. و چنانچه می‌خواهید از آکورد استفاده کنید بایستی ابتدا AForge.NET نصب نمایید.
AForge.NET یک کتابخانه بسیار عالی است اما در هر کدام از فضای نام هایش نقص هایی وجود دارد که در آکورد دات نت به آن افزوده شده است؛ به عنوان مثال در درختواره فضای نام MachineLearning مستندات دو پروژه مشاهده می‌کنیم که بسیاری از مفاهیم یادگیری ماشین از جمله : دسته بند نیو بیز، بوستینگ، بگینگ، درخت تصمیم، انواع مختلف اعتبارسنجی‌ها و ... در Accord.NET گنجانده شده است.





نویسنده: زواری

تاریخ: ۱۳۹۴/۰۵/۲۴ ۱۸:۱۶

همچنین یک توسعه دیگر بنام Accord.NET Extensions وجود دارد که توسط دکتر " دارکو یوریچ " برای آکورد نوشته شده است و ادعا می‌کند که با پیاده سازی "اساس شی تصویر" بعنوان آرایه محلی دات نت (مانند متلب)، سرعت پردازش‌ها بیشتر کرده است.

نویسنده: زواری
تاریخ: ۱۳۹۴/۰۵/۲۴ ۱۹:۰

اینو هم اضافه کنم که اگر نیاز هست با دات نت، پروژه پردازش تصویر بنویسیم؛ بهتر هست تا از فریم ورکهای فوق استفاده کنیم.
یک برنامه خیلی ابتدایی Emgu-V.S.-Aforge-V.S.-WICInterop.rar برای مقایسه سرعت بین "Emgu"، "AForge" و "WIC" نوشتم؛ به این ترتیب که یک تصویر خیلی بزرگ (حدود 10 مگابایت، تصویر یک نقشه) رو پویش میکنند. برای این پویش "aforg" دو ثانیه، "emgu" پنج و WIC بیست ثانیه زمان سپری شد.
درضمن ادعای برتری " Accord.NET Extensions Framework " رو هم بصورت مستند میتونید در لینک " Introducing Portable #Generic Image Library for C " مشاهده کنید.

عنوان: **Accord.NET #2**
نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۵/۲۶ ۰:۳۰
آدرس: www.dotnettips.info
گروه‌ها: Accord.NET, Machine Learning, Artificial Intelligence, SVM

در مطلب قبل با ساختار کلی کتابخانه Accord.NET آشنا شدیم. در این قسمت پس از فراگیری نحوه‌ی فراخوانی کتابخانه، به اجرای اولین برنامه‌ی کاربردی به کمک آکورد دات نت می‌پردازیم.

برای استفاده از Accord.NET می‌توان به یکی از دو صورت زیر اقدام کرد :
دریافت آخرین نسخه‌ی متن باز و یا dllهای پروژه‌ی Accord.NET از طریق گیت هاب

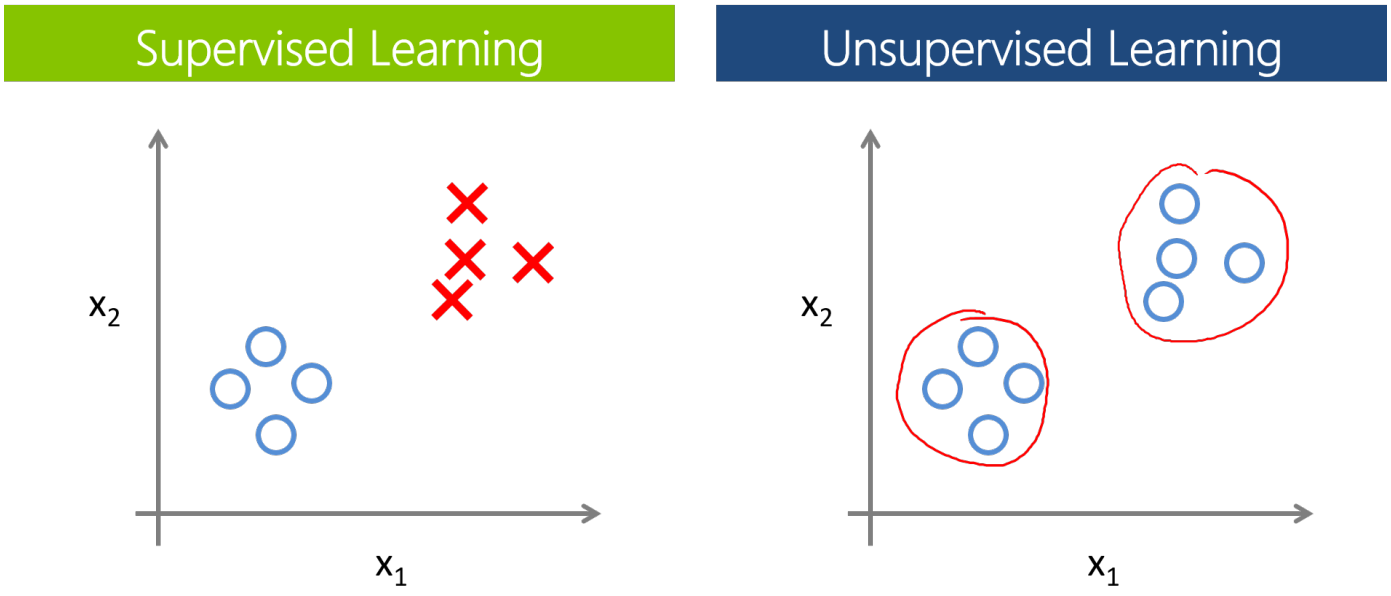
نصب از طریق NuGet (با توجه به این که در چارچوب Accord.NET کتابخانه‌های متنوعی وجود دارند و در هر پروژه نیاز به نصب همگی آنها نیست، فضای نام‌های مختلف در بسته‌های مختلف نیوگت قرار گرفته‌اند و برای نصب هر کدام می‌توانیم یکی از فرمان‌های زیر را استفاده کنیم)

```
PM> Install-Package Accord.MachineLearning
```

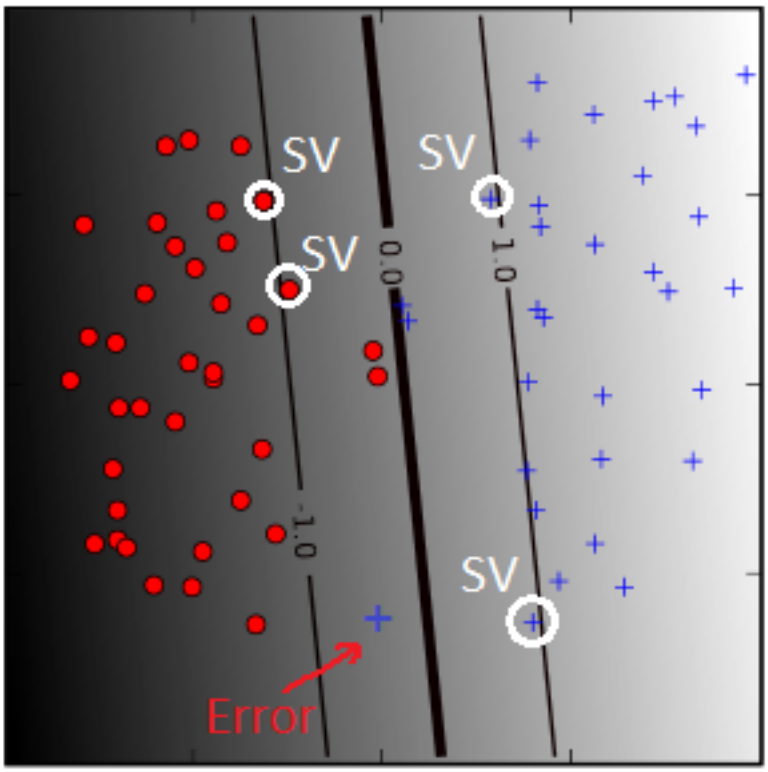
```
PM> Install-Package Accord.Imaging
```

```
PM> Install-Package Accord.Neuro
```

در اولین برنامه‌ی کاربردی خود می‌خواهیم الگوریتم ماشین بردار پشتیبان یا support vector machine را که یکی از روش‌های یادگیری **بانظارت** است برای **طبقه‌بندی** مورد استفاده قرار دهیم.
نکته : روش‌های یادگیری به دو دسته کلی با نظارت (Supervised learning) و بدون نظارت (Unsupervised learning) تقسیم بندی می‌شوند. در روش با نظارت، داده‌ها دارای برچسب یا label هستند و عملا نوع کلاس‌ها مشخص هستند و اصطلاحا برای طبقه بندی (Classification) استفاده می‌شوند. در روش بدون نظارت، داده‌هایمان بدون برچسب هستند و فقط تعداد کلاس ها و نیز یک معیار تفکیک پذیری مشخص است و برای خوشه بندی (Clustering) استفاده می‌شوند.

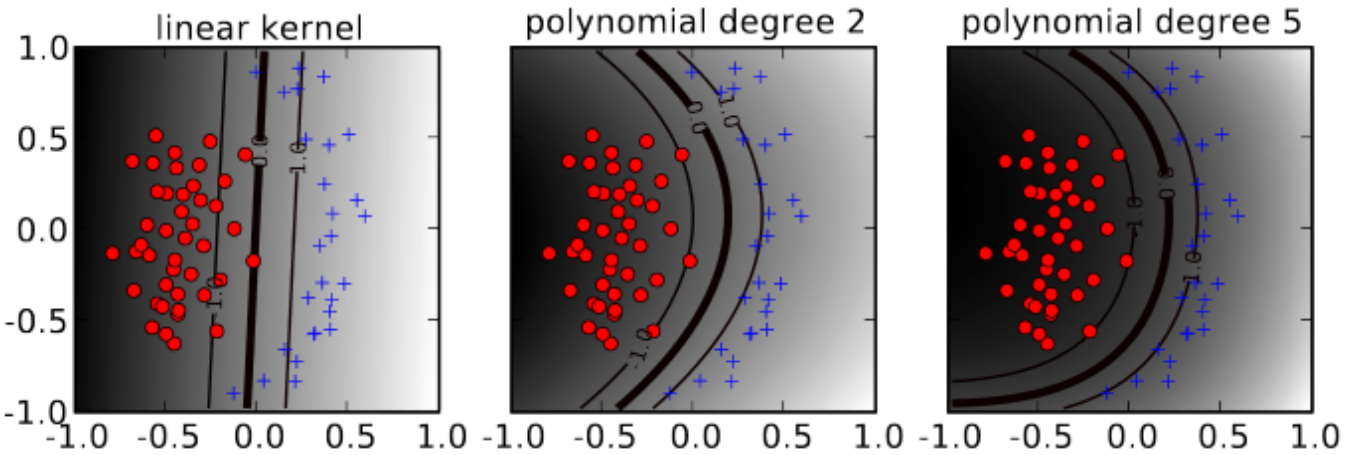


عملکرد SVM یا ماشین بردار پشتیبان به صورت خلاصه به این صورت است که با در نظر گرفتن یک خط یا ابرصفحه جدا کننده فرضی، ماشین یا دسته بندی را ایجاد می‌کند که از نقاط ابتدایی کلاس‌های مختلف که بردار پشتیبان یا SV نام دارند، بیشترین فاصله را دارند و در نهایت دادها را به دو کلاس مجزا تقسیم می‌کند.

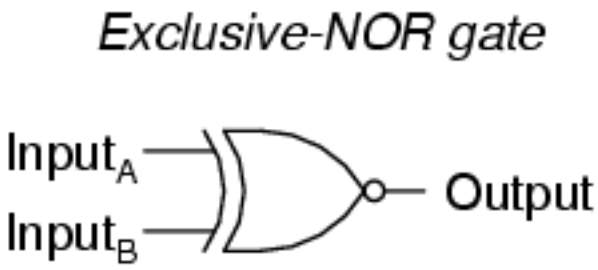


در تصویر بالا مقداری خطا مشاهده می‌شود که با توجه با خطی بودن جداساز مجبور به پذیرش این خطا هستیم.

در نسخه‌های جدیدتر این الگوریتم یک Kernel (از نوع خطی Linear ، چند جمله‌ای Polynomial، گوسین Gaussian و یا ...) برای آن در نظر گرفته شد که عملا نگاشتی را بین خط (نه صرفا فقط خطی) را با آن ابرصفحه جداکننده برقرار کند. در نتیجه دسته بندی با خطای کمتری را خواهیم داشت. (اطلاعات بیشتر در + و همچنین مطالب دکتر سعید شیری درباره SVM در +)



یک مثال مفهومی : هدف اصلی در این مثال شبیه سازی تابع XNOR به Kernel SVM می‌باشد.



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

برای شروع کار از فضای نام MachineLearning استفاده می‌کنیم و بسته‌ی نیوگت مربوطه را فرخوانی می‌کنیم. پس از اجرا، مشاهده می‌کنیم که فضای نام‌های Accord.Math و Accord.Statistics نیز به پروژه اضافه می‌شود.

```
PM> Install-Package Accord.MachineLearning
Attempting to resolve dependency 'Accord (≥ 3.0.2)'.
Attempting to resolve dependency 'Accord.Math (≥ 3.0.2)'.
Attempting to resolve dependency 'Accord.Statistics (≥ 3.0.2)'.
Installing 'Accord 3.0.2'.
```

در ابتدا مقادیر ورودی و برچسب‌ها را تعریف می‌کنیم

```
// ورودی
            double[][] inputs =
            {
                new double[] { 0, 0 }, // 0 xnor 0: 1 (label +1)
                new double[] { 0, 1 }, // 0 xnor 1: 0 (label -1)
                new double[] { 1, 0 }, // 1 xnor 0: 0 (label -1)
                new double[] { 1, 1 }  // 1 xnor 1: 1 (label +1)
            };

            // خروجی دسته بند ماشین بردار پشتیبان باید -1 یا +1 باشد
            int[] labels =
```

```
            {
                // 1,  0,  0, 1
                   1, -1, -1, 1
            };
```

پس از انتخاب نوع کرنل یا هسته، دسته‌بندمان را تعریف می‌کنیم :

```
// ساخت کرنل
            IKernel kernel = createKernel();

            // 2 ساخت دسته بند به کمک کرنل انتخابی و تنظیم تعداد ویژگی‌ها ورودی‌ها به مقدار
            KernelSupportVectorMachine machine = new KernelSupportVectorMachine(kernel, 2);
```

تابع ساخت کرنل :

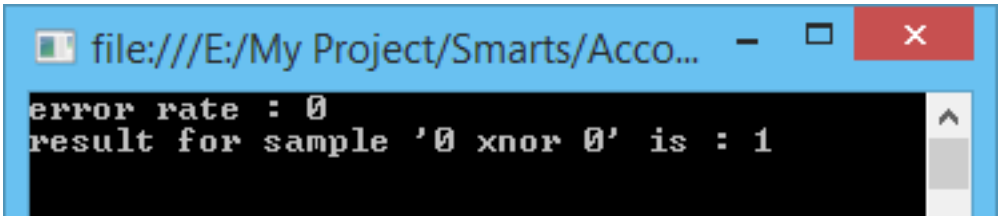
```
private static IKernel createKernel()
        {
            //var numPolyConstant = 1;
            //return new Linear(numPolyConstant);

            //var numDegree = 2;
            //var numPolyConstant = 1;
            //return new Polynomial(numDegree, numPolyConstant);

            //var numLaplacianSigma = 1000;
            //return new Laplacian(numLaplacianSigma);

            //var numSigAlpha = 7;
            //var numSigB = 6;
            //return new Sigmoid(numSigAlpha, numSigB);

            var numSigma = 0.1;
            return new Gaussian(numSigma);
        }
```

و سپس بایستی این Classifier را به یک الگوریتم یادگیری معرفی کنیم. الگوریتم بهینه سازی حداقلی ترتیبی (Sequential Minimal Optimization) یکی از از روش‌های یادگیری است که برای حل مسائل بزرگ درجه دوم بکار می‌رود و معمولا برای آموزش دسته بندی SVM از همین آموزنده استفاده می‌شود :

```
// معرفی دسته بندمان به الگوریتم یادگیری SMO
            SequentialMinimalOptimization teacher_smo = new SequentialMinimalOptimization(machine_svm,
inputs, labels);

            // اجرای الگوریتم یادگیری
            double error = teacher_smo.Run();
            Console.WriteLine(string.Format("error rate : {0}", error));
```

در نهایت می‌توانیم به عنوان نمونه برای آزمایش یکی از مقادیر ورودی را مورد بررسی قرار دهیم و خروجی کلاس را مشاهده کنیم.

```
// بررسی یکی از ورودی‌ها
            var sample = inputs[0];
            int decision = System.Math.Sign(machine_svm.Compute(sample));
            Console.WriteLine(string.Format("result for sample '0 xnor 0' is : {0}", decision));
```



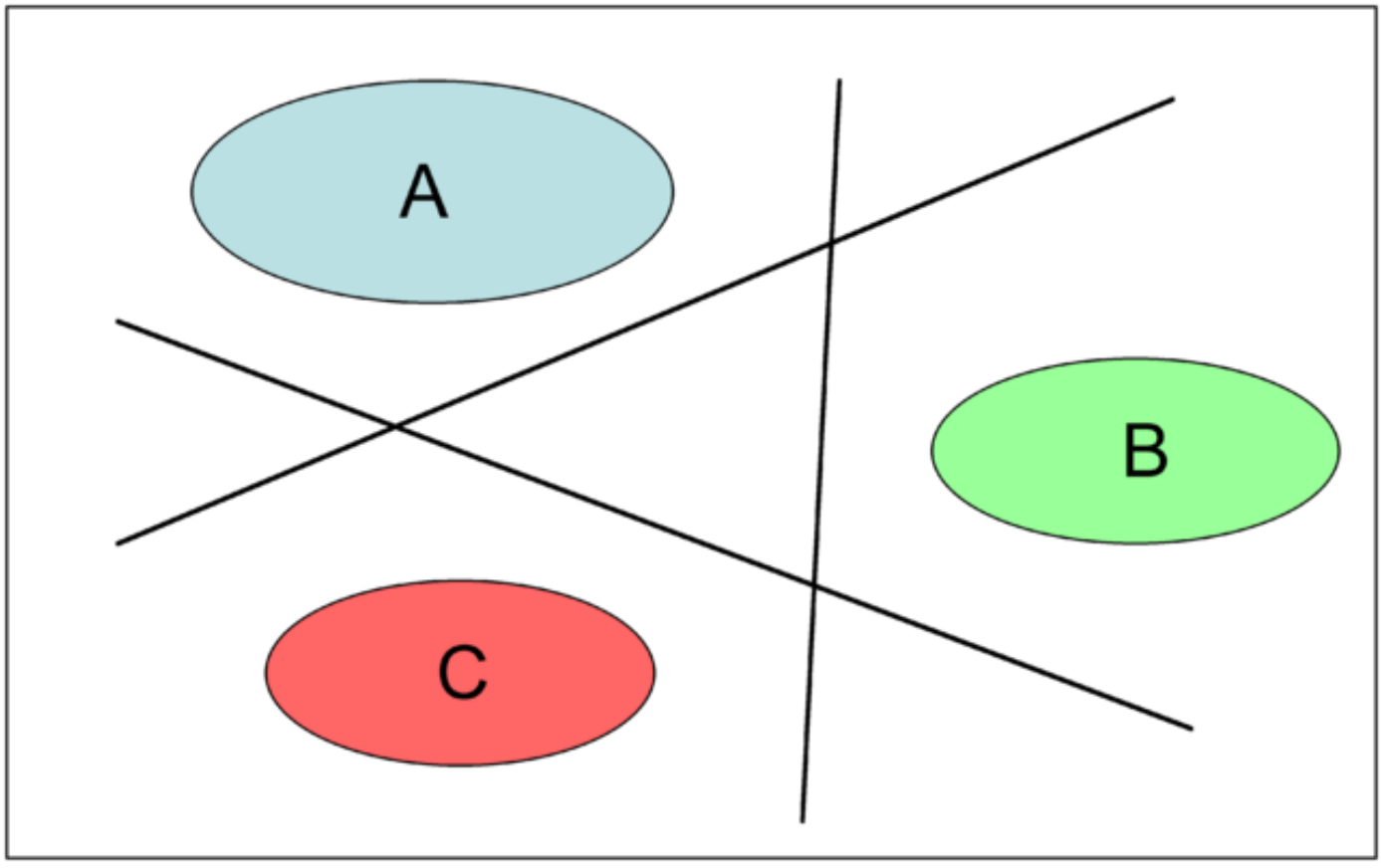
از این ساختار می‌توانیم برای طبقه بندی‌های با دو کلاس استفاده کنیم؛ مانند تشخیص جنسیت (مرد و زن) از طریق تصویر، تشخیص جنسیت (مرد و زن) از طریق صدا، تشخیص داشتن یا نداشتن یک بیماری خاص و ... . برای ایجاد هر کدام از این برنامه‌ها نیاز به یک مجموعه داده، استخراج ویژگی از آن و سپس نسبت دادن آن به الگوریتم داریم. در جلسات آینده با مفاهیم استخراج ویژگی و SVM چند کلاسه آشنا خواهیم شد.

دریافت کد

عنوان: **Accord.NET #3**
نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۶/۱۰ ۲۲:۵۰
آدرس: www.dotnettips.info
گروه‌ها: Accord.NET, Machine Learning, Artificial Intelligence

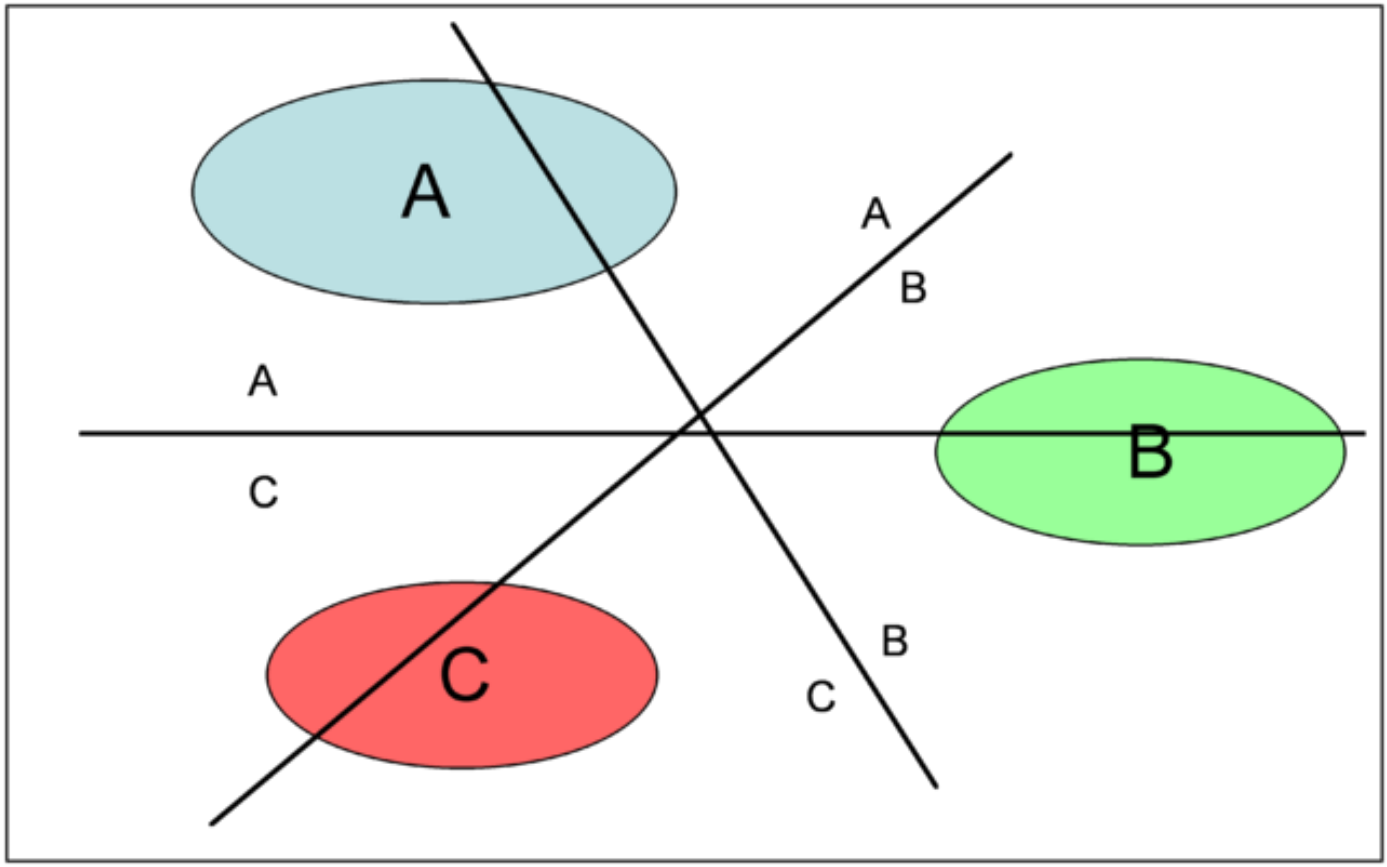
در مطلب قبل یک مثال مفهومی درباره کاربرد SVM بیان شد و دیدیم که این الگوریتم، یک روش دودویی است و عموما برای زمانی به کار می‌رود که مجموعه داده ما شامل دو کلاس باشد.
اگر بخواهیم نوع چهار میوه (سیب، گلابی، موز، پرتقال) را که از خط سورتینگ عبور می‌کنند، تشخیص دهیم و یا اینکه بخواهیم تشخیص اعداد دست نویس را داشته باشیم و یا اینکه حتی مطالب این وب سایت را که شامل چندین برچسب هستند، طبقه بندی کنیم، آیا در این تشخیص‌ها SVM به ما کمک می‌کند؟ پاسخ مثبت است.
در فضای نام یادگیری ماشین Accord.NET دو تابع خوب SupportVectorLearning **class** Multi و **label** Multi برای این گونه از مسائل تعبیه شده است. زمانیکه مسئله‌ی ما شامل مجموعه داده‌هایی بود که در چندین کلاس دسته بندی می‌شوند (مانند دسته بندی میوه، اعداد و ...) روش Multiclass و زمانیکه عناصر مجموعه داده ما به طور جداگانه شامل چندین برچسب باشند (مانند دسته بندی مطالب با داشتن چندین تگ، ...) روش Multilabel ابزار مفیدی خواهند بود. (+)

با توجه به **دودویی بودن ماشین بردار پشتیبان** ، دو استراتژی برای به کارگیری این الگوریتم برای دسته بندی‌های **چند کلاسه** وجود دارد:
روش یک در مقابل همه - One-against-all : در این روش عملا همان روش دودویی SVM را برای هر یک از کلاس‌ها به صورت جداگانه بررسی می‌کنیم. مثلا برای تشخیص میوه، یک بار دو کلاس سیب و غیر سیب (مابقی) بررسی می‌شوند و به همین ترتیب برای سایر کلاس‌ها و در مجموع صفحات ابرصفحه جدا کننده بین هر کلاس در مقابل سایر کلاس‌ها ایجاد می‌شود.



روش یک در مقابل یک - One-against-one (*) : در این روش هر کلاس، با هر یک از کلاس‌های دیگر به صورت تک تک بررسی

می‌شود و صفحات ابرصفحه جدا کننده مابین هر جفت کلاس متفاوت ایجاد می‌شود. (بیشتر در +)



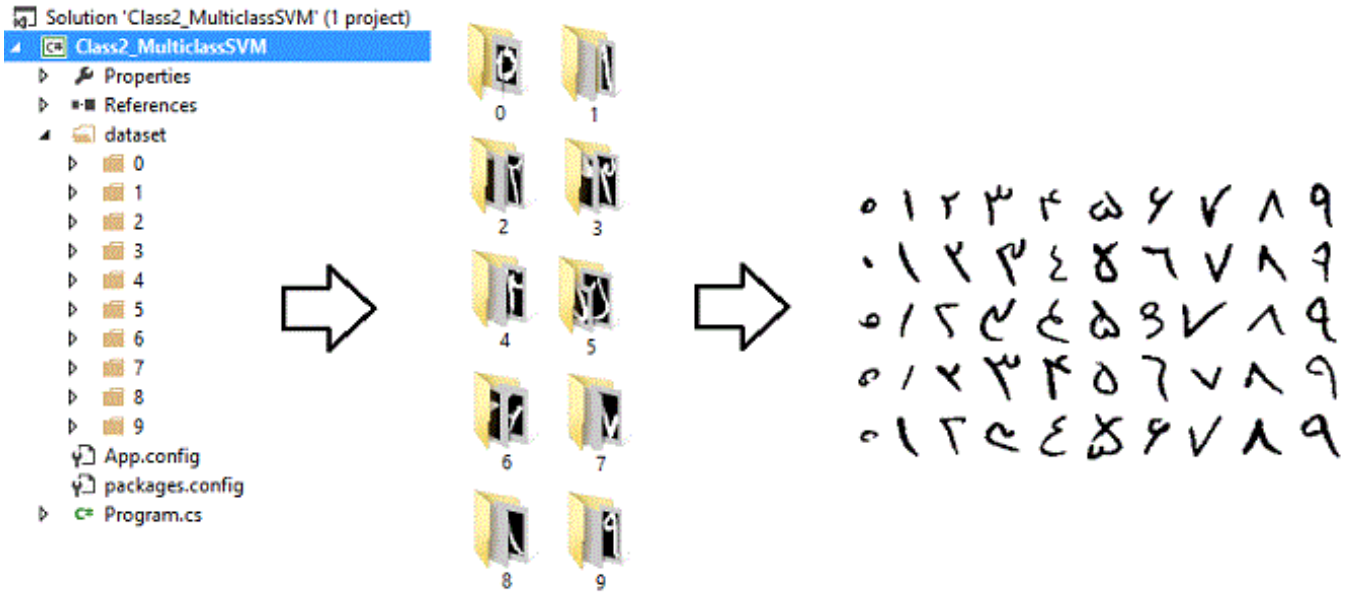
*روش "یک در مقابل یک" یا One-against-one اساس کار دسته بندی MulticlassSupportVectorMachine در فضای نام Accord.MachineLearning است.

یک مثال کاربردی : هدف در این مثال دسته بندی اعداد فارسی به کمک MulticlassSupportVectorMachine است.

به معرفی ابزار کار مورد نیاز می‌پردازیم.

1.مجموعه ارقام دستنویس هدی: مجموعه ارقام دستنویس هدی که اولین مجموعه‌ی بزرگ ارقام دستنویس فارسی است، مشتمل بر ۱۰۲۳۵۳ نمونه دستنوشته سیاه سفید است. این مجموعه طی انجام یک پروژه‌ی کارشناسی ارشد درباره بازشناسی فرم‌های دستنویس تهیه شده است. داده‌های این مجموعه از حدود ۱۲۰۰۰ فرم ثبت نام آزمون سراسری کارشناسی ارشد سال ۱۳۸۴ و آزمون کاردانی پیوسته‌ی دانشگاه جامع علمی کاربردی سال ۱۳۸۳ استخراج شده است. (اطلاعات بیشتر درباره مجموعه ارقام دستنویس هدی) .

تعداد 1000 نمونه (از هر عدد 100 نمونه) از این مجموعه داده، با فرمت bmp در این پروژه مورد استفاده قرار گرفته که به همراه پروژه در انتهای این مطلب قابل دریافت است.



2.استخراج ویژگی (Feature extraction) : در بازشناسی الگو و مفاهیم کلاس بندی، یکی از مهمترین گام‌ها، استخراج ویژگی است. ما موظف هستیم تا اطلاعات مناسبی را به عنوان ورودی برای دسته بندی‌مان معرفی نماییم. روش‌های مختلفی برای استخراج ویژگی وجود دارند. ویژگی‌ها به دو دسته‌ی کلی ویژگی‌های ظاهری (Appearance) و ویژگی‌های توصیف کننده (Descriptive) قابل تقسیم هستند. در تشخیص حروف و اعداد، ویژگی‌هایی مانند شدت نور نقاط (Intensity)، تعداد حلقه بسته، تعداد خطوط راست، تعداد دندانه، تعداد نقطه (برای حروف) و ... در دسته‌ی اول و ویژگی‌هایی مانند شیب خطوط، گرادیان، میزان افت یا شدت نور یک ناحیه، HOG و ... در دسته دوم قرار می‌گیرند. در این مطلب ما تنها از روش شدت نور نقاط برای استخراج ویژگی‌هایمان استفاده کرده‌ایم.

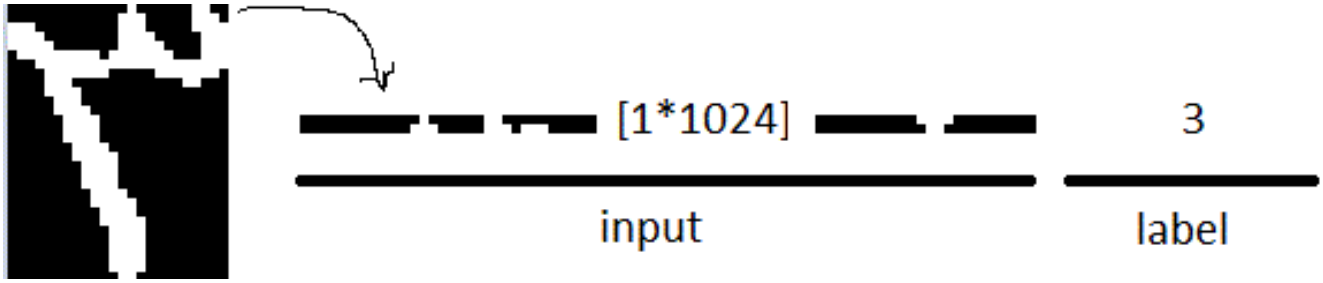
کد زیر با دریافت یک فایل Bitmap، ابتدا ابعاد را به اندازه 32*32 تغییر می‌دهد و سپس آن‌را به صورت یک بردار 1*1024 را بر می‌گرداند:

```
//تابع استخراج ویژگی
        private static double[] FeatureExtractor(Bitmap bitmap)
        {
            bitmap = BitmapResizer(bitmap, 32, 32);

            double[] features = new double[32 * 32];
            for (int i = 0; i < 32; i++)
                for (int j = 0; j < 32; j++)
                    features[i * 32 + j] = (bitmap.GetPixel(j, i).R == 255) ? 0 : 1;

            return features;
        }

        //تابع تغییر دهنده ابعاد عکس
        private static Bitmap BitmapResizer(Bitmap bitmap, int width, int height)
        {
            var newbitmap = new Bitmap(width, height);
            using (Graphics g = Graphics.FromImage((Image)newbitmap))
            {
                g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;
                g.DrawImage(bitmap, 0, 0, width, height);
            }
            return newbitmap;
        }
```

3.ایجاد ورودی‌ها و برچسب : در این مرحله ما باید ورودی‌های دسته بندی SVM را که عملا آرایه‌ای براساس تعداد نمونه‌های مجموعه آموزش (train) است، ایجاد نماییم.

ورودی‌ها (inputs) = با توجه به اینکه تعداد نمونه‌ها 50 مورد از هر عدد (مجموعا 500 نمونه) تعیین شده است و تعداد

ویژگی‌های هر نمونه یک بردار با طول 1024 است، ابعاد ماتریس ورودی مان [1024][500] می‌شود.
برچسب‌ها (labels) = تعداد برچسب مسلما به تعداد نمونه هایمان یعنی 500 مورد می‌باشد و مقادیر آن قاعدتا عدد متناظر آن تصویر است.



برای این کار از قطعه کد زیر استفاده می‌کنیم :

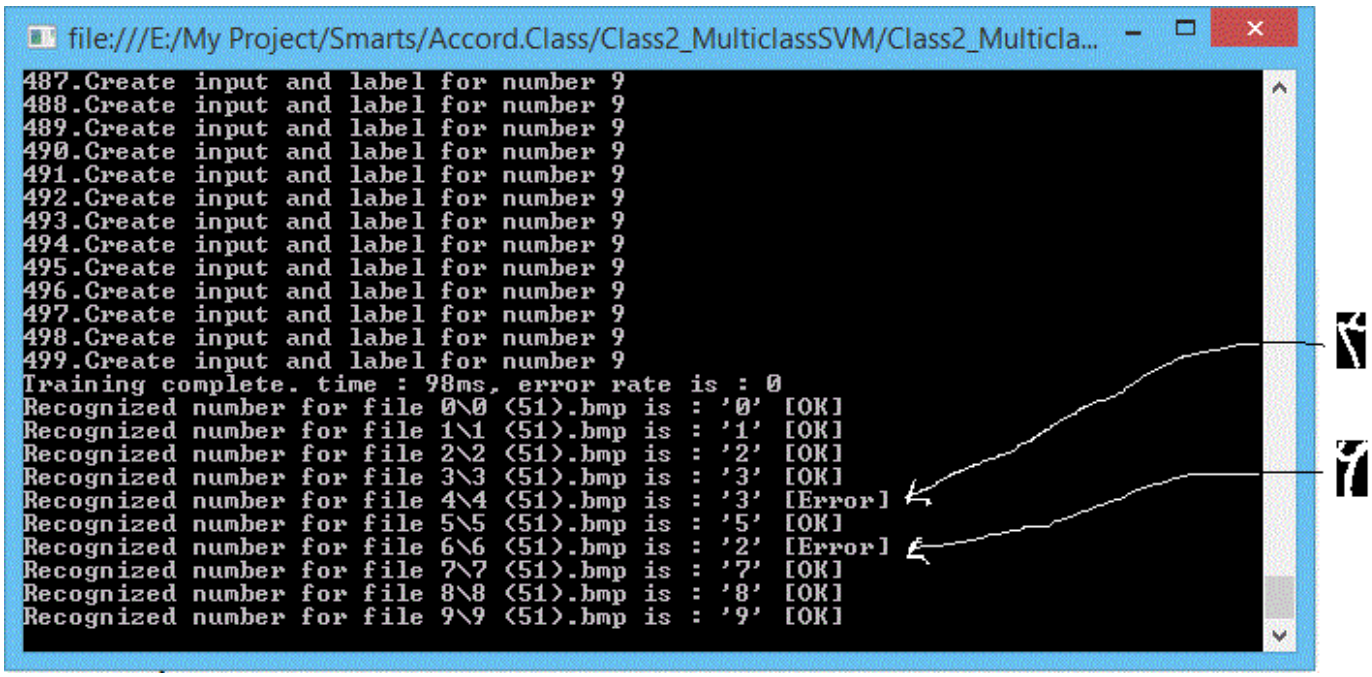
```
var path = new DirectoryInfo(Directory.GetCurrentDirectory()).Parent.Parent.FullName + @"\dataset\";

            // ایجاد ورودی و برچسب
            int trainingCount = 50;
            double[][] inputs = new double[trainingCount * 10][];
            int[] labels = new int[trainingCount * 10];

            var index = 0;
            var filename = "";
            Bitmap bitmap;
            double[] feature;

            for (int number = 0; number < 10; number++)
            {
                for (int j = 0; j < trainingCount; j++)
                {
                    index = (number * trainingCount) + j;
                    filename = string.Format(@"{0}\{0} ({1}).bmp", number, j + 1);
                    bitmap = new Bitmap(path + filename);

                    feature = FeatureExtractor(bitmap);

                    inputs[index] = feature;
                    labels[index] = number;

                    Console.WriteLine(string.Format("{0}.Create input and label for number {1}", index,
number));
                }
            }
```

4.در نهایت به دسته بندمان که همان MulticlassSupportVectorLearning است، خواهیم رسید. همانطور که در مطلب قبل مطرح شد، پس از تعریف پارامترهای Classifier مان، باید آن را به یک الگوریتم یادگیری که در اینجا هم همان روش SMO است، نسبت دهیم.

```
private static double MachineLearning(IKernel kernel, double[][] inputs, int[] labels)
        {
            machine_svm = new MulticlassSupportVectorMachine(1024, kernel, 10);

            // معرفی دسته بندمان به الگوریتم یادگیری SMO
            MulticlassSupportVectorLearning ml = new MulticlassSupportVectorLearning(machine_svm,
inputs, labels)
            {
                Algorithm = (svm, classInputs, classOutputs, i, j) =>
                    new SequentialMinimalOptimization(svm, classInputs, classOutputs)
            };

            var error = ml.Run();
            return error;
        }
```

می‌توانیم پس از اینکه ماشین دسته بندمان آماده شد، برای آزمایش تعدادی از نمونه‌های جدید و دیده نشده (UnSeen) را که در نمونه‌های آموزشی وجود نداشتند، مورد ارزیابی قرار دهیم. برای این کار اعداد 0 تا 9 از مجموعه داده مان را در نظر می‌گیریم و به وسیله کد زیر نتایج را مشاهده می‌کنیم :

```
// بررسی یک دسته از ورودی‌ها
            index = 51;
            for (int number = 0; number < 10; number++)
            {
                filename = string.Format(@"{0}\{0} ({1}).bmp", number, index);
                bitmap = new Bitmap(path + filename);

                feature = FeatureExtractor(bitmap);

                double[] responses;
                int recognizednumber = machine_svm.Compute(feature, out responses);

                Console.WriteLine
                (
                    String.Format
                    (
                        "Recognized number for file {0} is : '{1}' [{2}]",
                        filename,
                        recognizednumber,
                        (recognizednumber == number ? "OK" : "Error")
                    )
                );
                if (!machine_svm.IsProbabilistic)
                {
                    // Normalize responses
                    double max = responses.Max();
                    double min = responses.Min();

                    responses = Accord.Math.Tools.Scale(min, max, 0, 1, responses);
                    //int minIndex = Array.IndexOf(responses, 0);
                }
            }
```



مشاهده می‌شود که تنها بازشناسی تصاویر اعداد 4 و 6، به اشتباه انجام شده است که جای نگرانی نیست و می‌توان با افزایش تعداد نمونه‌های آموزشی و یا تغییرات پارامترها از جمله نوع کرنل و یا الگوریتم آموزنده این خطاها را نیز بر طرف کرد.

همانطور که دیدیم SVM گزینه‌ی بسیار مناسبی برای طبقه بندی خیلی از مسائل دو کلاسه و یا حتی چند کلاسه است. اما آکورد

دات نت Classifierهای خوب دیگری (مانند Naive Bayes و Decision Trees یا درخت تصمیم و ...) را نیز در چارچوب خود جای داده که در مطالب آینده معرفی خواهند شد.

دریافت پروژه

نظرات خوانندگان

نویسنده: زواری
تاریخ: ۱۳۹۴/۰۶/۱۱ ۲۱:۲۲

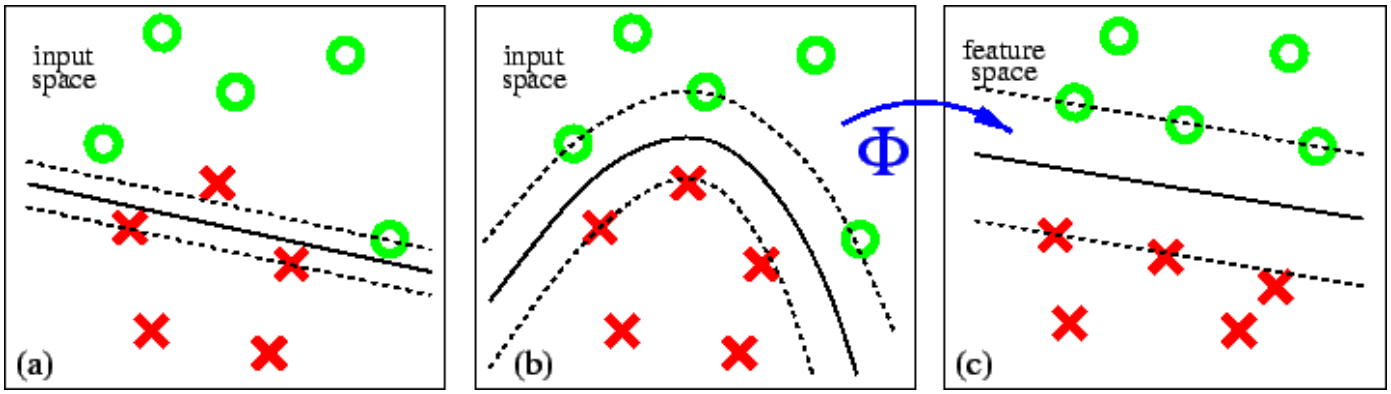
ممنون از مطالب مفیدتون.
اگه ممکنه در مورد کرنل و الگوریتم یادگیری کمی بیشتر توضیح بدید.
برای تغییر اندازه و استخراج ویژگی هم فکر میکنم بشه از توابع زیر استفاده کرد که شاید کاراتر باشن:

```
AForge.Imaging.Filters.ResizeBicubic
AForge.Imaging.Filters.ResizeBilinear
AForge.Imaging.Filters.ResizeNearestNeighbor

Accord.Imaging.Converters.ArrayToImage
Accord.Imaging.Converters.ImageToArray
Accord.Imaging.Converters.ImageToMatrix
Accord.Imaging.Converters.MatrixToImage
```

بازم تشکر و سپاس، درپناه حق موفق باشید.

نویسنده: محسن نجف زاده
تاریخ: ۱۳۹۴/۰۶/۱۷ ۲۱:۲۹

- در مورد کرنل همانطور که در مطلب قبل هم صحبت شد، می‌توان گفت که Kernel عملا نگاشتی را بین خط تفکیک کننده نمونه‌های کلاس‌ها با ابرصفحه جداکننده برقرار می‌کند و با این شرایط می‌توان SVM را به نوعی غیر خطی در نظر گرفت. مثلا در تصویر زیر با پارامتر ، فضای اولیه داده‌های ما را به فضای ویژگی هایی نگاشت می‌شود که می‌توان با همان SVM خطی دسته بندی کرد (+)



- اگر منظورتان از الگوریتم یادگیری روش Sequential Minimal Optimization است می‌توان گفت SMO یکی از روش‌های متداول و سریع برای آموزش SVM به حساب می‌آید که عملا یک مسئله بهینه سازی است که به دنبال بهترین ضرایب همان کرنل است (+)

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j K(x_i, x_j) \alpha_i \alpha_j,$$

- درباره پیشنهادتون در خصوص استفاده از فضای نام Accord.Imaging هم ضمن تشکر، می‌توان گفت که فعلا قصد ورود به فضای نام ی از آکورد دات نت به جز Accord.MachineLearning نداریم ولی در آینده حتما معرفی و استفاده خواهند شد.