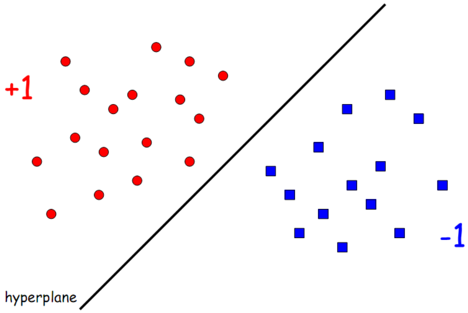
Machine Learning

Lecture 5: Implementing SVM

Konstantin Tretyakov, Phaedra Agius

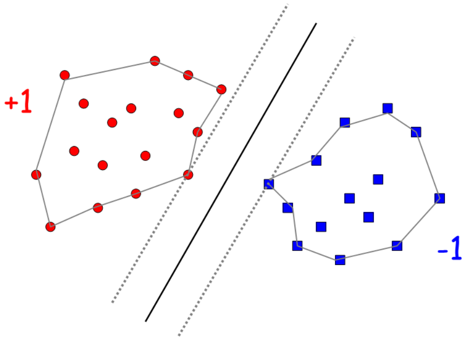
March 05, 2008

Last Time: Linear Classifier



$$f_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Last Time: Maximal Margin Linear Classifier



- Distance of point $\mathbf{x}$ to hyperplane: $\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- *Margin* of instance $(\mathbf{x}_i, y_i)$: $y_i\frac{\mathbf{w}^T\mathbf{x}_i+b}{\|\mathbf{w}\|}$

Last Time: Maximal Margin Linear Classifier

- Classifier margin: $\min_i \left(y_i \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right)$

Last Time: Maximal Margin Linear Classifier

- Classifier margin: $\min_i \left(y_i \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right)$
- We can safely rescale $\mathbf{w}$ and b without changing the solution.

Last Time: Maximal Margin Linear Classifier

- Classifier margin: $\min_i \left(y_i \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right)$
- We can safely rescale $\mathbf{w}$ and b without changing the solution.
- Let's fix the scale of $\mathbf{w}$ so that $\min_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) \right) = 1$

Last Time: Maximal Margin Linear Classifier

- Classifier margin: $\min_i \left(y_i \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right)$
- We can safely rescale **w** and b without changing the solution.
- Let's fix the scale of **w** so that $\min_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) \right) = 1$
- Then maximizing classifier margin is:

$$\max_{\mathbf{w},b} \left(\min_i \left(y_i \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|} \right) \right) = \max_{\mathbf{w},b} \left(\frac{\min_i (y_i (\mathbf{w}^T \mathbf{x}_i + b))}{\|\mathbf{w}\|} \right)$$

$$= \max_{\mathbf{w},b} \left(\frac{1}{\|\mathbf{w}\|} \right) \qquad \Rightarrow \text{need to minimize } \|\mathbf{w}\|$$

Last Time: Maximal Margin Linear Classifier

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } \forall i \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1.$$

Last Time: Maximal Margin Linear Classifier

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } \forall i \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1.$$

...or with slack variables:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$$

$$\begin{aligned} \text{s.t. } \forall i \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) &\geq 1 - \xi_i, \\ \xi_i &\geq 0. \end{aligned}$$

Today: How to Minimize That?

- Primal form
  - The primal form is a quadratic programme with linear constraints with m variables.
  - $\Rightarrow$ it is "easy".
  - As an example, we'll consider here something "simple": constrained gradient descent.

Today: How to Minimize That?

- Primal form
  - The primal form is a quadratic programme with linear constraints with m variables.
  - $\Rightarrow$ it is "easy".
  - As an example, we'll consider here something "simple": constrained gradient descent.
- Dual form (*the* SVM)
  - Also a quadratic programme with linear constraints with n variables.
  - Has advantages: sparse solution, allows to use kernels.
  - Lots of specific optimizations.

Constrained Gradient Descent

Let's consider the separable case:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } \forall i \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1.$$

Constrained Gradient Descent

Let's consider the separable case:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } \forall i \quad y_i \mathbf{x}_i^T \mathbf{w} + y_i b \geq 1.$$

Constrained Gradient Descent

Step of the descent:

$$-\frac{\partial f}{\partial \mathbf{w}} = -\mathbf{w} \qquad -\frac{\partial f}{\partial b} = 0$$

Projection to constrained region: ?

There are Better Methods

Most mathematical packages have built-in quadratic optimization routines you can easily use. E.g. Scilab:

```
X = [0 0; 0 1; 1 0; 1 1];
y = [1; 1; 1; -1];

dX = diag(y)*X;
fX = [dX y];
```

```
[w,lagr,f] = quapro(eye(3,3),zeros(3,1),-fX,-ones(y));
```

Dual: Reminder

The original problem

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t. } \forall i \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1.$$

is equivalent to

$$\min_{\mathbf{w}} \max_{\boldsymbol{\alpha}} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \alpha_i (1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \right)$$

$$\boldsymbol{\alpha} \geq 0$$

Dual: Reminder

Thanks to convexity we can switch min and max to get the equivalent *dual* problem:

$$\max_{\boldsymbol{\alpha}} \min_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{w}\|^2 + \sum_i \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)) \right)$$

$$\boldsymbol{\alpha} \geq 0$$

The inner min is easy to solve, and we're only left with one max:

$$\max_{\boldsymbol{\alpha}} \left(\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right)$$

$$\boldsymbol{\alpha}^T \mathbf{y} = 0 \qquad \boldsymbol{\alpha} \geq 0$$

Dual: Reminder

Or, in other terms:

$$\min_{\boldsymbol{\alpha}} \left(\frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1} \right)$$

$$\boldsymbol{\alpha}^T \mathbf{y} = 0 \qquad \boldsymbol{\alpha} \geq 0$$

For the version with slack variables there will also be the constraint

$$\boldsymbol{\alpha} \leq C$$

Dual: Reminder

Or, in other terms:

$$\min_{\boldsymbol{\alpha}} \left(\frac{1}{2} \boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1} \right)$$

$$\boldsymbol{\alpha}^T \mathbf{y} = 0 \qquad \boldsymbol{\alpha} \geq 0$$

For the version with slack variables there will also be the constraint

$$\boldsymbol{\alpha} \leq C$$

- This, again, is a quadratic programme, now with n variables, but much simpler constraints.
- This problem is *sparse*.

Dual: Problems

- The matrix $\mathbf{Q}$ is $n \times n$ which can be difficult to keep in memory for large n.
- The algorithm itself can be slow for large n.
- We can exploit sparseness to optimize the optimization:
  - Chunking
  - Decomposition

Chunking

- The solution actually depends only on the support vectors.

Chunking

- The solution actually depends only on the support vectors.
- Therefore:
    - Start with a small working set of points (hence small $\mathbf{Q}$). Find corresponding α.
    - Examine the margin for all other points. If it's ≥ 1 we're done.
    - Else, increase working set and retrain.

Chunking

- The solution actually depends only on the support vectors.
- Therefore:
  - Start with a small working set of points (hence small $\mathbf{Q}$). Find corresponding α.
  - Examine the margin for all other points. If it's ≥ 1 we're done.
  - Else, increase working set and retrain.
- The algorithm will definitely converge to the correct solution.
- However, if we're unlucky, the working set may still grow too large to handle.

Decomposition

- Update only a subset of the variables at each step.

Decomposition

- Update only a subset of the variables at each step.
- Let N be the set of variables, the values of which we keep fixed, and B be the set of variables to be updated.
- Arrange α, **y** and **Q** properly:

$$\boldsymbol{\alpha} = \begin{vmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{vmatrix} \qquad \mathbf{y} = \begin{vmatrix} \mathbf{y}_B \\ \mathbf{y}_N \end{vmatrix} \qquad \mathbf{Q} = \begin{vmatrix} \mathbf{Q}_{BB} & \mathbf{Q}_{BN} \\ \mathbf{Q}_{NB} & \mathbf{Q}_{NN} \end{vmatrix}$$

- The new problem:

$$\min_{\boldsymbol{\alpha}_B} \left(\frac{1}{2} \boldsymbol{\alpha}_B^T \mathbf{Q}_{BB} \boldsymbol{\alpha}_B - \alpha_B^T (\mathbf{1} - \mathbf{Q}_{BN} \boldsymbol{\alpha}_N) \right)$$

s.t.

$$\boldsymbol{\alpha}_B^T \mathbf{y}_B + \boldsymbol{\alpha}_N^T \mathbf{y}_N = 0 \qquad \mathbf{0} \leq \boldsymbol{\alpha} \leq C$$

Decomposition: Selecting the Working Set

- Consider the derivative of the objective function:

$$\frac{\partial \left(\frac{1}{2}\boldsymbol{\alpha}^T \mathbf{Q}\boldsymbol{\alpha} - \boldsymbol{\alpha}^T \mathbf{1}\right)}{\partial \boldsymbol{\alpha}} = \mathbf{Q}\boldsymbol{\alpha} - \mathbf{1}$$

- Pick the components of the derivative with largest absolute values*.

Decomposition: Shrinking

- It becomes clear fairly early in the iterations, which instances turn out *not* to be support vectors, these can be thrown away.
- Similarly, it becomes fairly early clear, for which instances will the α end up at bound (i.e. $\alpha = C$). These α values can be fixed to C and forgotten about too.

SVM^{light}

- Decomposition
- Shrinking
- Termination when all constraints satisfied to given precision
- LRU cache for kernel evaluations
- Rather reliable software. Current version: 6.01.

Sequential Minimal Optimization (SMO)

- Consider again the decomposition idea:

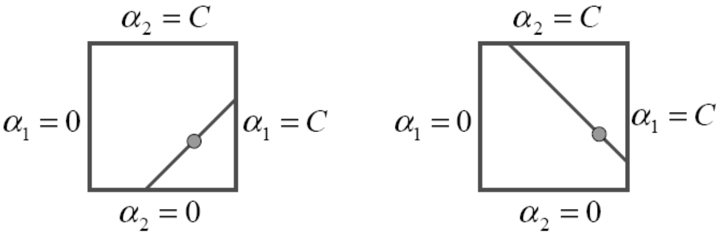
$$\min_{\boldsymbol{\alpha}_B} \left(\frac{1}{2} \boldsymbol{\alpha}_B^T \mathbf{Q}_{BB} \boldsymbol{\alpha}_B - \alpha_B^T (\mathbf{1} - \mathbf{Q}_{BN} \boldsymbol{\alpha}_N) \right)$$

  s.t.

$$\boldsymbol{\alpha}_B^T \mathbf{y}_B + \boldsymbol{\alpha}_N^T \mathbf{y}_N = 0 \qquad \mathbf{0} \leq \boldsymbol{\alpha} \leq C$$

- What if $|B| = 2$?

Sequential Minimal Optimization (SMO)



$y_1 \neq y_2 \Rightarrow \alpha_1 - \alpha_2 = \gamma$ $\qquad$ $y_1 = y_2 \Rightarrow \alpha_1 + \alpha_2 = \gamma$

SVM^{light} vs SMO (1999)

Examples	SVM^{light}	SMO	Chunking	Minimum	total SV	BSV
2477	18.0	26.3	64.9	3.6	431	47
3470	28.2	44.1	110.4	7.8	571	69
4912	46.2	83.6	372.5	13.2	671	96
7366	102.0	156.7	545.4	27.0	878	138
9888	174.6	248.1	907.6	46.8	1075	187
17188	450.0	581.0	3317.9	123.6	1611	363
24692	843.0	1214.0	6659.7	222.6	1994	506
49749	2834.4	3863.5	23877.6	706.2	3069	948
Scaling	1.7	1.7	2.0	1.7		

SMO vs SVM^{light} (1999)

Experiment	SMO Time (sec)	SVM^{light} Time (sec)	Chunking Time (sec)	SMO Scaling Exponent	SVM^{light} Scaling Exponent	Chunking Scaling Exponent
AdultLin	**13.7**	217.9	20711.3	**1.8**	2.1	3.1
AdultLinD	**21.9**	n/a	21141.1	**1.0**	n/a	3.0
WebLin	**339.9**	3980.8	17164.7	**1.6**	2.2	2.5
WebLinD	**4589.1**	n/a	17332.8	**1.5**	n/a	2.5
AdultGaussK	442.4	**284.7**	11910.6	2.0	2.0	2.9
AdultGauss	**523.3**	737.5	n/a	2.0	2.0	n/a
AdultGaussKD	**1433.0**	n/a	14740.4	**2.5**	n/a	2.8
AdultGaussD	1810.2	n/a	n/a	2.0	n/a	n/a
WebGaussK	**2477.9**	2949.5	23877.6	**1.6**	2.0	2.0
WebGauss	**2538.0**	6923.5	n/a	**1.6**	1.8	n/a
WebGaussKD	**23365.3**	n/a	50371.9	2.6	n/a	**2.0**
WebGaussD	24758.0	n/a	n/a	1.6	n/a	n/a
MNIST	**19387.9**	38452.3	33109.0	n/a	n/a	n/a

Questions?