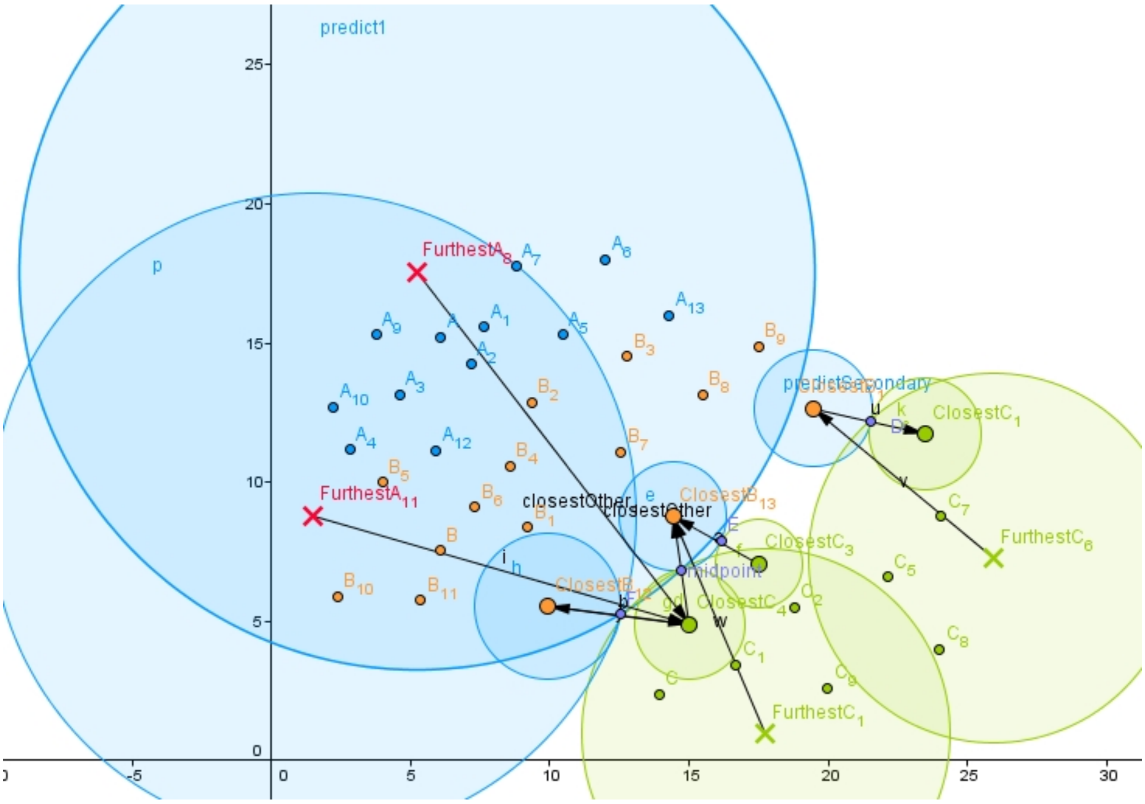
আরও ব্লগ তৈরি করুন প্রবেশ করুন



It is a simple guide or introduction to Biotechnology and Machine Learning with background research, Support Vector Machine explanation, its interpretation in Java called Least Similar Spheres, its implementation and results. Author hopes that this project could be useful to those who want to introduce themselves to bioinformatics, machine learning, SVM and how it could be interpreted.

Math of SVM from 2D to (n)D

2D SVM and line equations

SVM starts as 2D idea with math behind it looking like this.

Line equation:	Line or hyperplane can also be described as
$y = ax + b$	$w \cdot x + b = 0$
Or	This hyperplane is the center hyperplane.
$ax - y + b = 0$	To find its best orientation you need 2 more parallel hyperplanes with slightly different equalities.
Imagine it is made from two vectors	Positive class:
$w(a, 1, b)$ and $x(x, -y, 1)$	$w \cdot x + b = 1$
Then it can be described as dot product	Negative class:
$w \cdot x = 0$	$w \cdot x + b = -1$
Modified for easy use hyperplane function for Lagrange multiplier:	Function defining constraints of the hyperplane, its shift or distance from origin or zero, value b, associated class labels y_i and support vectors x_i
$f(x) = \frac{1}{2}\|w\|^2$	$g(x) = y_i(w^T \cdot x_i + b) - 1$

(n)D SVM and Lagrange multiplier

Then higher dimensional data and linearly inseparable problems are taken in to consideration, which starts using multidimensional variables with several constraints and leads to maximization ideas from Lagrange and there the line equation is fit in to Lagrange formula.

About Me

Gintaras

View my complete profile

Blog Archive

2016 (19)
June (19)
News, Comments and Updates
SVM diagrams (SVM 2D)
Multi Class Solution
Linearly Inseparable and Slack Variable
Linearly Inseparable and Higher Dimensions
Math of SVM from 2D to (n)D
SVM interpretation in Java and start of Least Simi...
Least Similar Spheres (LSS)
Abstract of the Research about Biotechnology and M...
The Introduction to Biotechnology and Machine Lear...
Literature Review (Machine Learning in early and l...
Literature Review (Machine Learning examples taken...
Literature Review (Machine Learning

Lagrange multiplier formula.	Multi-constraint
$L(x,\lambda) = f(x) - \lambda g(x)$	$L(x,\lambda) = f(x) - \sum \lambda g(x)$

Plugged in formulas.
$L(w,b,\lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \lambda_i (y_i(w^T \cdot x_i + b) - 1)$

Further this formula is gradually simplified to. Where K is a kernel or a function to which you pass vectors and measure their distances in higher/different dimensions.

SVM formula with constraints

$\sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i x_j$	$\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j$
Dot product distance or similarity measurement formula.	Kernel looks version
$\sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j f(x_i x_j)$	$\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i x_j)$
Constraints from derivatives	
$w = \sum_{i=1}^{n} \alpha_i y_i x_i$ $\sum_{i=1}^{n} \alpha_i y_i = 0$	

SVM Kernels

Some of the Kernels (Author may update this later with actual formulas). You could look at these and see that they are just a way to differently measure the distances between vectors, instances or rows. More about it in SVM interpretation section.

- Linear or dot product.
- Gaussian, which uses euclidean distance.
- RBF, which uses Gaussian.
- Exponential, which looks like simpler version of Gaussian.
- Polynomial, which uses probabilities and dot product.
- Hybrid, which is mix of polynomial and gaussian.
- Sigmoidal, step function used in neural networks.

Possible transformations between Lagrange SVM and final formula

Some possible transformations in between Lagrange and final SVM formula may look like this.

Produce partial derivatives to simplify formula:

With respect to w

$$L(w,b,\lambda)=\sum_{i=1}^{n}\lambda_i+\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{n}\lambda_i y_i(\mathbf{w}^T\cdot\mathbf{x}_i+\mathbf{b})$$

$$\frac{\partial L_P}{\partial w}=w-\sum_{i=1}^{n}\lambda_i y_i x_i$$

$$\mathbf{w}=\sum_{i=1}^{n}\lambda_i y_i x_i$$

With respect to b

$$L(w,b,\lambda)=\sum_{i=1}^{n}\lambda_i+\frac{1}{2}\|w\|^2-\sum_{i=1}^{n}\lambda_i y_i(\mathbf{w}^T\cdot\mathbf{x}_i+\mathbf{b})$$

$$\frac{\partial L_P}{\partial b}=-\sum_{i=1}^{n}\lambda_i y_i$$

$$\mathbf{b}=\sum_{i=1}^{n}\lambda_i y_i$$

Plug the derivatives back in

$$L(w,b,\lambda)=\sum_{i=1}^{n}\lambda_i+\frac{1}{2}\|\mathbf{w}\|^2-\sum_{i=1}^{n}\lambda_i y_i(w^T\cdot x_i+\mathbf{b})$$

$$L(w,b,\lambda)=\sum_{i=1}^{n}\lambda_i+\frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)^2-\sum_{i=1}^{n}\lambda_i y_i\left(\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\cdot x_i+\sum_{i=1}^{n}\lambda_i y_i\right)$$

Possible further reconfigurations

$$\begin{aligned}L(w,b,\lambda)=&\sum_{i=1}^{n}\lambda_i+\frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\\&-\left(\sum_{i=1}^{n}\lambda_i y_i\right)\left(\left(\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\cdot x_i\right)+\sum_{i=1}^{n}\lambda_i y_i\right)\end{aligned}$$

$$\begin{aligned}L(w,b,\lambda)=&\sum_{i=1}^{n}\lambda_i+\frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)-\left(\sum_{i=1}^{n}\lambda_i y_i\right)\left(\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\cdot x_i\right)\\&+\left(\sum_{i=1}^{n}\lambda_i y_i\right)\sum_{i=1}^{n}\lambda_i y_i\end{aligned}$$

$$\begin{aligned}L(w,b,\lambda)=&\sum_{i=1}^{n}\lambda_i+\left(\frac{1}{2}\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)-\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\right)\\&+\sum_{i=1}^{n}\lambda_i y_i\sum_{i=1}^{n}\lambda_i y_i\end{aligned}$$

$$L(w,b,\lambda)=\sum_{i=1}^{n}\lambda_i-\frac{1}{2}\left(\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\left(\sum_{i=1}^{n}\lambda_i y_i x_i\right)\right)+\left(\sum_{i=1}^{n}\lambda_i y_i\right)\left(\sum_{i=1}^{n}\lambda_i y_i\right)$$

Kernels in Java

Java code for the Kernel functions. (More of it in further posts)