Lecture 8 Multiclass classification

wbg231

December 2022

reduction to binary classification

One vs all/ one vs rest

- setting input space X, outputs space $y = \{1 \cdots k\}$
- we want to train k binary classifiers one for each class

$$h_1 \cdots h_k : X \to \mathbb{R}$$

  that is we are going to train k classifiers to take input sand produce real numbers
- each classifier will distinguish class $i = 1$ from the rest -1
- then we can predict using a majority vote

$$h(x) = argmax_{y \in Y} h(x)$$

- one versus all does not have to be linear necessarily

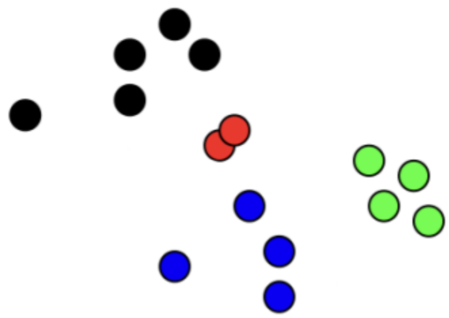
all versus all, one vs one, all pairs

- input space X, output space $Y = \{\cdots k\}$
- we train l chose 2 binary classifiers one for each $i \in [1, k], j \in [i+1, k]$
- classifier $h_{i,j}$ finds $P(y = i)$ as class 1 and inverse as -1
- predict using a majority vote

$$h(x) = argmax_{i \in (1 \cdots k)} \sum_{(j \neq k)} h_{i,j} \mathbb{I}(i < j) - h_{i,j} \mathbb{I}(j < i)$$

four class example

- consider the following dataset



- assume that each pair of classes is linerly sperable in the linear ava model

ava vs OVA

- AVA grows quadratically in the number iof classifers we need to train where as one versus all frows liniearly when training one versus all each classfier needs to train on total number of data point's (that is total data is marked for every class) when training all vs all each class is nly consdierd when the data that deal with it is on one of the two classes so we are dealing with $\frac{n}{k}$ training points for each of the two classes
- these lack theoretical strenght but are simple and work well in practice
- one vs all can have real class imballance issues
- ava has small training sets
- so caleberation is an issue for btoh models
- we do not likely abritary tie breaks

code word for labels

- we can encode labels as binary classes and predict the bits directly

-

class	h_1	h_2	h_3	h_4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

- so we can represent four classes in ova like the above this using 1 bit per class, how can we reduce this number?
- each bit length can fit 2^k classes at max
- so suppose we have 6 classes we are representing in 6 bits
- so that is $Y \in \{c_1 \cdots c_8\}$ then we find some bit representation of Y call it $B = \{0, 1\}^6$ and we are learning 6 binary classifiers $h_1 \cdots h_6 : X \to \{-1, 1\}$
- such that
  $$h_i(x) = P(\text{bit}_i = 1)$$
  in the binary representation
- predict the closest label in terms of hamming distance

error correcting output codes:summary

- this si more efficient than OVA
- but want to ballance number of bits (ie compression) and robustness. the fewer bits we use less bit combinations there are that do not correspond to a class, meaning we can recover form fewer of our binary classifiers making mistakes than we other wise could

review

- it is unclear how to generalize this to massive number of classes like image classification

multi class loss

binary logistic regression

- the task is given an input x we would like to output a classification between (0,1). we do this with a linear model with transformation function

$$P(X=1) = f(x) = sigmoid(X) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-w^t x - b}}$$

- the other class is represented as

$$P(y=-1|x,w) = 1 - f(x) = sigmoid(-z)$$

- so we are implicitly learning parameters for two classes w, b and $-w, -b$ (the second class is fully determined by the first in this case though)

multiclass logistic

- we can expand this with the softmax function where we learn $w_i \quad \forall i \in [1,c]$ and predict

$$P(y=c|x,w) = \frac{e^{w_c^t x + b_c}}{\sum_c w_c^t x + b_c}$$

- the loss function here is given by

$$L = \sum_i -y_c^i log(f_c(x^i))$$

this is more or less the sum of our negative likelihood times our true class label

compare this to one versus all

- this holds for both multiclass and ova
- our base hypothesis space are the linear combinations or score functions $\mathcal{H} = \{h : X \to \mathbb{R}\}$ so this of this as how we make a single predictor
- our multi class hypothesis space for k classes is given by

$$\mathcal{F} = \{x \to argmax_i h_i(x) | h_1 \cdots h_k \in \mathcal{H}\}$$

so that is we make our multiclass class prediction as a argmax of our score function that is we pick the most likely class

- ova objective $h_i(x) > 0$ for x with label i and $h_i(x) < x$ for x with all other labl
- then at test time to predict (x,i) correctly we need

$$h_i(x) > h_j(x) \quad \forall j \neq i$$

multiclass perceptron

- Base linear predictors: $h_i(x) = w_i^T x$ ($w \in \mathrm{R}^d$).
- Multiclass perceptron:

  Given a multiclass dataset $\mathcal{D} = \{(x, y)\}$;
  Initialize $w \leftarrow 0$;
  for $iter = 1, 2, \ldots, T$ **do**
      for $(x, y) \in \mathcal{D}$ **do**
          $\hat{y} = \arg\max_{y' \in \mathcal{Y}} w_{y'}^T x$;
          if $\hat{y} \neq y$ **then** `// We've made a mistake`
              $w_y \leftarrow w_y + x$; `// Move the target-class scorer towards` x
              $w_{\hat{y}} \leftarrow w_{\hat{y}} - x$; `// Move the wrong-class scorer away from` x
          end
      end
  end

- so here we are using a base linear predictor
- we initialize our hyperplanes at the origin (think of each w_i as defining a hyperplane) so then we are learning a set of hyplanes that can be stacked in a matrix as $W \in \mathbb{R}^{k \times d}$
- then for some number of iterations
- for all points in the dataset we set our predicted class as the class with highest score
- if we are wrong, then we move the w_y in the direction of the scorer.

re-write the score

- if we want this to scale we want to reduce W to a single vector w
- we can do a feature transformation

$$w_i^t x = w^t \phi(x, i)$$

$$h_i(x) = h(x, i)$$

- so the logic here is we are going to encode the labels in the feature space it's self
- so think of the score $w^t x = w^t \phi(x, i)$ as the compatibility for a label and input (this makes sense since we are taking an inner product)
- how do we form ϕ

- we can flatten the matrix $w \in \mathbb{R}^{k \times D}$ that is

$$W = \begin{pmatrix} w1,1 & \cdots & w_{1,d} \\ \cdots & \cdots & \cdots \\ wn,1 & \cdots & w_{n,d} \end{pmatrix} \Rightarrow w = (w_{1,1} \cdots w_{1,d}, w_{2,1} \cdots w_{n,d}) \in \mathbb{R}^{n*d}$$

- then we define $\phi : \mathbb{R}^d \times \{1 \cdots k\} \to \mathbb{R}^{n*d}$ such that

$$\phi(x,1) := (x_1 \cdots x_d, 0 \cdots 0)$$

  and

$$\phi(x,i) := (0, 0 \cdots x_1 \cdots x_d \cdots 0)$$

- so kind of think of ϕ as mapping x with something like bassi vectors for this new space
- also note that $w^t \phi(x,i)$ will be zeroes for all ellemets not corresponding to the class we are looking at so it is an orthoginal projection more or less

re-write multiclass perceptron

Multiclass perceptron using the multivector construction.

Given a multiclass dataset $\mathcal{D} = \{(x,y)\}$;
Initialize $w \leftarrow 0$;
for *iter* $= 1, 2, \ldots, T$ **do**
  for $(x,y) \in \mathcal{D}$ **do**
    $\hat{y} = \arg\max_{y' \in \mathcal{Y}} w^T \psi(x,y')$; `// Equivalent to` $\arg\max_{y' \in \mathcal{Y}} w_{y'}^T x$
    if $\hat{y} \neq y$ **then** `// We've made a mistake`
      $w \leftarrow w + \psi(x,y)$; `// Move the scorer towards` $\psi(x,y)$
      $w \leftarrow w - \psi(x,\hat{y})$; `// Move the scorer away from` $\psi(x,\hat{y})$
    end
  end
end

Exercise: What is the base binary classification problem in multiclass perceptron?

- that looks the same as before but it is conceptually distinct
- we initialize $w \in \mathbb{R}^{k \times d}$ as all zeros
- then for some number of iterations for all data points
- define our prediction as $\hat{y} = argmax_{y' \in y} w^t \phi(x, y')$ so we are taking the class that is most close to the x projected onto the bassis vecotr of the class space
- then if we got it wrong we move our hyperplane in teh direction of $\phi(x,y)$ which is the projection of x onto that bassis in the w space
- and away from the class we got wrong

- what is the base binary classification problem in multiclass perceptron i mean we define $w \in \mathbb{R}^{k \times 2}$ make a feature map $\phi(x, i)$ which projects onto the classes in the same way
- i think it fits the frame work with out much change at all

features

- for now let our running example be part of speech classification
- $X = \{\text{all words}\}$, $Y = \{\text{noun, verb,adj...}\}$
- the features (that is what) $x_i \in x \in X$ represent could be the word, what the word ends with etc
- note that $w \in \mathbb{R}^{d*K}$ (ie a weight vector) as we did above does not scale here, since both d and i are really large
- we could directly design features for each that is

$$\phi(x, y) = (\phi_1(x, y) \cdots \phi_d(x, y))$$

- so for example suppose our input is $x =$ the boy grabbed the apple and ran away

$$\begin{aligned}
\psi_1(x, y) &= 1(x = \text{apple AND } y = \text{NOUN}) \\
\psi_2(x, y) &= 1(x = \text{run AND } y = \text{NOUN}) \\
\psi_3(x, y) &= 1(x = \text{run AND } y = \text{VERB}) \\
\psi_4(x, y) &= 1(x \text{ ENDS_IN_ly AND } y = \text{ADVERB})
\end{aligned}$$

- $\cdots$
- we can design features that we think are logical, and output some binary representation like $\phi(X =$run, y=Noun$)=$(0,1,0$\cdots$)
- so the feature maps effectively one hot encode if characteristics are in the input vector than we project that times our w (to get a compatibility score)
- so we ultimately want to max $w_i^t \phi_i(x, y))$ when a prediction is correct
- we do not need to include features that are not in our training data
- this is a flexible model, we can capture a lot of things we are intrested
- we can just take features from our training data
- this is spare so quick for computation

- can use a hash function to map our templates to discrete values
- so so far we have done this with perceptron, but we can expand this to use an svm which gives a unique prediction that maximizes the functional margin, also svm allows for non-liniearly through kernel methods

1 multiclass svm

margin for multiclass

- recall in binary data our margin is

$$m = y(f(x)) = y(w^t x)$$

  we want a large positive margin (representing high confidence predictions that are correct)

- class specific margin for data points x^n, y^n

$$h(x^n, y^n) - h(x^n, y)$$

  so that is the divergence between the score of the correct class and another class

- we want the margin to be large and positive $\forall y \neq y^n, \forall y \in [1, n]$

multiclass separable svm

- the binary constrained svm objective is

$$min_w \frac{1}{2}||w||^2$$

$$\text{st} \quad m = y^n w^t x^n \geq 1, \quad \forall (x^n, y^n) \in \mathcal{D}$$

- kernel multiclass margin

$$m_{n,y}(w) =< w, \phi(x^n, y^n) > - < w, \phi(x^n, y) >$$

  that is the score of the true class minus the score of some other class

- multi class constrained svm objective

$$min_w \frac{1}{2}||w||^2$$

$$st \quad m(n, y)(w) \geq 1 \quad \forall (x^n, y^n) \in \mathcal{D}, \forall y \neq y^n \in [1 \cdots k]$$

- as in binary class take 1 are our target margin

generalizing hinge loss

- hinge loss is the convex paperbound of 0-1 zero one loss (meaning it is the min convex function that is always above zero one loss) given by

$$\ell_{hinge}(y, \hat{y} = max(0, 1 - yh(x)))$$

- multiclass zero one loss

$$\delta(y, y') = \mathbb{I}(y \neq y')$$

- what is the upper bound of $\Delta(y, y')$
- call $\hat{y} = argmax_{y \in Y} < w, \phi(x, y') >$
- we know that $< w, \phi(x, y) > \quad \leq \quad < w, \phi(x, \hat{y}) > \Rightarrow \Delta(y, \hat{y}) \leq \Delta(y, \hat{y}) - < w, (\phi(x, y) - \phi(x, \hat{y})) >$
- thus we have general hinge loss

$$\ell_{hinge}(y, x, w) = max_{y' \in Y}(\Delta(y, y') - < w, (\phi(x, y) - \phi(x, y')) >)$$

- so just substituting general hinge loss into the svm objective yields the multiclass svm objective

$$j(w) = max_{w \in R^d} \frac{1}{2}||w||^2 + C \sum_n max_{y' \in Y}(\Delta(y, y') - < w, (\phi(x, y) - \phi(x, y')) >)$$

- we call $\Delta(y, y')$ the target margin for each class if $m_{n,y'}(w) \geq \Delta(y^n, y') \forall y \in Y$ there is no loss on example n

recap

- so we are trying to solve multiclass problem
- solution 1: one vs all
  1. train k models $h_1(x), \cdots h_k(x) : X \rightarrow \mathbb{R}$
  2. predict with $\mathrm{argmax}_{y \in Y} h_y(x)$
  3. but this can fail with linear models pretty easily
- solution 2 multiclass loss
  1. train one model $h(x, y) : X \times y \rightarrow \mathbb{R}$
  2. predict as $argmax_y h(x, y)$
- one vs all does well in practice for what it is worth
- this generalizes to situating where k is really large and where one vs all fails
- the key idea is that we can generalize across output oy by using features of y

intro to structured prediction

part of speech tagging

- the task is given a give a part of speech tag for all words

x	$\underbrace{\text{[START]}}_{x_0}$	$\underbrace{\text{He}}_{x_1}$	$\underbrace{\text{eats}}_{x_2}$	$\underbrace{\text{apples}}_{x_3}$
y	$\underbrace{\text{[START]}}_{y_0}$	$\underbrace{\text{Pronoun}}_{y_1}$	$\underbrace{\text{Verb}}_{y_2}$	$\underbrace{\text{Noun}}_{y_3}$

- In this problem our input space is words of any sequence length so it is massive
- our output space is also large as it is the length of the sequence times the number of part of sempach tags

multiclass hypothesis space

- suppose we have a discrete output space $y(x)$ that can be very large but has some structure, and the size depends on x
- the basse hypothesis space is $\mathcal{H} = \{h : X \times Y \to \mathbb{R}\}$ where $h \in \mathcal{H}$ $h(x, y)$ is a compatibility score between input x and output y
- our hypothesis space is

$$\mathcal{F} = \{x \to argmax_y h(x, y) | h \in \mathcal{H}\}$$

  which yields a final prediction function $f \in \mathcal{F}$ which has a underlying compatibility score function $h \in \mathcal{H}$

- suppose we are trying to tag

x:	he	eats	apples
y:	pronoun	verb	noun

- our hypothesis space is a linear combinations of feature map $h(x, y) = w^t \phi(x, y)$
- how can we define the feature map

unary feature

- a unary feature only depends on the label at a singe position y_i and x
- so for instance it could be

$$\phi(x, y_i) = \mathbb{I}((x_i = \text{runs}) \wedge y_i = \text{verb})$$

- this is kinda like a nb assumption

markov features

- markov features only depend on the two adjacent labels y_{i-1} , y_i and x
- for instance

$$\theta(x, y_{i-1}, y_i) = 1(y_{i-1}\text{pronouns})1(y_i = \text{verb})$$

local feature vector

- at each position i in a sequence we define the local feature vector

$$\psi_i = (\theta_1(x, y_i), \cdots \phi_1(x, y_{i-1}, y_i, x))$$

  so that is the feature that has all teh feature that are relevant to that inptu

- the local compatibility score at position i is $< w, \phi_i(x, y_{i-1}, y_i) >$
- the compatibility score of (x, y) si the sum of there local compatibility scores

$$\sum_i < w, \phi_i(x, y_{i-1}, y_i) >=< w, \psi(x, y) >$$

- we can use the perceptron with on this set up to do structured prediction as well

going to svm structured

- we think of the zero one loss between two sequence as the hamming loss

$$\Delta(y, y') = \frac{1}{L}\sum_{i=1}^{L} \mathbb{I}(y_i \neq y')$$

- then plugging this into our svm and using the structured feature transformation we get structured svm

argmax problem

- to compute predictions we need $argmax_{y \in y(x)} < w, \psi(x,y) >$ and $|y(x)|$ is exponentially large (that is our prediction depends on the past)
- but note that $\psi(x,y) = \sum_i \psi_i(x,y)$ so we can marginalize and solve this in code with a dynamic programming algorithm

conditional random field

- general logistic function is given as

$$P(y|x) = \frac{1}{z(x)} e^{w^t \psi(x,y)}$$

  where z is for normalization
- if we plug markov features into this we can get a linear chain crf
- has a nice probabilistic interpretation