Machine Learning 2019: Week ⟨*insert large number*⟩ tutorial Structural SVMs and Ensemble Methods

Clara Meister

November 28, 2019

Institute for Machine Learning

Structural SVMs

(A brief review of) Structural SVMs

Motivation

- Structured learning is the subfield of machine learning concerned with computer programs that learn to map inputs to arbitrarily complex outputs.
- This stands in contrast to the simpler approaches of classification, where input data (instances) are mapped to "atomic" labels, i.e. symbols without any internal structure, and regression, where inputs are mapped to scalar numbers or vectors

(A brief review of) Structural SVMs

Support vector machine (soft margin)

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} \xi_i$$
$$s.t. \quad z_i \mathbf{w}^\top \mathbf{y}_i \geq 1 - \xi_i \quad \forall i$$

Where

- $z_i \in \mathbb{K}$ is the label. In this case, $\mathbb{K} = \{1, -1\}$
- $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^{n} \subset \mathbb{R}^k$ are training data derived from raw input data $\mathbf{x}_i$
- $\{\xi_i\}_{i=1}^{n}$ are slacks

(A brief review of) Structural SVMs

Support vector machine (soft margin)

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$
$$s.t. \quad z_i\mathbf{w}^\top\mathbf{y}_i \geq 1 - \xi_i \quad \forall i$$

- The margin (width 1) can be seen as a constant loss function
- $z_i\mathbf{y}_i$ is a *feature map*

(A brief review of) Structural SVMs

Structural/structured SVMs

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^\top\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \mathbf{w}^\top(\Psi(z_i,\mathbf{y}_i) - \Psi(z,\mathbf{y}_i)) \geq \Delta(z_i,z) - \xi_i \quad \forall i, \forall z \neq z_i$$

- Replace the margin with a general loss function $\Delta : \mathbb{K} \times \mathbb{K} \to \mathbb{R}$
- $\mathcal{Y}$ and $\mathbb{K}$ are generic
- $\Psi : \mathbb{K} \times \mathcal{Y} \to \mathbb{R}^k$ is a general feature map

(A brief review of) Structural SVMs

Need to define/understand the following for each problem:

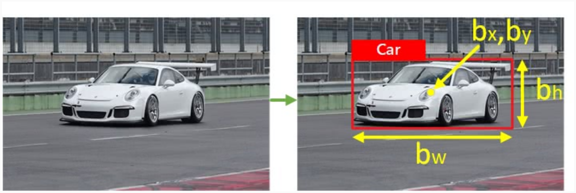
- Input and output spaces $\mathbb{K}$ and $\mathcal{Y}$
- Feature function $\Psi(z, \mathbf{y}) \rightarrow \mathbb{R}^k$
- Loss function $\Delta(z_i, z) \rightarrow \mathbb{R}$

(A brief review of) Structural SVMs

Examples?

(A brief review of) Structural SVMs

Object Localization in Images



[1]

[1] https://www.dlology.com/

(A brief review of) Structural SVMs

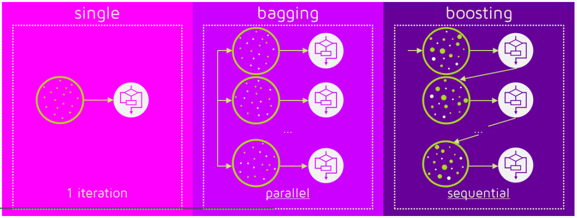
Object Localization in Images

- input space $\mathcal{Y}$ = images = $[0, 255]^{3 \cdot M \cdot N}$
- output space $\mathbb{R}^4$ = bounding box coordinates (x, y, width, height)
- feature function Ψ = e.g. "bag-of-words histogram of region z in image y"
- loss function $\Delta(z_i, z)$ = e.g. measure of overlap

Ensemble methods

Ensemble methods

1. Bagging
   - **B**ootstrap **agg**regation
   - Each classifier trained on a **different bootstrap sample** of the training data
   - e.g. random forests: bagging decision trees
2. Boosting
   - Every classifier trained on same data, but with different weights
   - Each classifier weighted during aggregation
   - e.g. AdaBoost



[2]

[2] https://quantdare.com

Ensemble methods

Motivation

The idea behind: One thousand mediocre opinions, averaged wisely, are better than a really good one!

- Computationally less expensive (faster)
- Single estimators are usually very simple ones (easy to implement)
- Often outperforms very complicated models!
- Low variance *and* (potentially) low bias ...

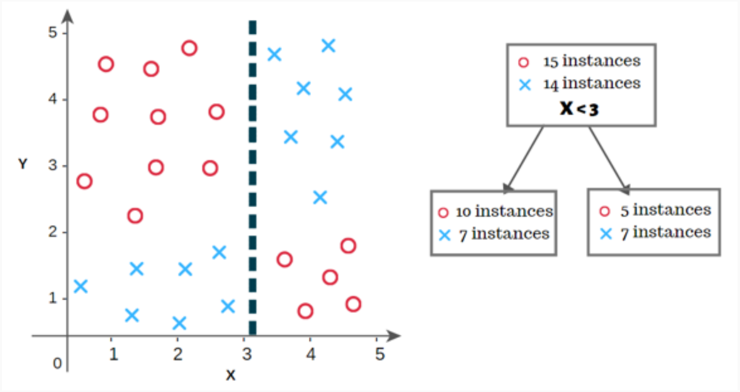
Decision trees

Recap: how they work...

- Recursive partitioning of training data, trained greedily starting at root
- At each step, pick a random subset of features and define a decision boundary/split
- Terminate when the data assigned to a node are all in the same class (other termination criteria also possible)
- If height set to 1 and only one feature, then *decision stump*
- Several training algorithms: one of the most common are `CART` trees (classification and regression trees)

Decision trees

Recap: how they work...

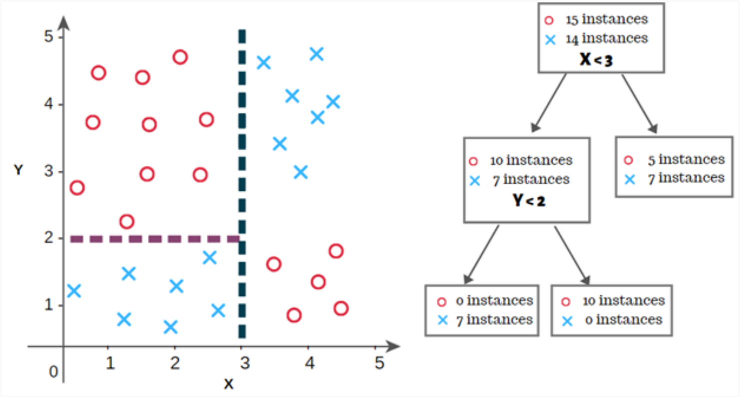


[3]

[3] https://towardsdatascience.com/decision-tree-overview-with-no-maths-66b256281e2b

Decision trees

Recap: how they work...

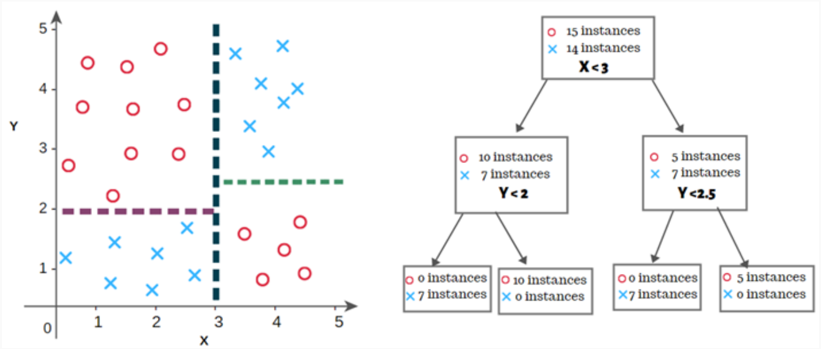


[4]

[4] https://towardsdatascience.com/decision-tree-overview-with-no-maths-66b256281e2b

Decision trees

Recap: how they work...



[5]

[5] https://towardsdatascience.com/decision-tree-overview-with-no-maths-66b256281e2b

Decision trees

Mathematical Definition

- Training data: $(X, \mathbf{y}) \in \mathbb{R}^{n \times d} \times \mathbb{R}^n$
- Let $T = (V, E)$ be a decision tree, $\mathcal{I} = \{1, \dots, |V|\}$
- Let $C : \mathcal{I} \to \mathcal{I} \times \mathcal{I}$ map nodes to their children
- For each node $v_i \in V$, let $(X_i, \mathbf{y}_i)$ denote the submatrix/sublabel tuple (subset of the training data) assigned to that node, where $X_i \in \mathbb{R}^{n_i \times d}$, $\mathbf{y}_i \in \mathbb{R}^{n_i}$
    - For the root v_0, $(X_0, \mathbf{y}_0) := (X, \mathbf{y})$

Decision trees

Mathematical Definition: decision boundary computation

Suppose we are at node v_i with decision boundary scoring function Δ

1. Terminate if $\mathbf{y}_i \in \{0\}^{n_i} \cup \{1\}^{n_i}$ (all labels are the same)
2. Pick a random subset of features $\{f_{i1}, \ldots, f_{ik}\} \subseteq \{1, \ldots, d\}$
3. Define the set of valid decision boundaries $\mathcal{B}_i \subset \{0,1\}^{n_i}$
    - For each feature f_{ij}
        3.1 define a boundary/cutoff $c_{ij} \in \{1, \ldots, n_i\}$
        3.2 induce a binary vector $\mathbf{b}_{ij} \in \{0,1\}^{n_i}$, where $(\mathbf{b}_{ij})_\ell = 1 \Leftrightarrow (X_i)_{\ell, f_{ij}} \geq c_{ij}$
4. Find the optimal decision boundary

$$\mathbf{b}_i^* = \arg\max_{\mathbf{b}_{ij} \in \mathcal{B}_i} \Delta(\mathbf{y}_i, \mathbf{b}_{ij})$$

5. Let $X_{C(i)_1} := X_i[\mathbf{b}_i^*]$ and $X_{C(i)_2} := X_i[\neg \mathbf{b}_i^*]$ (partition the rows)

Exact algorithm for computing the c_{ij} and $\mathbf{b}_i^*$ varies

Decision trees

Mathematical Definition: decision boundary scoring

- Usually, an *impurity* measure I is computed on the parent node v_i and its children
- Given $\mathbf{b}_i$, let $s_i = \sum_{l=1}^{n_i} (\mathbf{b}_i)_l$
- Compute a weighted difference in impurity to score a choice of decision boundary

$$\Delta(\mathbf{y}_i, \mathbf{b}_i) := I(\mathbf{y}_i) - \left(\frac{s_i}{n_i} I(\mathbf{y}_i[\mathbf{b}_i]) + \frac{n_i - s_i}{n_i} I(\mathbf{y}_i[\neg \mathbf{b}_i]) \right)$$

Decision trees

Mathematical Definition: impurity measures

- Let $\mathcal{Y}_i := \{(\mathbf{y}_i)_j : j \in \{1, \ldots, n_i\}\}$ (the set of all class labels in $\mathbf{y}_i$)
- Let $\mathbf{p}_i$ be a probability distribution on $\mathcal{Y}_i$, i.e.

$$(\mathbf{p}_i)_j = \frac{|\{\ell : (\mathbf{y}_i)_\ell = y\}|}{n_i} \quad \forall y \in \mathcal{Y}_i$$

- Impurity measures
  1. Gini impurity

$$I_G(\mathbf{y}_i) = \sum_{j=1}^{|\mathcal{Y}_i|} (\mathbf{p}_i)_j (1 - (\mathbf{p}_i)_j)$$

  2. Shannon entropy

$$I_H(\mathbf{y}_i) = H(\mathbf{p}_i) = -\sum_{j=1}^{|\mathcal{Y}_i|} (\mathbf{p}_i)_j \log_2 (\mathbf{p}_i)_j$$

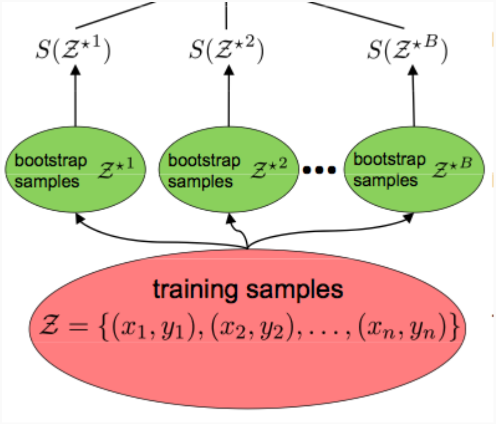
- The weighted difference in Shannon entropy is the *information gain*

Decision trees

- To classify a new data point, use the optimal decision boundaries trained to traverse the tree until a leaf is reached. The label of that leaf is then the prediction
- The selection of a random subset of features at each step acts as a form of feature selection
    - Given a feature $f \in \{1, \dots, d\}$, let $\mathcal{J}(f) = \{i : f = f_{ij} \quad \forall j\}$ (the set of all nodes where f was used to define an optimal decision boundary)
    - Let $FI(f) = \frac{1}{|\mathcal{J}(f)|} \sum_{i \in \mathcal{J}(f)} \Delta(\mathbf{y}_i, \mathbf{b}_i^*)$
    - Normalize to sum to 1: $\tilde{FI} = \frac{FI(f)}{\sum_{j=1}^{d} FI(j)}$
- If train until leaves are pure, then low bias, but high variance (overfitting)
    - applies to both classification and feature importance calculation

Random forests

- A collection of decision trees trained on bootstrap samples
- Reduce variance (ensemble), slightly increase bias (since full data set not used and only subset of variables used for splitting)



$S(\mathcal{Z}^{\star 1})$
$S(\mathcal{Z}^{\star 2})$
$S(\mathcal{Z}^{\star B})$
bootstrap samples $\mathcal{Z}^{\star 1}$
bootstrap samples $\mathcal{Z}^{\star 2}$
bootstrap samples $\mathcal{Z}^{\star B}$
training samples
$\mathcal{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

Random forests: Out-of-bag error

- Built-in form of validation error
- Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be a random forest of size m trained on bootstrap samples $X^{(1)}, \ldots, X^{(m)}$
- For each row X^i in the training set X, define the assignment function

$$\mathcal{A}(i) = \{m' : \exists j, (X^{(m')})^j = X^i\}$$

- The out-of-bag (OOB) error of X^i is then the average prediction error ($1 -$ accuracy) on

$$\{T_{m'} : m' \notin \mathcal{A}(i)\}$$

- The OOB error of the trained forest is the average out-of-bag error of the X^is

Random forests: imbalanced training data

- If the *imbalance ratio* of the classes is high (e.g. there are 1000$\times$ as many in class 1 as in class 0), then an optimal classifier will just predict that everything is in class 1
- Two methods proposed for dealing with imbalanced training data
    1. Class weights
        - Give greater weight to minority class samples
    2. balanced bootstrapping (balanced random forests)
        2.1 Take bootstrap sample of minority class
        2.2 Sample of same size from majority class

Chen, C., Liaw, A., & Breiman, L. (2004). Using random forest to learn imbalanced data. University of California, Berkeley, 110.

Random forests

Feature Importance

- Can get better estimate of feature importance by averaging feature importance vectors from each decision tree
- More robust than feature importance calculations from each trees
- Allows random forests to be used as a preprocessing step for cheap feature selection
    - Note that if features are highly correlated, then in expectation, their feature importances will be equal
    - In practice, you may want to pick only one of them

Random forests

- Pros
    - Fast/easy to train, used as baselines for more complex classifiers
    - No need for data splitting for cross-validation (use OOB)
    - Useful for highly-imbalanced data sets
- Cons
    - Less interpretable than decision trees
    - Work best on low-dimensional data
    - Bad at learning more complex relationships between (i.e. convolutions of) features

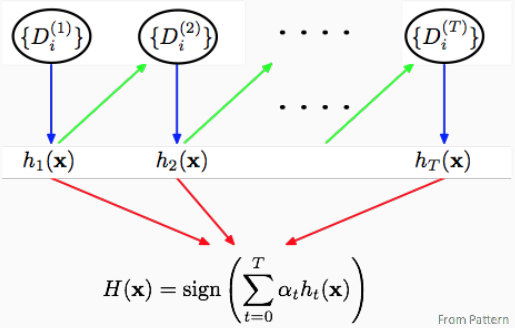
Boosting

In words

- *Sequentially* training weak classifiers
- If data point x is misclassified, increase its weight. This means later classifiers focus on points that are difficult for the former ones

Boosting

AdaBoost



From Pattern Recognition and Machine Learning. C. Bishop

- Mathematically, the training distribution is shifted to emphasize the 'hard' cases. This is achieved by changing the weight (prob distr) of the samples