Support Vector Machines

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

NetDB-ML, Spring 2014

Outline

Outline

Why Sparse Models?

- We have seen ***dense*** kernel machines 一般的Kernel Machine是dense
  - $h = \sum_{t=1}^{N} c_t k(\boldsymbol{x}^{(t)}, \cdot) = \sum_{t=1}^{N} c_t \Phi(\boldsymbol{x}^{(t)})$ where $c_t \neq 0$ for most t
- To make a prediction $h(\boldsymbol{x}') = \sum_{t=1}^{N} c_t k(\boldsymbol{x}^{(t)}, \boldsymbol{x}')$, we need to store ***all*** examples
- May be infeasible in practice due to
  - Big dataset (large N)
  - Time limit
  - Space limit
- Solution: make the kernel machines ***sparse***;
  - I.e., make $c_t \neq 0$ for only a small fraction of the examples called ***support vectors***
- How?

Creating Sparsity

如果r^1恆等於(任何g)$g(x^1)$，並且constriants也都滿足，
那麼$c^1=0$，因為x^1並不影響h

- Recall that a kernel machine $h = \sum_{t=1}^{N} c_t k(\boldsymbol{x}^{(t)}, \cdot)$ is the solution to

$$\arg\min_{g\in\mathcal{H}} L((\boldsymbol{x}^{(1)}, r^{(1)}, g(\boldsymbol{x}^{(1)})), \cdots, (\boldsymbol{x}^{(N)}, r^{(N)}, g(\boldsymbol{x}^{(N)}))) + \Omega(\|g\|_{\mathcal{RKHS}})$$
$$\text{subject to } C_k((\boldsymbol{x}^{(1)}, r^{(1)}, g(\boldsymbol{x}^{(1)})), \cdots, (\boldsymbol{x}^{(N)}, r^{(N)}, g(\boldsymbol{x}^{(N)}))) \leqslant 0, \forall k$$

- In what situation, will we have $c_1 = 0$? If for different g, $g(\boldsymbol{x}^{(1)})$ neither changes the value of L nor violates the constrains $C_k(\cdots) \leqslant 0$, then minimizing the regularization term makes $c_1 = 0$
- Idea:
    - Suppose $L((\boldsymbol{x}^{(1)}, r^{(1)}, g(\boldsymbol{x}^{(1)})), \cdots, (\boldsymbol{x}^{(N)}, r^{(N)}, g(\boldsymbol{x}^{(N)}))) = \sum_{t=1}^{N} l(\boldsymbol{x}^{(t)}, r^{(t)}, g(\boldsymbol{x}^{(t)}))$, refine l such that $l(\boldsymbol{x}^{(t)}, r^{(t)}, g(\boldsymbol{x}^{(t)})) = 0$ for most t's no matter what the g is
    - Similarly, refine constrains (if any) such that they hold for most t's no matter what the g is

Outline

Support Vector Regression

ε : Hyper-parameter,誤差容忍

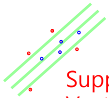
- Idea: to create an ϵ-***insensitive loss***:

error<ε → error=0
太小的error就當作沒error

$$\arg\min_{\boldsymbol{w},b,\xi^+,\xi^-} \sum_{t=1}^{N}(\xi_t^+ + \xi_t^-) + \lambda\|\boldsymbol{w}\|^2$$

subject to
$$(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-r^{(t)} \leqslant \epsilon+\xi_t^+$$
$$(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-r^{(t)} \geqslant -\epsilon-\xi_t^-$$
$$\xi_t^+ \geqslant 0,\ \xi_t^- \geqslant 0,\ \forall t=1,\cdots,N$$

ξ_t^+,ξ_t^+ : 離預測最遠的距離



  - λ and ϵ are hyperparameters

- No matter what the $\boldsymbol{w}$ and b are, most examples (those falling inside the "ϵ-tube") will have $\xi_t^+=0$, $\xi_t^-=0$, and the constrains always hold

「誤差免除」距離以內

- So, if the representation theorem applies, the solution will be sparse
  - Examples falling outside the ϵ-tube are support vectors
- Is the representation theorem applicable?

ξ_t^+,ξ_t^+ : 離預測太遠的資料會影響w的走向

Applying Representer Theorem

- Note that $\xi_t^+ \geqslant 0$ and $(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) - r^{(t)} \leqslant \epsilon + \xi_t^+$ implies
  $\xi_t^+ = \max(0, (\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) - (r^{(t)} + \epsilon))$ 預測值偏差太大→正數，否則0
  - Similarly, $\xi_t^- = \max(0, (r^{(t)} - \epsilon) - (\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b))$預測值偏差太小→正數，否則0
- We can eliminate $\boldsymbol{\xi}^+$ and $\boldsymbol{\xi}^-$ by rewriting the objective as

$$\arg\min_{\boldsymbol{w},b} \quad \sum_{t=1}^N (\max(0, (\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) - (r^{(t)} + \epsilon)) + \max(0, (r^{(t)} - \epsilon) - (\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b)) + \lambda \|\boldsymbol{w}\|^2$$

  - ***Convex***! ($\max(a,b)$ is convex if a and b are convex)
  - Shape of the loss function l? 線性：既 Convex 也 Concave
    Maximize a convex : convex

$\|w\|^2$: convex

- The semiparametric representation theorem applies here:
  $h = \sum_{t=1}^N c_t k(\boldsymbol{x}^{(t)}, \cdot) + b$
- Feature-dimension-independent problem:

$$\arg\min_{\boldsymbol{c}\in\mathbb{R}^N,b} \quad \sum_{t=1}^N (\max(0, (\boldsymbol{K}_{t,\cdot}\boldsymbol{c} + b) - (r^{(t)} + \epsilon)) + \max(0, (r^{(t)} - \epsilon) - (\boldsymbol{K}_{t,\cdot}\boldsymbol{c} + b)) + \lambda \boldsymbol{c}^\top \boldsymbol{K}\boldsymbol{c}$$

  - $\boldsymbol{K}_{t,\cdot}$ is the t-th row of $\boldsymbol{K}$

Ragularized，
並且使用ε-tube，
讓越少的x參與決策
(成為Support Vextor)
就可以讓越多K_t是0
從而產生稀疏性

Outline

Outline

Support Vector Classification (1)

- The idea of ϵ-insensitive loss still results in dense solution when applied to the classification problem. Why?
  - Model is linear in feature space, where
  - examples close to or far away from the decision boundary (i.e., $\{\boldsymbol{x} : h(\boldsymbol{x}) = 0\}$, a hyperplane) will all be support vectors 與需求相反
- Holds even if we adopt a non-linear kernel
  - Non-linear kernel "re-positions" instances (by creating feature space)
  - Makes instances more separable
  - But does not change the label values $+1$'s and -1's 會這樣做
- Any new idea?

Support Vector Classification (2)

- We can borrow the idea of the perceptron to remove support vectors far from the decision boundary:

$$\arg\min_{\boldsymbol{w},b,\xi} \sum_{t=1}^{N} \xi_t + \lambda \|\boldsymbol{w}\|^2$$
$$\text{subject to } r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) \geqslant -\xi_t \text{ and } \xi_t \geqslant 0,\ \forall t = 1, \cdots, N$$

- Examples not on the decision boundary nor misclassified will have $\xi_t = 0$ and the constrains hold 被分得「太對」的，不會是Support Vector
- Only examples on the decision boundary and outliers will be support vectors 差點被分錯，或是已經被分錯的，才會是Support Vector

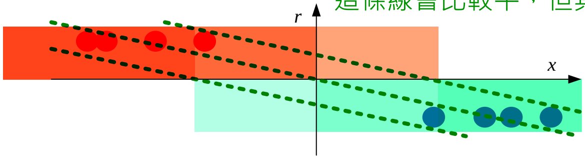
Support Vector Classification (3)

$$\arg\min_{\boldsymbol{w},b,\xi} \sum_{t=1}^{N} \xi_t + \lambda \|\boldsymbol{w}\|^2$$

subject to $r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) \geqslant -\xi_t$ and $\xi_t \geqslant 0$, $\forall t = 1, \cdots, N$

- With a lifting/kernel function, instances are more separable
- The above objective has problems in dealing with a perfectly separable dataset:
  1. Multiple hyperplanes can be equally good. Which one is the best?
  2. The hyperplane could be arbitrary flat, causing numerical problems

因為要盡可能讓||w||小一點，所以
這條線會比較平，但其實不應該這麼平



3條線都不會產生emp error，但左右2條只要
有東西落在靠中間，很容易就分錯了

- Solution?

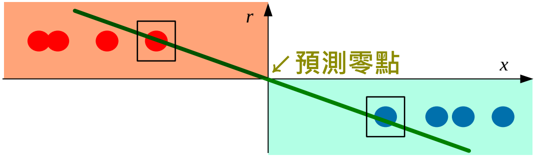
Support Vector Classification (4)

- Refined objective:

$$\arg\min_{\boldsymbol{w},b,\xi} \sum_{t=1}^{N} \xi_t + \lambda \|\boldsymbol{w}\|^2$$ 找盡可能smooth的w

subject to $r^{(t)}((\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) - r^{(t)}) \geqslant -\xi_t$ and $\xi_t \geqslant 0$, $\forall t = 1, \cdots, N$

並且　讓正examples和負examples離預測零點有一定的，相同的距離
即，把線放在最負的正examples和最正的負examples的正中間



- Always find the hyperplane that sits “in the middle” between positive and negative examples
- Not arbitrarily flat anymore: the flatter the hyperplane, the more the slacks Slack variables,鬆弛參數,$\xi^{(t)}$是用來衡量標記錯誤的樣本，其錯誤的程度

Large Margin Perspective

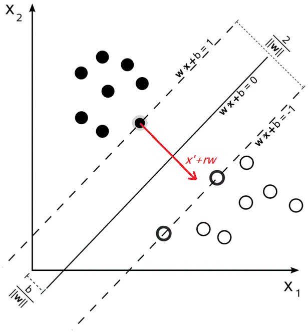
- Form geometric point of view, minimizing the term $\|\boldsymbol{w}\|^2$ in the problem:

C是Regularize用的 Hyper-Parameter

$$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{t=1}^{N}\xi_t$$
$$\text{subject to } r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) \geqslant 1-\xi_t \text{ and } \xi_t \geqslant 0,\ \forall t=1,\cdots,N$$

amounts to finding a hyperplane that ***maximizes the margin***

意味著



- The distance between $\{\boldsymbol{x}:\boldsymbol{w}^\top\boldsymbol{x}+b=1\}$ and $\{\boldsymbol{x}:\boldsymbol{w}^\top\boldsymbol{x}+b=-1\}$ is $\frac{2}{\|\boldsymbol{w}\|}$ [Homework]
- Examples touching or falling inside the margin are support vectors
- The distance between $\{\boldsymbol{x}:\boldsymbol{w}^\top\boldsymbol{x}+b=a\}$ and $\{\boldsymbol{x}:\boldsymbol{w}^\top\boldsymbol{x}+b=-a\}$ is $\frac{2a}{\|\boldsymbol{w}\|}$. Why pick $a=1$? Changing a does ***not*** change the hyperplane we find 不管a怎麼選，重點是要minimize ||w||

Applying Representer Theorem

- Again, $r^{(t)}((\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})+b)-r^{(t)}) \geqslant -\xi_t$ and $\xi_t \geqslant 0$ implies
  $\xi_t = \max(0, -r^{(t)}((\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})+b)-r^{(t)}))$ 跑對邊,ξ_t=0 跑錯邊,ξ_t=錯得多離譜
- Also, $r^{(t)}((\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})+b)-r^{(t)})$ can be written as
  $r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})+b)-1$ $r^t r^t=1$
- Objective with ξ_t eliminated:

$$\arg\min_{\boldsymbol{w},b} \sum_{t=1}^{N} \max(0, 1-r^{(t)}\boxed{(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})}+b))+\boxed{\lambda\|\boldsymbol{w}\|^2}$$

- Semiparametric representation theorem applies:
  $h=\sum_{t=1}^{N} c_t k(\boldsymbol{x}^{(t)}, \cdot)+b$
- Feature-dimension-independent objective:

$$\arg\min_{\boldsymbol{c}\in\mathbb{R}^N,b} \sum_{t=1}^{N} \max(0, 1-r^{(t)}\boxed{(\boldsymbol{K}_{t,\cdot}\boldsymbol{c}+b))}+\boxed{\lambda\boldsymbol{c}^\top\boldsymbol{K}\boldsymbol{c}}$$

  - Still ***convex***

Outline

Dual Problem

- Primal problem:

  最平滑(Margin最大)，分錯的錯誤(slack)總和最低

  $$\arg\min_{\boldsymbol{w},b,\xi} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{t=1}^{N}\xi_t$$

  subject to $r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) \geqslant 1-\xi_t$ and $\xi_t \geqslant 0$, $\forall t = 1,\cdots,N$

  對於錯誤的容忍

- Dual problem:

  先找一條線分開
  再把margin掰大

  $$\arg\max_{\alpha,\beta} \inf_{\boldsymbol{w},b,\xi} \mathcal{L}(\boldsymbol{w},b,\xi,\boldsymbol{\alpha},\boldsymbol{\beta})$$

  subject to $\boldsymbol{\alpha} \geqslant \mathbf{0}, \boldsymbol{\beta} \geqslant \mathbf{0}$,

  where $\mathcal{L}(\boldsymbol{w},b,\xi,\boldsymbol{\alpha},\boldsymbol{\beta}) =$ Constraints with Lagrange Multipliers

  $$\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{t=1}^{N}\xi_t + \sum_{t=1}^{N}\alpha_t(1-r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-\xi_t) + \sum_{t=1}^{N}\beta_t(-\xi_t)$$

- Why dual? We can distinguish SVs ***on*** the margin $\{\boldsymbol{x}^{(t)} : r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) = 1\}$ from those ***inside*** the margin $\{\boldsymbol{x}^{(t)} : r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) < 1\}$

- How to solve it?

Solving the Dual Problem

偏微分，導出最優化條件

- $dual(\alpha,\beta) = \inf_{w\in\mathbb{R}^n} \mathcal{L}(w,b,\xi,\alpha,\beta)$ is convex in terms of w, b, and ξ_t
- $\frac{\partial\mathcal{L}}{\partial w} = w - \sum_{t=1}^{N}\alpha_t r^{(t)} x^{(t)} = 0 \Rightarrow \boxed{w = \sum_{t=1}^{N}\alpha_t r^{(t)}\Phi(x^{(t)})}$
- $\frac{\partial\mathcal{L}}{\partial b} = \boxed{\sum_{t=1}^{N}\alpha_t r^{(t)} = 0}$ 零和 (zero-sum)
- $\frac{\partial\mathcal{L}}{\partial \xi_t} = \boxed{C - \alpha_t - \beta_t = 0} \Rightarrow \boxed{\beta_t = C - \alpha_t}$ 對每一個Example而言，Slack的大小決定α_t和C的大小關係
  - Additionally, since $\beta_t \geqslant 0$, we have $\alpha_t \leqslant C$
- $dual(\alpha,\beta;x) =$

最優化條件

$$\begin{cases} \sum_{t=1}^{N}\alpha_t - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j r^{(i)} r^{(j)}\Phi(x^{(i)})^\top\Phi(x^{(j)}), & \text{if } \sum_{t=1}^{N}\alpha_t r^{(t)} = 0 \text{ and } \forall i\ \alpha_i \le C \\ -\infty, \text{就不會被外面的maximize選到} & \text{otherwise} \end{cases}$$

- Dual problem:

$$\arg\max_{\alpha}\ \alpha\mathbf{1} - \frac{1}{2}\alpha^\top\widetilde{K}\alpha$$
$$\text{subject to } \mathbf{0} \leqslant \alpha \leqslant C\mathbf{1},\ r^\top\alpha = 0$$

  - $\widetilde{K}_{i,j} = r^{(i)} r^{(j)} k(x^{(i)}, x^{(j)})$

Primal vs. Dual

- Primal problem:

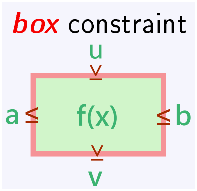
$$\arg\min_{\boldsymbol{w},b,\boldsymbol{\xi}} primal(\boldsymbol{w},b,\boldsymbol{\xi}) := \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \xi_t$$
$$\text{subject to } r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) \geqslant 1 - \xi_t \text{ and } \xi_t \geqslant 0,\ \forall t = 1,\cdots,N$$

- Dual problem:

$$\arg\max_{\boldsymbol{\alpha}} dual(\boldsymbol{\alpha}) := \boldsymbol{\alpha}\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \widetilde{\boldsymbol{K}}\boldsymbol{\alpha} = \sum_t \alpha_t - \frac{1}{2}\sum_{i,j} r^{(i)}\alpha_i r^{(j)}\alpha_j \boldsymbol{K}_{i,j}$$
$$\text{subject to } \mathbf{0} \leqslant \boldsymbol{\alpha} \leqslant C\mathbf{1},\ \boldsymbol{r}^\top\boldsymbol{\alpha} = 0$$

C是Hyper-Parameter

- Why solving dual problem?
- Constrains become much simpler: only ***box*** constraint and ***zero-sum*** constraint
- Better understanding of the data



Types of SVs

C是Hyper-Parameter

- By $\boldsymbol{w} = \sum_{t=1}^{N} \alpha_t r^{(t)} \Phi(\boldsymbol{x}^{(t)})$, examples whose $\alpha_t > 0$ will be support vectors (SVs)
  - Non-SVs are those examples whose $\alpha_t = 0$
  - ***Free SVs***: $0 < \alpha_t < C$
  - ***Bounded SVs***: $\alpha_t = C$

margin外 $r^{(t)}\Phi(x^{(t)}) > 1 : \alpha_t = 0$

margin上 $r^{(t)}\Phi(x^{(t)}) = 1 : \alpha_t = [0,C]$

margin內 $r^{(t)}\Phi(x^{(t)}) < 1 : \alpha_t = C$

- Where do the corresponding examples $\boldsymbol{x}^{(t)}$ appear in the graph of $f(\cdot) = \boldsymbol{w}^\top \Phi(\cdot) + b$?

KKT Conditions

- By KKT conditions, we have:

定義
- Primal feasibility: $r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) \geqslant 1-\xi_t$ and $\xi_t \geqslant 0$, $\forall t=1,\cdots,N$

Dual
正確運作
- Complementary slackness: $\alpha_t(1-r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-\xi_t)=0$ and $\beta_t(-\xi_t)=0$ 無Slack:非SV，α_t=0 有Slack:是SV，α_t=C

- Non SVs whose $\alpha_t=0$: $r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) \geqslant 1$ C是Hyper-Parameter
  - $1-r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-\xi_t \leqslant 0$
  - Since $\beta_t=C-\alpha_t \neq 0$, we have $\xi_t=0$

margin外(非Support Vector)
$r^{(t)}\Phi(x^{(t)}) > 1 : \alpha_t = 0$

- ***Free SVs*** whose $0<\alpha_t<C$: $r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)=1$
  - $1-r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-\xi_t = 0$
  - Since $\beta_t=C-\alpha_t \neq 0$, we have $\xi_t=0$

margin上
$r^{(t)}\Phi(x^{(t)}) = 1 : \alpha_t = [0,C]$

- ***Bounded SVs*** whose $\alpha_t=C$: $r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b) \leqslant 1$ (usually strick)
  - $1-r^{(t)}(\boldsymbol{w}^\top\Phi(\boldsymbol{x}^{(t)})+b)-\xi_t = 0$
  - Since $\beta_t=C-\alpha_t=0$, hence $\xi_t \geqslant 0$

margin內
$r^{(t)}\Phi(x^{(t)}) < 1 : \alpha_t = C$

差點被分錯邊(只要不夠對)，就有slack了(真的分錯也有)

Solving b

- How to obtain b?
- By complementary slackness in KKT conditions, we have $\alpha_t(1 - r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) - \xi_t) = 0$ for any t
- For a free SV $\boldsymbol{x}^{(t)}$ (whose $0 < \alpha_t < C$, thus $\xi_t = 0$), we have $1 = r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) \Rightarrow b = r^{(t)} - \boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)})$
    - Can be computed based on ***any*** free SV
- In practice, we usually take the average over ***all*** free SVs to avoid numeric error

Outline

SVM Solvers

- SVM primal/dual objectives can be solved by standard convex or quadratic programming software
- In practice, dedicated solvers are developed, since
- Quadratic optimization packages were often designed to take advantage of sparsity in the quadratic part of the objective function
    - Unfortunately, the SVM kernel matrix is rarely sparse; sparsity occurs in the solution of the SVM problem
- The specification of a SVM problem rarely fits in memory. Kernel matrix coefficient must be cached or computed on the fly
    - Vast speedups are achieved by accessing the kernel matrix coefficients carefully
- Generic optimization packages sometimes make extra work to locate the optimum with high accuracy
    - The accuracy requirements of a learning problem are unusually low

Dual Problem Revisited

- Primal problem:

$$\arg\min_{\boldsymbol{w},b,\boldsymbol{\xi}} primal(\boldsymbol{w},b,\boldsymbol{\xi}) := \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \xi_t$$
$$\text{subject to } r^{(t)}(\boldsymbol{w}^\top \Phi(\boldsymbol{x}^{(t)}) + b) \geqslant 1 - \xi_t \text{ and } \xi_t \geqslant 0,\ \forall t = 1,\cdots,N$$

- Dual problem:

$$\arg\max_{\boldsymbol{\alpha}} dual(\boldsymbol{\alpha}) := \boldsymbol{\alpha}\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\top \widetilde{\boldsymbol{K}}\boldsymbol{\alpha} = \sum_t \alpha_t - \frac{1}{2}\sum_{i,j} r^{(i)}\alpha_i r^{(j)}\alpha_j \boldsymbol{K}_{i,j}$$
$$\text{subject to } \mathbf{0} \leqslant \boldsymbol{\alpha} \leqslant C\mathbf{1},\ \boldsymbol{r}^\top\boldsymbol{\alpha} = 0$$

- ***Box*** constraint and ***zero-sum*** constraint

Sequential Minimal Optimization

- A special case of ***coordinate descent*** (or ***decomposition methods***) which picks only few coordinates to update during each iteration
- What's the minimal #coordinates for SVM dual? 2, due to the zero-sum constraint $\boldsymbol{r}^\top \boldsymbol{\alpha} = 0$

```
Repeat until convergence {
    1. Select some coordinate pair $\alpha_i$ and $\alpha_j$ to update next using some heuristic (e.g., to pick the two that allow us to make the biggest progress towards the global maximum); 選讓最佳化問題可以最有進展的兩個點
    2. Re-optimize $dual(\boldsymbol{\alpha})$ with respect to $\alpha_i$ and $\alpha_j$, while holding all the other coordinates $\alpha_k$'s, $k \neq i,j$, fixed;
}
```

- As compared to the batch descent: more iterations, but each iteration can run very fast!

相較於Gradient decent，雖然iteration多，但是每個iteration快

Outline

Optimality Conditions Revisited (1)

- It is sometimes convenient to rewrite the box constraint $0 \leqslant \alpha_t \leqslant C$ as:

先不要管Free Support Vectors

$$r^{(t)}\alpha_t \in [A^{(t)}, B^{(t)}] = \begin{cases} [0, C], & r^{(t)} = 1 \\ [-C, 0], & r^{(t)} = -1 \end{cases}$$

總之乘起來就是在0~±C之間

- Let $\boldsymbol{\alpha}^*$ be the solution to the dual problem
- Consider a pair of subscripts (i, j) such that $r^{(i)}\alpha_i^* < B^{(i)}$ and $A^{(j)} < r^{(j)}\alpha_j^*$
- Define $\boldsymbol{\alpha}^\epsilon$ such that $\alpha_t^\epsilon = \alpha_t^* + \begin{cases} \epsilon r^{(t)}, & \text{if } t = i \\ -\epsilon r^{(t)}, & \text{if } t = j \\ 0, & \text{otherwise} \end{cases}$, clearly $\boldsymbol{\alpha}^\epsilon$ is also a feasible solution when ϵ is positive and sufficiently small

I_{up}和I_{down}的定義

I_{up}：在負Margin上方

正Instance : $r^{(i)}\alpha_i<C$，$\alpha_i<C$，$r^{(i)}$在線上或線外

負Instance : $r^{(i)}\alpha_i<0$，$\alpha_i>0$，$r^{(i)}$在線上或線內

I_{down}：在正Margin下方

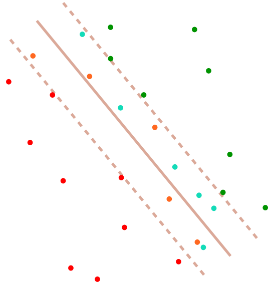
正Instance : $r^{(i)}\alpha_i>0$，$\alpha_i>0$，$r^{(i)}$在線上或線內

負Instance : $r^{(i)}\alpha_i>-C$，$\alpha_i<C$，$r^{(i)}$在線上或線外

By definition

$I_{up} := \{ i \mid r^{(i)}\alpha^*_i < B^{(i)} \}$

$I_{down} := \{ j \mid A^{(j)} < r^{(j)}\alpha^*_j \}$



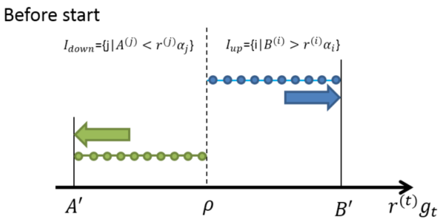
Optimality Conditions Revisited (2)

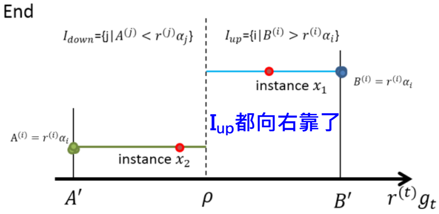
- Also, define the gradient g_t of *dual* at $\boldsymbol{\alpha}$ as 泰勒展開式的前2項
  $g_t := \frac{\partial dual(\boldsymbol{\alpha})}{\partial \alpha_t} = 1 - r^{(t)} \sum_i r^{(i)} \alpha_i \boldsymbol{K}_{t,i}$ [Proof]
- By the first order expansion, we have
  $dual(\boldsymbol{\alpha}^{\epsilon}) - dual(\boldsymbol{\alpha}^*) = \epsilon(r^{(i)} g_i^* - r^{(j)} g_j^*) + o(\epsilon)$
  因為α*已是最佳解下的α
  - $r^{(i)} g_i^* - r^{(j)} g_j^* \leqslant 0$ as $dual(\boldsymbol{\alpha}^{\epsilon}) \leqslant dual(\boldsymbol{\alpha}^*)$
- Since this holds for all pairs (i,j) such that $r^{(i)} \alpha_i^* < B^{(i)}$ Iup and
  Idown $A^{(j)} < r^{(j)} \alpha_j^*$, we have another ***necessary optimality condition***:

$$\exists \rho \in \mathbb{R} \text{ such that } \max_{i \in I_{up}} r^{(i)} g_i^* \leqslant \rho \leqslant \min_{j \in I_{down}} r^{(j)} g_j^*$$

where $I_{up} := \{i | r^{(i)} \alpha_i^* < B^{(i)}\}$ and $I_{down} := \{j | A^{(j)} < r^{(j)} \alpha_j^*\}$
  - ρ is unique, due to the existence of free SVs that exist in both I_{up} and I_{down}

Iup □□□□□□ Idown □□□□□□ 這兩個之間的差距


Before start
I_{down}={j | $A^{(j)} < r^{(j)}\alpha_j$}
I_{up}={i | $B^{(i)} > r^{(i)}\alpha_i$}
A'
ρ
B'
$r^{(t)}g_t$


End
I_{down}={j | $A^{(j)} < r^{(j)}\alpha_j$}
I_{up}={i | $B^{(i)} > r^{(i)}\alpha_i$}
instance x_1
$B^{(i)} = r^{(i)}\alpha_i$
I_{up}都向右靠了
$A^{(i)} = r^{(i)}\alpha_i$
instance x_2
A'
ρ
B'
$r^{(t)}g_t$

$$\exists \rho \in \mathbb{R} \text{ such that } \forall t, \begin{cases} r^{(t)}\alpha_t^* = B^{(t)}, & \text{if } r^{(t)}g_t^* > \rho \\ r^{(t)}\alpha_t^* = A^{(t)}, & \text{if } r^{(t)}g_t^* < \rho \end{cases}$$

$$\exists \rho \in \mathbb{R} \text{ such that } \forall t, \begin{cases} \alpha_t^* = C, & \text{if } g_t^* > r^{(t)}\rho \\ \alpha_t^* = 0, & \text{if } g_t^* < r^{(t)}\rho \end{cases}$$

If $r^{(t)}g_t^* > \rho$ and $r^{(t)}\alpha_t^* < B^{(t)}$, then the instance t must be in I_{up}(consider instance x_1), meaning $\max_{i \in I_{up}} r^{(i)}g_i > \rho$ in Condition 1, a contradiction. Similarly, if $r^{(t)}g_t^* < \rho$ and $r^{(t)}\alpha_t^* > A^{(t)}$, then the instance t must be in I_{down}(consider instance x_2), leading to another contradiction $\rho > \min_{j \in I_{down}} r^{(j)}g_j$ to Condition 1.

Optimality Conditions Revisited (3)

- Actually the above condition is also ***sufficient***
- By letting $\boldsymbol{w}^* := \sum_{t=1}^{N} \alpha_t^* r^{(t)} \Phi(\boldsymbol{x}^{(t)})$, $b^* := \rho$, and $\xi_t^* := \max\{0, g_t^* - r^{(t)}\rho\}$, we can see that [Proof]
  - All primal constrains are satisfied, and
  - $primal(\boldsymbol{w}^*, b^*, \xi^*) - dual(\boldsymbol{\alpha}^*) = 0$, so no duality gap

Sequential Minimal Optimization

Algorithm 6.2 SMO with Maximum Violating Pair working set selection

```
1: $\forall k \in \{1 \ldots n\} \quad \alpha_k \leftarrow 0$    ▷ Initial coefficients
2: $\forall k \in \{1 \ldots n\} \quad g_k \leftarrow 1$    ▷ Initial gradient
3: loop
4:     $i \leftarrow \arg\max_i y_i g_i \quad$ subject to $\quad y_i \alpha_i < B_i$
5:     $j \leftarrow \arg\min_j y_j g_j \quad$ subject to $\quad A_j < y_j \alpha_j$    ▷ Maximal violating pair
6:     if $y_i g_i \le y_j g_j$ stop.    ▷ Optimality criterion (11)
7:     $\lambda \leftarrow \min \left\{ B_i - y_i \alpha_i,\ y_j \alpha_j - A_j,\ \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$    ▷ Direction search
8:     $\forall k \in \{1 \ldots n\} \quad g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}$    ▷ Update gradient
9:     $\alpha_i \leftarrow \alpha_i + y_i \lambda \qquad \alpha_j \leftarrow \alpha_j - y_j \lambda$    ▷ Update coefficients
10: end loop
```

- $I_{up} := \{i | r^{(i)} \alpha_i < B^{(i)}\}$ and $I_{down} := \{j | A^{(j)} < r^{(j)} \alpha_j\}$ are indexes of α_t's who's values can still be moved up and down along the directions $r^{(t)}$ and $-r^{(t)}$ respectively

Outline

Directional Search

- Suppose there is no constraint in the dual problem
- Let $\boldsymbol{\alpha}^{new} := \boldsymbol{\alpha} + \lambda \boldsymbol{d}$, where $\boldsymbol{d}$ is a search direction
- Linear search: $\lambda^* := \arg\max_{\lambda \geqslant 0} dual(\boldsymbol{\alpha} + \lambda \boldsymbol{d})$ can be easily solved, yielding $\lambda^* = \max(0, \frac{\boldsymbol{g}^\top \boldsymbol{d}}{\boldsymbol{d}^\top \widetilde{\boldsymbol{K}} \boldsymbol{d}})$ [Proof]
- When $\boldsymbol{d}$ points to an ascending direction (or, equivalently, $\boldsymbol{g}^\top \boldsymbol{d} \geqslant 0$), $\lambda^* = \frac{\boldsymbol{g}^\top \boldsymbol{d}}{\boldsymbol{d}^\top \widetilde{\boldsymbol{K}} \boldsymbol{d}}$
  - This is true in SMO (to be explained in working set selection)

Directional Search in SMO (1)

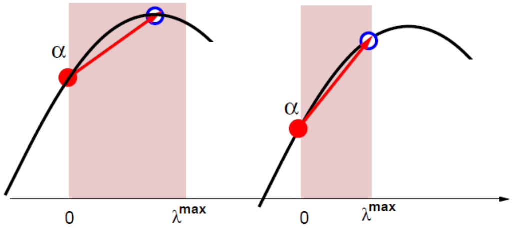
- Assume that the indexes of working set (i,j) is known (explained in working set selection), SMO picks the search direction $\boldsymbol{d}^{(i,j)}$ that moves $\boldsymbol{\alpha}$ along the ith and jth coordinates only (i.e., $d_t^{(i,j)} = 0$ for all $t \neq i,j$)
- In the presence of box constraint $\mathbf{0} \leqslant \boldsymbol{\alpha} \leqslant C$ and zero-sum constraint $\boldsymbol{r}^\top \boldsymbol{\alpha} = 0$, we need to ensure that $\boldsymbol{\alpha}^{new} = \boldsymbol{\alpha} + \lambda \boldsymbol{d}$ is still feasible

Directional Search in SMO (2)

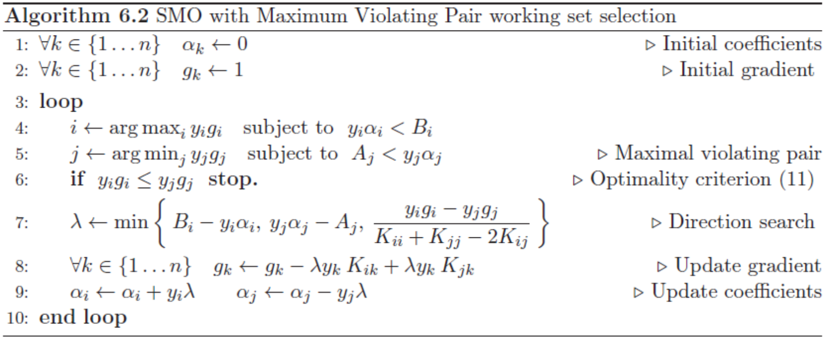
- SMO deals with the zero-sum constraint by letting
$$d_t^{(i,j)} = \begin{cases} r^{(t)}, & \text{if } t = i \\ -r^{(t)}, & \text{if } t = j \\ 0, & \text{otherwise} \end{cases}$$
- So, $\boldsymbol{r}^\top \boldsymbol{\alpha}^{new} = r^{(1)}\alpha_1 + \cdots + r^{(i)}(\alpha_i + \lambda r^{(i)}) + \cdots + r^{(j)}(\alpha_j - \lambda r^{(j)}) + \cdots + r^{(N)}\alpha_N = \boldsymbol{r}^\top \boldsymbol{\alpha} + \lambda - \lambda = 0$
- Line search can be simplified: $\lambda^* = \frac{\boldsymbol{g}^\top \boldsymbol{d}^{(i,j)}}{\boldsymbol{d}^{(i,j)\top}\widetilde{\boldsymbol{K}}\boldsymbol{d}^{(i,j)}} = \frac{r^{(i)}g_i - r^{(j)}g_j}{\boldsymbol{K}_{i,i} + \boldsymbol{K}_{j,j} - 2\boldsymbol{K}_{i,j}}$

Directional Search in SMO (3)

- SMO respects the box constraint by letting $\lambda^* = \min(B^{(i)} - r^{(i)}\alpha_i, r^{(j)}\alpha_j - A^{(j)}, \frac{r^{(i)}g_i - r^{(j)}g_j}{\boldsymbol{K}_{i,i}+\boldsymbol{K}_{j,j}-2\boldsymbol{K}_{i,j}})$
- So, $r^{(i)}(\alpha_i + \lambda r^{(i)}) = r^{(i)}\alpha_i + \lambda \leqslant B^{(i)}$ and $r^{(j)}(\alpha_j - \lambda r^{(j)}) = r^{(j)}\alpha_j - \lambda \geqslant A^{(j)}$



Sequential Minimal Optimization



Algorithm 6.2 SMO with Maximum Violating Pair working set selection

```
1: $\forall k \in \{1 \ldots n\} \quad \alpha_k \leftarrow 0$                                  ▷ Initial coefficients
2: $\forall k \in \{1 \ldots n\} \quad g_k \leftarrow 1$                                       ▷ Initial gradient
3: loop
4:     $i \leftarrow \arg\max_i y_i g_i$ subject to $y_i \alpha_i < B_i$
5:     $j \leftarrow \arg\min_j y_j g_j$ subject to $A_j < y_j \alpha_j$                        ▷ Maximal violating pair
6:     if $y_i g_i \leq y_j g_j$ stop.                                                        ▷ Optimality criterion (11)
7:     $\lambda \leftarrow \min \left\{ B_i - y_i \alpha_i,\ y_j \alpha_j - A_j,\ \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$   ▷ Direction search
8:     $\forall k \in \{1 \ldots n\} \quad g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}$      ▷ Update gradient
9:     $\alpha_i \leftarrow \alpha_i + y_i \lambda \qquad \alpha_j \leftarrow \alpha_j - y_j \lambda$        ▷ Update coefficients
10: end loop
```

Outline

Maximizing the Gain

- How to select the indexes (i,j) for the working set?

Maximizing the Gain

- How to select the indexes (i,j) for the working set?
- Idea: pick the one that maximizes the gain: $\boldsymbol{d}^{(i,j)*} := \arg\max_{\boldsymbol{d}^{(i,j)}} \max_{\lambda \geqslant 0} dual(\boldsymbol{\alpha} + \lambda \boldsymbol{d}^{(i,j)}) - dual(\boldsymbol{\alpha})$ subject to $r^{(i)}\alpha_i + \lambda \leqslant B^{(i)}$ and $A^{(j)} \leqslant r^{(j)}\alpha_j - \lambda$
- Unfortunately the search of the best direction $\boldsymbol{d}^{(i,j)*}$ requires iterating over the $N(N-1)$ possible pairs of indexes
  - The maximization of λ then amounts to performing a direction search
  - These repeated direction searches would virtually access all the kernel matrix during each SMO iteration
  - Not acceptable for a fast algorithm

Maximal Violating Pair

- By the first order approximation, we have $dual(\boldsymbol{\alpha}+\lambda \boldsymbol{d}^{(i,j)}) - dual(\boldsymbol{\alpha}) \approx \lambda \boldsymbol{g}^\top \boldsymbol{d}^{(i,j)}$ if λ is sufficiently small
- $\boldsymbol{d}^{(i,j)*} = \arg\max_{\boldsymbol{d}^{(i,j)}} \max_{0 \leqslant \lambda \leqslant \epsilon} \lambda \boldsymbol{g}^\top \boldsymbol{d}^{(i,j)}$
- We can assume that there is a direction $\boldsymbol{d}^{(i,j)}$ such that $\boldsymbol{g}^\top \boldsymbol{d}^{(i,j)} > 0$ because we would otherwise have reached the optimum
- $\boldsymbol{d}^{(i,j)*} = \arg\max_{\boldsymbol{d}^{(i,j)}} \boldsymbol{g}^\top \boldsymbol{d}^{(i,j)} = \arg\max_{i \in I_{up}, j \in I_{down}} (r^{(i)} g_i - r^{(j)} g_j)$
- Therefore, $i := \max_{k \in I_{up}} r^{(k)} g_k$ and $j := \min_{k \in I_{down}} r^{(k)} g_k$

Sequential Minimal Optimization

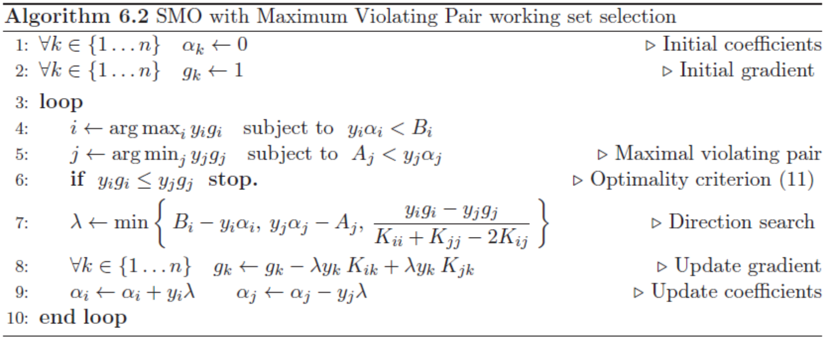
Algorithm 6.2 SMO with Maximum Violating Pair working set selection

```
 1: $\forall k \in \{1 \dots n\} \quad \alpha_k \leftarrow 0$                                   ▷ Initial coefficients
 2: $\forall k \in \{1 \dots n\} \quad g_k \leftarrow 1$                                       ▷ Initial gradient
 3: loop
 4:     $i \leftarrow \arg\max_i y_i g_i$   subject to  $y_i \alpha_i < B_i$
 5:     $j \leftarrow \arg\min_j y_j g_j$   subject to  $A_j < y_j \alpha_j$                   ▷ Maximal violating pair
 6:     if $y_i g_i \leq y_j g_j$ stop.                                                      ▷ Optimality criterion (11)
 7:     $\lambda \leftarrow \min \left\{ B_i - y_i \alpha_i,\ y_j \alpha_j - A_j,\ \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$   ▷ Direction search
 8:     $\forall k \in \{1 \dots n\} \quad g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}$     ▷ Update gradient
 9:     $\alpha_i \leftarrow \alpha_i + y_i \lambda \qquad \alpha_j \leftarrow \alpha_j - y_j \lambda$           ▷ Update coefficients
10: end loop
```

- The selected i and j are useful for ***both*** termination check and working set selection
- Compare the running time of SMO with that of CVX [Homework]

Outline

The Cost of SMO



Algorithm 6.2 SMO with Maximum Violating Pair working set selection

```
1: $\forall k \in \{1 \ldots n\}$   $\alpha_k \leftarrow 0$                                          ▷ Initial coefficients
2: $\forall k \in \{1 \ldots n\}$   $g_k \leftarrow 1$                                           ▷ Initial gradient
3: loop
4:     $i \leftarrow \arg\max_i y_i g_i$  subject to  $y_i \alpha_i < B_i$
5:     $j \leftarrow \arg\min_j y_j g_j$  subject to  $A_j < y_j \alpha_j$                       ▷ Maximal violating pair
6:     if  $y_i g_i \leq y_j g_j$  stop.                                        ▷ Optimality criterion (11)
7:     $\lambda \leftarrow \min \left\{ B_i - y_i \alpha_i,\ y_j \alpha_j - A_j,\ \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$      ▷ Direction search
8:     $\forall k \in \{1 \ldots n\}$   $g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}$          ▷ Update gradient
9:     $\alpha_i \leftarrow \alpha_i + y_i \lambda$      $\alpha_j \leftarrow \alpha_j - y_j \lambda$                        ▷ Update coefficients
10: end loop
```

- Which line costs the most time?

The Cost of SMO

Algorithm 6.2 SMO with Maximum Violating Pair working set selection

```
1: $\forall k \in \{1 \ldots n\} \quad \alpha_k \leftarrow 0$    ▷ Initial coefficients
2: $\forall k \in \{1 \ldots n\} \quad g_k \leftarrow 1$    ▷ Initial gradient
3: loop
4:     $i \leftarrow \arg\max_i y_i g_i$ subject to $y_i \alpha_i < B_i$
5:     $j \leftarrow \arg\min_j y_j g_j$ subject to $A_j < y_j \alpha_j$    ▷ Maximal violating pair
6:     if $y_i g_i \leq y_j g_j$ stop.    ▷ Optimality criterion (11)
7:     $\lambda \leftarrow \min \left\{ B_i - y_i \alpha_i,\ y_j \alpha_j - A_j,\ \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$    ▷ Direction search
8:     $\forall k \in \{1 \ldots n\} \quad g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}$    ▷ Update gradient
9:     $\alpha_i \leftarrow \alpha_i + y_i \lambda \qquad \alpha_j \leftarrow \alpha_j - y_j \lambda$    ▷ Update coefficients
10: end loop
```

- Which line costs the most time? Line 8: updating of the gradient, which requires the computation of two full rows of the kernel matrix

Computation of Kernel Values

- Computing the kernel values $\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ only appear as “constant factors” in the asymptotic complexity of solving the SVM problem
- But in practice it accounts for ***more than half*** the total running time!

Observations

- Computing kernels is expensive: data can be high-dimensional, e.g., images have thousands of pixels, documents have thousands of words

Observations

- Computing kernels is expensive: data can be high-dimensional, e.g., images have thousands of pixels, documents have thousands of words
- Computing the full kernel matrix is wasteful: the expression of the gradient $g^{(i)} = 1 - r^{(i)} \sum_{j=1}^{N} r^{(j)} \alpha_j \boldsymbol{K}_{i,j}$ only depend on kernel values $\boldsymbol{K}_{i,j}$ that involve at least one support vector (the other kernel values are multiplied by zero)

Observations

- Computing kernels is expensive: data can be high-dimensional, e.g., images have thousands of pixels, documents have thousands of words
- Computing the full kernel matrix is wasteful: the expression of the gradient $g^{(i)} = 1 - r^{(i)} \sum_{j=1}^{N} r^{(j)} \alpha_j \boldsymbol{K}_{i,j}$ only depend on kernel values $\boldsymbol{K}_{i,j}$ that involve at least one support vector (the other kernel values are multiplied by zero)
- The kernel matrix may not fit in memory: large-scale problems are common today

Kernel Caching

- In practice, SMO solvers implement a ***cache*** that keeps the most recently used kernel values
- A kernel value is computed on the fly when it is missed from cache
- Kernel cache hit rate becomes a major factor of the training time
- How much is the speedup? [Homework bonus]

Outline

Structural Risk Minimization

TBA