Online Learning

Roberto Paredes

Pattern Recognition and Human Language Technologies
Instituto Tecnológico de Informática
Universidad Politécnica de Valencia

Index

- Introduction
- Online Learning: Perceptron
- Online Learning: Kernel Perceptron
- Passive-Aggressive (PA) Online Learning
- Online Learning on a Budget
- Online Learning Applications

Introduction

Introduction

- Online Learning: A procedure for obtaining a machine learning model that uses an unique sample (new) at each iteration
- Moreover: The distribution of the data is unknown (or change over the time), the data have not ever seen before, and *batch* procedure is not feasible
- Online Learning problems, Where?
- But *pure* online learning problems?

- Which is the motivation? → Computational Efficiency and Shifting problem

Could we get good models processing an unique sample at each iteration?

Introduction

- Online Learning: A procedure for obtaining a machine learning model that uses an unique sample (new) at each iteration
- Moreover: The distribution of the data is unknown (or change over the time), the data have not ever seen before, and *batch* procedure is not feasible
- Online Learning problems, Where?
- But *pure* online learning problems?

- Which is the motivation?$\rightarrow$ Computational Efficiency and Shifting problem

  Could we get good models processing an unique sample at each iteration?

Introduction - Notation

- The student should know:
  - Basic Machine Learning concepts
  - Linear models
  - Kernel methods

- The notation:
  - Samples are vectors: $\mathbf{x} \in \Re^d$
  - Weight vector: $\mathbf{w} \in \Re^d$
  - Class-label: $y \in \{-1, +1\}$
  - Class-label (multiclass): $y \in [1 \dots M]$
  - Loss function: $\ell(\cdot)$
  - Kernels: $K(\cdot, \cdot)$
  - Set of indexes (of support vectors): $S = \{\cdots\}$

Linear Models

- Linear models:

$$y = \mathrm{sgn}(\mathbf{w}' \cdot \mathbf{x}' + w'_0) \tag{1}$$

where $w_0 \in \Re$ and $\mathbf{x}', \mathbf{w}' \in \Re^{d'}$

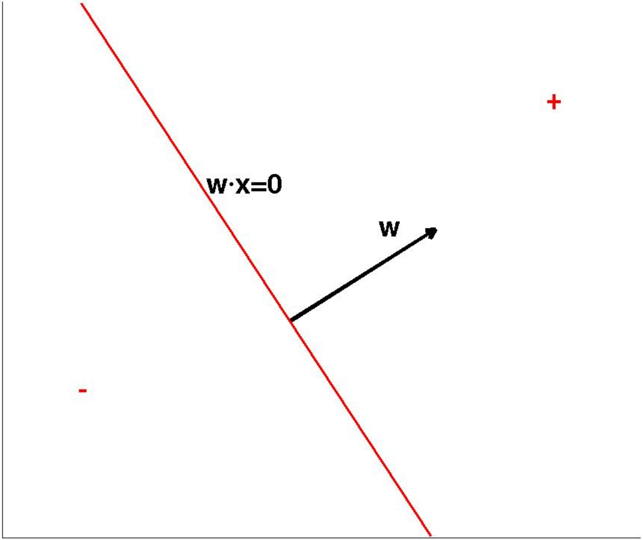
- Normally we use a *compact* notation:

$$y = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x}) \tag{2}$$

where $\mathbf{w} = \{w'_0, w'_1, w'_2, \ldots, w'_{d'}\}$ and $\mathbf{x} = \{1, x'_1, x'_2, \ldots, x'_{d'}\}$

- Let be $d = d' + 1$ then $\mathbf{w}, \mathbf{x} \in \Re^d$

Linear Models



- The goal: Given a set of data $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ find a **w** that gives the minimum classification error

Classifier:	$\mathrm{sgn}(\mathbf{wx})$
Decision boundary:	$\mathbf{wx} = 0$
Margin (sample i):	$y_i(\mathbf{wx}_i)$
Error criterion (sample i):	$y_i(\mathbf{wx}_i) < 0$

- **Perceptron**: update the model for the misclassified labels following the rule:

$$\mathbf{w}_{new} = \mathbf{w} + y_i \mathbf{x}_i$$

Why this updating rule?

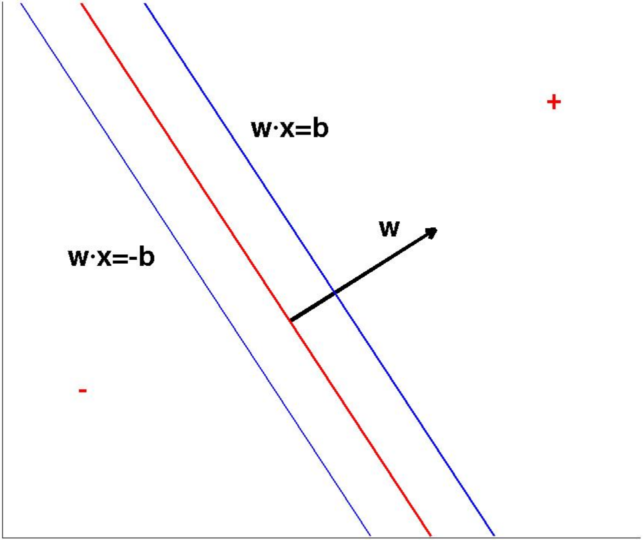
Linear Models: Perceptron

- The goal: Given a set of data $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ find a **w** that gives the minimum classification error

Classifier:	$\text{sgn}(\mathbf{wx})$
Decision boundary:	$\mathbf{wx} = 0$
Margin (sample i):	$y_i(\mathbf{wx}_i)$
Error criterion (sample i):	$y_i(\mathbf{wx}_i) < b \ , \ b \in \Re^+$

- **Perceptron**: update the model for the misclassified labels following the rule:

$$\mathbf{w}_{new} = \mathbf{w} + y_i \mathbf{x}_i$$


+
w·x=b
w
w·x=-b
-

Linear Models: Perceptron

- The goal: Given a set of data $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ find a **w** that gives the minimum classification error

Classifier:	$\mathrm{sgn}(\mathbf{wx})$
Decision boundary:	$\mathbf{wx} = 0$
Margin (sample i):	$y_i(\mathbf{wx}_i)$
Error criterion (sample i):	$y_i(\mathbf{wx}_i) < 0$

If

$$\exists\ \mathbf{u} \in \Re^d \quad y_i \mathbf{u} \mathbf{x}_i > 0 \quad \forall i = 1 \ldots n$$

then the problem is linearly separable. Note: $||\ \mathbf{u}\ ||$ no matters

Online Learning: Perceptron

Online Learning: Perceptron

- Perceptron Online Learning:
  - Initialize $\mathbf{w}_1 = \mathbf{0}$
  - For all $t = 1 \ldots T$ do:
    - Receives $\mathbf{x}_t$ and compute $y = sign(\mathbf{w}_t \mathbf{x}_t)$
    - If $y \neq y_t$ then $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$
      else $\mathbf{w}_{t+1} = \mathbf{w}_t$
  - The algorithm returns $\mathbf{w}_{T+1}$

Online Learning: Perceptron - Bounding the number of errors

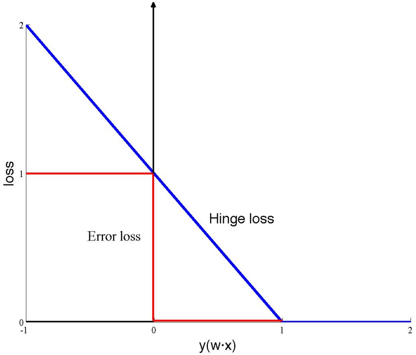
- Let be $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ a finite data set
- Let be $\mathbf{u}^*$ the linear model with minimum number of errors for X
- Let be $\mathbf{w}_{T+1}$ the linear model obtained for X using the Perceptron
- Which is the relation between the number of errors of $\mathbf{u}^*$ and the Perceptron?

$$\sum_{t=1}^{T} \varepsilon(\mathbf{w}_t) \;\leq\; \sum_{t=1}^{T} \varepsilon(\mathbf{u}^*) \;+\; \textit{constant}$$

- Is the constant value small?
- Note: $\sum_{t=1}^{T} \varepsilon(\mathbf{w}_t)$ is an online error while $\sum_{t=1}^{T} \varepsilon(\mathbf{u}^*)$ is the error of some $\mathbf{u}^*$ with all the samples available

Online Learning: Perceptron - Bounding the number of errors

- To find $\mathbf{u}^*$ that minimizes the number of errors for given set X is a NP-hard problem, we have to relax the expression introducing some convex loss: Hinge loss
- The hinge loss is $\ell(\mathbf{w}; (\mathbf{x}, y)) = max(0, 1 - y(\mathbf{wx}))$



Online Learning: Perceptron - Bounding the number of errors

- We redefine the previous relations as:

$$\sum_{t=1}^{T} \varepsilon(\mathbf{w}_t) \;\leq\; \sum_{t=1}^{T} \ell(\mathbf{u}^*) \;+\; \textit{constant}$$

- And we get:

$$\sum_{t=1}^{T} \varepsilon(\mathbf{w}_t) \;\leq\; \sum_{t=1}^{T} \ell(\mathbf{u}) + \;\| \, u \, \|^2 \;+\; \| \, u \, \| \sqrt{\sum_{t=1}^{T} \ell(\mathbf{u})}$$

- Note: for any **u**
- It is worth to see how to get this relation

Online Learning: Perceptron - General Model

- General model:

$$y = sign(\mathbf{w}_t \mathbf{x}_t) = sign(\sum_{i=1}^{t-1} y_i \alpha_i \mathbf{x}_i \mathbf{x}_t)$$

- Common algorithmic structure:
  - Receives $\mathbf{x}_t$ and compute y
  - If $y \cdot y_t > \beta_t$ then $\alpha_t = 0$
  - else $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t y_t \mathbf{x}_t$, where $\alpha_t > 0$
  - optionally **w** is scaled.: $\mathbf{w}_{t+1} \leftarrow c_t \mathbf{w}_{t+1}$

- Perceptron: $\alpha_t = 1$,$\beta_t = 0$ and $c_t = 1$
- Well-known algorithms like: *Relaxed Online Maximum Margin Algorithm (ROMMA)*,*Approximate Maximal Margin Classification Algorithm (ALMA)* and *Margin Infused Relaxed Algorithm (MIRA)*

- MIRA for two-class problem: (*Crammer and Singer (2003)*)
  - Apply the common algorithmic structure presented before
  - For each $\mathbf{x}_t$ define α_t as:

$$\alpha_t = G\left(-\frac{y_t(\mathbf{w}_t\mathbf{x}_t)}{\|\,\mathbf{x}_t\,\|^2}\right) \quad \text{where}$$

$$G(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } 0 \leq z \leq 1 \\ 1 & \text{if } 1 < z \end{cases}$$

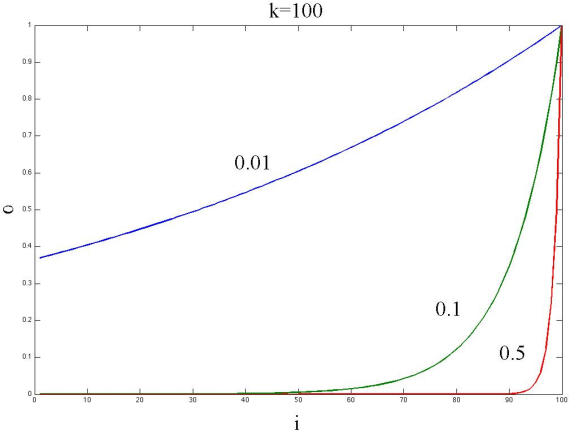
Online Learning: Perceptron - The shifting Perceptron

- The Shifting Perceptron Algorithm (SPA) (*Cavallanti, Cesa-Bianchi and Gentile (2006)*)
- Goal: The tracking ability $\rightarrow$ weak dependence on the past:
  - Memory boundeness (Online Learning on a Budget)
  - Weight decay:

$$\text{If} \;\; y_t \neq sign(\mathbf{w}_t \mathbf{x}_t) \;\; \text{then}$$

$$\mathbf{w}_{t+1} = (1 - \lambda_k)\mathbf{w}_t + y_t \mathbf{x}_t, \;\; k \leftarrow k + 1, \;\; \lambda_k = \frac{\lambda}{\lambda + k}$$

- The Shifting Perceptron implements an exponential decaying scheme
- Let be x_i the $i - th$ sample with mistake
- $\alpha_i = (1 - \lambda)^{k-i}$, where k is the total mistakes at the moment



Online Learning: Perceptron - Extension to Multiclass

- Let be M the number of classes
- Kesler's construction: $\mathbf{x} \in \Re^d$ is transformed into $M - 1$ samples $\mathbf{x}' \in \Re^{M \times d}$
- Useless under the practical point of view
- Very useful for converting multiclass problems into two class problems for the purpose of obtaining a convergence proof

- In a practical scenario:
  - $\mathbf{w} \in \Re^d \to \mathbf{W} \in \Re^{M \times d}$
  - Given a pair $(\mathbf{x}_t, y_t)$ compute: $y = \arg\max_{i=1\dots M} \mathbf{W}^i \mathbf{x}$
  - If $y \neq y_t$ then an error is produced

Online Learning: Perceptron - Multiclass algorithms

- A family of additive multiclass algorithms: (*Crammer and Singer (2003)*)
- Given $(\mathbf{x}_t, y_t),\;\; y_t \in \{1, 2, \ldots, M\}$
- Compute $y = \arg\max_{i=1\ldots M} \mathbf{W}^i \mathbf{x}$
- If $y \neq y_t$:
  - $\mathbf{W}^{y_t} \leftarrow \mathbf{W}^{y_t} + \alpha_{y_t} \mathbf{x}_t$
  - $\mathbf{W}^r \leftarrow \mathbf{W}^r + \alpha_r \mathbf{x}_t,\;\; \forall r \in E,\;\; \text{where} E = \{r : \mathbf{W}^r \mathbf{x}_t > \mathbf{W}^{y_t} \mathbf{x}_t\}$
  - Imposing the constrain $\alpha_{y_t} = -\sum_{r \in E} \alpha_r$
- Some examples:

$$\alpha_r = \begin{cases} -\frac{1}{|E|} & \text{if } \; r \in E \\ 1 & \text{if } \; r = y_t \\ 0 & \textit{otherwise} \end{cases} \qquad \alpha_r = \begin{cases} -1 & \text{if } \; r = \arg\max_{s \in E} \mathbf{W}^s \mathbf{x}_t \\ 1 & \text{if } \; r = y_t \\ 0 & \textit{otherwise} \end{cases}$$

- Error Bound for this family of algorithms:

$$\sum_{t=1}^{T} \varepsilon(\mathbf{W}_t) \leq 2\frac{(R+D)^2}{\gamma^2}$$

where

$D^2 = \sum_{t=1}^{T}(d^t)^2$
$d^t = \max\{0, \gamma - (\hat{\mathbf{W}}^{y_t}\mathbf{x}_t - \max_{i \neq y_t} \hat{\mathbf{W}}^i \mathbf{x}_t\})\}$
$R = max_t \mid\mid \mathbf{x}_t \mid\mid$

- In particular for the best:

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}:||\mathbf{W}||=1} \sum_{t=1}^{T} 2\frac{(R+D)^2}{\gamma^2}$$

Online Learning: Perceptron - MIRA

- MIRA for multi-class problem:

Given $(\mathbf{x}_t, y_t)$, $y_t \in \{1, 2, \ldots, M\}$

Compute $y = \arg\max_{i=1\ldots M} \mathbf{W}^i \mathbf{x}$

If $y \neq y_t$

Find τ that solves the optimization problem:

$$min_\tau \ \frac{1}{2} \sum_{i=1}^{M} \mid\mid \mathbf{W}^i + \tau_i \mathbf{x}_t \mid\mid$$

$$\text{subject to:} \begin{cases} \tau_i \leq \delta_{r,y_t} & i = 1 \ldots M \\ \sum_{i=1}^{M} \tau_i = 0 \end{cases}$$

Update $\mathbf{W}^i = \mathbf{W}^i \ + \ \tau_i \mathbf{x}_t$

- Is still MIRA an ultraconservative algorithm?

Online Learning: Kernel Perceptron

Online Learning: Kernel Perceptron

- A linear Perceptron in a RKHS: *Referring Kernel Hilbert Space*
- The Perceptron model becomes a linear combination of kernels
- All past mistaken samples $\mathbf{x}_t$ become support vectors
- The number of support vectors in not bounded in principle

- General model: Kernel Perceptron
  - Linear Perceptron: $y = sign(\mathbf{w}_t \mathbf{x}_t) = sign(\sum_{i=1}^{t-1} y_i \alpha_i \mathbf{x}_i \mathbf{x}_t)$
  - Kernel extension: $y = sign(\mathbf{w}_t \mathbf{x}_t) = sign(\sum_{i=1}^{t-1} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_t))$
- The weight α_i can be seen as the *importance* of $\mathbf{x}_i$
- All the previous algorithms can be applied, for instance MIRA:

$$y = sign(\mathbf{w}_t \mathbf{x}_t) = sign(\sum_{i=1}^{t-1} y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}_t))$$

$$\alpha_i = G\left(-\frac{y_i(\mathbf{w}_i \mathbf{x}_i)}{\|\ \mathbf{x}_i\ \|^2}\right)$$

Passive-Aggressive Online Learning

Passive-Aggressive (PA) Online Learning

- Online Learning:
  1. At each time t we received a sample $\mathbf{x}_t$
  2. The class-label y for this $\mathbf{x}_t$ is obtained from our model
  3. The real class-label y_t is then received
  4. Some *loss* is measured (divergence between y_t and y)
  5. Modify the model to get zero loss
  6. go to 1

- Some important considerations:
  - At each time t we only observe an unique pair $(\mathbf{x}_t, y_t)$
  - The modifications to the model should preserve what was learned from previous pairs: $\{(\mathbf{x}_1, y_1) \ldots (\mathbf{x}_{t-1}, y_{t-1})\}$

Passive-Aggressive (PA) Online Learning

- Things to do:
  - We have to define how to measure the loss, loss function

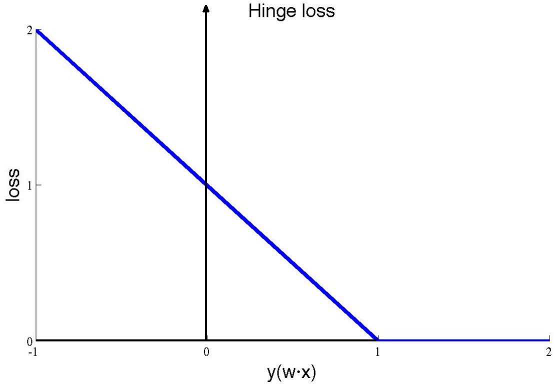
    $\rightarrow$ The loss for the pair $(\mathbf{x}_t, y_t)$ should be 0
  - We have to solve how to preserve the previous learning

    $\rightarrow$ Define a *distance* between the models

    $\rightarrow$ The distance between models should be minimum

Passive-Aggressive (PA) Online Learning

- Using a *linear* model and the *hinge-loss* function:
  - The class label is $y = \mathrm{sgn}(\mathbf{w}_t \mathbf{x}_t)$
  - The hinge loss is $\ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = max(0, 1 - y_t(\mathbf{w}\mathbf{x}_t))$



  - The model divergence can be computed as $\| \mathbf{w}' - \mathbf{w} \|^2$

Passive-Aggressive (PA) Online Learning

- Minimization problem (Crammer et al. 2006):

$$\mathbf{w}_{t+1} = \underset{w \in \Re^d}{\arg\min} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 \quad s.t. \;\; \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0$$

- Find a vector **w** *near* to the current $\mathbf{w}_t$ that classifies correctly (and with some margin) the new sample $\mathbf{x}_t$

Passive-Aggressive (PA) Online Learning

- Lagrangian:

$$\mathcal{L}(\mathbf{w}, \tau) = \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \;\; + \;\; \tau(1 - y_t(\mathbf{w}\mathbf{x}_t))$$

- Setting the derivatives of $\mathcal{L}$ with respect to **w** to zero:

$$0 = \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \tau) = \mathbf{w} - \mathbf{w}_t - \tau y_t \mathbf{x}_t \;\; \rightarrow \;\; \boxed{\mathbf{w} = \mathbf{w}_t + \tau y_t \mathbf{x}_t}$$

- Plugging back to the Lagrangian equation:

$$\mathcal{L}(\tau) = -\frac{1}{2}\tau^2 \mid\mid \mathbf{x}_t \mid\mid^2 \;\; + \;\; \tau(1 - y_t(\mathbf{w}_t\mathbf{x}_t))$$

- Setting the derivatives w.r.t τ to zero:

$$0 = \frac{\partial \mathcal{L}(\tau)}{\partial \tau} = -\tau \mid\mid \mathbf{x}_t \mid\mid^2 \;\; + \;\; (1 - y_t \mathbf{w}_t \mathbf{x}_t) \;\; \rightarrow \;\; \boxed{\tau = \frac{1 - y_t(\mathbf{w}_t\mathbf{x}_t)}{\mid\mid \mathbf{x}_t \mid\mid^2}}$$

Exercise: Check these expressions

Passive-Aggressive (PA) Online Learning

- Solution:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t \mathbf{x}_t \qquad\qquad \tau = \frac{\ell(\mathbf{w}_t; (\mathbf{x}_t, y_t))}{\|\mathbf{x}_t\|^2}$$

- Geometrical interpretation

Passive-Aggressive (PA) Online Learning

- Advantage:
  - The model modification: $\mathbf{w}_{t+1} - \mathbf{w}_t = \tau_t y_t \mathbf{x}_t$ is as much as needed to get $\ell_t = 0$
  - Certainly such modification leads to the minimum of $\frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2$
- Problem:
  - But this minimum could be too much in case of outliers or problems that are not linearly separable
  - In some iteration t the model could *forget* what has learned before, $\mid\mid \mathbf{w}_{t+1} - \mathbf{w}t \mid\mid^2 \uparrow\uparrow$
- Solution: Introduce a parameter that controls the *Aggressiveness* of the algorithm

Passive-Aggressive (PA) Online Learning

- Applying the same ideas introduced previously (Vapnik, 1998) to derive soft-margin classifiers
- New minimization:

$$\mathbf{w}_{t+1} = \arg\min_{w \in \Re^d} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi \quad s.t. \ \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \ \ and \ \ \xi \geq 0$$

- Larger values of C imply a more aggressive update strategy

Passive-Aggressive (PA) Online Learning

- Two models:
  - PA-I

$$\mathbf{w}_{t+1} = \underset{w \in \Re^d}{\arg\min} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi \quad s.t.\ \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \ \ and \ \ \xi \geq 0$$

  - PA-II

$$\mathbf{w}_{t+1} = \underset{w \in \Re^d}{\arg\min} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi^2 \quad s.t.\ \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi$$

Exercise: Obtain the PA-I and PA-II updating rules

Passive-Aggressive (PA) Online Learning

- Solutions to the two proposed models:
  - PA-I

$$\tau_t = min\left\{C, \frac{\ell_t}{\|\mathbf{x}_t\|^2}\right\}$$

  - PA-II

$$\tau_t = \frac{\ell_t}{\|\mathbf{x}_t\|^2 + \frac{1}{2C}}$$

- In both cases: $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$

Passive-Aggressive (PA) Online Learning

- PA Algorithm:
  - Initialize $\mathbf{w}_1 = (0, \ldots, 0)$
  - For $t = 1, 2, \ldots$
    - Receive sample $\mathbf{x}_t$
    - Compute $y = \text{sgn}(\mathbf{w}_t \mathbf{x}_t)$
    - Receive correct label y_t
    - Compute loss, $\ell_t = max\{0, 1 - y_t(\mathbf{w}_t \mathbf{x}_t)\}$
    - Compute $\tau_t = min\left\{C, \frac{\ell_t}{||\mathbf{x}_t||^2}\right\}$ (PA-I)
    - Update $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$

Passive-Aggressive (PA) Online Learning

- Some demos

PA with kernels

- The linear model is compact, all the model is stored in **w**

$$\mathbf{w}_t = \sum_{i=1}^{t-1} \tau_i y_i \mathbf{x}_i$$

$$\mathbf{w}_t \mathbf{x}_t = \sum_{i=1}^{t-1} \tau_i y_i (\mathbf{x}_t \mathbf{x}_i)$$

- The inner product can be replaced with a general Mercel kernel $K(x_i, x_t)$

$$\mathbf{w}_t \mathbf{x}_t = \sum_{i=1}^{t-1} \tau_i y_i K(\mathbf{x}_t, \mathbf{x}_i)$$

- How is the algorithm affected ?

PA with kernels

- PA Algorithm:
  - Initialize $\mathbf{w}_1 = (0, \ldots, 0)$
  - For $t = 1, 2, \ldots$
    - Receive sample $\mathbf{x}_t$
    - Compute $y = \text{sgn}(\mathbf{w}_t \mathbf{x}_t)$
    - Receive correct label y_t
    - Compute loss, $\ell_t = max\{0, 1 - y_t(\mathbf{w}_t \mathbf{x}_t)\}$
    - Compute $\tau_t = min\left\{C, \frac{\ell_t}{||\mathbf{x}_t||^2}\right\}$ (PA-I)
    - Update $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t$

PA with kernels

- PA Algorithm:
  - Initialize $\mathbf{w}_1 = (0, \ldots, 0)$
  - For $t = 1, 2, \ldots$
    - Receive sample $\mathbf{x}_t$
    - Compute $y = \text{sgn}(\sum_{i=1}^{t} \tau_i y_i K(\mathbf{x}_t, \mathbf{x}_i))$
    - Receive correct label y_t
    - Compute loss, $\ell_t = max\{0, 1 - y_t(\sum_{i=1}^{t} \tau_i y_i K(\mathbf{x}_t, \mathbf{x}_i))\}$
    - Compute $\tau_t = min\left\{C, \frac{\ell_t}{||\mathbf{x}_t||^2}\right\}$ (PA-I)
    - ~~Update~~ $\cancel{\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t y_t \mathbf{x}_t}$

PA with kernels

- Some important issues:
  - The weight vector **w** is not used anymore
  - If $\tau_i = 0$ we can avoid the kernel $K(\mathbf{x}_t, \mathbf{x}_i)$
  - Those vectors $\mathbf{x}_t$ that produce some loss $\tau_t > 0$ become *support vectors*
- Some disadvantages:
  - This model is more *expensive*
  - The value τ_i associated to previous sample $\mathbf{x}_i$ is no reconsidered
  - The number of support vectors used to be higher than the necessary

Passive-Aggressive (PA) Online Learning

- Some demos

PA for Regression

PA for Regression

- Modify the PA for regression problems
- A different loss is required:

$$\ell_\epsilon = \max(0, \mid \mathbf{wx} - y \mid - \epsilon)$$

- Similar optimization problem:

$$\mathbf{w}_{t+1} = \arg\min_{w \in \Re^d} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \quad s.t. \;\; \ell_\epsilon(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0$$

- Solution:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + sign(y_t - \hat{y}_t)\tau_t \mathbf{x}_t \quad \text{where} \quad \tau_t = \frac{\ell_t}{\mid\mid \mathbf{x} \mid\mid^2}$$

PA for Regression

- PA-I and PA-II can also be obtained for the regression model

$$\mathrm{PA-I} \qquad \tau_t = min\left\{C, \frac{\ell_\epsilon}{\|\ \mathbf{x}_t\ \|^2}\right\}$$

$$\mathrm{PA-II} \qquad \tau_t = \frac{\ell_\epsilon}{\|\ \mathbf{x}_t\ \|^2 + \frac{1}{2C}}$$

PA for multiclass problems

- $\mathbf{w} \in \Re^d$
- For each class m, the sample $\mathbf{x}$ is mapped $\Phi(\mathbf{x}, m) \in \Re^d$
- Given a pair $(\mathbf{x}_t, y_t)$ compute the M mappings: $\Phi(\mathbf{x}, 1) \ldots \Phi(\mathbf{x}, M)$
- Simplified constrained optimization:

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 \quad s.t. \;\; \mathbf{w}(\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, s_t)) \geq 1$$

where $s_t = argmax_{i \in \{1 \ldots M\},\ i \neq y_t} \;\; \mathbf{w}_t \Phi(\mathbf{x}_t, i)$

PA for multiclass problems

- The solution to this multiclass optimization problem is:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t(\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, s_t))$$

where $\tau_t = \frac{\ell_t}{||\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, s_t)||^2}$

PA for multiclass problems

- Another approximation $\mathbf{w}_t\Phi(\mathbf{x}_t, r) = \mathbf{W}_t^r\mathbf{x}_t$
- Then $\mathbf{W} \in \Re^{M\times d}$
- For each class m, the sample $\mathbf{x}$ is mapped $\Phi(\mathbf{x}, m) \in \Re^d$
- Simplified constrained optimization:

$$\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \frac{1}{2} \mid\mid \mathbf{W} - \mathbf{W}_t \mid\mid^2 \quad s.t. \;\; (\mathbf{W}^{y_t}\mathbf{x}_t - \mathbf{W}^{s_t}\mathbf{x}_t) \geq 1$$

where $s_t = argmax_{i\in\{1\ldots M\},\ i\neq y_t}\ \ \mathbf{W}^i\mathbf{x}_t$

- The solution to this multiclass optimization problem is:

$$\mathbf{W}_{t+1}^{y_t} = \mathbf{W}_t^{y_t} + \tau_t \mathbf{x}_t$$

$$\mathbf{W}_{t+1}^{s_t} = \mathbf{W}_t^{s_t} - \tau_t \mathbf{x}_t$$

where $\tau_t = \frac{\ell_t}{2||\mathbf{x}_t||^2}$

Online Learning on a Budget

Online Classification on a Budget

- Every time a sample produces $\ell_t > 0$ this sample is added to the support vector set
- Under certain circumstances the set of support vectors grows considerably
- The computational efficiency decreases, for the following samples and for the test phase
- In real applications usually the memory resources could be very limited
- Solution $\rightarrow$ limit the number of support vectors to *B*
- Moreover in changing tasks budget algorithms uses to outperform non-budget algorithms

Online Classification on a Budget: Budget Perceptron

- Budget Perceptron (BP) *Crammer, Kandola and Singer (2004)*
- Linear version:

For $t = 1 \ldots$

  Get new sample $\mathbf{x}_t$

  Compute $y = \mathbf{w}_t \mathbf{x}_t$

  If $y_t y < \beta$ then

    $S = S \cup \{t\}$
    $\alpha_t = 1$
    $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \alpha_t \mathbf{x}_t$
    DC(S,$\mathbf{w}_{t+1}$)

DC(S,**w**)

  For all $i \in S$

    If $\beta \leq y_i(\mathbf{w} - y_i \alpha_i \mathbf{x}_i)$ then

      $\mathbf{w} = \mathbf{w} - y_i \alpha_i \mathbf{x}_i$
      $S = S / \{i\}$

  return $S, \mathbf{w}$

Online Classification on a Budget: Budget Perceptron

- Budget Perceptron (BP) *Crammer, Kandola and Singer (2004)*

For $t = 1 \ldots$

  Get new sample $\mathbf{x}_t$

  Compute
  $y = \sum_{i \in S} y_i \alpha_i K(\mathbf{x}_t, \mathbf{x}_i)$

  If $y_t y < \beta$ then

    $S = S \cup \{t\}$
    $\alpha_t = 1$
    DC(S)

DC(S)

  For all $i \in S$

    if $\beta \leq y_i (\sum_{j \in S, j \neq i} y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j))$
    then

      $S = S / \{i\}$

  return S

Online Classification on a Budget: Budget Perceptron

- Budget Perceptron (BP) with fixed-size S

  For $t = 1 \ldots$

  Get new sample $\mathbf{x}_t$

  Compute $y = \sum_{i \in S} y_i \alpha_i K(\mathbf{x}_t, \mathbf{x}_i)$

  If $y_t y < \beta$ then

  if $\mid S \mid = B$ then Remove(S)
  $S = S \cup \{t\}$
  $\alpha_t = 1$

  Remove(S)

  Find $s = \arg\max_{i \in S} \{y_i (\sum_{j \in S, j \neq i} y_j \alpha_j K(\mathbf{x}_i, \mathbf{x}_j))\}$
  $S = S / \{s\}$
  return S

Online Classification on a Budget: Forgetron

- Forgetron: *Dekel, Shalev-Shwartz and Singer (2006)*

- First, consider the *Remove-Oldest* Perceptron
- Can be seen as a simple modification of the kernel Perceptron
- Algorithm:
  1. if $\ell_t < 0$ do nothing
  2. if $\ell_t > 0$ and *nsv* $<$ *B* then add sample $\mathbf{x}_t$, $nsv = nsv + 1$
  3. if $\ell_t > 0$ and *nsv* $>=$ *B* then add sample $\mathbf{x}_t$ but remove oldest sample in sv set
- Dekel et al. discussed the *damage* caused by removing the oldest sample

- The key for controlling this damage is to ensure that the sample being removed has small influence

Online Classification on a Budget: Forgetron

- Second, *Shrinking* Perceptron:

$$\mathbf{w}_t\mathbf{x}_t = \sum_{i=1}^{t-1} \sigma_i y_i K(\mathbf{x}_t, \mathbf{x}_i)$$

  where $\sigma_i \in [0, 1]$
- When a new $\mathbf{x}_t$ is added to the sv set:
  - Its associated weight $\sigma_t = 1$
  - The weights of previous sample in sv are decreased $\sigma_i = \phi\sigma_i$ for $0 < i < t$
  - where $0 < \phi < 1$
- If the weights decrease rapidly enough the contribution of older samples becomes negligible
- But again, a damage is produced on the accuracy of the online algorithm

Online Classification on a Budget: Forgetron

- Forgetron combines two approaches, *Remove-Oldest* Perceptron and *Shrinking* Perceptron

- Very important to define the value of ϕ, more concretely ϕ_t
- Self-tuned Forgetron:

$$\phi_t = \begin{cases} min\{1, \frac{-b+\sqrt{d}}{2a}\} & \text{if } a > 0 \vee (a < 0 \wedge d > 0 \wedge \frac{-b-\sqrt{d}}{2a} > 1) \\ min\{1, -c/b\} & \text{if } a = 0 \\ 1 & \text{otherwise} \end{cases}$$

Online Classification on a Budget: LBP

- Least recent Budget Perceptron (LBP) (*Cavallanti, Cesa-Bianchi and Gentile (2007)*)
- An aggressive variant of Forgetron

For $t = 1 \ldots$

Get new sample $\mathbf{x}_t$

Compute $y = \sum_{i \in S} y_i \alpha_i K(\mathbf{x}_t, \mathbf{x}_i)$

If $y_t y < \beta$ then

if $| S | < B$ then $\{S = S \cup \{t\};\ \alpha_t = 1\}$
else $S = S / min\{S\}$

Online Classification on a Budget: Stoptron

- Stop learning when budget is exceeded (*Orabona, Keshet and Caputo (2008)*)

  For $t = 1 \ldots$

  Get new sample $\mathbf{x}_t$

  Compute $y = \sum_{i \in S} y_i \alpha_i K(\mathbf{x}_t, \mathbf{x}_i)$

  If $y_t y < \beta$ then

  if $| S | < B$ then $\{S = S \cup \{t\};\ \alpha_t = 1\}$
  else $S = S$

Online Classification on a Budget: Randomized Budget Perceptron

- Randomized Budget Perceptron (RBP) (*Cavallanti, Cesa-Bianchi and Gentile (2007)*)

For $t = 1 \ldots$

  Get new sample $\mathbf{x}_t$

  Compute $y = \sum_{i \in S} y_i \alpha_i K(\mathbf{x}_t, \mathbf{x}_i)$

  If $y_t y < \beta$ then

    $S = S \cup \{t\}$
    $\alpha_t = 1$
    if $\mid S \mid = B$ then Remove(S)

Remove(S)

  Select *randomly* $s \in S$
  $S = S / \{s\}$
  return S

Online PA on a Budget

- Budget Passive Aggressive (BPA) (*Wang and Vucetic (2010)*)
- The key idea is to add a new constrain to the PA optimization problem:

$$\mathbf{w}_{t+1} = \underset{w \in \Re^d}{\arg\min} \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C\xi \quad s.t. \ \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \ \ and \ \ \xi \geq 0$$

- The new constrain is:

$$\mathbf{w} = \mathbf{w}_t - \alpha_r \phi(\mathbf{x}_r) + \sum_{i \in V} \beta_i \phi(\mathbf{x}_i)$$

where $\phi(\mathbf{x})$ denotes a mapping from original input space to the feature space: $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi(\mathbf{x}')$

Online PA on a Budget

- The new constraint:

$$\mathbf{w} = \mathbf{w}_t - \alpha_r \phi(\mathbf{x}_r) + \sum_{i \in V} \beta_i \phi(\mathbf{x}_i)$$

- Intuitively BPA is removing support vector $\mathbf{x}_r$ but add new support vector as a linear combination of the support vectors that belongs to V
- This set $V \subseteq S \cup \{t\} - \{r\}$ to be defined
- Then no *new* support vector are added at all if $\{t\}$ is not included in V:

$$\mathbf{w}_t = \sum_{i \in S} \alpha_i \phi(\mathbf{x}_i)$$

$$\mathbf{w} = \sum_{i \in V} (\alpha_i + \beta_i) \phi(\mathbf{x}_i) + \sum_{i \in S - V} \alpha_i \phi(\mathbf{x}_i) \;\; - \;\; \alpha_r \phi(\mathbf{x}_r)$$

Online PA on a Budget

- Denote $\mathbf{w}_{t+1}^{r}$ the solution of the new optimization problem when $\mathbf{x}_r$ is removed
- Find the r^* that minimizes the PA objective function:

$$Q(\mathbf{w}) = \frac{1}{2} \parallel \mathbf{w} - \mathbf{w}_t \parallel^2 + C \cdot H(\mathbf{w}; (\mathbf{x}_t, y_t))$$

$$r^* = \arg\min_{r \in S \cup \{t\}} Q(\mathbf{w}_{t+1}^{r})$$

- Assuming r is known w_{t+1}^{r} is:

$$\mathbf{w}_{t+1}^{r} = \mathbf{w}_t - \alpha_r \phi(\mathbf{x}_r) + \sum_{i \in V} \beta_i \phi(\mathbf{x}_i)$$

where

$$\beta = \alpha_r \mathbf{K}^{-1} \mathbf{k}_r + \tau y_t \mathbf{K}^{-1} \mathbf{k}_t$$

$$\tau = min \left(C, max \left(\frac{1 - y_t(f_t(\mathbf{x}_t) - \alpha_r k_{tr} + \alpha_r (\mathbf{K}^{-1}\mathbf{k}_r)^T \mathbf{k}_t)}{(\mathbf{K}^{-1}\mathbf{k}_t)^T \mathbf{k}_t} \right) \right)$$

- Several choices to define the set V
  - BPA-Simple (BPA-S) $\rightarrow V = \{t\} \rightarrow O(B)$

$$\beta_t = \frac{\alpha_r k_{rt}}{k_{tt}} + \tau y_t$$

$$\tau = min\left(C, \frac{H(\mathbf{w}_t; (\mathbf{x}_t, y_t))}{k_{tt}}\right)$$

  - BPA-Projection (BPA-P) $\rightarrow V = S + \{t\} - \{r\} \rightarrow O(B \cdot B^2) \rightarrow O(B^3)$
  - BPA-Nearest-Neighbor (BPA-NN) $\rightarrow V = \{t\} + NN(r) \rightarrow O(B^2)$

Online PA on a Budget

Table 1: Results on 7 benchmark datasets

Time	Algs	Adult	Banana	Checkerb	NCheckerb	Cover	Phoneme	USPS	Avg
		21K×123	4.3K×2	10K×2	10K×2	10K×54	10K×41	7.3K×256	
		75%	55%	50%	50%	51%	50%	52%	
					Memory-unbounded online algorithms				
$O(N)$	Pcptrn	80.2±0.2	87.4±1.5	96.3±0.6	83.4±0.7	76.0±0.4	78.9±0.6	94.6±0.1	85.3
	(#SV)	(4.5K)	(0.6K)	(0.5K)	(2.8K)	(2.8K)	(2.4K)	(0.4K)	
	PA	83.6±0.2	89.1±0.7	97.2±0.1	95.8±1.0	81.6±0.2	82.6±0.9	96.7±0.1	89.5
	(#SV)	(15K)	(2K)	(2.6K)	(5.9K)	(9.9K)	(7.2K)	(4.5K)	
	PA^R	**84.1**±0.1	**89.3**±0.7	**97.5**±0.1	**96.2**±0.8	**82.7**±0.3	**83.7**±0.7	**96.7**±0.1	90.0
	(#SV)	(4.4K)	(1.5K)	(2.6K)	(3.3K)	(9.8K)	(6.5K)	(4.5K)	
					Budgeted online algorithms (B=100)				
$O(B)$	Stptrn	76.5±2.0	86.7±2.1	87.3±0.9	75.4±4.3	64.2±1.7	67.6±2.7	89.1±1.2	78.1
	Rand	76.2±3.6	84.1±2.6	85.6±1.2	69.4±2.9	61.3±3.2	65.0±4.4	87.1±0.9	75.5
	Fogtrn	72.8±6.1	82.8±2.4	86.1±1.0	68.2±3.5	60.8±2.7	65.6±1.2	86.2±2.1	74.6
	PA+Rnd	78.4±1.9	84.9±2.1	83.3±1.4	75.1±3.6	63.1±1.5	64.0±3.9	86.2±1.1	76.4
	BPA-S	82.4±0.1	89.4±1.3	90.0±0.8	87.4±0.7	68.6±1.9	67.4±3.0	89.6±1.3	82.1
	BPA^R-S	82.4±0.1	89.5±1.7	90.0±1.0	88.2±1.2	69.3±1.8	67.0±3.2	89.3±1.2	82.2
	BPA-NN	82.8±0.4	89.6±1.4	94.0±1.2	90.2±1.3	69.1±1.8	74.3±0.7	90.8±0.9	84.4
	BPA^R-NN	**83.1**±0.0	89.8±1.1	94.2±0.9	**92.3**±0.5	70.3±0.8	74.6±0.8	90.8±0.6	85.0
$O(B^2)$	Pjtrn++	80.1±0.1	89.5±1.1	**95.4**±0.7	88.1±0.7	68.7±1.0	74.6±0.7	89.2±0.7	83.7
$O(B^3)$	BPA-P	83.0±0.2	**89.6**±1.1	**95.4**±0.7	91.7±0.8	**74.3**±1.4	**75.2**±1.0	**92.8**±0.7	86.0
	BPA-P^R	**84.0**±0.0	**89.6**±0.8	95.2±0.8	**94.1**±0.9	**75.0**±1.0	**74.9**±0.6	**92.6**±0.7	86.5
					Budgeted online algorithms (B=200)				
$O(B)$	Stptrn	78.7±1.8	85.6±1.5	92.8±1.1	76.0±3.1	65.5±2.3	70.5±2.6	92.3±0.7	80.2
	Rand	76.4±2.8	83.6±2.0	90.3±1.3	74.5±2.1	62.4±2.4	67.3±2.5	89.8 ±1.1	77.8
	Fogtrn	72.9±6.8	85.0±1.3	90.9±1.7	72.2±4.4	62.1±2.8	68.0±2.3	90.3±0.9	77.3
	PA+Rnd	80.1±2.4	86.7±1.9	87.0±1.3	78.3±1.8	64.2±2.7	68.7±4.3	88.8±0.8	79.1
	BPA-S	82.7±0.2	89.5±0.7	93.4±0.5	89.7±0.9	71.7±1.7	71.3±2.3	92.6±0.9	84.4
	BPA^R-S	83.1±0.1	89.5±0.9	93.9±0.6	90.8±0.8	71.7±1.2	71.6±2.2	92.1±0.6	84.7
	BPA-NN	83.1±0.4	89.6±1.1	95.5±0.4	91.7±1.3	72.7±1.0	75.8±1.0	92.8±0.6	85.9
	BPA^R-NN	83.3±0.4	89.5±1.4	95.2±0.5	**93.3**±0.6	72.7±1.4	77.2±1.7	94.0±0.4	86.5
$O(B^2)$	Pjtrn++	82.9±0.1	89.5±1.2	**95.8**±0.5	92.5±1.0	75.1±2.0	75.2±0.6	93.2±0.6	86.3
$O(B^3)$	BPA-P	**83.8**±0.0	**89.7**±0.7	**95.9**±0.6	92.8±0.7	**76.0**±1.3	**78.0**±0.3	**94.8**±0.3	87.3
	BPA^R-P	**84.6**±0.0	**90.3**±1.5	95.6±1.2	**94.5**±1.1	**76.3**±1.0	**77.6**±0.6	**94.8**±0.3	87.7

Online Learning Applications

Online Learning for Video Tagging

Introduction

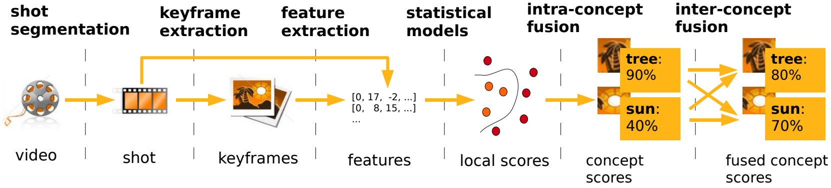
- Video Tagging (concept detection) is a key block of video retrieval systems
- Generally solved by means of casting the concept detection as a binary classification problem
- SVM's can be considered state-of-the-art to solve such binary problems
- Video Tagging must cover a wide range of potential users to gain attraction
- SVM's scales poorly in such scenario

Video Tagging

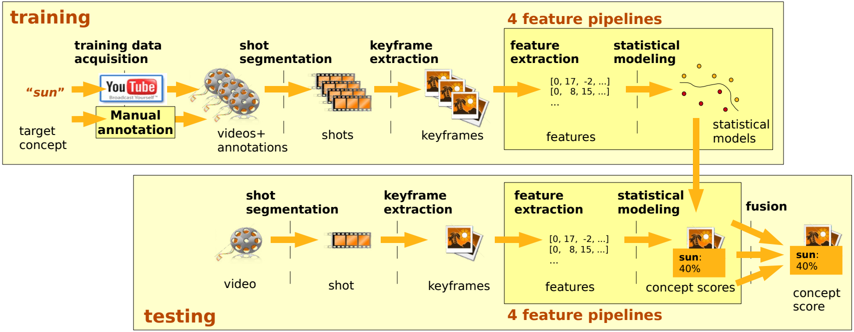
- Each concept is treated as a binary problem
- The video is processed and *shots* are detected
- For each shot one (or more) key-frames are extracted
- Each key-frame (image) is usually represented by bag of words (visual terms)
- For each key-frame the concept presence is evaluated
- Final decision for the whole video is evaluated by means of the fusion of the key-frame scores

Video Tagging

- Tagging process:



- Training and Testing:



Video Tagging

- A video $\mathcal{X}$ is represented by a set of key-frames $x_1, .., x_n$
- A score $sc(c, x_i)$ is assigned to each pair key-frame x_i and concept c
- The score at video level can be computed by fusing the scores of the different key frames:

$$sc(c, \mathcal{X}) = Fusion(sc(c, x_1), \cdots, sc(c, x_n))$$

- In the present work we propose the following avg+max fusion:

$$sc(c, \mathcal{X}) = \frac{1}{n} \sum_{i=1}^{n} sc(c, x_i) \; + \max_{1<=i<=n} sc(c, x_i)$$

Linear Approaches

- Assuming a vectorial representation of the key-frames x_i, $\mathbf{x}_i \in \Re^d$
- Computing the key-frame score as follows:

$$sc(c, x_i) = \mathbf{w}_c \mathbf{x}_i$$

  $\mathbf{w}_c$ is a weight vector associated to concept c

- This *linear* approach has the following properties:
  - is very compact, few bytes per concept
  - is very fast and simple to compute
  - can take profit of the sparsity of the vectorial representation $\mathbf{x}_i$

Online Learning: PAMIR

- We propose to maximize the following discriminative index[1] :

$$J(\mathbf{w}_c) = \sum_{\forall \mathbf{x}_p \in X_p} \sum_{\forall \mathbf{x}_n \in X_n} (\mathbf{w}_c \mathbf{x}_p - \mathbf{w}_c \mathbf{x}_n)$$

  $\mathbf{x}_p \in X_p$ is a key-frame of a *positive* video
  $\mathbf{x}_n \in X_n$ is a key-frame of a *negative* video

- very costly optimization problem, $O(\mid X_p \mid \mid X_n \mid)$

[1] D. Grangier and S. Bengio. “A discriminative kernel-based model to rank images from text queries”. IEEE Transactions on PAMI, 30(8):1371–1384, 2008.

- The weight vector $\mathbf{w}_c$ is estimated using an online iterative procedure solving:

$$\mathbf{w}_c^i = \arg\min \ \frac{1}{2} \mid\mid \mathbf{w}_c - \mathbf{w}_c^{i-1} \mid\mid^2 \ + \ \mathcal{C} \ l(\mathbf{w}_c; \mathbf{x}_p, \mathbf{x}_n)$$

where i is the iteration and $\mathcal{C}$ is the *aggressiveness* parameter

- The cost function $l(\cdot)$ is the hinge loss function:

$$l(\mathbf{w}_c; \mathbf{x}_p, \mathbf{x}_n) = \begin{cases} 0 & \mathbf{w}_c(\mathbf{x}_p - \mathbf{x}_n) > 1 \\ 1 - \mathbf{w}_c(\mathbf{x}_p - \mathbf{x}_n) & \textit{otherwise} \end{cases}$$

- The solution to this minimization is:

$$\mathbf{w}_c^i = \mathbf{w}_c^{i-1} + \Gamma^i(\mathbf{x}_p - \mathbf{x}_n)$$

where the Lagrange multiplier Γ^i is:

$$\Gamma^i = \min\left\{\mathcal{C}, \frac{l(\mathbf{w}_c; \mathbf{x}_p, \mathbf{x}_n)}{\|\ \mathbf{x}_p - \mathbf{x}_n\ \|^2}\right\}$$

- When the loss $l(\mathbf{w}_c; \mathbf{x}_p, \mathbf{x}_n)$ is zero no model update is performed

Stochastic Gradient Descent: MROC

- We proposed to maximize the AROC for the binary problem defined by the positive and negative key-frames [2]:

$$J(\mathbf{w}_c) = \frac{1}{|X_p||X_n|} \sum_{\forall \mathbf{x}_p \in X_p} \sum_{\forall \mathbf{x}_n \in X_n} step(\mathbf{w}_c \mathbf{x}_p - \mathbf{w}_c \mathbf{x}_n)$$

  where $step(\cdot)$ is the step function centered at 0.

- This index is optimized following a gradient descent approach
- Sigmoid function instead of step:

$$S_\beta(z) = \frac{1}{1 + \exp(-\beta z)} .$$

[2] M. Villegas and R. Paredes. "Score Fusion by Maximizing the Area Under the ROC Curve". IbPria'09, volume 5524 of LNCS, pages 473–480,June 2009

Stochastic Gradient Descent: MROC

- The index gradient is:

$$\frac{\partial J(\mathbf{w}_c)}{\partial \mathbf{w}_c} = \frac{1}{\mid X_p \mid \mid X_n \mid} \sum_{\forall \mathbf{x}_p \in X_p} \sum_{\forall \mathbf{x}_n \in X_n} sigm'(\mathbf{w}_c \mathbf{x}_p - \mathbf{w}_c \mathbf{x}_n)(\mathbf{x}_p - \mathbf{x}_n)$$

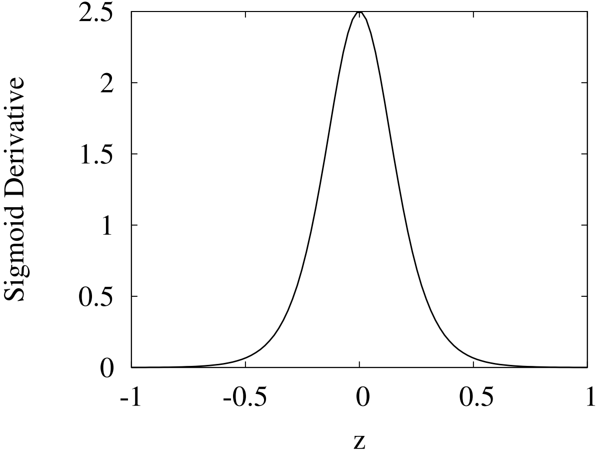
- The weight vector $\mathbf{w}_c$ update is:

$$\mathbf{w}'_c = \mathbf{w}_c + \mu \frac{\partial J(\mathbf{w}_c)}{\partial \mathbf{w}_c}$$

- Pairs $(\mathbf{x}_p, \mathbf{x}_n)$ are randomly selected from the pool of all the possible pairs, stochastic gradient descent.

Stochastic Gradient Descent: MROC

- Plot of sigmoid derivative using $\beta = 10$:



- Saving $\mathbf{w}_c$ updates discarding pairs with low values of $sigm'(\cdot)$

Experiments

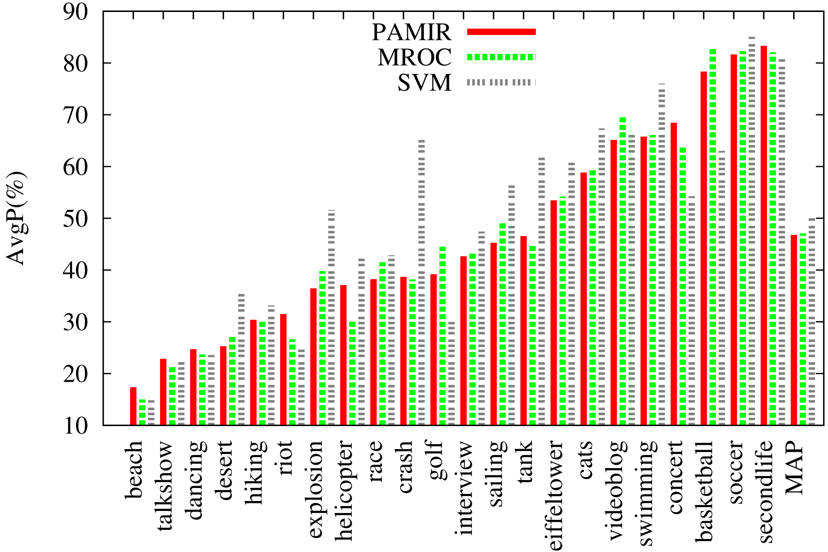
- Dataset: Youtube-22Concepts
- The overall length of the dataset is about 194 hrs
- 2,200 real-world online video clips for 22 concepts
- 75% training and 25% testing was used
- SIFT descriptors clustered to a vocabulary of 2,000 visual words
- Performance: *Average Precision* averaged for all concepts, *mean* average precision (MAP)
- Comparing SVM(libsvm), linearSVM(liblinear), PAMIR and MROC (C-implementation)

Results

- MAP results and time (secs) required for training one concept:

Method / #samples		MAP	Time (secs)
SVM(χ^2)	200	25.5%	94
SVM(χ^2)	1500	42.5%	525
SVM(lin.)	1500	36.2%	117
SVM(liblinear)	1500	34.5%	17
SVM(χ^2)	9000	52.3%	16540
SVM(liblinear)	9000	42.6%	91
PAMIR	9000	46.8%	37
MROC	9000	47.0%	24

Results per concept



Results

- Test time (secs) required for all the test key-frames:

Method / training samples		Time (secs)
SVM(χ^2)	200	89
SVM(χ^2)	1500	456
SVM(χ^2)	9000	2040
linear	9000	0.2

Conclusions

- Linear approaches can be applied to Video Tagging with *relatively* good results
- We obtain a fast and compact classifier
- The linear model is easy to update, new pairs
- Better (high dimensional) representations of the images could provide a linear separation of the concept space

Online Learning for Relevance Feedback on Image Retrieval

- Relevance Feedback on Image Retrieval can be considered an online learning problem
- For a given set of images retrieved the users judge the relevance of each image to the query introduced
- The set of relevant and non-relevant images forms an online set of samples
- Then, a linear classifier must be found in order to discriminate between relevant and non-relevant images
- This linear classifier is applied to the complete set of images available
- Notation and methodology similar to the Video Tagging problem
- An example: RISE demo

Datasets

- Two different datasets, Corel and MSRC
  - Corel has $1,000$ images and 10 different classes
  - MSRC has $4,325$ images and 33 different classes
- MSRC is a more challenging task
- Evaluation measure: Average Precision (AvgP%)

Results

Table: Average Precision for Corel dataset along 5 iterations of RF

Method	It 1	It 2	It 3	It 4	It 5
Relevance Score	50.0	57.6	61.2	62.62	62.96
PA - Linear	50.0	46.6	57.9	60.8	61.6
PA - RBF	50.0	47.4	50.5	52.0	52.3
PA - HI	50.0	59.5	62.1	63.6	64.5

Results

Table: Average Precision for MSRC dataset along 5 iterations of RF

Method	It 1	It 2	It 3	It 4	It 5
Relevance Score	21.3	22.8	23.9	24.4	24.7
PA - Linear	21.3	18.4	21.2	22.0	22.4
PA - RBF	21.3	20.4	21.2	21.5	21.6
PA - HI	21.3	24.9	25.7	26.9	26.3

References

- F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain”. Psychological Review, 65:386-407, 1958
- B. Novikoff. “On convergence proofs on perceptrons”. In Proceedings of the Symposium on the Mathematical Theory of Automata, 1962
- V. N. Vapnik. “Statistical Learning Theory”. Wiley, 1998.
- R. O. Duda, P. E. Hart and D. G. Stork. “Pattern Classification (2nd ed.)”. John Wiley and Sons, 2001.
- Claudio Gentile. “A New Approximate Maximal Margin Classification Algorithm”. JMLR, 2(Dec):213-242, 2001.
- Koby Crammer and Yoram Singer. “Ultraconservative Online Algorithms for Multiclass Problems”. JMLR, 3:951-991, 2003.
- Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, Yoram Singer. “Online Passive-Aggressive Algorithms”. JMLR, 7(Mar):551–585, 2006.
- Giovanni Cavallanti, Nicolò Cesa-Bianchi and Claudio Gentile. “Tracking the best hyperplane with a simple budget Perceptron”. Machine Learning, 69:143 - 167 , 2007
- Ofer Dekel, Shai Shalev-Shwartz and Yoram Singer. “The Forgetron: A Kernel-Based Perceptron on a Budget”. SIAM J. Comput, 37: 1342-1372, 2008
- Zhuang Wang and Slobodan Vucetic. “Online Passive-Aggressive Algorithms on a Budget”. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010.