Notes on understanding classifiers

[work in progress]

Tomasz Golan

February 21, 2016

Contents

Abstract

This note is related to Simple Classifier project (https://github.com/TomaszGolan/simpleClassifiers) which was an exercise to understand two classifiers: k-Nearest Neighbors (kNN) and Support Vector Machine (SVM) . Two classes of 2D points are considered within the project: *separable* (below or above $f(x) = x$)) and *inseparable* (inside or outside circle). The purpose of this note is to explain technical details of both algorithms and demonstrate how they work on simple examples. Please note, I am not an expert in the field and the note is rather how I understand the problem.

1 Support Vector Machine

1.1 Sequential Minimal Optimization

1.1.1 Summary of linear SVM problem

The hyperplane, given by:

$$z = \omega_0 + \langle\vec{\omega}, \vec{x}\rangle \tag{1}$$

separates classes for $z = 0$. The nearest points lie on $z = \pm 1$. The normal vector can be calculated from:

$$\vec{\omega} = \sum_{i=1}^{N} \lambda_i y_i \vec{x}_i \tag{2}$$

where N is the number of learning samples, $\vec{x}_i$ is i-th feature vector and y_i - corresponding class membership ($\{-1, 1\}$). λ_i are Lagrange (KKT) multipliers, which can be obtained from Eq. ??. Free parameter ω_0 can be calculated from (for any support vector):

$$\omega_0 = y_i - \langle \vec{\omega}, \vec{x}_i \rangle \tag{3}$$

Using Eq. 2, the hyperplane equation can be rewritten in the following form:

$$z(\vec{x}) = \sum_{i=1}^{N} \lambda_i y_i \langle \vec{x}_i, \vec{x} \rangle + \omega_0 \tag{4}$$

Each training sample must fulfill the KKT conditions, in this case given by:

$$\begin{aligned} \lambda_i = 0 \quad &\Leftrightarrow \quad y_i z_i \geq 1 &(5)\\ 0 < \lambda_i < C \quad &\Leftrightarrow \quad y_i z_i = 1 &(6)\\ \lambda_i = C \quad &\Leftrightarrow \quad y_i z_i \leq 1 &(7) \end{aligned}$$

where $z_i = z(\vec{x}_i)$ is the output for i-th training sample. Lets understand why. From complementary slackness condition (Eq. ??):

$$\begin{aligned} \lambda_i(y_i z_i - 1 + \xi_i) &= 0 &(8)\\ \mu_i \xi_i &= 0 &(9) \end{aligned}$$

If $\lambda_i = 0$, then $\mu_i = C$ (from Eq. ??), so $\xi_i = 0$ and the constraint (Eq. ??) becomes:

$$y_i z_i - 1 + \xi_i \geq 0 \Rightarrow y_i z_i - 1 \geq 0 \tag{10}$$

If $\lambda_i > 0$, then $y_i z_i - 1 + \xi_i = 0$. If $0 < \lambda < C$, then $\mu_i > 0$, so $\xi_i = 0$ and $y_i z_i - 1 = 0$. If $\lambda = C$, then $\mu_i = 0$, so $\xi_i \geq 0$ and $y_i z_i - 1 \leq 0$.

1.1.2 SMO algorithm

Sequential Minimal Optimization (SMO) algorithm was developed by John C. Platt[1]. The method solves the smallest possible optimization problem at a time. In this case, it means updating two Lagrange multipliers in a step. Why two? To preserve a linear equality constraint ($\sum\limits_{i=1}^{N} \lambda_i y_i = 0$). It guarantees convergence through Osuna's theorem[2], which states that the global training problem can be broken down into a sequence of smaller subproblems.

Updating two Lagrange multipliers

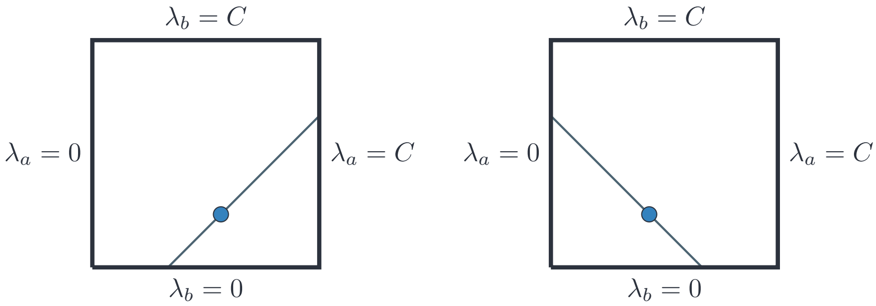
Lets assume at least one of Lagrange multipliers (λ_a, λ_b) violates KKT conditions. Both multipliers must be updated to preserve $\gamma = y_a \lambda_a + y_b \lambda_b$. Also, both are restricted by $0 < \lambda_i < C$ constraint. It is illustrated on Fig. 1.

As only λ_a and λ_b are going to be changed, lets extract them from sums in the Lagrangian (for convenience $k_{ij} \equiv K(\vec{x}_i, \vec{x}_j) = \langle \vec{x}_i, \vec{x}_j \rangle$ is introduced)[3]:

[1] *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, J.C. Platt, Microsoft Research, Technical Report MSR-TR-98-14.

[2] *An improved training algorithm for support vector machines*, E. Osuna et al., In Proc. of IEEE NNSP'97, 1997

[3] Note, that k symmetry ($k_{ij} = k_{ji}$) is used.



(a) $y_a \neq y_b \Rightarrow \lambda_a - \lambda_b = \gamma$ (b) $y_a = y_b \Rightarrow \lambda_a + \lambda_b = \gamma$

Figure 1: Possible relations for two Lagrange multipliers. Picture "cloned" from *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*, J.C. Platt, Microsoft Research, Technical Report MSR-TR-98-14.

$$\begin{aligned}
\mathcal{L}(\lambda) &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j k_{ij} + \sum_{i=1}^{N}\lambda_i \qquad (11)\\
&= \lambda_a + \lambda_b + \sum_{\substack{i=1\\ i\neq a,b}}^{N}\lambda_i - \frac{1}{2}\lambda_a^2 k_{aa} - \frac{1}{2}\lambda_b^2 k_{bb} - \lambda_a\lambda_b y_a y_b k_{ab}\\
&- \lambda_a y_a \sum_{\substack{i=1\\ i\neq a,b}}^{N}\lambda_i y_i k_{ia} - \lambda_b y_b \sum_{\substack{i=1\\ i\neq a,b}}^{N}\lambda_i y_i k_{ib} - \frac{1}{2}\sum_{\substack{i=1\\ i\neq a,b}}^{N}\sum_{\substack{j=1\\ j\neq a,b}}^{N}\lambda_i\lambda_j y_i y_j k_{ij}
\end{aligned}$$

For convenience lets introduce:

$$\begin{aligned}
\tilde{\omega}_j &= \sum_{\substack{i=1\\ i\neq a,b}}^{N}\lambda_i y_i k_{ij} \qquad (12)\\
&= \sum_{i=1}^{N}\lambda_i y_i k_{ij} + \omega_0 - \lambda_a y_a k_{aj} - \lambda_b y_b k_{bj} - \omega_0\\
&= z_j - \omega_0 - \lambda_a y_a k_{aj} - \lambda_b y_b k_{bj}
\end{aligned}$$

so the Lagrangian can be rewritten as:

$$\mathcal{L}(\lambda_a,\lambda_b) = \lambda_a + \lambda_b - \frac{1}{2}\lambda_a^2 k_{aa} - \frac{1}{2}\lambda_b^2 k_{bb} - \lambda_a\lambda_b y_a y_b k_{ab} - \lambda_a y_a\tilde{\omega}_a - \lambda_b y_b\tilde{\omega}_b + const \qquad (13)$$

where *const* does not depend on $\lambda_{\{a,b\}}$. As mentioned before, there is an linear dependence between two Lagrange multipliers: $\lambda_a = \gamma - s\lambda_b$, where $s = y_a y_b$. Therefore, the Lagrange can be expressed only by one multiplier (note, $s^2 = 1$ and $sy_a = y_b$):

$$
\begin{aligned}
\mathcal{L}(\lambda_b) &= \gamma - s\lambda_b + \lambda_b - \frac{1}{2}\left(\gamma - s\lambda_b\right)^2 k_{aa} - \frac{1}{2}\lambda_b^2 k_{bb} \\
&- s\left(\gamma - s\lambda_b\right)\lambda_b k_{ab} - \left(\gamma - s\lambda_b\right) y_a \tilde{\omega}_a - \lambda_b y_b \tilde{\omega}_b + const \\
&= \gamma - s\lambda_b + \lambda_b - \frac{1}{2}\gamma^2 k_{aa} - \frac{1}{2}s^2\lambda_b^2 k_{aa} + s\gamma\lambda_b k_{aa} - \frac{1}{2}\lambda_b^2 k_{bb} \\
&- s\gamma\lambda_b k_{ab} + s^2\lambda_b^2 k_{ab} - \gamma y_a \tilde{\omega}_a + s\lambda_b y_a \tilde{\omega}_a - \lambda_b y_b \tilde{\omega}_b + const \\
&= \frac{1}{2}\lambda_b^2\left(2k_{ab} - k_{aa} - k_{bb}\right) + \lambda_b\left[1 - s + s\gamma(k_{aa} - k_{ab}) + y_b(\tilde{\omega}_a - \tilde{\omega}_b)\right] + const
\end{aligned} \tag{14}
$$

where last *const* are terms without λ_b. As the goal is to maximize $\mathcal{L}$, lets check derivatives

$$
\begin{aligned}
\frac{\partial \mathcal{L}(\lambda_b)}{\partial \lambda_b} &= \lambda_b\left(2k_{ab} - k_{aa} - k_{bb}\right) + \left[1 - s + s\gamma(k_{aa} - k_{ab}) + y_b(\tilde{\omega}_a - \tilde{\omega}_b)\right] \\
\frac{\partial^2 \mathcal{L}(\lambda_b)}{\partial \lambda_b^2} &= \left(2k_{ab} - k_{aa} - k_{bb}\right)
\end{aligned} \tag{15}
$$

$\frac{\partial^2 \mathcal{L}(\lambda_b)}{\partial \lambda_b^2} < 0$ is required for maximum, which is fulfilled by inner product (or most kernels) until $\vec{x}_a \neq \vec{x}_b$. One must be careful when two training samples are the same as second derivative is zero then. New λ_b is obtained from $\frac{\partial \mathcal{L}(\lambda_b)}{\partial \lambda_b} = 0$:

$$
\lambda_b^{new} = \frac{s - 1 - s\gamma(k_{aa} - k_{ab}) - y_b(\tilde{\omega}_a - \tilde{\omega}_b)}{2k_{ab} - k_{aa} - k_{bb}} \tag{16}
$$

where

$$
\begin{aligned}
\tilde{\omega}_a - \tilde{\omega}_b &= z_a - \omega_0 - \lambda_a y_a k_{aa} - \lambda_b y_b k_{ab} + z_b + \omega_0 + \lambda_a y_a k_{ab} + \lambda_b y_b k_{bb} \\
&= z_a + z_b - \left(\gamma - s\lambda_b\right) y_a k_{aa} - \lambda_b y_b k_{ab} + \left(\gamma - s\lambda_b\right) y_a k_{ab} + \lambda_b y_b k_{bb} \\
&= z_a + z_b - \gamma y_a k_{aa} + \gamma y_a k_{ab} + y_b\lambda_b\left(k_{aa} + k_{bb} - 2k_{ab}\right)
\end{aligned} \tag{17}
$$