UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Chapter 6. Online Learning

Neural Networks

2023/2024

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Departamento de Sistemas Informáticos y Computación

Index

Index

Background: offline or batch approach

- Let $X = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_N, t_N)\}$, $\mathbf{x}_n \in \mathbb{R}^D$, $t_n \in Y$ ($Y \equiv \{1, \ldots, C\}$ or $Y \equiv \mathbb{R}$) (supervised) be a training set and $g_{\mathbf{w}} : \mathbb{R}^D \to Y$ be a model with parameters $\mathbf{w}$.

- $q_X(\mathbf{w})$ be an objective function to be optimized:

$$q_X(\mathbf{w}) \;=\; \frac{1}{N}\sum_{n=1}^{N} q_n(\mathbf{w})$$

  where:
  - Negative likelihood: $q_n(\mathbf{w}) \;=\; -\log p(t_n \mid \mathbf{x}_n; \mathbf{w})$
  - A loss function: $q_n(\mathbf{w}) \;=\; L(t_n, g_{\mathbf{w}}(\mathbf{x}_n))$

- Solution: Applying gradient descent to $q_X(\mathbf{w})$:

$$\mathbf{w}_{k+1} \;=\; \mathbf{w}_k - \rho\, \nabla_{\mathbf{w}}\, q_X(\mathbf{w})\,|_{\mathbf{w}_k} \;=\; \mathbf{w}_k - \rho\, \sum_{n=1}^{N} \nabla_{\mathbf{w}} q_n(\mathbf{w})\,|_{\mathbf{w}_k}$$

Introduction

- With streaming data, offline learning is not adequate.
- The distribution of the data is unknown (or change over the time), the data have not ever seen before.
- Online Learning: A procedure for obtaining a machine learning model that uses an unique sample (new) at each iteration.
- *Could we get good models processing an unique sample at each iteration?*

Basic scheme of the Online Learning

- Repeat as long as new samples exist

  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 decrease the loss.

Online Learning

- Problem: Catastrophic forgetting.
- Other related ML topics:
  - Continual learning (to prevent catastrophic forgetting)
  - Lifelong Machine Learning (Multi-task learning)
  - Incremental learning.
- Incremental and Offline Learning

$$q_X(\mathbf{w}) = \sum_{n=1}^{N} q_n(\mathbf{w}) = q_k(\mathbf{w}) + \sum_{n=1:n\neq k}^{N} q_n(\mathbf{w}) = q_k(\mathbf{w}) + \epsilon$$

Therefore $\nabla_{\mathbf{w}} q_k(\mathbf{w})$ can be seen as a rude (noisy) approach of $\nabla_{\mathbf{w}} q_X(\mathbf{w})$.

Online Learning: Optimizing with respect to the past

- Given a sample $(\mathbf{x}_n, t_n)$, a new model $g_{\mathbf{w}_n}$ with parameter $\mathbf{w}_n$ should be learnt for $1 \leq n \leq k$.

- Optimizing with respect to past: The regret concept $R(\mathbf{w}_1, \ldots, \mathbf{w}_k)$:

$$R(\mathbf{w}_1, \ldots, \mathbf{w}_k) \; = \; \frac{1}{k} \sum_{n=1}^{k} q_n(\mathbf{w}_n) - \min_{\mathbf{w}} \frac{1}{k} \sum_{n=1}^{k} q_n(\mathbf{w})$$

- Online gradient descent: $\mathbf{w}_{n+1} \; = \; \mathbf{w}_n - \rho_n \, \nabla_{\mathbf{w}} \, q_n(\mathbf{w}) \,|_{\mathbf{w}_n}$

- The rule online gradient descent minimizes $R(\mathbf{w}_1, \ldots, \mathbf{w}_k)$.

Online Learning: Optimizing with respect to the expected loss

- Given a sample $(\mathbf{x}, t)$ a new model $g_{\mathbf{w}}$ with parameter $\mathbf{w}$ should be learnt.

- Optimizing with respecto to the expected loss in the future:

$$q(\mathbf{w}) \;=\; \mathbb{E}_{\mathbf{x},t}[q_{\mathbf{x},t}(\mathbf{w})] \;=\; \int_{\mathbf{x}} \sum_{t} p(\mathbf{x}, t)\; q_{\mathbf{x},t}(\mathbf{w}) d\mathbf{x}$$

- As an approximation, given a stream data $(\mathbf{x}_n, t_n)$, for $1 \leq n \leq k$ we apply online gradient descent to obtain a sequence $\mathbf{w}_n$: $\mathbf{w}_{n+1} \;=\; \mathbf{w}_n - \rho_n\, \nabla_{\mathbf{w}}\, q_n(\mathbf{w})\, |_{\mathbf{w}_n}$

- To minimize the expected loss $q(\mathbf{w})$: A running average (Polyak-Ruppert averaging):

$$\overline{\mathbf{w}}_k \;=\; \frac{1}{k} \sum_{n=1}^{k} \mathbf{w}_n$$

- Recursive computation: $\overline{\mathbf{w}}_k \;=\; \overline{\mathbf{w}}_{k-1} - \frac{1}{k}(\overline{\mathbf{w}}_{k-1} - \mathbf{w}_k)$

Index

Linear Models

- Linear discriminant models:

$$g : \mathbb{R}^{d_o} \rightarrow \mathbb{R} : g(\mathbf{x}) \;=\; \mathbf{w} \cdot \mathbf{x} + w_0$$

  where $w_0 \in \mathbb{R}$ and $\mathbf{x}, \mathbf{w} \in \mathbb{R}^{d_o}$

- Normally we use a *compact* notation:

$$g(\mathbf{x}) \;=\; \mathbf{w} \cdot \mathbf{x}$$

  where the new $\mathbf{w} = \{w_0, w_1, w_2, \ldots, w_{d_o}\}$ and $\mathbf{x} = \{1, x_1, x_2, \ldots, x_{d_o}\}$

- Let be the new $d = d_o + 1$ then $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$

Linear Models

- Linear models for classification (2 classes):

$$G(\mathbf{x}) \;=\; \mathsf{sgn}(g(\mathbf{x}))$$

- Linear models for classification (C classes):

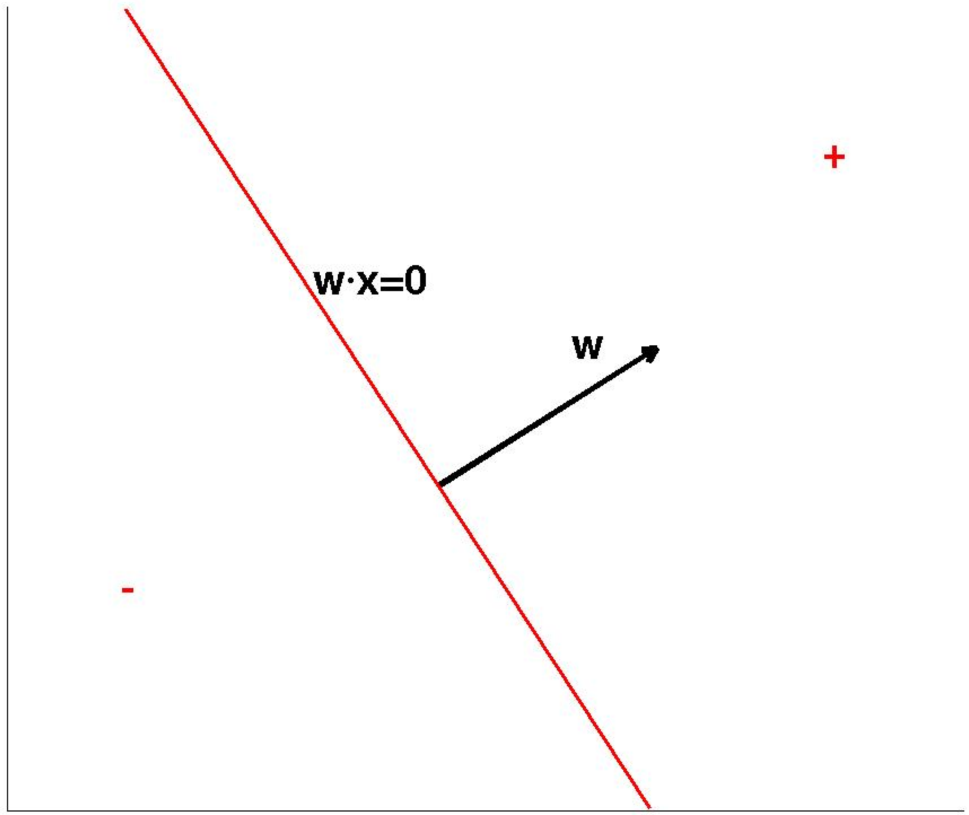
$$g_c(\mathbf{x}) \;=\; \mathbf{w}_c \cdot \mathbf{x} + w_{c,0} \;\text{ for }\; 1 \leq c \leq C$$

$$G(\mathbf{x}) \;=\; \underset{c}{\mathsf{argmax}}\, g_c(\mathbf{x})$$

- Linear models for regression:

$$g : \mathbb{R}^d \rightarrow \mathbb{R}$$

Linear Models for 2 class classification



Index

Linear Models: Perceptron

- The goal: Given a set of data $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{-1. + 1\}$ for $1 \leq n \leq T$

  find a $\mathbf{w}$ that gives the minimum classification error:

$$y_n \ \mathbf{w}^t \mathbf{x}_n \geq 0 \ \text{ for } 1 \leq n \leq T$$

- Minimize $q_X : \mathbb{R}^d \rightarrow \mathbb{R}^{\geq 0}$

$$q_X(\mathbf{w}) \ = \ \sum_{\substack{(\mathbf{x},y) \in X \\ y \ \mathbf{w}^t \mathbf{x} < 0}} -y \mathbf{w}^t \mathbf{x}$$

- Solution (batch perceptron): Gradient descent.

$$\mathbf{w}_{t+1} \ = \ \mathbf{w}_t + \rho_k \sum_{\substack{(\mathbf{x},y) \in X \\ y \ \mathbf{w}_t^t \mathbf{x} < 0}} y \ \mathbf{x}$$

Linear Models: Batch Perceptron

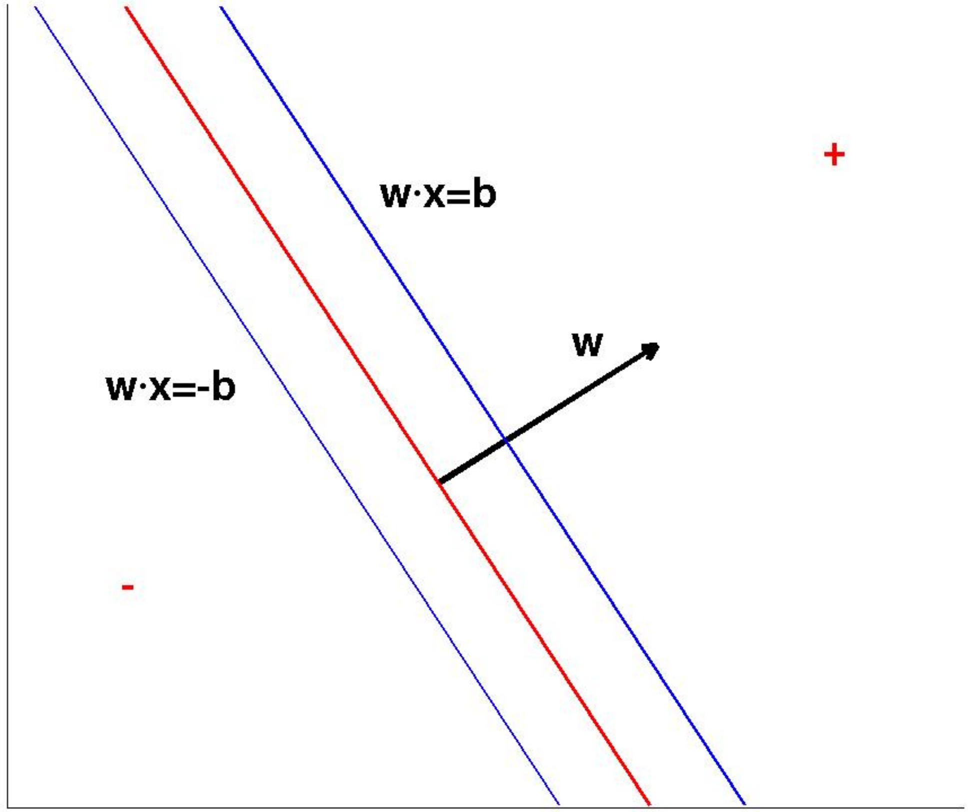
- Given a set of training data: X.
- Initialization $\mathbf{w} \;=\;$ random
- While no convergence
  - Initialize $\Delta \;=\; 0$
  - for each training sample $(\mathbf{x}, y) \in X$
    * $g \;=\; y\,\mathbf{w}^t\mathbf{x}$
    * If $g < 0$ then $\Delta += \; y\,\mathbf{x}$ # error
  - $\mathbf{w} += \; \rho\,\Delta$ # uptating the weights
- Output: $\mathbf{w}$

One weight updating by epoch

Linear Models: Perceptron with margin

- Given a set of training data: X.
- Given a margin $b \in \mathbb{R}^{\geq 0}$
- Initialization $\mathbf{w} \;=\;$ random
- While no convergence
  - Initialize $\Delta \;=\; 0$
  - for each training sample $(\mathbf{x}, y) \in X$
    * $g \;=\; y\ \mathbf{w}^t\mathbf{x}$
    * If $g < b$ then $\Delta += \; y\ \mathbf{x}$ # error
  - $\mathbf{w} += \; \rho\ \Delta$ # uptating the weights
- Output: $\mathbf{w}$

Linear Models: Perceptron



Linear Models: Incremental Perceptron

- Given a set of training data: X.

- An incremental approach:

$$\mathbf{w}_{t+1} = \begin{cases} \mathbf{w}_t & \text{if } \; y \; \mathbf{w}_t^t \mathbf{x} \geq 0 \\ \mathbf{w}_t + \rho_k \; y_t \; \mathbf{x}_t & \text{if } \; y \; \mathbf{w}_t^t \mathbf{x} < 0 \end{cases}$$

  where $(\mathbf{x}_t, y_t)$ is random sampled from X.

- Incremental algorithm (In the worst case one weight updating by sample):
  - Given a set of training data: X.
  - Initialization $\mathbf{w} = $ random
  - While no convergence
    * for each training sample $(\mathbf{x}, y) \in X$
      · $g = y \; \mathbf{w}^t \mathbf{x}$
      · If $g < 0$ then $\Delta = y \; \mathbf{x}$ # error
      · $\mathbf{w} += \rho \; \Delta$ # uptating the weights
  - Output: $\mathbf{w}$

Linear Models: Online Perceptron

- An online approach: The same updating rule

- Online algorithm:
  - Given a set of weights: $\mathbf{w}$
  - Given a training sample: $(\mathbf{x}, y)$.
  - $g \;=\; y\; \mathbf{w}^t\mathbf{x}$
  - If $g < 0$ then $\Delta \;=\; y\; \mathbf{x}$ # error
  - $\mathbf{w} \mathrel{+}= \;\rho\; \Delta$ # uptating the weights
  - Output: $\mathbf{w}$

Linear Models: Properties

- Convergence: If the training set X is linearly separable the batch and incremental perceptron converge in a finite number of iterations.

- Given a training set $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ the general form of the output $\mathbf{w}$ of the batch and incremental perceptron is:

$$\mathbf{w} \;=\; \sum_{i=1}^{T} \alpha_i \; y_i \; \mathbf{x}_i \qquad \alpha_i \in \mathbb{R}^{\geq 0}$$

- After the processing of streaming data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_t, y_t)\}$ the general form of the output $\mathbf{w}$ of the online perceptron is:

$$\mathbf{w} \;=\; \sum_{i=1}^{t} \alpha_i \; y_i \; \mathbf{x}_i \qquad \alpha_i \in \{0, 1\}$$

- Well-known algorithms like: *Relaxed Online Maximum Margin Algorithm (ROMMA)*, *Approximate Maximal Margin Classification Algorithm (ALMA)*, *Margin Infused Relaxed Algorithm (MIRA)* and *Pocket Perceptron*.

Index

Linear Models: Adaline

- The goal: Given a set of data $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_T, y_T)\}$ with $\mathbf{x}_1 \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for $1 \leq i \leq T$

  find a $\mathbf{w}$ that gives the minimum classification error:

$$\mathbf{w}^t \mathbf{x}_n \;=\; y_n \;\text{ or }\; (\mathbf{w}^t \mathbf{x}_n \;\approx\; y_n) \;\text{ for } 1 \leq n \leq T$$

- Minimize $q_X : \mathbb{R}^d \to \mathbb{R}^{\geq 0}$

$$q_X(\mathbf{w}) \;=\; \frac{1}{2} \sum_{n=1}^{T} (\mathbf{w}^t \mathbf{x}_n - y_n)^2$$

- Solution (batch): Gradient descent.

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t + \rho_k \sum_{n=1}^{T} (y_n - \mathbf{w}_t^t \, \mathbf{x}_n) \, \mathbf{x}_n$$

Linear Models: Batch Adaline

- Given a set of training data: X.
- Initialization $\mathbf{w} \ = \ $ random
- While no convergence
  - Initialize $\Delta \ = \ 0$
  - for each training sample $(\mathbf{x}, y) \in X$
    - $\Delta += \ (y - \mathbf{w}^t\mathbf{x})\ \mathbf{x}$ # error
  - $\mathbf{w} += \ \rho\, \Delta$ # uptating the weights
- Output: $\mathbf{w}$

One weight updating by epoch

Asymptotic convergence

Linear Models: Incremental Adaline

- Given a set of training data: X.

- An incremental approach:

$$\mathbf{w}_{t+1} \;=\; \mathbf{w}_t + \rho_k \left(y_t - \mathbf{w}_t^t \mathbf{x}_t\right) \mathbf{x}_t$$

  where $(\mathbf{x}_t, y_t)$ is random sampled from X.

- Incremental algorithm: (one weight updating by sample):
  - Given a set of training data: X.
  - Initialization $\mathbf{w} \;=\;$ random
  - While no convergence
    - for each training sample $(\mathbf{x}, y) \in X$
      - $\Delta \;=\; (y - \mathbf{w}^t \mathbf{x})\, \mathbf{x}$ # error
      - $\mathbf{w} \mathrel{+}= \rho\, \Delta$ # uptating the weights
  - Output: $\mathbf{w}$

Linear Models: Online Adaline

- An online approach: The same updating rule

- Online algorithm:
  - Given a set of weights: $\mathbf{w}$
  - Given a training sample: $(\mathbf{x}, y)$.
  - $\Delta \ = \ (y - \mathbf{w}^t\mathbf{x})\ \mathbf{x}$ # error
  - $\mathbf{w} += \ \rho\, \Delta$ # uptating the weights
  - Output: $\mathbf{w}$

Index

Online Learning: Kernel Perceptron

- Perceptron and Support Vector Machines.
- 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.

Kernel Perceptron

- General model: Kernel Perceptron
  - Linear Perceptron. From a training set X:

$$\mathbf{w} \;=\; \sum_{i=1}^{T} \alpha_i \; y_i \; \mathbf{x}_i \Rightarrow g(\mathbf{x}) \;=\; \sum_{i=1}^{T} \alpha_i \; y_i \; \mathbf{x}_i^t \; \mathbf{x}$$

  - Kernel extension:

$$g(\mathbf{x}) \;=\; \sum_{i=1}^{T} \alpha_i \; y_i \; K(\mathbf{x}_i, \mathbf{x})$$

- The weight α_i can be seen as the *importance* of $\mathbf{x}_i$.

Kernel Perceptron

- Incremental Kernel Perceptron algorithm:
  - Given a training set X
  - Initialization $\mathbf{w} \;=\;$ random
  - While no convergence
    - for each training sample $(\mathbf{x}_n, y_n) \in X$
      - $g \;=\; y_n \displaystyle\sum_{i=1}^{T} \alpha_i \; y_i \; K(\mathbf{x}_i, \mathbf{x}_n)$
      - If $g < 0$ then $\alpha_n{+}{+}$ # error
  - Output: $\mathbf{w}$

Online Learning: Kernel Perceptron

- General model: Kernel Perceptron
  - Linear Online Perceptron. For the sample $\mathbf{x}_t$:

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

  - Kernel extension:

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

- The weight α_i can be seen as the *importance* of $\mathbf{x}_i$.

Online Kernel Perceptron

- Given de previous $t-1$ training samples $(\mathbf{x}_i, y_i)$ and α_i for $1 \leq i \leq t-1$
- Given a new training sample $(\mathbf{x}, y)$
- $g \;=\; y \displaystyle\sum_{i=1}^{t-1} \alpha_i \; y_i \; K(\mathbf{x}_i, \mathbf{x})$
- If $g < 0$ then $\alpha_t = 1$ else $\alpha_t = 0$ # error
- Output: α_t

Index

Passive-Aggressive (PA) Online Learning

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

- 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

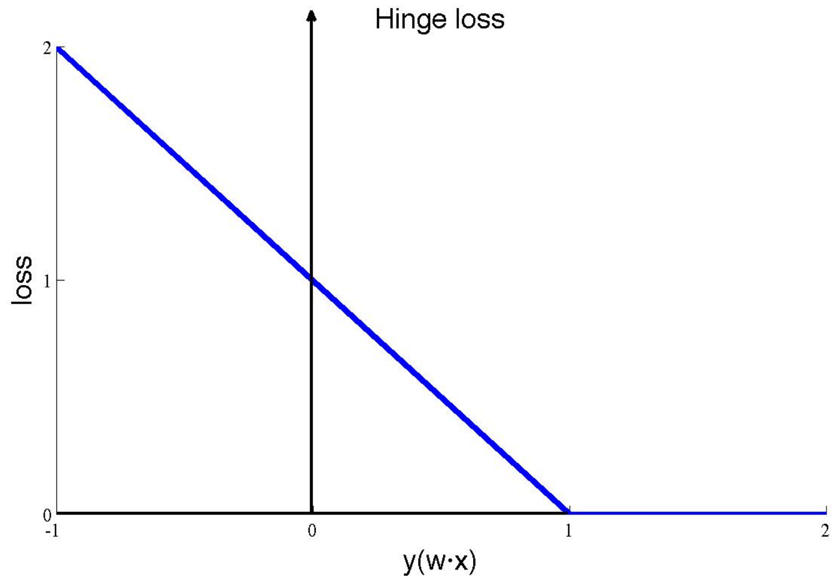
  - We have to solve how to preserve the previous learning

  $\rightarrow$ 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 $y = \text{sgn}(\mathbf{w}_t^t \mathbf{x}_t)$
  - The hinge loss is
    $$\ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = \max(0, 1 - y_t(\mathbf{w}^t \mathbf{x}_t)) = \begin{cases} 0 & y_t(\mathbf{w}^t \mathbf{x}_t) \geq 1 \\ 1 - y_t(\mathbf{w}^t \mathbf{x}_t) & \text{otherwise} \end{cases}$$



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

Passive-Aggressive (PA) Online Learning

- Minimization problem (Crammer et al. 2006):

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \qquad \text{s.t.} \;\; \ell_t \; = \; \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0$$

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

- If $\ell(\mathbf{w}_t; (\mathbf{x}_t, y_t)) = 0$, the minimum is $\mathbf{x}_t$, the problem appears when $\ell(\mathbf{w}_t; (\mathbf{x}_t, y_t)) > 0$

Passive-Aggressive (PA) Online Learning

- Lagrange function:

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

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

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

- Dual Lagrange function, plugging back to the Lagrangian equation:

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

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

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

- Final solution: $\mathbf{w}_{t+1} \ = \ \mathbf{w}_t + \tau y_t \mathbf{x}_t \qquad \tau = \dfrac{\ell(\mathbf{w}_t; (\mathbf{x}_t, y_t))}{\mid\mid \mathbf{x}_t \mid\mid^2}$

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 k the model could *forget* what has learned before, $\mid\mid \mathbf{w}_{t+1} - \mathbf{w}_t \mid\mid^2 \quad \uparrow\uparrow$

- 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:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 + C\xi \quad \text{s.t.} \;\; \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \;\; \text{and} \;\; \xi \geq 0$$

- Larger values of C imply a more aggressive update strategy

Passive-Aggressive (PA) Online Learning

- Two models:
  - PA-I

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 + C\xi \quad \text{s.t.} \;\; \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi \;\; \text{and} \;\; \xi \geq 0$$

  - PA-II

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 + C\xi^2 \quad \text{s.t.} \;\; \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi$$

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

Passive-Aggressive (PA) Online Learning

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

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

  - PA-II

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

- In both cases: $\mathbf{w}_{t+1} = \mathbf{w}_t + \tau_t \, y_t \, \mathbf{x}_t$ and $\ell_t \; = \; \ell(\mathbf{w}_t; (\mathbf{x}_t, y_t))$

Passive-Aggressive (PA) Online Learning

- Given a set of weights: $\mathbf{w}$
- Receive sample $\mathbf{x}$
- Compute $g = \mathbf{w}^t\mathbf{x}$
- Receive correct label y
- If $y \neq \text{sgn}(g)$ then
  - Compute loss, $\ell = \max\{0, 1 - yg\}$
  - Compute $\tau = \min\left\{C, \frac{\ell}{||\mathbf{x}||^2}\right\}$ (PA-I)
  - Update $\mathbf{w} = \mathbf{w} + \tau\, y\, \mathbf{x}$
- Output $\mathbf{w}$

PA with kernels

- The linear model is compact, all the model is stored in $\mathbf{w}$

$$\mathbf{w}_t = \sum_{i=1}^{k-1} \tau(i) y(i)\, \mathbf{x}(i)$$

$$\mathbf{w}_t\, \mathbf{x}_t = \sum_{i=1}^{k-1} \tau(i)\, y(i)\, (\mathbf{x}_t \mathbf{x}(i))$$

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

$$\mathbf{w}_t^t \mathbf{x}_t = \sum_{i=1}^{k-1} \tau(i)\, y(i)\, K(\mathbf{x}_t, \mathbf{x}(i))$$

- How is the algorithm affected ?

PA for Regression

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

$$\ell \;=\; \max(0, |\, \mathbf{w}\mathbf{x} - y \,|)$$

- Similar optimization problem:

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2} \,||\, \mathbf{w} - \mathbf{w}_t \,||^2 \qquad s.t. \;\; \ell(\mathbf{w}; (\mathbf{x}_t, y_t)) = 0$$

- Solution:

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

PA for Regression

- A different loss is required:

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

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

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

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

PA for multiclass problems

- Define $\mathbf{w}^r \in \mathbb{R}^d$ with $1 \leq r \leq C$.

- Notation: Let $\mathbf{W}$ be a matrix which the r-th row is $\mathbf{w}^r$.

- Simplified constrained optimization:

$$\mathbf{W}_{t+1} = \underset{\mathbf{W}}{\text{argmin}} \frac{1}{2} \mid\mid \mathbf{W} - \mathbf{W}_t \mid\mid^2 \quad \text{s.t.} \;\; \ell_t \; = \; (\mathbf{w}^{y_t}\mathbf{x}_t - \mathbf{w}^{s_t}\mathbf{x}_t) \geq 1$$

where $s_t = \text{argmax}_{i \in \{1 \ldots M\},\ i \neq y_t} \;\; \mathbf{w}^i \mathbf{x}_t$

- Solution:

$$\mathbf{w}_{t+1}^{y_t} = \mathbf{w}_t^{y_t} + \tau_t \, \mathbf{x}_t \;\; \text{and} \;\; \mathbf{w}_{t+1}^{s_t} = \mathbf{w}_t^{s_t} - \tau_t \mathbf{x}_t$$

where $\tau_t = \dfrac{\ell_t}{2 \mid\mid \mathbf{x}_t \mid\mid^2}$

Generalization for multiclass problems

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

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}}{\text{argmin}} \frac{1}{2} \mid\mid \mathbf{w} - \mathbf{w}_t \mid\mid^2 \quad \text{s.t. } \ \ell_t \ = \ \mathbf{w}^t(\Phi(\mathbf{x}_t, y_t) - \Phi(\mathbf{x}_t, s_t)) \geq 1$$

where $s_t = \text{argmax}_{i \in \{1 \ldots M\},\ i \neq y_t} \quad \mathbf{w}_t \Phi(\mathbf{x}_t, i)$

- The solution to this multiclass optimization problem is:

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

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

Other applications

- Multiclass and multilabel classification.
- Learning with structured output: graphs, trees, strings.
- Uniclass prediction.

Index

Online learning techniques

- Online learning for deep networks:
  - Hedge Backpropagation [Sahoo IJCAI 18]
  - Online BackPropagation.
  - Online learning with Transformer [Peris CSL 19]

- Online learning for statistical log-linear models:

$$p(y \mid \mathbf{x}) = \frac{\exp(\sum_m \lambda_m h_m(y, \mathbf{x}))}{\sum_{y'} \exp(\sum_m \lambda_m h_m(y', \mathbf{x}))}$$

  - Discriminative online adaptation algorithm based on ridge regression technique [Martinez PR 12] [Chinea PAA 19]
  - Generative vs Discriminative models [Wäschle MTS 12] [Ortiz CL 16].
  - Bayesian adaptation [Sanchis CSL 15]

Online BackPropagation (regresion with Multilayer Perceptron)

- Given a set of weights: $\mathbf{w} \equiv \{w_{i,j}^l\}$
- Given a training sample: $(\mathbf{x}, y)$.
- From layer $l = 0$ to $l = L$, for each unit $1 \leq i \leq M_l$ compute the total input z_i^l and $s_i^l = f(z_i^l)$
- From layer $l = L$ to $l = 1$ and for each unit $1 \leq i \leq M_l$
  - Compute the error δ_i^l:
    if $l == L$ then $\delta_i^L = f'(z_i^l)(y_i - s_i^L)$ else $\delta_i^l = f'(z_i^l) \sum_r \delta_r^{l+1} w_{ri}^{l+1}$
  - Compute $\Delta w_{ij}^l = \rho \; \delta_i^l \; s_i^{l-1}$
- For each l, i and j update $w_{ij}^l = w_{ij}^l + \Delta w_{ij}^l$
- Output $\mathbf{w}$

Transformer (Vaswani 2017)

- Goal: Given an input token sequence x_1^J and an output token sequence y_1^I, compute:

$$p(y_1^I \mid x_1^J) = \prod_{i=1}^{I} p(y_i \mid y_1^{i-1}, u(x_1^J))$$

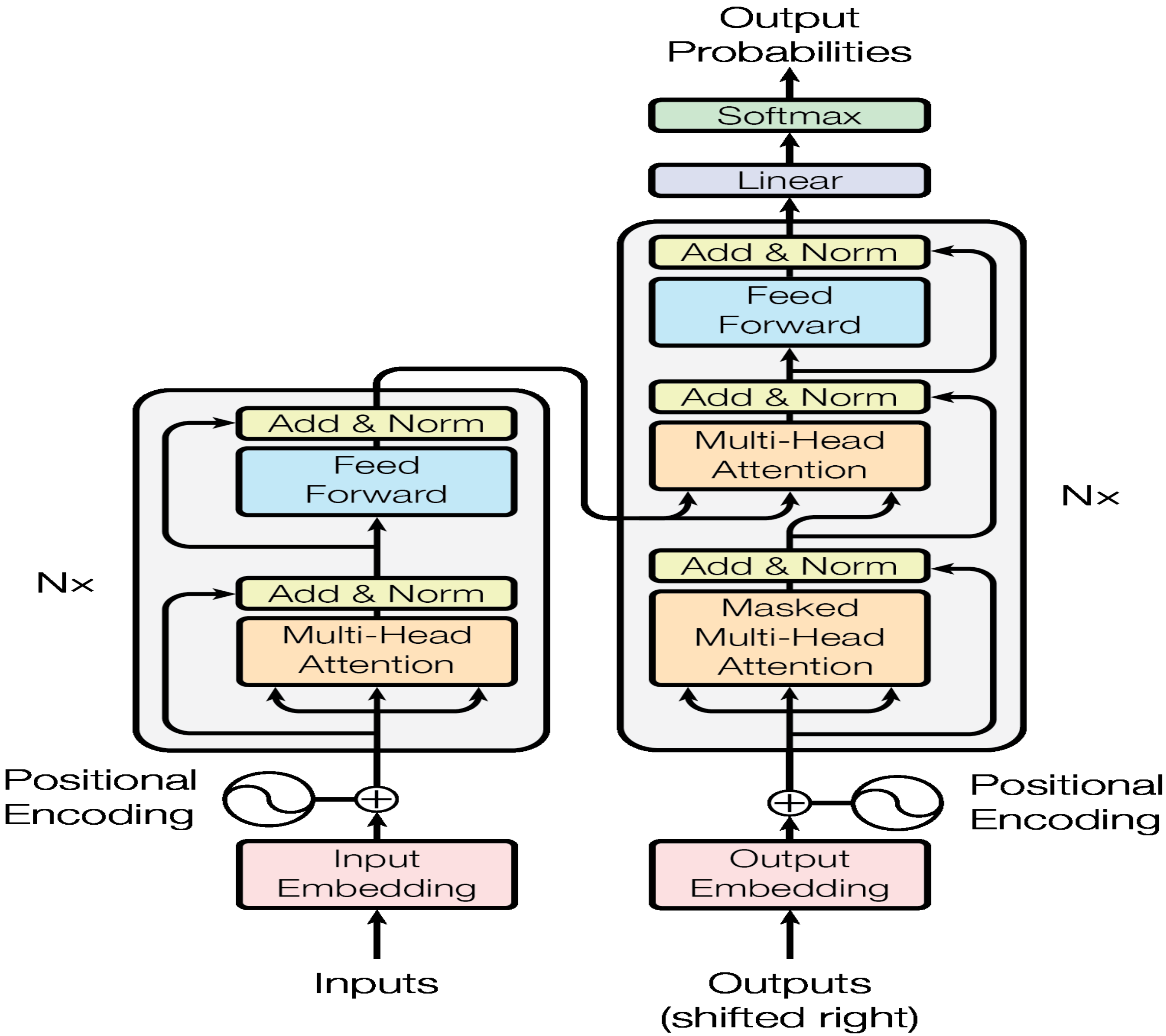
- Feed-forward networks.

- Self-attention or intra-sentence attention: (j, j') & (i, i') in addition to the cross-attention (i, j).

- Position encoding.

- Multi-head attention.

Transformer (Vaswani 2017)



On-line learning with in NMT [Peris et al. CSL 2019]

- Given a new sentence pair (x_1^J, y_1^I) validated by the user, the weights of the model are updated.
- To do this, one iteration of the ADAGRAD or ADADELTA is performed.
- An increase of 2-3 BLUE points w.r.t. not perform the update.
- The toolkit implements a fully NMT SMT system
  https://github.com/lvapeab/nmt-keras

Index

Bibliography (I)

- M Chinea-Rios, G Sanchis-Trilles, F Casacuberta. Discriminative ridge regression algorithm for adaptation in statistical machine translation. Pattern Analysis and Applications 22 (4):1293-1305. 2019.
- Koby Crammer, Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz. Online Passive-Aggressive Algorithms. Journal of Machine Learning Research. 7:551-585. 2006.
- Pascual Martínez-Gómez, Germán Sanchis-Trille, Francisco Casacuberta. Online adaptation strategies for statistical machine translation in post-editing scenarios. Pattern Recognition. 45 (9):3193-3203. 2012.
- Kevin P. Murphy. Machine Learning, A Probabilistic Perspective. The MIT Press. 2012.
- Daniel Ortiz-Martínez. Online learning for statistical machine translation. Computational Linguistics. 42 (1):121-161. 2016.

Bibliography (II)

- Álvaro Peris, Francisco Casacuberta. Online learning for effort reduction in interactive neural machine translation. Comput. Speech Lang. 58:98-126. 2019.
- Doyen Sahoo, Quang Pham, Jing Lu, Steven C. H. Hoi. Online Deep Learning: Learning Deep Neural Networks on the Fly. Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence. 2018.
- G Sanchis-Trilles, F Casacuberta. Improving translation quality stability using Bayesian predictive adaptation. Computer Speech & Language 34 (1): 1-17 2015.
- K. Waschle, P. Simianer, N. Bertoldi, S. Riezler. M. Federico. Generative and Discriminative Methods for Online Adaptation in SMT. Proceedings of the XIV Machine Translation Summit. 11-18. 2013.