Structured Prediction

Ningshan Zhang

Advanced Machine Learning, Spring 2016
Outline

- Ensemble Methods for Structured Prediction[1]
  - On-line learning
  - Boosting
Structured Prediction Problem

- Structured Output: $\mathcal{Y} = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_l$.
- Loss Function: $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$ decomposable.
- Training data: sample drawn i.i.d. from $\mathcal{X} \times \mathcal{Y}$ according to some distribution $\mathcal{D}$,

$$S = ((x_1, y_1), \ldots, (x_m, y_m)) \in \mathcal{X} \times \mathcal{Y}$$

- Problem: find hypothesis $h : \mathcal{X} \rightarrow \mathcal{Y}$ with small generalization error

$$\mathbb{E}_{(x, y) \sim \mathcal{D}} [L(h(x), y)]$$
Ensemble Methods: Further assumptions

- The number of substructures $l \geq 1$ is fixed.

- Loss function $L$ can be decomposed as a sum of loss functions $l_k : \mathcal{Y}_k \to \mathbb{R}^+$, i.e.
  
  for all $y = (y^1, ..., y^l)$ and $y' = (y'^1, ..., y'^l)$,

  $$L (y, y') = \sum_{k=1}^{l} l_k \left(y^k, y'^k\right)$$

- A set of structured prediction experts $h_1, ..., h_p$:

  $$h_j (x) = \left(h_j^1 (x), ..., h_j^l (x)\right), \forall j = 1, ..., p$$
Path Experts

- Intuition: one particular expert maybe better at predicting k-th substructure, but not elsewhere.

![Path Expert Diagram]

**Figure:** Path Expert

- Construct a larger set of experts, 'path experts':

$$
\mathcal{H} = \left\{ (h_{j_1}^1, ..., h_{j_l}^l) \mid j_k \in \{1, ..., p\}, k = 1, ..., l \right\}, |\mathcal{H}| = p^l
$$

- When selecting the best $h^*$ from $\mathcal{H}$, it may not coincide with any of original experts.
Review: Randomized Weighted Majority (WM)

- Task: find a distribution over experts.
- Initialize weights for all experts:
  \[ w_{1,j} = \frac{1}{p}, j = 1, \ldots, p \]
- After receiving sample \((x_t, y_t)\),
  - Update weight for expert \(j, j = 1, \ldots, p\):
    \[ w_{t+1,j} = \frac{w_{t,j} e^{-\eta L(\hat{y}_{t,j}, y_t)}}{\sum_{j=1}^{p} w_{t,j} e^{-\eta L(\hat{y}_{t,j}, y_t)}}, \eta > 0 \text{ constant} \]
- Return \(\mathbf{w}_{T+1} = (w_{T+1,1}, \ldots, w_{T+1,p})\)
Randomized Weighted Majority (WM)

- Apply the Weighted Majority algorithm to path experts set $\mathcal{H}$:

$$\forall h \in \mathcal{H}, \ w_{t+1}(h) = \frac{w_t(h) e^{-\eta L(h(x_t), y_t)}}{\sum_{h \in \mathcal{H}} w_t(h) e^{-\eta L(h(x_t), y_t)}}$$

- $p'$ updates per round!

- Using the structure of $\mathcal{Y}$ and additive loss assumption, we can reduce to $pl$ updates per round.
Weighted Majority Weight Pushing (WMWP)

- For every $h = (h_{j_1}^1, ..., h_{j_l}^l)$, $w(h) = \prod_{k=1}^{l} w_{j_k}^k$
- Enforce the weights at each substructure sum to one:

$$\sum_{j=1}^{p} w_{j}^k = 1, \forall k \in \{1, ..., l\}$$

- The overall weights $\sum_{h \in \mathcal{H}} w(h)$ will automatically sum to one!

Example: $p = 2, l = 2$,

$$w_1^1 w_1^2 + w_1^1 w_2^2 + w_2^1 w_1^2 + w_2^1 w_2^2 = (w_1^1 + w_2^1) (w_1^2 + w_2^2) = 1$$
Weighted Majority Weight Pushing (WMWP)

- Initialize weights: \( w_{1,j}^k = 1/p, \forall (k,j) \in \{1, ..., l\} \times \{1, ..., p\} \),
- At round \( t \), \( \forall (k,j) \in \{1, ..., l\} \times \{1, ..., p\} \), update weight

\[
w_{t+1,j}^k = \frac{w_{t,j}^k \cdot e^{-\eta l_k(h_j^k(x_t), y_t^k)}}{\sum_{j=1}^{p} w_{t,j}^k \cdot e^{-\eta l_k(h_j^k(x_t), y_t^k)}},
\]

- Return \( W_1, ..., W_{T+1} \), where \( W_t = (w_{t,j}^k) \).
On-line to batch conversion

How to define a hypothesis based on \( \{ W_1, ..., W_{T+1} \} \)?

Two steps:

1. Choose a good collection of distributions. \( P = \{ W_1, ..., W_{T+1} \} \) is one choice, but not necessarily the best.
2. Use \( P \) to define hypotheses for prediction.
On-line to batch conversion

- Step 1: Choose a good collection of distributions.
- For any collection $P$, define score

$$
\Gamma(P) = \frac{1}{|P|} \sum_{W_t \in P} \sum_{h \in \mathcal{H}} W_t(h) L(h(x_t), y_t) + M \sqrt{\frac{\log \frac{1}{\delta}}{|P|}}
$$

- Chose $P_\delta = \arg \min_{P \in \mathcal{P}} \Gamma(P)$
On-line to batch conversion

- Step 2: define hypotheses.
- Randomized: randomly select a path expert $h \in \mathcal{H}$ according to
  
  $$p(h) = \frac{1}{|P_\delta|} \sum_{W_t \in P_\delta} W_t(h),$$

  denote the set of generated hypotheses as $\mathcal{H}_{Rand}$.
- Deterministic: defined a scoring function

  $$\tilde{h}_{MVote}(x, y) = \prod_{k=1}^{l} \left( \frac{1}{|P_\delta|} \sum_{W_t \in P_\delta} \sum_{j=1}^{p} w_{t,j}^k 1_{h_j^k(x) = y^k} \right)$$

  $$\mathcal{H}_{MVote}(x) = \arg \max_{y \in Y} \tilde{h}_{MVote}(x, y)$$
Learning Guarantees

- Regret $R_T$

$$R_T = \sum_{t=1}^{T} \mathbb{E}_{h \sim W_t} [L (h (x_t), y_t)] - \inf_{h \in H} \sum_{t=1}^{T} L (h (x_t), y_t)$$

- For any $\delta > 0$, with probability at least $1 - \delta$ over sample $(x_i, y_i)_{i=1}^{T}$ drawn i.i.d from $D$, the following hold:

$$\mathbb{E} [L (\mathcal{H}_{Rand} (x), y)] \leq \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{h \sim W_t} [L (h (x_t), y_t)] + M \sqrt{\frac{\log \frac{T}{\delta}}{T}},$$

$$\mathbb{E} [L (\mathcal{H}_{Rand} (x), y)] \leq \inf_{h \in H} \mathbb{E} [L (h (x), y)] + \frac{R_T}{T} + 2M \sqrt{\frac{\log \frac{2T}{\delta}}{T}}.$$
Learning Guarantees

- The following inequality always hold: with expectations taken over \((x, y) \sim D\) and \(h \sim p\) for \(\mathcal{H}_{\text{Rand}}\),

\[
\mathbb{E} [L_{\text{Ham}}(\mathcal{H}_{\text{MVote}}(x), y)] \leq 2 \mathbb{E} [L_{\text{Ham}}(\mathcal{H}_{\text{Rand}}(x), y)]
\]

- Proof: by definition of \(\mathcal{H}_{\text{MVote}}\), the following always hold:

\[
\frac{1}{2} 1_{\mathcal{H}_{\text{MVote}}^k(x) \neq y^k} \leq \frac{1}{|P_\delta|} \sum_{W_t \in P_\delta} \sum_{j=1}^p w_{t,j}^k 1_{h_j^k(x) \neq y^k}
\]

summing over \(k\) and take expectations over \(D\) yields the desired results. □
AdaBoost: Review

- Hypothesis space: $\mathcal{H} = \left\{ \sum_{j=1}^{N} \alpha_j h_j, \alpha_j \geq 0 \right\}$, where $h_j \in \mathcal{H}_0$ are base classifiers, $h_j : \mathcal{X} \to \mathbb{R}$.

- Objective Function: $F(\alpha) = \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{N} \alpha_j h_j(x_i)}$

- Apply Coordinate Descent to $F(\alpha)$

- Return hypothesis: $h(x) = \text{sgn} \left( \sum_{j=1}^{N} \alpha_j h_j(x) \right)$
ESPBoost: hypothesis space

- How to make a convex combination of path experts? Prediction → Score → Convex Combination of Score → ’Combined’ Prediction

- For each path experts $h_t \in \mathcal{H}$, define the score $\tilde{h}_t : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$:

$$\forall (x, y) \in \mathcal{X} \times \mathcal{Y}, \tilde{h}_t (x, y) = \sum_{k=1}^{l} 1_{h_t^k(x) = y^k}$$

- Convex combination of score:

$$\left\{ \tilde{h} = \sum_{t=1}^{T} \alpha_t \tilde{h}_t : \tilde{h}_t \text{ derived from path experts, } \alpha_t \geq 0 \right\}$$

- ’Combined’ Prediction:

$$h(x) = \arg \max_{y \in \mathcal{Y}} \tilde{h}(x, y) = \arg \max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \sum_{k=1}^{l} \alpha_t 1_{h_t^k(x) = y^k}$$
Loss Function and Upper Bound

- Normalized Hamming Loss:

\[
\frac{1}{m} \sum_{i=1}^{m} \left[ L_{Ham} (\mathcal{H}_{ESPBoost} (x_i), y_i) \right]
\]

\[
= \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} 1_{\tilde{h}^k (x_i, y_i) - \max_{y \neq y_i} \tilde{h}^k (x_i, y) < 0}
\]

\[
\leq \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp \left\{ - \sum_{t=1}^{T} \alpha_t \rho \left( \tilde{h}^k_t, x_i, y_i \right) \right\} \quad := F (\alpha)
\]

- Margin: \( \rho \left( \tilde{h}^k, x_i, y_i \right) = \tilde{h}^k (x_i, y_i^k) - \arg \max_{y^k \neq y_i^k} \tilde{h}^k (x_i, y^k) \)
**ESPBoost**

- **Hypothesis space**: the set of combined predictions

\[ \mathcal{H}_{ESPBoost} (x) = \left\{ h(x) : \arg \max_{y \in \mathcal{Y}} \tilde{h}(x, y) \right\} \]

- **Objective Function** \( F : \mathbb{R}^T_+ \rightarrow \mathbb{R} \)

\[
F(\alpha) = \frac{1}{ml} \sum_{i=1}^{m} \sum_{k=1}^{l} \exp \left\{ - \sum_{t=1}^{T} \alpha_t \rho \left( \tilde{h}^k_t, x_i, y_i \right) \right\}
\]

- **Apply coordinate descent.**
Algorithm 2 ESPBoost Algorithm.

Inputs: $S = ((x_1, y_1), \ldots, (x_m, y_m))$; set of experts $\{h_1, \ldots, h_p\}$.

for $i = 1$ to $m$ and $k = 1$ to $l$ do
    $D_1(i, k) \leftarrow \frac{1}{ml}$
end for

for $t = 1$ to $T$ do
    $h_t \leftarrow \arg\min_{h \in H} E(i, k) \sim D_t \left[1_{h^k(x_i) \neq y_i^k}\right]$
    $\epsilon_t \leftarrow E(i, k) \sim D_t \left[1_{h_t^k(x_i) \neq y_i^k}\right]$
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$
    $Z_t \leftarrow 2 \sqrt{\epsilon_t(1 - \epsilon_t)}$
    for $i = 1$ to $m$ and $k = 1$ to $l$ do
        $D_{t+1}(i, k) \leftarrow \frac{\exp(-\alpha_t \rho(h_t^k, x_i, y_i)) D_t(i, k)}{Z_t}$
    end for
end for

Return $\tilde{h} = \sum_{t=1}^{T} \alpha_t \tilde{h}_t$
Generalized Kernel Approach

- Problem Setting: Given \((x_i, y_i)_{i=1}^n \in \mathcal{X} \times \mathcal{Y}\), we want to learn a mapping \(f\) from \(\mathcal{X}\) to \(\mathcal{Y}\).

- \(l : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}\) a kernel function
- \(\mathcal{F}_Y\): RKHS associated with \(l\)
- \(\Phi_l\): the mapping from \(\mathcal{Y}\) to \(\mathcal{F}_Y\)

- Step 1: Learn the mapping \(g\) from \(\mathcal{X}\) to \(\mathcal{F}_Y\)
- Step 2: Find the pre-image of \(g(x)\)
Generalized Kernel Approach

- Step 1: learn the mapping \( g : \mathcal{X} \rightarrow \mathcal{F}_Y \)
- Preliminaries
  - Operator-valued kernels
  - Function-valued RKHS
Operator-valued Kernels

\[ \mathcal{L}(\mathcal{F}_Y) \text{ be the set the bounded operators } \mathcal{T} : \mathcal{F}_Y \to \mathcal{F}_Y \]

Non-negative \( \mathcal{L}(\mathcal{F}_Y) - \text{valued kernel} \) \( K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{F}_Y) \)

such that:

- \( \forall x_i, x_j \in \mathcal{X}, K(x_i, x_j) = K(x_j, x_i)^* \).
- For any \( m > 0, \left\{ (x_i, \varphi_i)_{i=1,...,m} \right\} \subseteq \mathcal{X} \times \mathcal{F}_Y, \)
  \[ \sum_{i,j=1}^{m} \langle K(x_i, x_j) \varphi_j, \varphi_i \rangle_{\mathcal{F}_Y} \geq 0. \]

Note: * denotes adjoint operator, i.e. \( \forall \varphi_1, \varphi_2 \in \mathcal{F}_Y, \)

\[ \langle K(\varphi_1), \varphi_2 \rangle_{\mathcal{F}_Y} = \langle \varphi_1, K^*(\varphi_2) \rangle_{\mathcal{F}_Y} \]
A Hilbert space $\mathcal{F}_{XY}$ of functions $g : \mathcal{X} \rightarrow \mathcal{F}_Y$ is a $\mathcal{F}_Y$–valued RKHS if there is a non-negative $\mathcal{L}(\mathcal{F}_Y)$–valued kernel $K$ with the following properties:

- $\forall x \in \mathcal{X}, \forall \varphi \in \mathcal{F}_Y$, the function $K(x, \cdot)\varphi \in \mathcal{F}_{XY}$
- $\forall g \in \mathcal{F}_{XY}, \forall x \in \mathcal{X}, \forall \varphi \in \mathcal{F}_Y$, $\langle g, K(x, \cdot)\varphi \rangle_{\mathcal{F}_{XY}} = \langle g(x), \varphi \rangle_{\mathcal{F}_Y}$

Theorem: Bijection between $\mathcal{F}_{XY}$ and $K$, as long as $K$ is non-negative.
Kernel Ridge Regression

- Step 1: Learning $g : \mathcal{X} \rightarrow \mathcal{F}_Y$
- Kernel Ridge Regression, with closed form solution:

$$\arg \min_{g \in \mathcal{F}_{xy}} \sum_{i=1}^{n} \left\| g \left( x_i \right) - \Phi_l \left( y_i \right) \right\|_{\mathcal{F}_Y}^2 + \lambda \left\| g \right\|_{\mathcal{F}_{xy}}^2$$

$$g \left( x \right) = K_x \left( K + \lambda I \right)^{-1} \Phi_l$$

- $K_x$: a row vector of operators, $\left[ K \left( \cdot, x_i \right) \in \mathcal{L} \left( \mathcal{F}_Y \right) \right]_{i=1}^{n}$
- $K$: a matrix of operators, $\left[ K \left( x_i, x_j \right) \in \mathcal{L} \left( \mathcal{F}_Y \right) \right]_{i,j=1}^{n}$
- $\Phi_l$: a column vector of functions $\left[ \Phi_l \left( y_i \right) \in \mathcal{F}_Y \right]_{i=1}^{n}$
Find Pre-image

- Step 2: Find pre-image of $g(x)$

$$ f(x) = \arg \min_{y \in \mathcal{Y}} \| g(x) - \Phi_l(y) \|^2_{\mathcal{F}_Y} $$

$$ = \arg \min_{y \in \mathcal{Y}} \left\| K_x (K + \lambda I)^{-1} \Phi_l - \Phi_l(y) \right\|^2_{\mathcal{F}_Y} $$

$$ = \arg \min_{y \in \mathcal{Y}} l(y, y) - 2 \left\langle K_x (K + \lambda I)^{-1} \Phi_l, \Phi_l(y) \right\rangle_{\mathcal{F}_Y} $$

- $\Phi_l(y)$ unknown, use a generalized kernel trick:

$$ \left\langle T \Phi_l(y_i), \Phi_l(y) \right\rangle = [T l(y_i, \cdot)](y) $$

- Express $f(x)$ using only kernel functions:

$$ f(x) = \arg \min_{y \in \mathcal{Y}} l(y, y) - 2 \left[ K_x (K + \lambda I)^{-1} L \right](y) $$

where $L$ is a column vector of $[l(y_i, \cdot)]_{i=1}^n$.
Covariance-based Operator-valued Kernels

- Covariance-based operator-valued kernels:

\[ K(x_i, x_j) = k(x_i, x_j) \mathcal{C}_{YY} \]

- \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) a scalar-valued kernel;

- \( \mathcal{C}_{YY} : \mathcal{F}_Y \to \mathcal{F}_Y \) a covariance operator defined by a random variable \( Y \in \mathcal{Y} \):

\[ \langle \varphi_1, \mathcal{C}_{YY} \varphi_2 \rangle_{\mathcal{F}_Y} = \mathbb{E} [\varphi_1(Y) \varphi_2(Y)] \]

- Empirical covariance operator

\[ \hat{\mathcal{C}}_{YY}(\varphi) = \frac{1}{n} \sum_{i=1}^{n} \varphi(y_i) I(\cdot, y_i) \]

- To account for the effects of input, we could also use conditional covariance operator

\[ \mathcal{C}_{YY|X} = \mathcal{C}_{YY} - \mathcal{C}_{YX} \mathcal{C}_{XX}^{-1} \mathcal{C}_{XY} \]
Conclusion

- Ensemble methods: ensemble learning with expended 'path experts'.
  - On-line algorithm (WMWP): efficient for learning and inference by exploiting the output structure.
  - On-line-to-batch-conversion: randomized and deterministic algorithms with learning guarantees.
  - Boosting: efficient for output structure.

- Kernel method:
  - Use a joint feature space.
  - Covariance-based operator-valued kernel to encode interactions between outputs.
  - Conditional Covariance-based operator to correlate input with 'interaction between outputs'.
  - Express the final hypothesis with only kernel functions.