Introduction to Machine Learning (67577)
Lecture 9

Shai Shalev-Shwartz

School of CS and Engineering,
The Hebrew University of Jerusalem

Multiclass, Ranking, and Complex Prediction Problems

Outline

Multiclass Categorization

- So far, we mainly dealt with binary problems $h : \mathcal{X} \to \{\pm 1\}$ or regression problems, $h : \mathcal{X} \to \mathbb{R}$

Multiclass Categorization

- So far, we mainly dealt with binary problems $h : \mathcal{X} \to \{\pm 1\}$ or regression problems, $h : \mathcal{X} \to \mathbb{R}$
- Multiclass problems: $h : \mathcal{X} \to \{1, \ldots, k\}$

Multiclass Categorization

- So far, we mainly dealt with binary problems $h : \mathcal{X} \to \{\pm 1\}$ or regression problems, $h : \mathcal{X} \to \mathbb{R}$
- Multiclass problems: $h : \mathcal{X} \to \{1, \ldots, k\}$
- E.g.: $x \in \mathcal{X}$ is an image and $\{1, \ldots, k\}$ represents k possible objects

Multiclass Categorization

- So far, we mainly dealt with binary problems $h : \mathcal{X} \to \{\pm 1\}$ or regression problems, $h : \mathcal{X} \to \mathbb{R}$
- Multiclass problems: $h : \mathcal{X} \to \{1, \ldots, k\}$
- E.g.: $x \in \mathcal{X}$ is an image and $\{1, \ldots, k\}$ represents k possible objects
- We'll later consider problems in which k is extremely large, E.g., in translation, $x \in \mathcal{X}$ is a sentence in Hebrew and $\{1, \ldots, k\}$ is all possible sentences in English

One-vs-All reduction

- Suppose we have a learner A for binary classification and we need to solve a multiclass problem

One-vs-All reduction

- Suppose we have a learner A for binary classification and we need to solve a multiclass problem
- Let $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$

One-vs-All reduction

- Suppose we have a learner A for binary classification and we need to solve a multiclass problem
- Let $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
- One-vs-All: train k binary problems, where problem i discriminates between class i and the rest of the classes. That is:

One-vs-All reduction

- Suppose we have a learner A for binary classification and we need to solve a multiclass problem
- Let $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
- One-vs-All: train k binary problems, where problem i discriminates between class i and the rest of the classes. That is:
  - For every i, feed the binary training set $S_i = (\mathbf{x}_1, (-1)^{\mathbb{1}_{[y_1 \neq i]}}), \ldots, (\mathbf{x}_m, (-1)^{\mathbb{1}_{[y_m \neq i]}})$ into A to get h_i

One-vs-All reduction

- Suppose we have a learner A for binary classification and we need to solve a multiclass problem
- Let $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
- One-vs-All: train k binary problems, where problem i discriminates between class i and the rest of the classes. That is:
  - For every i, feed the binary training set $S_i = (\mathbf{x}_1, (-1)^{\mathbb{1}_{[y_1 \neq i]}}), \ldots, (\mathbf{x}_m, (-1)^{\mathbb{1}_{[y_m \neq i]}})$ into A to get h_i
  - Output the multiclass classifier

$$h(\mathbf{x}) = \operatorname*{argmax}_{i \in [k]} h_i(\mathbf{x})$$

One-vs-All reduction

- Suppose we have a learner A for binary classification and we need to solve a multiclass problem
- Let $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
- One-vs-All: train k binary problems, where problem i discriminates between class i and the rest of the classes. That is:
  - For every i, feed the binary training set $S_i = (\mathbf{x}_1, (-1)^{\mathbb{1}_{[y_1 \neq i]}}), \ldots, (\mathbf{x}_m, (-1)^{\mathbb{1}_{[y_m \neq i]}})$ into A to get h_i
  - Output the multiclass classifier

$$h(\mathbf{x}) = \operatorname*{argmax}_{i \in [k]} h_i(\mathbf{x})$$

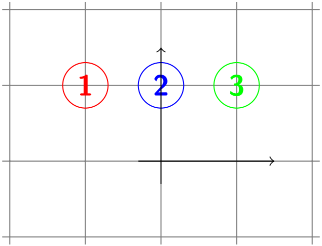
- In case of ties (more than one h_i predicts 1), and if h_i outputs a confidence as well (e.g. SVM), we can use the confidence of h_i to break ties

All-Pairs reduction

- All pairs of classes are compared to each other.
- For every $1 \le i < j \le k$, construct a binary sample, $S_{i,j}$, containing examples from class i against examples from class j.
- Call the binary learner to get $h_{i,j}$
- Output the multiclass classifier by predicting the class which had the highest number of "wins"

Sub-optimality of reductions

- One-vs-All over halfspace binary classifier will fail on the sample below, although it is separable by the resulting hypothesis class



Linear Multiclass Predictors

- Recall, binary linear predictor is

$$h(x) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle$$

Linear Multiclass Predictors

- Recall, binary linear predictor is

$$h(x) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle$$

- Can rewrite as

$$h(\mathbf{x}) \;=\; \operatorname*{argmax}_{y \in \{\pm 1\}} \langle \mathbf{w}, y\mathbf{x} \rangle$$

Linear Multiclass Predictors

- Recall, binary linear predictor is

$$h(x) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle$$

- Can rewrite as

$$h(\mathbf{x}) \;=\; \underset{y \in \{\pm 1\}}{\mathrm{argmax}}\, \langle \mathbf{w}, y\mathbf{x} \rangle$$

- Natural generalization:

$$h(\mathbf{x}) \;=\; \underset{y \in \mathcal{Y}}{\mathrm{argmax}}\, \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle,$$

where $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is a class-sensitive feature mapping.

Linear Multiclass Predictors

- Recall, binary linear predictor is

$$h(x) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle$$

- Can rewrite as

$$h(\mathbf{x}) \;=\; \operatorname*{argmax}_{y \in \{\pm 1\}} \langle \mathbf{w}, y\mathbf{x} \rangle$$

- Natural generalization:

$$h(\mathbf{x}) \;=\; \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle,$$

  where $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is a class-sensitive feature mapping.

- Intuitively, we can think of the elements of $\Psi(\mathbf{x}, y)$ as score functions that assess how good the label y fits the instance $\mathbf{x}$.

Linear Multiclass Predictors

- Recall, binary linear predictor is

$$h(x) = \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle$$

- Can rewrite as

$$h(\mathbf{x}) \;=\; \operatorname*{argmax}_{y \in \{\pm 1\}} \langle \mathbf{w}, y\mathbf{x} \rangle$$

- Natural generalization:

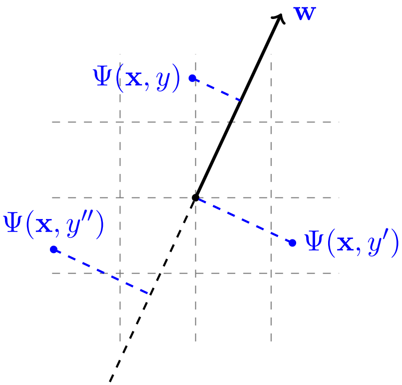
$$h(\mathbf{x}) \;=\; \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle ,$$

  where $\Psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ is a class-sensitive feature mapping.

- Intuitively, we can think of the elements of $\Psi(\mathbf{x}, y)$ as score functions that assess how good the label y fits the instance $\mathbf{x}$.
- The immediate question, how to construct Ψ ?

Linear Multiclass Predictors

$$h(\mathbf{x}) \;=\; \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle,$$



Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge

Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge
- Example: TF-IDF:

Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge
- Example: TF-IDF:
  - $\mathcal{X}$ = text documents, $\mathcal{Y}$ = topics, d = size of a dictionary of words.

Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge
- Example: TF-IDF:
  - $\mathcal{X}$ = text documents, $\mathcal{Y}$ = topics, d = size of a dictionary of words.
  - Term Frequency: $TF(j, \mathbf{x})$ = number of times word j appears in document $\mathbf{x}$.

Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge
- Example: TF-IDF:
  - $\mathcal{X}$ = text documents, $\mathcal{Y}$ = topics, d = size of a dictionary of words.
  - Term Frequency: $TF(j, \mathbf{x})$ = number of times word j appears in document $\mathbf{x}$.
  - Document Frequency: $DF(j, y)$ = number of times word j appears in documents in our training set that are not about topic y.
    (measures if word j is frequent in other topics)

Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge
- Example: TF-IDF:
  - $\mathcal{X}$ = text documents, $\mathcal{Y}$ = topics, d = size of a dictionary of words.
  - Term Frequency: $TF(j, \mathbf{x})$ = number of times word j appears in document $\mathbf{x}$.
  - Document Frequency: $DF(j, y)$ = number of times word j appears in documents in our training set that are not about topic y. (measures if word j is frequent in other topics)
  - Term-Frequency-Inverse-Document-Frequency:

$$\Psi_j(\mathbf{x}, y) = TF(j, \mathbf{x}) \log\left(\frac{m}{DF(j,y)}\right)$$

Class-sensitive feature mapping

- Choosing Ψ is a "feature learning" problem, and is similar to choosing a kernel in SVM. Hence, in general, it requires some prior knowledge
- Example: TF-IDF:
  - $\mathcal{X}$ = text documents, $\mathcal{Y}$ = topics, d = size of a dictionary of words.
  - Term Frequency: $TF(j, \mathbf{x})$ = number of times word j appears in document $\mathbf{x}$.
  - Document Frequency: $DF(j, y)$ = number of times word j appears in documents in our training set that are not about topic y.
    (measures if word j is frequent in other topics)
  - Term-Frequency-Inverse-Document-Frequency:

$$\Psi_j(\mathbf{x}, y) = TF(j, \mathbf{x}) \log\left(\frac{m}{DF(j,y)}\right)$$

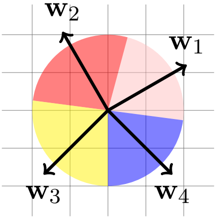
  - Intuitively, $\Psi_j(\mathbf{x}, y)$ should be large if word j appears a lot in $\mathbf{x}$ but does not appear at all in documents that are not on topic y
    In such case, we tend to believe that the document $\mathbf{x}$ is on topic y

The Multi-vector Construction

$$h(\mathbf{x}) = \underset{y \in [k]}{\operatorname{argmax}} (W\mathbf{x})_y = \underset{y \in [k]}{\operatorname{argmax}} \langle \mathbf{w}_y, \mathbf{x} \rangle$$

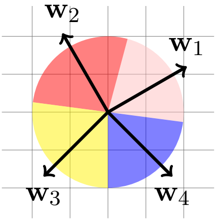
The Multi-vector Construction

$$h(\mathbf{x}) = \underset{y \in [k]}{\operatorname{argmax}} (W\mathbf{x})_y = \underset{y \in [k]}{\operatorname{argmax}} \langle \mathbf{w}_y, \mathbf{x} \rangle$$



The Multi-vector Construction

$$h(\mathbf{x}) = \underset{y \in [k]}{\operatorname{argmax}} (W\mathbf{x})_y = \underset{y \in [k]}{\operatorname{argmax}} \langle \mathbf{w}_y, \mathbf{x} \rangle$$



Can be written as $\operatorname{argmax}_y \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$ for

$$\Psi(\mathbf{x}, y) = [\ \underbrace{0, \ldots, 0}_{\in \mathbb{R}^{(y-1)n}}, \underbrace{x_1, \ldots, x_n}_{\in \mathbb{R}^n}, \underbrace{0, \ldots, 0}_{\in \mathbb{R}^{(k-y)n}}\ ] \ .$$

Cost-sensitive losses

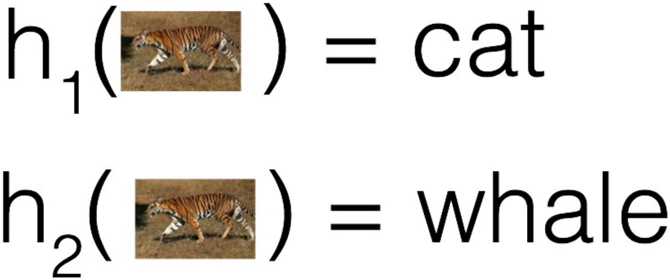
Which prediction is worse?

$h_1($  $) = \text{cat}$

$h_2($  $) = \text{whale}$

Cost-sensitive losses

Which prediction is worse?



- Cost function: $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$
- Zero-one loss is a special case: $\Delta(y, y') = \mathbb{1}_{[y' \neq y]}$

ERM

- ERM problem: find $\mathbf{w}$ that minimizes

$$L_S(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} \Delta(h_{\mathbf{w}}(\mathbf{x}_i), y_i) \ .$$

ERM

- ERM problem: find $\mathbf{w}$ that minimizes

$$L_S(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} \Delta(h_{\mathbf{w}}(\mathbf{x}_i), y_i) \ .$$

- In the realizable case, equivalent to the linear programming problem:

$$\forall i \in [m],\ \forall y \in \mathcal{Y} \setminus \{y_i\},\quad \langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i)\rangle > \langle \mathbf{w}, \Psi(\mathbf{x}_i, y)\rangle$$

ERM

- ERM problem: find $\mathbf{w}$ that minimizes

$$L_S(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} \Delta(h_{\mathbf{w}}(\mathbf{x}_i), y_i) \ .$$

- In the realizable case, equivalent to the linear programming problem:

$$\forall i \in [m],\ \forall y \in \mathcal{Y} \setminus \{y_i\},\ \ \langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i)\rangle > \langle \mathbf{w}, \Psi(\mathbf{x}_i, y)\rangle$$

- NP hard in the non-realizable case so we'll use a surrogate convex loss

Generalized Hinge Loss

Multiclass predictor:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y'}) \rangle \ .$$

Generalized Hinge Loss

Multiclass predictor:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}') \rangle \ .$$

By definition:

$$\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle \leq \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) \rangle \ .$$

Generalized Hinge Loss

Multiclass predictor:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y'}) \rangle \ .$$

By definition:

$$\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle \leq \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) \rangle \ .$$

Therefore,

$$\Delta(h_{\mathbf{w}}(\mathbf{x}), y) \ \leq \ \Delta(h_{\mathbf{w}}(\mathbf{x}), y) + \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) - \Psi(\mathbf{x}, y) \rangle$$

Generalized Hinge Loss

Multiclass predictor:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y'}) \rangle \ .$$

By definition:

$$\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle \leq \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) \rangle \ .$$

Therefore,

$$\begin{aligned} \Delta(h_{\mathbf{w}}(\mathbf{x}), y) &\leq \Delta(h_{\mathbf{w}}(\mathbf{x}), y) + \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) - \Psi(\mathbf{x}, y) \rangle \\ &\leq \max_{y' \in \mathcal{Y}} \left(\Delta(y', y) + \langle \mathbf{w}, \Psi(\mathbf{x}, y') - \Psi(\mathbf{x}, y) \rangle \right) \end{aligned}$$

Generalized Hinge Loss

Multiclass predictor:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}') \rangle \ .$$

By definition:

$$\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle \le \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) \rangle \ .$$

Therefore,

$$\begin{aligned} \Delta(h_{\mathbf{w}}(\mathbf{x}), y) &\le \Delta(h_{\mathbf{w}}(\mathbf{x}), y) + \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) - \Psi(\mathbf{x}, y) \rangle \\ &\le \max_{y' \in \mathcal{Y}} \left(\Delta(y', y) + \langle \mathbf{w}, \Psi(\mathbf{x}, y') - \Psi(\mathbf{x}, y) \rangle \right) \\ &\stackrel{\text{def}}{=} \ell(\mathbf{w}, (\mathbf{x}, y)) \end{aligned}$$

Generalized Hinge Loss

Multiclass predictor:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{y' \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}') \rangle \ .$$

By definition:

$$\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle \le \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) \rangle \ .$$

Therefore,

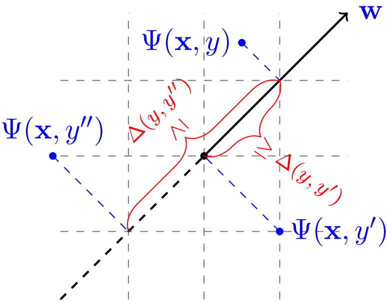
$$\begin{aligned}
\Delta(h_{\mathbf{w}}(\mathbf{x}), y) &\le \Delta(h_{\mathbf{w}}(\mathbf{x}), y) + \langle \mathbf{w}, \Psi(\mathbf{x}, h_{\mathbf{w}}(\mathbf{x})) - \Psi(\mathbf{x}, y) \rangle \\
&\le \max_{y' \in \mathcal{Y}} \left(\Delta(y', y) + \langle \mathbf{w}, \Psi(\mathbf{x}, y') - \Psi(\mathbf{x}, y) \rangle \right) \\
&\stackrel{\text{def}}{=} \ell(\mathbf{w}, (\mathbf{x}, y))
\end{aligned}$$

- The generalized hinge loss is convex and ρ-Lipschitz, for $\rho = \max_{y' \in \mathcal{Y}} \|\Psi(\mathbf{x}, y') - \Psi(\mathbf{x}, y)\|$.

Generalized Hinge Loss

The generalized hinge loss equals zero when:

$$\forall y' \in \mathcal{Y} \setminus \{y\}, \quad \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle \geq \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}') \rangle + \Delta(y', y) \ .$$



Multiclass SVM

Parameters:

- class sensitive feature mapping, Ψ
- cost function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$
- regularization parameter $\lambda > 0$

Solve:

$$\underset{\mathbf{w} \in \mathbb{R}^d}{\operatorname{argmin}} \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max_{y' \in \mathcal{Y}} \left(\Delta(y', y_i) + \langle \mathbf{w}, \Psi(\mathbf{x}_i, y') - \Psi(\mathbf{x}_i, y_i) \rangle \right) \right)$$

Output:

$$h_{\mathbf{w}}(\mathbf{x}) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$$

SGD implementation

- Loss function:

$$\ell(\mathbf{w},(\mathbf{x},y)) = \max_{y' \in \mathcal{Y}} \left(\Delta(y',y) + \langle \mathbf{w}, \Psi(\mathbf{x},y') - \Psi(\mathbf{x},y) \rangle \right)$$

- Sub-gradient calculation
  - find $\hat{y} \in \operatorname{argmax}_{y' \in \mathcal{Y}} \left(\Delta(y',y) + \langle \mathbf{w}^{(t)}, \Psi(\mathbf{x},y') - \Psi(\mathbf{x},y) \rangle \right)$
  - set $\mathbf{v}_t = \Psi(\mathbf{x},\hat{y}) - \Psi(\mathbf{x},y)$

Outline

Structured Output Prediction

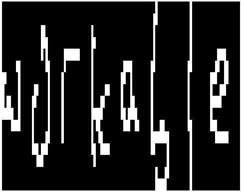
Structured Output Prediction = Multiclass problems in which $\mathcal{Y}$ is very large but is endowed with a predefined structure

Structured Output Prediction

Structured Output Prediction = Multiclass problems in which $\mathcal{Y}$ is very large but is endowed with a predefined structure

Example — Optical Character Recognitio (OCR):

- $\mathcal{X}$ = set of images
- $\mathcal{Y}$ all possible words in English



$\longrightarrow$ workable

Structured Output Prediction

The good news: sample complexity of multiclass SVM does not depend on $|\mathcal{Y}|$ but rather on $\|\Psi(\mathbf{x}, y)\|$ and $\|\mathbf{w}\|$.

Structured Output Prediction

The good news: sample complexity of multiclass SVM does not depend on $|\mathcal{Y}|$ but rather on $\|\Psi(\mathbf{x}, y)\|$ and $\|\mathbf{w}\|$.

However, the huge size of $\mathcal{Y}$ poses computational challenges:

1. To apply the multiclass prediction we need to solve a maximization problem over $\mathcal{Y}$. How can we predict efficiently when $\mathcal{Y}$ is so large?
2. How do we train $\mathbf{w}$ efficiently? In particular, to apply the SGD rule we again need to solve a maximization problem over $\mathcal{Y}$.

Structured Output Prediction

The good news: sample complexity of multiclass SVM does not depend on $|\mathcal{Y}|$ but rather on $\|\Psi(\mathbf{x}, y)\|$ and $\|\mathbf{w}\|$.

However, the huge size of $\mathcal{Y}$ poses computational challenges:

1. To apply the multiclass prediction we need to solve a maximization problem over $\mathcal{Y}$. How can we predict efficiently when $\mathcal{Y}$ is so large?
2. How do we train $\mathbf{w}$ efficiently? In particular, to apply the SGD rule we again need to solve a maximization problem over $\mathcal{Y}$.

Solution:

- Endow Ψ and Δ with structure that allows fast maximization over $\mathcal{Y}$

Modeling for OCR

- For simplicity assume $\mathcal{Y} =$ words of length r

Modeling for OCR

- For simplicity assume $\mathcal{Y} =$ words of length r
- Define $\Delta(\mathbf{y}', \mathbf{y}) = \frac{1}{r} \sum_{i=1}^{r} \mathbb{1}_{[y_i \neq y'_i]}$

Modeling for OCR

- For simplicity assume $\mathcal{Y} =$ words of length r
- Define $\Delta(\mathbf{y}', \mathbf{y}) = \frac{1}{r} \sum_{i=1}^{r} \mathbb{1}_{[y_i \neq y'_i]}$
- Think about $\mathbf{x} \in \mathcal{X}$ as a matrix of size $n \times r$ (after segmentation to the r words)

Modeling for OCR

- For simplicity assume $\mathcal{Y}$ = words of length r
- Define $\Delta(\mathbf{y}', \mathbf{y}) = \frac{1}{r}\sum_{i=1}^{r} \mathbb{1}_{[y_i \neq y'_i]}$
- Think about $\mathbf{x} \in \mathcal{X}$ as a matrix of size $n \times r$ (after segmentation to the r words)
- Type 1 features: (capture pixels in the image whose gray level values are indicative to a certain letter)

$$\Psi_{i,j,1}(\mathbf{x}, \mathbf{y}) = \frac{1}{r}\sum_{t=1}^{r} x_{i,t}\, \mathbb{1}_{[y_t = j]} \ .$$

Modeling for OCR

- For simplicity assume $\mathcal{Y} =$ words of length r
- Define $\Delta(\mathbf{y}', \mathbf{y}) = \frac{1}{r}\sum_{i=1}^{r} \mathbb{1}_{[y_i \neq y'_i]}$
- Think about $\mathbf{x} \in \mathcal{X}$ as a matrix of size $n \times r$ (after segmentation to the r words)
- Type 1 features: (capture pixels in the image whose gray level values are indicative to a certain letter)

$$\Psi_{i,j,1}(\mathbf{x}, \mathbf{y}) = \frac{1}{r}\sum_{t=1}^{r} x_{i,t}\, \mathbb{1}_{[y_t = j]} \ .$$

- Type 2 features: (capture "it is likely to see the pair 'qu' in a word")

$$\Psi_{i,j,2}(\mathbf{x}, \mathbf{y}) = \frac{1}{r}\sum_{t=2}^{r} \mathbb{1}_{[y_t = i]}\, \mathbb{1}_{[y_{t-1} = j]} \ .$$

Modeling for OCR

- For simplicity assume $\mathcal{Y}$ = words of length r
- Define $\Delta(\mathbf{y}', \mathbf{y}) = \frac{1}{r}\sum_{i=1}^{r} \mathbb{1}_{[y_i \neq y'_i]}$
- Think about $\mathbf{x} \in \mathcal{X}$ as a matrix of size $n \times r$ (after segmentation to the r words)
- Type 1 features: (capture pixels in the image whose gray level values are indicative to a certain letter)

$$\Psi_{i,j,1}(\mathbf{x}, \mathbf{y}) = \frac{1}{r}\sum_{t=1}^{r} x_{i,t}\, \mathbb{1}_{[y_t = j]} \ .$$

- Type 2 features: (capture "it is likely to see the pair 'qu' in a word")

$$\Psi_{i,j,2}(\mathbf{x}, \mathbf{y}) = \frac{1}{r}\sum_{t=2}^{r} \mathbb{1}_{[y_t = i]}\, \mathbb{1}_{[y_{t-1} = j]} \ .$$

- Claim: The problem $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y})\rangle$ can be solved efficiently using dynamic programming

Dynamic Programming

- Can rewrite the problem as:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{t=1}^{r} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y_t, y_{t-1}) \rangle \ .$$

Dynamic Programming

- Can rewrite the problem as:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{t=1}^{r} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y_t, y_{t-1}) \rangle \ .$$

- Maintain a matrix $M \in \mathbb{R}^{q,r}$ such that

$$M_{s,\tau} = \max_{(y_1,\ldots,y_\tau):y_\tau = s} \sum_{t=1}^{\tau} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y_t, y_{t-1}) \rangle$$

and observe: Maximum of $\langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$ equals to $\max_s M_{s,r}$

Dynamic Programming

- Can rewrite the problem as:

$$h_{\mathbf{w}}(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{t=1}^{r} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y_t, y_{t-1}) \rangle \ .$$

- Maintain a matrix $M \in \mathbb{R}^{q,r}$ such that

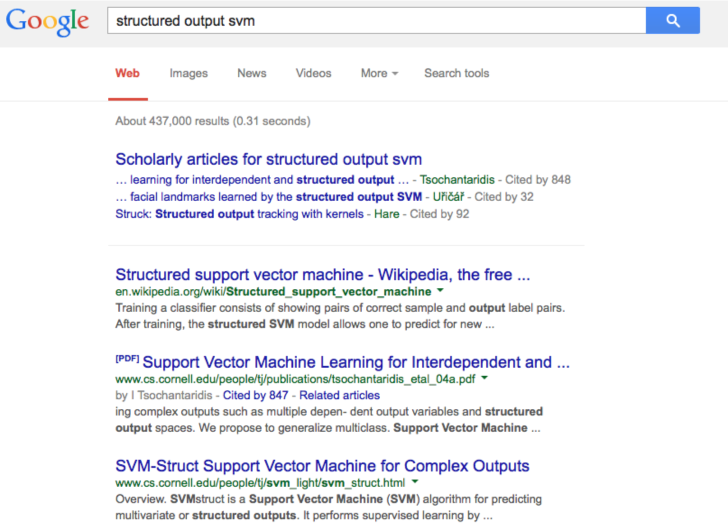
$$M_{s,\tau} = \max_{(y_1,\ldots,y_\tau):y_\tau = s} \sum_{t=1}^{\tau} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, y_t, y_{t-1}) \rangle$$

  and observe: Maximum of $\langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$ equals to $\max_s M_{s,r}$
- Calculate M in a recursive manner:

$$M_{s,\tau} \ = \ \max_{s'} \ \left(M_{s',\tau-1} + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, s, s') \rangle \right) \ .$$

Outline

Ranking


Google
structured output svm
Web
Images
News
Videos
More
Search tools
About 437,000 results (0.31 seconds)
Scholarly articles for structured output svm
… learning for interdependent and structured output … - Tsochantaridis - Cited by 848
… facial landmarks learned by the structured output SVM - Uřičář - Cited by 32
Struck: Structured output tracking with kernels - Hare - Cited by 92
Structured support vector machine - Wikipedia, the free …
en.wikipedia.org/wiki/Structured_support_vector_machine
Training a classifier consists of showing pairs of correct sample and output label pairs. After training, the structured SVM model allows one to predict for new …
[PDF] Support Vector Machine Learning for Interdependent and …
www.cs.cornell.edu/people/tj/publications/tsochantaridis_etal_04a.pdf
by I Tsochantaridis - Cited by 847 - Related articles
ing complex outputs such as multiple depen- dent output variables and structured output spaces. We propose to generalize multiclass. Support Vector Machine …
SVM-Struct Support Vector Machine for Complex Outputs
www.cs.cornell.edu/people/tj/svm_light/svm_struct.html
Overview. SVMstruct is a Support Vector Machine (SVM) algorithm for predicting multivariate or structured outputs. It performs supervised learning by …

Ranking

- Ranking = ordering instances according to relevance

Ranking

- Ranking = ordering instances according to relevance
- Let $\mathcal{X}^* = \bigcup_{n=1}^{\infty} \mathcal{X}^n$ be the set of all sequences of instances from $\mathcal{X}$ of arbitrary length.

Ranking

- Ranking = ordering instances according to relevance
- Let $\mathcal{X}^* = \bigcup_{n=1}^{\infty} \mathcal{X}^n$ be the set of all sequences of instances from $\mathcal{X}$ of arbitrary length.
- Ranking hypothesis: a function that receives a sequence of instances $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_r) \in \mathcal{X}^*$, and returns a permutation of $[r]$

Ranking

- Ranking = ordering instances according to relevance
- Let $\mathcal{X}^* = \bigcup_{n=1}^{\infty} \mathcal{X}^n$ be the set of all sequences of instances from $\mathcal{X}$ of arbitrary length.
- Ranking hypothesis: a function that receives a sequence of instances $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_r) \in \mathcal{X}^*$, and returns a permutation of $[r]$
- Equivalently, $h(\bar{\mathbf{x}}) = \mathbf{y} \in \mathbb{R}^r$, where $\pi(\mathbf{y})$ is the induced permutation.

Ranking

- Ranking = ordering instances according to relevance
- Let $\mathcal{X}^* = \bigcup_{n=1}^{\infty} \mathcal{X}^n$ be the set of all sequences of instances from $\mathcal{X}$ of arbitrary length.
- Ranking hypothesis: a function that receives a sequence of instances $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_r) \in \mathcal{X}^*$, and returns a permutation of $[r]$
- Equivalently, $h(\bar{\mathbf{x}}) = \mathbf{y} \in \mathbb{R}^r$, where $\pi(\mathbf{y})$ is the induced permutation.
- E.g.

$\mathbf{y}$	sorted $\mathbf{y}$	$\pi(\mathbf{y})$
2	-1	4
1	0.5	3
6	1	5
-1	2	1
0.5	6	2

Ranking

- Ranking = ordering instances according to relevance
- Let $\mathcal{X}^* = \bigcup_{n=1}^{\infty} \mathcal{X}^n$ be the set of all sequences of instances from $\mathcal{X}$ of arbitrary length.
- Ranking hypothesis: a function that receives a sequence of instances $\bar{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_r) \in \mathcal{X}^*$, and returns a permutation of $[r]$
- Equivalently, $h(\bar{\mathbf{x}}) = \mathbf{y} \in \mathbb{R}^r$, where $\pi(\mathbf{y})$ is the induced permutation.
- E.g.

$\mathbf{y}$	sorted $\mathbf{y}$	$\pi(\mathbf{y})$
2	-1	4
1	0.5	3
6	1	5
-1	2	1
0.5	6	2

- $\pi(\mathbf{y})_i$ is the position of y_i in the sorted vector. Top-ranked instances are those that achieve the highest values in $\pi(\mathbf{y})$.

Loss functions for Ranking

Kendall-tau loss:

- Count the number of pairs (i, j) that are in different order in the two permutations:

$$\Delta(\mathbf{y}', \mathbf{y}) = \frac{2}{r(r-1)} \sum_{i=1}^{r-1} \sum_{j=i+1}^{r} \mathbb{1}_{[\mathrm{sign}(y'_i - y'_j) \neq \mathrm{sign}(y_i - y_j)]} \ .$$

  More useful than the 0-1 loss as it reflects the level of similarity between the two rankings.

Loss functions for Ranking

Normalized Discounted Cumulative Gain (NDCG):

- Emphasizes correctness at the top of the list by using a discount
- Define a discounted cumulative gain measure:

$$G(\mathbf{y}', \mathbf{y}) = \sum_{i=1}^{r} D(\pi(\mathbf{y}')_i) \; y_i$$

  where D is a decreasing function
- Normalized discounted cumulative gain

$$\Delta(\mathbf{y}', \mathbf{y}) = 1 - \frac{G(\mathbf{y}', \mathbf{y})}{G(\mathbf{y}, \mathbf{y})}$$

- NDCG is often used to evaluate the performance of search engines since in such applications it makes sense to completely ignore elements which are not at the top of the ranking.

Discounted cumulative gain — example

$$G(\mathbf{y}',\mathbf{y}) = \sum_{i=1}^{r} D(\pi(\mathbf{y}')_i)\ y_i$$

$\mathbf{y}'$	sorted $\mathbf{y}'$	$\pi(\mathbf{y}')$	$D(\pi(\mathbf{y}'))$	$\mathbf{y}$	$D(\pi(\mathbf{y}))$	$\pi(\mathbf{y})$	sorted $\mathbf{y}$
2	-1	4	1	5	1	4	-2
1	0.5	3	0	-2	0	1	1
6	1	5	2	6	2	5	3
-1	2	1	0	1	0	2	5
0.5	6	2	0	3	0	3	6

$$\Delta(\mathbf{y}',\mathbf{y}) = 1 - \frac{G(\mathbf{y}',\mathbf{y})}{G(\mathbf{y},\mathbf{y})} = 1 - \frac{17}{17} = 0$$

Linear Predictors for Ranking

- Linear predictor

$$h_{\mathbf{w}}((\mathbf{x}_1, \ldots, \mathbf{x}_r)) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

Linear Predictors for Ranking

- Linear predictor

$$h_{\mathbf{w}}((\mathbf{x}_1, \ldots, \mathbf{x}_r)) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

- Let V be the set of all permutations as vectors in $[r]^r$

Linear Predictors for Ranking

- Linear predictor

$$h_{\mathbf{w}}((\mathbf{x}_1, \ldots, \mathbf{x}_r)) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

- Let V be the set of all permutations as vectors in $[r]^r$
- Observe:

$$\pi(\mathbf{y}') \; = \; \operatorname*{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i \, y'_i$$

Linear Predictors for Ranking

- Linear predictor

$$h_{\mathbf{w}}((\mathbf{x}_1, \ldots, \mathbf{x}_r)) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

- Let V be the set of all permutations as vectors in $[r]^r$
- Observe:

$$\pi(\mathbf{y}') = \operatorname*{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i \, y'_i$$

- Therefore, with $\Psi(\bar{\mathbf{x}}, \mathbf{v}) = \sum_{i=1}^{r} v_i \mathbf{x}_i$, we have

$$\begin{aligned} \pi(h_{\mathbf{w}}(\bar{\mathbf{x}})) &= \operatorname*{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x}_i \rangle \\ &= \operatorname*{argmax}_{\mathbf{v} \in V} \langle \mathbf{w}, \sum_{i=1}^{r} v_i \mathbf{x}_i \rangle \\ &= \operatorname*{argmax}_{\mathbf{v} \in V} \langle \mathbf{w}, \Psi(\bar{\mathbf{x}}, \mathbf{v}) \rangle \ . \end{aligned}$$

Linear Predictors for Ranking

- We can now rely on the structured output formulation

Linear Predictors for Ranking

- We can now rely on the structured output formulation
- The resulting hinge loss will be

$$\max_{\mathbf{v} \in V} \left[\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - \pi(\mathbf{y})_i) \langle \mathbf{w}, \mathbf{x}_i \rangle \right]$$

Linear Predictors for Ranking

- We can now rely on the structured output formulation
- The resulting hinge loss will be

$$\max_{\mathbf{v}\in V} \left[\Delta(\mathbf{v},\mathbf{y}) + \sum_{i=1}^{r}(v_i - \pi(\mathbf{y})_i)\,\langle \mathbf{w}, \mathbf{x}_i\rangle\right]$$

- Each step of SGD for the resulting learning problem boils down to "the assignment problem" and can be solved efficiently using the "Hungarian method"

Outline

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?
- Not quite, because of the loss function

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?
- Not quite, because of the loss function
- Example: Fraud detection

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?
- Not quite, because of the loss function
- Example: Fraud detection
  - Each day we get r transactions and need to predict fraud or benign

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?
- Not quite, because of the loss function
- Example: Fraud detection
  - Each day we get r transactions and need to predict fraud or benign
  - 99.9% of the transactions are benign

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?
- Not quite, because of the loss function
- Example: Fraud detection
  - Each day we get r transactions and need to predict fraud or benign
  - 99.9% of the transactions are benign
  - So, the constant "benign" predictor will have zero-one error of 0.1%

Bipartite Ranking

- In ranking, the output is $\mathbf{y} \in \mathbb{R}^r$
- If $y_i = y_j$ we obtain a partial order
- In the extreme case, $\mathbf{y} \in \{\pm 1\}^r$ we obtain just two options (relevant or non-relevant)
- So, is it a binary classification problem ?
- Not quite, because of the loss function
- Example: Fraud detection
  - Each day we get r transactions and need to predict fraud or benign
  - 99.9% of the transactions are benign
  - So, the constant "benign" predictor will have zero-one error of 0.1%
  - But, it's a poor predictor ...

Multivariate Performance Measures

- Suppose we predict $\mathbf{y}' \in \mathbb{R}^r$ and the truth is $\mathbf{y} \in \{\pm 1\}^r$

Multivariate Performance Measures

- Suppose we predict $\mathbf{y}' \in \mathbb{R}^r$ and the truth is $\mathbf{y} \in \{\pm 1\}^r$
- Given a threshold θ, we have 4 numbers of interest:

$$\begin{aligned}
\text{True Positives:} \quad & a = |\{i : y_i = +1 \wedge \operatorname{sign}(y'_i - \theta) = +1\}| \\
\text{False Positives:} \quad & b = |\{i : y_i = -1 \wedge \operatorname{sign}(y'_i - \theta) = +1\}| \\
\text{False Negatives:} \quad & c = |\{i : y_i = +1 \wedge \operatorname{sign}(y'_i - \theta) = -1\}| \\
\text{True Negatives:} \quad & d = |\{i : y_i = -1 \wedge \operatorname{sign}(y'_i - \theta) = -1\}|
\end{aligned}$$

Multivariate Performance Measures

- Suppose we predict $\mathbf{y}' \in \mathbb{R}^r$ and the truth is $\mathbf{y} \in \{\pm 1\}^r$
- Given a threshold θ, we have 4 numbers of interest:

$$\begin{aligned}
\text{True Positives:} \quad & a = |\{i : y_i = +1 \wedge \mathrm{sign}(y'_i - \theta) = +1\}| \\
\text{False Positives:} \quad & b = |\{i : y_i = -1 \wedge \mathrm{sign}(y'_i - \theta) = +1\}| \\
\text{False Negatives:} \quad & c = |\{i : y_i = +1 \wedge \mathrm{sign}(y'_i - \theta) = -1\}| \\
\text{True Negatives:} \quad & d = |\{i : y_i = -1 \wedge \mathrm{sign}(y'_i - \theta) = -1\}|
\end{aligned}$$

- Recall (a.k.a. sensitivity or true positive rate) is the fraction of true positives $\mathbf{y}'$ "catches", namely, $\frac{TP}{TP+FN}$. Recall increases with θ

Multivariate Performance Measures

- Suppose we predict $\mathbf{y}' \in \mathbb{R}^r$ and the truth is $\mathbf{y} \in \{\pm 1\}^r$
- Given a threshold θ, we have 4 numbers of interest:

$$\begin{aligned}
\text{True Positives:} \quad & a = |\{i : y_i = +1 \wedge \mathrm{sign}(y'_i - \theta) = +1\}| \\
\text{False Positives:} \quad & b = |\{i : y_i = -1 \wedge \mathrm{sign}(y'_i - \theta) = +1\}| \\
\text{False Negatives:} \quad & c = |\{i : y_i = +1 \wedge \mathrm{sign}(y'_i - \theta) = -1\}| \\
\text{True Negatives:} \quad & d = |\{i : y_i = -1 \wedge \mathrm{sign}(y'_i - \theta) = -1\}|
\end{aligned}$$

- Recall (a.k.a. sensitivity or true positive rate) is the fraction of true positives $\mathbf{y}'$ "catches", namely, $\frac{TP}{TP+FN}$. Recall increases with θ
- Precision is the fraction of correct predictions among the positive labels we predict, namely, $\frac{TP}{TP+FP}$. Precision decreases with θ

Multivariate Performance Measures

- Suppose we predict $\mathbf{y}' \in \mathbb{R}^r$ and the truth is $\mathbf{y} \in \{\pm 1\}^r$
- Given a threshold θ, we have 4 numbers of interest:

$$\begin{aligned}
\text{True Positives:} \quad & a = |\{i : y_i = +1 \wedge \operatorname{sign}(y'_i - \theta) = +1\}| \\
\text{False Positives:} \quad & b = |\{i : y_i = -1 \wedge \operatorname{sign}(y'_i - \theta) = +1\}| \\
\text{False Negatives:} \quad & c = |\{i : y_i = +1 \wedge \operatorname{sign}(y'_i - \theta) = -1\}| \\
\text{True Negatives:} \quad & d = |\{i : y_i = -1 \wedge \operatorname{sign}(y'_i - \theta) = -1\}|
\end{aligned}$$

- Recall (a.k.a. sensitivity or true positive rate) is the fraction of true positives $\mathbf{y}'$ "catches", namely, $\frac{TP}{TP+FN}$. Recall increases with θ
- Precision is the fraction of correct predictions among the positive labels we predict, namely, $\frac{TP}{TP+FP}$. Precision decreases with θ
- Specificity is the fraction of true negatives that our predictor "catches", namely, $\frac{TN}{TN+FP}$. Specificity decreases with θ

Multivariate Performance Measures

- Suppose we predict $\mathbf{y}' \in \mathbb{R}^r$ and the truth is $\mathbf{y} \in \{\pm 1\}^r$
- Given a threshold θ, we have 4 numbers of interest:

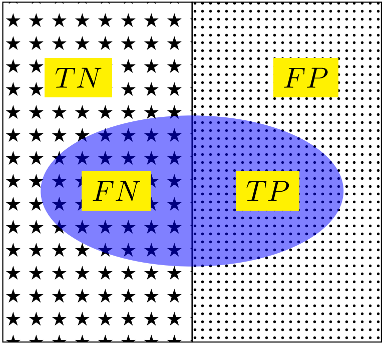
$$\begin{aligned}
\text{True Positives:} \quad & a = |\{i : y_i = +1 \wedge \operatorname{sign}(y'_i - \theta) = +1\}| \\
\text{False Positives:} \quad & b = |\{i : y_i = -1 \wedge \operatorname{sign}(y'_i - \theta) = +1\}| \\
\text{False Negatives:} \quad & c = |\{i : y_i = +1 \wedge \operatorname{sign}(y'_i - \theta) = -1\}| \\
\text{True Negatives:} \quad & d = |\{i : y_i = -1 \wedge \operatorname{sign}(y'_i - \theta) = -1\}|
\end{aligned}$$

- Recall (a.k.a. sensitivity or true positive rate) is the fraction of true positives $\mathbf{y}'$ "catches", namely, $\frac{TP}{TP+FN}$. Recall increases with θ
- Precision is the fraction of correct predictions among the positive labels we predict, namely, $\frac{TP}{TP+FP}$. Precision decreases with θ
- Specificity is the fraction of true negatives that our predictor "catches", namely, $\frac{TN}{TN+FP}$. Specificity decreases with θ
- θ controls the tradeoff between precision and recall

Multivariate Performance Measures

Example:

- y' predicts by the halfspace ('dots' $=$ positive, 'stars' $=$ negative)
- y predicts by the ellipse (within $=$ positive, outside $=$ negative)

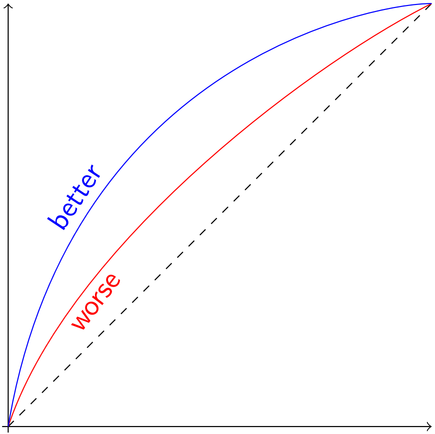


$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

Receiver operating characteristic (ROC) curve

Recall=TPR=$\frac{TP}{TP+FN}$



1-sensitivity=FPR=$\frac{FP}{FP+TN}$

Multivariate Performance Measures

For every θ we can define a single number that measures the performance.

- Averaging sensitivity and specificity: $\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$. This is the accuracy on positive examples averaged with the accuracy on negative examples.
- F_1-score: The F_1 score is the harmonic mean of the precision and recall: $\frac{2}{\frac{1}{\text{Precision}}+\frac{1}{\text{Recall}}}$.
- Recall at k: We measure the recall while the prediction must contain at most k positive labels. That is, we should set θ so that $TP+FP \le k$. E.g., convenient for fraud detection
- Precision at k: We measure the precision while the prediction must contain at least k positive labels. That is, we should set θ so that $TP+FP \ge k$.

Multivariate Performance Measures

- Zero-one loss can be written as $\frac{FP+TN}{m}$

Multivariate Performance Measures

- Zero-one loss can be written as $\frac{FP+TN}{m}$
- If 99.9% of the examples are negatively labeled, zero-one loss of the "all negative" predictor is 0.1%

Multivariate Performance Measures

- Zero-one loss can be written as $\frac{FP+TN}{m}$
- If 99.9% of the examples are negatively labeled, zero-one loss of the "all negative" predictor is 0.1%
- But, the recall of such predictor is 0, hence the F_1 score is also 0, which means that the loss $1 - F_1$ will be 1 (worst possible).

Multivariate Performance Measures

- Zero-one loss can be written as $\frac{FP+TN}{m}$
- If 99.9% of the examples are negatively labeled, zero-one loss of the "all negative" predictor is 0.1%
- But, the recall of such predictor is 0, hence the F_1 score is also 0, which means that the loss $1 - F_1$ will be 1 (worst possible).
- Conclusion: we need to train our predictor with an appropriate loss function for the problem

Linear Predictors for Multivariate Measures

- Linear predictor:

$$h_{\mathbf{w}}(\bar{\mathbf{x}}) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

Linear Predictors for Multivariate Measures

- Linear predictor:

$$h_{\mathbf{w}}(\bar{\mathbf{x}}) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

- Actual prediction:

$$\mathbf{b}(\mathbf{y}') = (\mathrm{sign}(y'_1 - \theta), \ldots, \mathrm{sign}(y'_r - \theta)) \ \in \ \{\pm 1\}^r \ .$$

Linear Predictors for Multivariate Measures

- Linear predictor:

$$h_{\mathbf{w}}(\bar{\mathbf{x}}) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

- Actual prediction:

$$\mathbf{b}(\mathbf{y}') = (\mathrm{sign}(y'_1 - \theta), \ldots, \mathrm{sign}(y'_r - \theta)) \ \in \ \{\pm 1\}^r \ .$$

- Observe, for all performance measures defined before, exists some $V \subseteq \{\pm 1\}^r$ s.t.

$$\mathbf{b}(\mathbf{y}') = \operatorname*{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i y'_i \ .$$

Linear Predictors for Multivariate Measures

- Linear predictor:

$$h_{\mathbf{w}}(\bar{\mathbf{x}}) = (\langle \mathbf{w}, \mathbf{x}_1 \rangle, \ldots, \langle \mathbf{w}, \mathbf{x}_r \rangle) \ .$$

- Actual prediction:

$$\mathbf{b}(\mathbf{y}') = (\mathrm{sign}(y'_1 - \theta), \ldots, \mathrm{sign}(y'_r - \theta)) \ \in \ \{\pm 1\}^r \ .$$

- Observe, for all performance measures defined before, exists some $V \subseteq \{\pm 1\}^r$ s.t.

$$\mathbf{b}(\mathbf{y}') = \underset{\mathbf{v} \in V}{\mathrm{argmax}} \ \sum_{i=1}^{r} v_i y'_i \ .$$

- E.g., for recall at k take V to be all vectors in $\{\pm 1\}^r$ that has at most k pluses

Linear Predictors for Multivariate Measures

- Hinge loss for multivariate:

$$\begin{aligned}
\Delta(h_{\mathbf{w}}(\bar{\mathbf{x}}), \mathbf{y}) &= \Delta(\mathbf{b}(h_{\mathbf{w}}(\bar{\mathbf{x}})), \mathbf{y}) \\
&\le \Delta(\mathbf{b}(h_{\mathbf{w}}(\bar{\mathbf{x}})), \mathbf{y}) + \sum_{i=1}^{r} (b_i(h_{\mathbf{w}}(\bar{\mathbf{x}})) - y_i) \langle \mathbf{w}, \mathbf{x}_i \rangle \\
&\le \max_{\mathbf{v} \in V} \left[\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - y_i) \langle \mathbf{w}, \mathbf{x}_i \rangle \right]
\end{aligned}$$

SGD for Multivariate Measures

- Applying SGD involves solving the maximization problem in the definition of the loss

$$\operatorname*{argmax}_{\mathbf{v}\in V} \left[\Delta(\mathbf{v},\mathbf{y}) + \sum_{i=1}^{r}(v_i - y_i)\langle \mathbf{w}, \mathbf{x}_i\rangle\right]$$

SGD for Multivariate Measures

- Applying SGD involves solving the maximization problem in the definition of the loss

$$\underset{\mathbf{v}\in V}{\operatorname{argmax}} \left[\Delta(\mathbf{v},\mathbf{y}) + \sum_{i=1}^{r}(v_i - y_i)\langle \mathbf{w}, \mathbf{x}_i\rangle\right]$$

- Key idea, suppose Δ only depends on $a = TP, b = FP$ and partition V into sets of the form

$$\bar{\mathcal{Y}}_{a,b} = \{\mathbf{v} \;:\; |\{i : v_i = 1 \wedge y_i = 1\}| = a \,\wedge\, |\{i : v_i = 1 \wedge y_i = -1\}| = b\}$$

SGD for Multivariate Measures

- Applying SGD involves solving the maximization problem in the definition of the loss

$$\underset{\mathbf{v}\in V}{\operatorname{argmax}} \left[\Delta(\mathbf{v},\mathbf{y}) + \sum_{i=1}^{r}(v_i - y_i)\left\langle \mathbf{w}, \mathbf{x}_i \right\rangle\right]$$

- Key idea, suppose Δ only depends on $a = TP, b = FP$ and partition V into sets of the form

$$\bar{\mathcal{Y}}_{a,b} = \{\mathbf{v} \ : \ |\{i : v_i = 1 \wedge y_i = 1\}| = a \ \wedge \ |\{i : v_i = 1 \wedge y_i = -1\}| = b\}$$

- Withint each $\bar{\mathcal{Y}}_{a,b}$ the value of Δ is constant, so we only need to maximize the expression

$$\max_{\mathbf{v}\in\bar{\mathcal{Y}}_{a,b}} \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x}_i \rangle \ .$$

SGD for Multivariate Measures

- Applying SGD involves solving the maximization problem in the definition of the loss

$$\underset{\mathbf{v} \in V}{\operatorname{argmax}} \left[\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - y_i) \langle \mathbf{w}, \mathbf{x}_i \rangle \right]$$

- Key idea, suppose Δ only depends on $a = TP, b = FP$ and partition V into sets of the form

$$\bar{\mathcal{Y}}_{a,b} = \{\mathbf{v} \; : \; |\{i : v_i = 1 \wedge y_i = 1\}| = a \wedge |\{i : v_i = 1 \wedge y_i = -1\}| = b\}$$

- Withint each $\bar{\mathcal{Y}}_{a,b}$ the value of Δ is constant, so we only need to maximize the expression

$$\max_{\mathbf{v} \in \bar{\mathcal{Y}}_{a,b}} \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x}_i \rangle \; .$$

- This can be done efficiently by sorting the examples according to $\langle \mathbf{w}, \mathbf{x}_i \rangle$

Summary

- Multiclass problems
- Simple reductions
- Linear predictors and Multiclass SVM
- Structured Output Prediction
- Ranking
- Bipartite Ranking and Multivariate Performance Measures
- Efficient sub-gradient calculations using combinatorial optimization (dynamic programming, assignment problems, sorting)