Lecture 8
Online and Incremental Learning

Pavel Laskov[1] Blaine Nelson[1]

[1]Cognitive Systems Group
Wilhelm Schickard Institute for Computer Science
Universität Tübingen, Germany

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

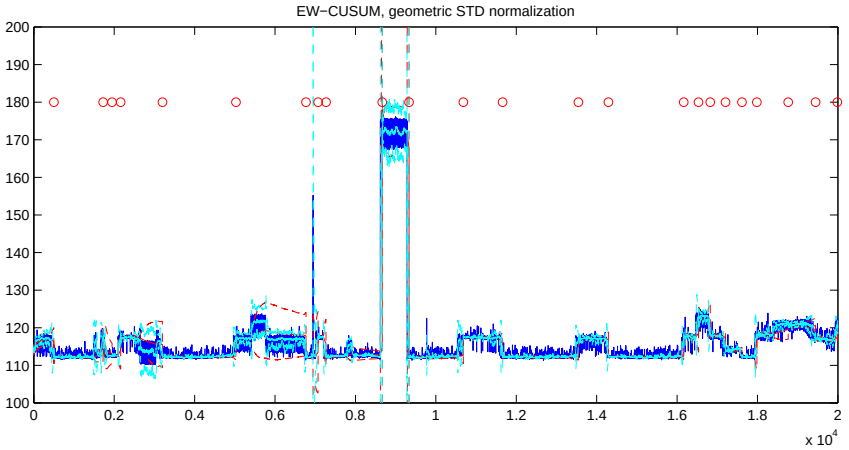
Advanced Topics in Machine Learning, 2012

Why Learning Online?

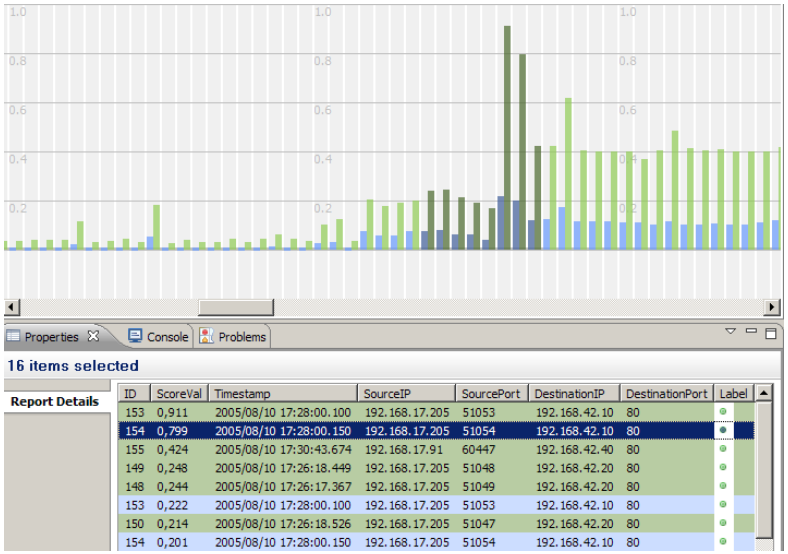
- Non-stationarity of real data
  - $\Rightarrow$ Update of models is needed when new data is available
  - $\Rightarrow$ Evaluation of relevance of past data
- Learning on a tight budget
  - $\Rightarrow$ Some data may be irrelevant
  - $\Rightarrow$ Cost-sensitive learning

Applications of Online Learning

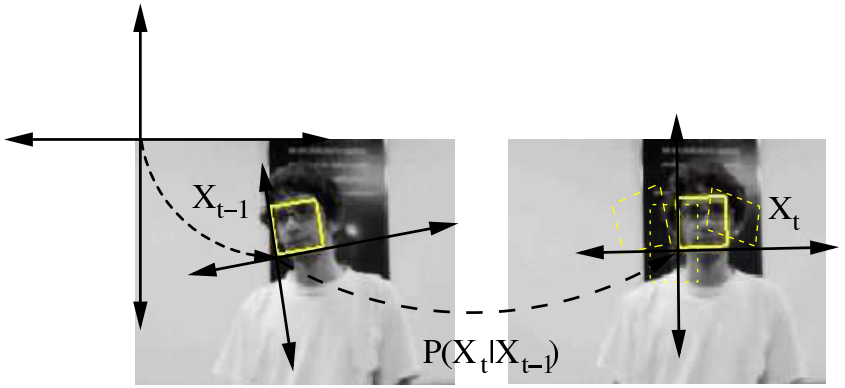
Time series analysis



Anomaly and change point detection



Security event monitoring



Object tracking

Computational Considerations

How much do we pay for training?

Computational Considerations

How much do we pay for training?

- PCA: $O(n^3)$ (eigenvalue decomposition)
- Ridge regression: $O(n^3)$ (matrix inversion)
- SVM: $O(n^2 \log n)$ (theoretical bound on feasible direction decomposition)

Computational Considerations

How much do we pay for training?

- PCA: $O(n^3)$ (eigenvalue decomposition)
- Ridge regression: $O(n^3)$ (matrix inversion)
- SVM: $O(n^2 \log n)$ (theoretical bound on feasible direction decomposition)

How much are we willing to pay?

Computational Considerations

How much do we pay for training?

- PCA: $O(n^3)$ (eigenvalue decomposition)
- Ridge regression: $O(n^3)$ (matrix inversion)
- SVM: $O(n^2 \log n)$ (theoretical bound on feasible direction decomposition)

How much are we willing to pay?

- An order of magnitude less: to match batch learning
- Constant or linear time: if we are really greedy!

Incremental SVM: Problem Setup

- Initial state: An SVM has been trained on a data set $X = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_k, y_k)\}$, and the initial solution $\boldsymbol{\alpha}_0$ is available.

Incremental SVM: Problem Setup

- Initial state: An SVM has been trained on a data set $X = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_k, y_k)\}$, and the initial solution $\boldsymbol{\alpha}_0$ is available.
- Goal: Given a new data point x_c, find an solution $\boldsymbol{\alpha}_*$ which is optimal for the extended training set $\{X \cup x_c\}$.

Incremental SVM: Problem Setup

- Initial state: An SVM has been trained on a data set $X = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_k, y_k)\}$, and the initial solution $\boldsymbol{\alpha}_0$ is available.
- Goal: Given a new data point x_c, find an solution $\boldsymbol{\alpha}_*$ which is optimal for the extended training set $\{X \cup x_c\}$.
- Key idea: Ensure that optimality conditions are enforced at every step of updating the solution from $\boldsymbol{\alpha}_0$ to $\boldsymbol{\alpha}_*$.

Incremental SVM: Problem Setup

- Initial state: An SVM has been trained on a data set $X = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_k, y_k)\}$, and the initial solution $\boldsymbol{\alpha}_0$ is available.
- Goal: Given a new data point x_c, find an solution $\boldsymbol{\alpha}_*$ which is optimal for the extended training set $\{X \cup x_c\}$.
- Key idea: Ensure that optimality conditions are enforced at every step of updating the solution from $\boldsymbol{\alpha}_0$ to $\boldsymbol{\alpha}_*$.
- Notation: All training data will be grouped into three sets according to the values of α:
  - set $\mathcal{S}$: $0 < \alpha_i < C$
  - set $\mathcal{O}$: $\alpha_i = 0$
  - set $\mathcal{E}$: $\alpha_i = C$

KKT Re-visited (once again ☺)

Consider the saddle-point formulation of the SVM training problem:

$$\max_{\mu} \min_{0 \leq \boldsymbol{\alpha} \leq C} W = -\mathbf{1}^\top \boldsymbol{\alpha} + \tfrac{1}{2} \boldsymbol{\alpha}^\top H \boldsymbol{\alpha} + \mu(\mathbf{y}^\top \boldsymbol{\alpha})$$

KKT Re-visited (once again ☺)

Consider the saddle-point formulation of the SVM training problem:

$$\max_{\mu} \min_{0 \leq \boldsymbol{\alpha} \leq C} W = -\mathbf{1}^\top \boldsymbol{\alpha} + \tfrac{1}{2} \boldsymbol{\alpha}^\top H \boldsymbol{\alpha} + \mu(\mathbf{y}^\top \boldsymbol{\alpha})$$

The KKT conditions for this problem can be expressed as follows:

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_j H_{ij}\alpha_j + y_i \mu - 1 \begin{cases} \geq 0, & i \in \mathcal{O} \\ = 0, & i \in \mathcal{S} \\ \leq 0, & i \in \mathcal{E} \end{cases}$$

$$\frac{\partial W}{\partial \mu} = \sum_j y_j \alpha_j = 0$$

KKT Re-visited (once again ☺)

Consider the saddle-point formulation of the SVM training problem:

$$\max_{\mu} \min_{0 \leq \boldsymbol{\alpha} \leq C} W = -\mathbf{1}^\top \boldsymbol{\alpha} + \tfrac{1}{2} \boldsymbol{\alpha}^\top H \boldsymbol{\alpha} + \mu(\mathbf{y}^\top \boldsymbol{\alpha})$$

The KKT conditions for this problem can be expressed as follows:

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_j H_{ij}\alpha_j + y_i\mu - 1 \begin{cases} \geq 0, & i \in \mathcal{O} \\ = 0, & i \in \mathcal{S} \\ \leq 0, & i \in \mathcal{E} \end{cases}$$

$$\frac{\partial W}{\partial \mu} = \sum_j y_j \alpha_j = 0$$

These conditions must be met for both $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_*$, i.e., before and after the inclusion of x_c, and at every single intermediate step!

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Let's try $\alpha_c = 0$. What happens?

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Let's try $\alpha_c = 0$. What happens?

- Constraints for all g_i, $i \neq c$ remain intact.

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Let's try $\alpha_c = 0$. What happens?

- Constraints for all g_i, $i \neq c$ remain intact.
- Equality constraint remains intact.

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Let's try $\alpha_c = 0$. What happens?

- Constraints for all g_i, $i \neq c$ remain intact.
- Equality constraint remains intact.
- A new constraint gets added for g_c:

$$g_i = \sum_j H_{cj}\alpha_j + y_c\mu - 1 \geq 0, \text{ since } \alpha_c = 0$$

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Let's try $\alpha_c = 0$. What happens?

- Constraints for all g_i, $i \neq c$ remain intact.
- Equality constraint remains intact.
- A new constraint gets added for g_c:

$$g_i = \sum_j H_{cj}\alpha_j + y_c\mu - 1 \geq 0, \text{ since } \alpha_c = 0$$

- If the last constraint is satified for $\alpha_c = 0$, then nothing needs to be done, and $\boldsymbol{\alpha}_* = \boldsymbol{\alpha}_0$.

Incremental SVM: Initialization

Which weight α_c should be assigned to x_c at the beginning?

Let's try $\alpha_c = 0$. What happens?

- Constraints for all g_i, $i \neq c$ remain intact.
- Equality constraint remains intact.
- A new constraint gets added for g_c:

$$g_i = \sum_j H_{cj}\alpha_j + y_c\mu - 1 \geq 0, \text{ since } \alpha_c = 0$$

- If the last constraint is satified for $\alpha_c = 0$, then nothing needs to be done, and $\boldsymbol{\alpha}_* = \boldsymbol{\alpha}_0$.
- Otherwise we need to increase α_c and update all other α's so as to keep all conditions except for g_c satisfied.

KKT Conditions in Differential Form

Consider the difference between the optimality conditions in two different SVM states:

$$\Delta g_i = H_{ic}\Delta\alpha_c + \sum_{j\in\mathcal{S}} H_{ij}\Delta\alpha_j + y_i\Delta\mu, \quad \forall i \in D \cup \{c\}$$

$$0 = y_c\Delta\alpha_c + \sum_{j\in\mathcal{S}} y_j\Delta\alpha_j$$

KKT Conditions in Differential Form

Consider the difference between the optimality conditions in two different SVM states:

$$\Delta g_i = H_{ic}\Delta\alpha_c + \sum_{j\in\mathcal{S}} H_{ij}\Delta\alpha_j + y_i\Delta\mu, \quad \forall i \in D \cup \{c\}$$

$$0 = y_c\Delta\alpha_c + \sum_{j\in\mathcal{S}} y_j\Delta\alpha_j$$

For $i \in \mathcal{S}$, $g_i = 0$; hence, these conditions can be written in matrix form:

$$\underbrace{\begin{bmatrix} 0 & y_{s_1} & \cdots & y_{s_k} \\ y_{s_1} & H_{s_1 s_1} & \cdots & H_{s_1 s_k} \\ \vdots & \vdots & \ddots & \vdots \\ y_{s_k} & H_{s_k s_1} & \cdots & H_{s_k s_k} \end{bmatrix}}_{Q} \underbrace{\begin{bmatrix} \Delta\mu \\ \Delta\alpha_{s_1} \\ \vdots \\ \Delta\alpha_{s_k} \end{bmatrix}}_{\Delta\tilde{\boldsymbol{\alpha}}_{\mathcal{S}}} = -\underbrace{\begin{bmatrix} y_c \\ H_{s_1 c} \\ \vdots \\ H_{s_k c} \end{bmatrix}}_{\eta_c} \Delta\alpha_c$$

Sensitivity Conditions

- The conditions for $g_{\mathcal{S}}$ imply that

$$\Delta\tilde{\boldsymbol{\alpha}}_{\mathcal{S}} = \underbrace{-Q^{-1}\eta_c}_{\beta}\Delta\alpha_c$$

$\Rightarrow$ vector β measures sensitivity of $\tilde{\boldsymbol{\alpha}}$ with respect to $\Delta\alpha_c$.

Sensitivity Conditions

- The conditions for $g_{\mathcal{S}}$ imply that

$$\Delta\tilde{\boldsymbol{\alpha}}_{\mathcal{S}} = \underbrace{-Q^{-1}\eta_c}_{\beta}\Delta\alpha_c$$

  $\Rightarrow$ vector β measures sensitivity of $\tilde{\boldsymbol{\alpha}}$ with respect to $\Delta\alpha_c$.

- Substituting the sensitivity of $\tilde{\boldsymbol{\alpha}}_{\mathcal{S}}$ back into the conditions for g, we obtain:

$$\Delta g_i = \gamma_i \Delta\alpha_c, \quad \forall i \in D \cup \{c\}$$
$$\text{where } \gamma_i = H_{ic} + \sum_{j\in\mathcal{S}} H_{ij}\beta_j + y_i\beta_0$$

  $\Rightarrow$ vector γ measures sensitivity of g_i, for $i \in \{\mathcal{O}, \mathcal{E}\}$, with respect to $\Delta\alpha_c$.

"Bookkeeping"

- Knowing the increment $\Delta\alpha_c$ we directly forecast:
  - the change $\Delta\tilde{\alpha}_{\mathcal{S}}$ of the weights for points in $\mathcal{S}$,
  - the change Δg_i of the gradient for points in $\{\mathcal{O}, \mathcal{E}\}$.

"Bookkeeping"

- Knowing the increment $\Delta\alpha_c$ we directly forecast:
  - the change $\Delta\tilde{\alpha}_{\mathcal{S}}$ of the weights for points in $\mathcal{S}$,
  - the change Δg_i of the gradient for points in $\{\mathcal{O}, \mathcal{E}\}$.

☹ Unfortunately, these forecasts hold only as long as no structural changes occur between $\mathcal{S}$, $\mathcal{O}$ and $\mathcal{E}$.

"Bookkeeping"

- Knowing the increment $\Delta\alpha_c$ we directly forecast:
  - the change $\Delta\tilde{\alpha}_{\mathcal{S}}$ of the weights for points in $\mathcal{S}$,
  - the change Δg_i of the gradient for points in $\{\mathcal{O}, \mathcal{E}\}$.
- ☹ Unfortunately, these forecasts hold only as long as no structural changes occur between $\mathcal{S}$, $\mathcal{O}$ and $\mathcal{E}$.
- Potential structural changes:
  - for $i \in \mathcal{S}, \alpha_i \to C$: i joins $\mathcal{E}$.
  - for $i \in \mathcal{S}, \alpha_i \to 0$: i joins $\mathcal{O}$.
  - for $i \in \mathcal{O}, g_i > 0 \to 0$: i joins $\mathcal{S}$.
  - for $i \in \mathcal{E}, g_i < 0 \to 0$: i joins $\mathcal{S}$.

"Bookkeeping"

- Knowing the increment $\Delta\alpha_c$ we directly forecast:
  - the change $\Delta\tilde{\alpha}_{\mathcal{S}}$ of the weights for points in $\mathcal{S}$,
  - the change Δg_i of the gradient for points in $\{\mathcal{O}, \mathcal{E}\}$.
- ☹ Unfortunately, these forecasts hold only as long as no structural changes occur between $\mathcal{S}$, $\mathcal{O}$ and $\mathcal{E}$.
- Potential structural changes:
  - for $i \in \mathcal{S}, \alpha_i \to C$: i joins $\mathcal{E}$.
  - for $i \in \mathcal{S}, \alpha_i \to 0$: i joins $\mathcal{O}$.
  - for $i \in \mathcal{O}, g_i > 0 \to 0$: i joins $\mathcal{S}$.
  - for $i \in \mathcal{E}, g_i < 0 \to 0$: i joins $\mathcal{S}$.
- Key idea: we can predict the smallest $\Delta\alpha_c$ such that for some i, the first structural change occurs.

Predicting Structural Changes

1. $i \in \mathcal{S}, \beta_i > 0$: α grows up to C.
   - find $\Delta\alpha_c^1 = \min_i \frac{C-\alpha_i}{\beta_i}$

Predicting Structural Changes

1. $i \in \mathcal{S}, \beta_i > 0$: α grows up to C.
   - find $\Delta\alpha_c^1 = \min_i \frac{C-\alpha_i}{\beta_i}$
2. $i \in \mathcal{S}, \beta_i < 0$: α falls down to 0.
   - find $\Delta\alpha_c^2 = \min_i \frac{-\alpha_i}{\beta_i}$

Predicting Structural Changes

1. $i \in \mathcal{S}, \beta_i > 0$: α grows up to C.
   - find $\Delta\alpha_c^1 = \min_i \frac{C-\alpha_i}{\beta_i}$
2. $i \in \mathcal{S}, \beta_i < 0$: α falls down to 0.
   - find $\Delta\alpha_c^2 = \min_i \frac{-\alpha_i}{\beta_i}$
3. $i \in \mathcal{E}, \gamma_i > 0$ or $i \in \mathcal{O}, \gamma_i < 0$: g_i grows up / falls down to 0
   - find $\Delta\alpha_c^3 = \min_i \frac{-g_i}{\gamma_i}$

Predicting Structural Changes

1. $i \in \mathcal{S}, \beta_i > 0$: α grows up to C.
   - find $\Delta\alpha_c^1 = \min_i \frac{C-\alpha_i}{\beta_i}$
2. $i \in \mathcal{S}, \beta_i < 0$: α falls down to 0.
   - find $\Delta\alpha_c^2 = \min_i \frac{-\alpha_i}{\beta_i}$
3. $i \in \mathcal{E}, \gamma_i > 0$ or $i \in \mathcal{O}, \gamma_i < 0$: g_i grows up / falls down to 0
   - find $\Delta\alpha_c^3 = \min_i \frac{-g_i}{\gamma_i}$
4. g_c becomes 0 (termination condition 1):
   - compute $\Delta\alpha_c^4 = \frac{-g_c}{\gamma_c}$

Predicting Structural Changes

1. $i \in \mathcal{S}, \beta_i > 0$: α grows up to C.
   - find $\Delta\alpha_c^1 = \min_i \frac{C-\alpha_i}{\beta_i}$
2. $i \in \mathcal{S}, \beta_i < 0$: α falls down to 0.
   - find $\Delta\alpha_c^2 = \min_i \frac{-\alpha_i}{\beta_i}$
3. $i \in \mathcal{E}, \gamma_i > 0$ or $i \in \mathcal{O}, \gamma_i < 0$: g_i grows up / falls down to 0
   - find $\Delta\alpha_c^3 = \min_i \frac{-g_i}{\gamma_i}$
4. g_c becomes 0 (termination condition 1):
   - compute $\Delta\alpha_c^4 = \frac{-g_c}{\gamma_c}$
5. α_c reaches C (termination condition 2):
   - compute $\Delta\alpha_c^5 = C - \alpha_c$

Predicting Structural Changes

1. $i \in \mathcal{S}, \beta_i > 0$: α grows up to C.
   - find $\Delta\alpha_c^1 = \min_i \frac{C-\alpha_i}{\beta_i}$
2. $i \in \mathcal{S}, \beta_i < 0$: α falls down to 0.
   - find $\Delta\alpha_c^2 = \min_i \frac{-\alpha_i}{\beta_i}$
3. $i \in \mathcal{E}, \gamma_i > 0$ or $i \in \mathcal{O}, \gamma_i < 0$: g_i grows up / falls down to 0
   - find $\Delta\alpha_c^3 = \min_i \frac{-g_i}{\gamma_i}$
4. g_c becomes 0 (termination condition 1):
   - compute $\Delta\alpha_c^4 = \frac{-g_c}{\gamma_c}$
5. α_c reaches C (termination condition 2):
   - compute $\Delta\alpha_c^5 = C - \alpha_c$

The smallest increment $\Delta\alpha_c$ among the 5 possible cases yields the first structural change.

Structural Change: Post-processing

The main operation to be performed after the structural change is the update of the inverse matrix Q^{-1}.

- Example k joins $\mathcal{S}$:
  - $\Rightarrow$ add a row and a column with zero entries to Q^{-1}
  - $\Rightarrow$ perform a recursive update (next slide)
- Example k leaves $\mathcal{S}$:
  - $\Rightarrow$ re-arrange Q^{-1} to put row and column k "at the end"
  - $\Rightarrow$ update the values (follows)

Recursive Expansion of Q^{-1}

Consider the specific structure of Q:

$$\tilde{Q}^{-1} = \begin{bmatrix} 0 & y_s^\top & y_k \\ y_s & H_{ss} & H_{sk} \\ y_k & H_{ks} & H_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} Q & \eta_k \\ \eta_k^\top & H_{kk} \end{bmatrix}^{-1}$$

where

$$\eta_k^\top = \begin{bmatrix} y_k & H_{ks} \end{bmatrix}$$

Recursive Expansion of Q^{-1}

Consider the specific structure of Q:

$$\tilde{Q}^{-1} = \begin{bmatrix} 0 & y_s^\top & y_k \\ y_s & H_{ss} & H_{sk} \\ y_k & H_{ks} & H_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} Q & \eta_k \\ \eta_k^\top & H_{kk} \end{bmatrix}^{-1}$$

where

$$\eta_k^\top = \begin{bmatrix} y_k & H_{ks} \end{bmatrix}$$

We will make use of the Sherman-Woodbury-Morrison formula:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BZCA^{-1} & -A^{-1}BZ \\ -ZCA^{-1} & Z \end{bmatrix}$$

where $Z = (D - CA^{-1}B)^{-1} = (H_{kk} - \eta_k^\top Q^{-1}\eta_k)^{-1}$ (scalar !)

Recursive Expansion of Q^{-1} (ctd.)

Putting everything together and using the fact that $\beta_k = -Q^{-1}\eta_k$ (see slide 9), we obtain:

$$\begin{bmatrix} 0 & y_s^\top & y_k \\ y_s & H_{ss} & H_{sk} \\ y_k & H_{ks} & H_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} Q^{-1} + Z(Q^{-1}\eta_k)(Q^{-1}\eta_k)^\top & -Z(Q^{-1}\eta_k) \\ -Z(Q^{-1}\eta_k)^\top & Z \end{bmatrix}$$

$$= \begin{bmatrix} Q^{-1} & 0 \\ 0 & 0 \end{bmatrix} + Z \begin{bmatrix} \beta_k \\ 1 \end{bmatrix} \begin{bmatrix} \beta_k^\top & 1 \end{bmatrix}$$

Recursive Expansion of Q^{-1} (ctd.)

Putting everything together and using the fact that $\beta_k = -Q^{-1}\eta_k$ (see slide 9), we obtain:

$$\begin{bmatrix} 0 & y_s^\top & y_k \\ y_s & H_{ss} & H_{sk} \\ y_k & H_{ks} & H_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} Q^{-1} + Z(Q^{-1}\eta_k)(Q^{-1}\eta_k)^\top & -Z(Q^{-1}\eta_k) \\ -Z(Q^{-1}\eta_k)^\top & Z \end{bmatrix}$$

$$= \begin{bmatrix} Q^{-1} & 0 \\ 0 & 0 \end{bmatrix} + Z \begin{bmatrix} \beta_k \\ 1 \end{bmatrix} \begin{bmatrix} \beta_k^\top & 1 \end{bmatrix}$$

Hence, expansion of Q^{-1} amounts to:

- extending Q^{-1} with a zero row and column ($O(s)$)
- computing Z ($O(s)^2$)
- computing and adding the outer product of $[\beta_k^\top \; 1]$ ($O(s)^2$)

Recursive Contraction of Q^{-1}

Before contraction of Q^{-1}, we have:

$$Q^{-1} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} = \begin{bmatrix} \tilde{Q}^{-1} + Z\beta_k\beta_k^\top & Z\beta_k \\ Z\beta_k^\top & Z \end{bmatrix}$$

Recursive Contraction of Q^{-1}

Before contraction of Q^{-1}, we have:

$$Q^{-1} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} = \begin{bmatrix} \tilde{Q}^{-1} + Z\beta_k\beta_k^\top & Z\beta_k \\ Z\beta_k^\top & Z \end{bmatrix}$$

Writing this relation component-wise, we obtain:

$$\begin{aligned} \tilde{Q}^{-1} &= q_{11} - Z\beta_k\beta_k^\top \\ q_{21}^\top = q_{12} &= Z\beta_k \\ q_{22} &= Z \end{aligned}$$

Recursive Contraction of Q^{-1}

Before contraction of Q^{-1}, we have:

$$Q^{-1} = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} = \begin{bmatrix} \tilde{Q}^{-1} + Z\beta_k\beta_k^\top & Z\beta_k \\ Z\beta_k^\top & Z \end{bmatrix}$$

Writing this relation component-wise, we obtain:

$$\begin{aligned} \tilde{Q}^{-1} &= q_{11} - Z\beta_k\beta_k^\top \\ q_{21}^\top = q_{12} &= Z\beta_k \\ q_{22} &= Z \end{aligned}$$

Combination of the three constraints above yields:

$$\tilde{Q}^{-1} = q_{11} - \frac{q_{12}q_{21}}{q_{22}}$$

Incremental SVM: High-Level Algorithm

1: Read example x_c, compute g_c.
2: **while** $g_c < 0$ & $\alpha_c < C$ **do**
3: Compute β and γ.
4: Compute $\Delta\alpha_c^1, \ldots, \Delta\alpha_c^5$.
5: $\Delta\alpha_c^{\max} \leftarrow \min\{\Delta\alpha_c^1, \ldots, \Delta\alpha_c^5\}$
6: $\alpha_c \leftarrow \alpha_c + \Delta\alpha_c^{\max}$
7: $\alpha_s \leftarrow \beta\Delta\alpha_c^{\max}$
8: $g_{c,e,o} \leftarrow \gamma\Delta\alpha_c^{\max}$
9: Let k be the index of the example yielding the minimum in step 5.
10: **if** $k \in \mathcal{S}$ **then**
11: Move k from $\mathcal{S}$ to either $\mathcal{E}$ or $\mathcal{O}$.
12: **else if** $k \in \mathcal{E} \cup \mathcal{O}$ **then**
13: Move k from either $\mathcal{E}$ or $\mathcal{O}$ to $\mathcal{S}$.
14: **else**
15: $\{k = c$: do nothing, the algorithm terminates.$\}$
16: **end if**
17: Update Q^{-1} recursively.
18: **end while**

Incremental SVM: Summary

- The only general method to exactly update an existing SVM solution
- Can be reversed for incremental "unlearning"
- Can be extended for regression and anomaly detection
- Moderate performance:
  - $O(s^2)$ time and space (s being the size of the set $\mathcal{S}$)
  - suitable for working sets of up to 50,000 examples
- Further usage:
  - Computation of the leave-one-out error
  - Poisoning attacks against SVM

Online Learning

An online learning algorithm looks at every example exactly once.

- ☺ Easy update of existing solutions
- ☺ Low storage requirements
- ☺ (Sometimes) faster learning of batch datasets
- ☹ Inexact: may converge to a different solution
- ☹ Potentially slow convergence rate

LASVM: a Simple Online Extension of SMO

- Function PROCESS(k):
  1: $\alpha_k \leftarrow 0$, compute g_k, add k to $\mathcal{S}$.
  2: $j \leftarrow \text{argmin}_{s \in \mathcal{S}} y_s g_s$
  3: Bail out if (j, k) is not a τ-violating pair.
  4: Perform an SMO update step on (j, k).
- Function REPROCESS:
  1: $i \leftarrow \text{argmax}_{s \in \mathcal{S}} g_s$ with $\alpha_s < C$
  2: $j \leftarrow \text{argmin}_{s \in \mathcal{S}} g_s$ with $\alpha_s > 0$
  3: Bail out if (j, k) is not a τ-violating pair.
  4: Perform an SMO update step on (i, j).

LASVM: Main Algorithm and Analysis

- Initialization
  - Seed $\mathcal{S}$ with a few examples of each class and compute the initial gradient g
- Online iterations: repeat a predefined number of times:
  - Pick an example k_t
  - Run PROCESS(k_t)
  - Run REPROCESS once
- Finishing:
  - Repeat REPROCESS until $\delta \leq \tau$

☺ The online part brings us close enough to the optimal solution at low linear cost.

☹ Still no hard guarantees on the solution quality: the final stage has an uncontrollable number of iterations.

Stochastic Gradient Descent (SGD)

The Best Pill for an Easy Case

Consider a linear SVM in d-dimensional case. A large number of simple online algorithms implement the following simple strategy:

- Pick up a random example $\mathbf{x}_t$
- Update the weight as:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \gamma_t B_t \nabla_{\mathbf{w}} Q(\mathbf{x}_t, \mathbf{w}_t)$$

where γ_t is a learning rate and B_t is a scaling matrix.

Under mild regularity conditions and with appropriate learning rates (decreasing gains satisfying $\sum_t \gamma_t^2 < \infty$), the stochastic gradient descent converges to an optimal solution.

SGD Variants and their Convergence Rates

- First order, $B = \lambda^{-1} I$:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{1}{\lambda(t + t_0)} g_t(\mathbf{w}_t)$$

  - Iterations to reach accuracy ρ: $\frac{\nu\kappa^2}{\rho} + O(\frac{1}{\rho})$, iteration cost: $O(d)$

- Second order, $B = H^{-1}$:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \frac{1}{t + t_0} H^{-1} g_t(\mathbf{w}_t)$$

  - Iterations to reach accuracy ρ: $\frac{\nu}{\rho} + O(\frac{1}{\rho})$, iteration cost: $O(d^2)$

Summary

- In contrast to incremental learning, online learning attempts to achieve fixed cost per example, potentially at the expense of learning accuracy.
- Theoretical analysis shows that high optimization accuracy is not always necessary for learning accuracy.
- Efficient algorithms for online learning exist mainly for linear learning problems.
- For non-linear methods, online learning currently cannot guarantee perfect covergence to the optimal solution.

Bibliography I

[1] Antoine Bordes, Léon Bottou, and Patrick Gallinari. SGD-QN: Careful quasi-newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.

[2] Antoine Bordes, Seyda Ertekin, Jason Weston, and Léon Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

[3] Gert Cauwenberghs and Tommaso Poggio. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems 13*, pages 409–415, 2001.

[4] Pavel Laskov, Christian Gehl, Stefan Krüger, and Klaus-Robert Müller. Incremental support vector learning: Analysis, implementation and applications. *Journal of Machine Learning Research*, 7:1909–1936, 2006.