Reductions

COMS 4721 Spring 2022
Daniel Hsu

Reductions in ML

- Some ML problems are apparently more "complex" than binary classification or regression
  - Multi-class classification (which bird is depicted in the image?)
  - Multi-label prediction (which bird**s** are depicted in the image?)
  - Ranking search results
  - Parsing sentences
  - Online decision-making
  - ...

Reductions in ML

- Some ML problems are apparently more "complex" than binary classification or regression
  - Multi-class classification (which bird is depicted in the image?)
  - Multi-label prediction (which bird**s** are depicted in the image?)
  - Ranking search results
  - Parsing sentences
  - Online decision-making
  - ...
- Possible approach: **reduce** the "complex" problem to a "simpler" problem like binary classification or regression (or a bunch of such simpler problems)

Reductions in ML

- Some ML problems are apparently more "complex" than binary classification or regression
  - Multi-class classification   (which bird is depicted in the image?)
  - Multi-label prediction   (which bird**s** are depicted in the image?)
  - Ranking search results
  - Parsing sentences
  - Online decision-making
  - ...
- Possible approach: **reduce** the "complex" problem to a "simpler" problem like binary classification or regression (or a bunch of such simpler problems)
  - E.g., To learn good ranking functions, exploit technology for learning binary classifiers

    Combine the learned binary classifiers to form a ranking function

Multi-class classification

Statistical model for multi-class classification:

- Outcome/label Y is random variable taking values in a finite, unordered set $\{1, 2, \ldots, K\}$
- Feature vector is a vector of d random variables $\vec{X} := (X_1, \ldots, X_d)$
- Joint distribution of $(\vec{X}, Y)$ reflects the population of examples we anticipate encountering in the future for the present application

Multi-class classification

Statistical model for multi-class classification:

- Outcome/label Y is random variable taking values in a finite, unordered set $\{1, 2, \ldots, K\}$
- Feature vector is a vector of d random variables $\vec{X} := (X_1, \ldots, X_d)$
- Joint distribution of $(\vec{X}, Y)$ reflects the population of examples we anticipate encountering in the future for the present application

- Distribution of Y models outcome of rolling a K-sided die
  (But values on die faces are ignored; may as well be "apple", "banana", "cantaloupe", etc.)

Multi-class classification

Statistical model for multi-class classification:

- Outcome/label Y is random variable taking values in a finite, unordered set $\{1, 2, \ldots, K\}$
- Feature vector is a vector of d random variables $\vec{X} := (X_1, \ldots, X_d)$
- Joint distribution of $(\vec{X}, Y)$ reflects the population of examples we anticipate encountering in the future for the present application

- Distribution of Y models outcome of rolling a K-sided die
  (But values on die faces are ignored; may as well be "apple", "banana", "cantaloupe", etc.)
- Standard benchmark: Error rate (same as in binary classification)

Multi-class $\longrightarrow$ binary classification: One-against-all

One-against-all (a.k.a. one-versus-rest)

Multi-class $\longrightarrow$ binary classification: One-against-all

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

Multi-class $\longrightarrow$ binary classification: One-against-all

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

2. Use binary classification learning technology to learn K different binary classifiers $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_K : \mathbb{R}^d \to \{0, 1\}$

Multi-class $\longrightarrow$ binary classification: One-against-all

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

2. Use binary classification learning technology to learn K different binary classifiers $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_K \colon \mathbb{R}^d \to \{0, 1\}$

3. Combine these binary classifiers into a single multi-class classifier $\hat{F} \colon \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \operatorname*{arg\,max}_{\ell \in \{1,2,\ldots,K\}} \hat{f}_\ell(\vec{x})$$

Multi-class $\longrightarrow$ binary classification: One-against-all

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

2. Use binary classification learning technology to learn K different binary classifiers $\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_K \colon \mathbb{R}^d \to \{0, 1\}$

3. Combine these binary classifiers into a single multi-class classifier $\hat{F} \colon \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \operatorname*{arg\,max}_{\ell \in \{1,2,\ldots,K\}} \hat{f}_\ell(\vec{x})$$

OOPS: Only get correct prediction if *all* K binary classifiers work predict correctly!

Multi-class $\longrightarrow$ binary classification: One-against-all (attempt #2)

One-against-all (a.k.a. one-versus-rest)

Multi-class $\longrightarrow$ binary classification: One-against-all (attempt #2)

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

Multi-class $\longrightarrow$ binary classification: One-against-all (attempt #2)

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_K : \mathbb{R}^d \to [0,1]$ (predict cond. prob. of "positive" class)

Multi-class $\longrightarrow$ binary classification: One-against-all (attempt #2)

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_K \colon \mathbb{R}^d \to [0,1]$ (predict cond. prob. of "positive" class)

3. Combine these predictors into a single multi-class classifier $\hat{F} \colon \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \operatorname*{arg\,max}_{\ell \in \{1,2,\ldots,K\}} \hat{\eta}_\ell(\vec{x})$$

Multi-class $\longrightarrow$ binary classification: One-against-all (attempt #2)

One-against-all (a.k.a. one-versus-rest)

1. Pretend there are K binary classification problems

   ℓ-th problem: class ℓ is treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_K \colon \mathbb{R}^d \to [0,1]$ (predict cond. prob. of "positive" class)

3. Combine these predictors into a single multi-class classifier $\hat{F} \colon \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \operatorname*{arg\,max}_{\ell \in \{1,2,\ldots,K\}} \hat{\eta}_\ell(\vec{x})$$

Better than first attempt: Can tolerate some errors in conditional probability estimates!

Multi-class $\longrightarrow$ binary classification: Error correcting output codes

ECOC (Dietterich & Bakiri, JAIR 1995; Langford & Beygelzimer, COLT 2005)

Multi-class $\longrightarrow$ binary classification: Error correcting output codes

ECOC (Dietterich & Bakiri, JAIR 1995; Langford & Beygelzimer, COLT 2005)

1. Pretend there are T binary classification problems, defined by $S_1, S_2, \ldots S_T \subseteq \{1, 2, \ldots, K\}$

   t-th problem: classes in S_t are treated as "positive"; all other classes are treated as "negative"

Multi-class $\longrightarrow$ binary classification: Error correcting output codes

ECOC (Dietterich & Bakiri, JAIR 1995; Langford & Beygelzimer, COLT 2005)

1. Pretend there are T binary classification problems, defined by $S_1, S_2, \ldots S_T \subseteq \{1, 2, \ldots, K\}$

   t-th problem: classes in S_t are treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_T \colon \mathbb{R}^d \to [0, 1]$ (predict cond. prob. of "positive" class)

Multi-class $\longrightarrow$ binary classification: Error correcting output codes

ECOC (Dietterich & Bakiri, JAIR 1995; Langford & Beygelzimer, COLT 2005)

1. Pretend there are T binary classification problems, defined by $S_1, S_2, \ldots S_T \subseteq \{1, 2, \ldots, K\}$

   t-th problem: classes in S_t are treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_T : \mathbb{R}^d \to [0, 1]$ (predict cond. prob. of "positive" class)

3. Combine these predictors into a single multi-class classifier $\hat{F} : \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \mathrm{decode}(\hat{\eta}_1(\vec{x}), \ldots, \hat{\eta}_T(\vec{x}))$$

   where $\mathrm{decode}(\cdots)$ is based on how one decodes (possibly noisy) messages in telecommunications

Multi-class $\longrightarrow$ binary classification: Error correcting output codes

ECOC (Dietterich & Bakiri, JAIR 1995; Langford & Beygelzimer, COLT 2005)

1. Pretend there are T binary classification problems, defined by $S_1, S_2, \ldots S_T \subseteq \{1, 2, \ldots, K\}$

   t-th problem: classes in S_t are treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_T \colon \mathbb{R}^d \to [0, 1]$ (predict cond. prob. of "positive" class)

3. Combine these predictors into a single multi-class classifier $\hat{F} \colon \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \mathrm{decode}(\hat{\eta}_1(\vec{x}), \ldots, \hat{\eta}_T(\vec{x}))$$

   where $\mathrm{decode}(\cdots)$ is based on how one decodes (possibly noisy) messages in telecommunications

If $S_1, S_2, \ldots, S_T$ are cleverly chosen, this is more robust to errors than OAA!

Multi-class $\longrightarrow$ binary classification: Error correcting output codes

ECOC (Dietterich & Bakiri, JAIR 1995; Langford & Beygelzimer, COLT 2005)

1. Pretend there are T binary classification problems, defined by $S_1, S_2, \ldots S_T \subseteq \{1, 2, \ldots, K\}$

   t-th problem: classes in S_t are treated as "positive"; all other classes are treated as "negative"

2. Use *conditional probability* learning technology to learn K different conditional probability predictors $\hat{\eta}_1, \hat{\eta}_2, \ldots, \hat{\eta}_T : \mathbb{R}^d \to [0, 1]$ (predict cond. prob. of "positive" class)

3. Combine these predictors into a single multi-class classifier $\hat{F} : \mathbb{R}^d \to \{1, 2, \ldots, K\}$

$$\hat{F}(\vec{x}) = \mathrm{decode}(\hat{\eta}_1(\vec{x}), \ldots, \hat{\eta}_T(\vec{x}))$$

   where $\mathrm{decode}(\cdots)$ is based on how one decodes (possibly noisy) messages in telecommunications

If $S_1, S_2, \ldots, S_T$ are cleverly chosen, this is more robust to errors than OAA!
...BUT: Step 2 in ECOC might be harder than Step 2 in OAA

ECOC with Hadamard code

ECOC example:
Reducing multi-class classification with $K = 8$ classes to $T = 7$ binary classification problems

	1	2	3	4	5	6	7	8
S_1	1	0	1	0	1	0	1	0
S_2	1	1	0	0	1	1	0	0
S_3	1	0	0	1	1	0	0	1
S_4	1	1	1	1	0	0	0	0
S_5	1	0	1	0	0	1	0	1
S_6	1	1	0	0	0	0	1	1
S_7	1	0	0	1	0	1	1	0

Decoding (i.e, computing $\hat{F}(\vec{x})$ for new $\vec{x}$):
Compute vector of predictions $\vec{p} := (\hat{\eta}_1(\vec{x}), \hat{\eta}_2(\vec{x}), \ldots, \hat{\eta}_7(\vec{x}))$; return class whose column is closest to $\vec{p}$.

Other reductions

- Multi-label $\longrightarrow$ binary classification

  Similar to multi-class $\longrightarrow$ binary (OAA, ECOC)

- Ranking $\longrightarrow$ binary classification
  - Learn to predict $(\vec{x}_1, \vec{x}_2) \mapsto$ "is $\vec{x}_1$ better than $\vec{x}_2$?"
  - Combine binary predictions using a robust version of comparison-based sorting
- ...