CPSC 340:
Machine Learning and Data Mining

Linear Classifiers: multi-class

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Admin

- Assignment 4:
  - Due in a week

- Midterm:
  - The deadline has passed for grading clarifications
  - All issues should soon be fixed in your grades repos

Motivation: Part of Speech (POS) Tagging

- Consider problem of finding the verb in a sentence:
  - "The 340 students jumped at the chance to hear about POS features."

- Part of speech (POS) tagging is the problem of labeling all words.
  - 45 common syntactic POS tags.
  - Current systems have ~97% accuracy.
  - You can achieve this by applying "word-level" classifier to each word.

- What features of a word should we use for POS tagging?

But first…

- Recall we can convert categorical feature to set of binary features:

Age	City	Income
23	Van	22,000.00
23	Bur	21,000.00
22	Van	0.00
25	Sur	57,000.00
19	Bur	13,500.00
22	Van	20,000.00

Age	Van	Bur	Sur	Income
23	1	0	0	22,000.00
23	0	1	0	21,000.00
22	1	0	0	0.00
25	0	0	1	57,000.00
19	0	1	0	13,500.00
22	1	0	0	20,000.00

- This how we use a categorical feature ("city") in regression models.

POS Features

- Regularized multi-class logistic regression with 19 features gives ~97% accuracy:
  - Categorical features whose domain is all words (“lexical” features):
    - The word (e.g., “jumped” is usually a verb).
    - The previous word (e.g., “he” hit vs. “a” hit).
    - The previous previous word.
    - The next word.
    - The next next word.
  - Categorical features whose domain is combinations of letters (“stem” features):
    - Prefix of length 1 (“what letter does the word start with?”)
    - Prefix of length 2.
    - Prefix of length 3.
    - Prefix of length 4 (“does it start with JUMP?”)
    - Suffix of length 1.
    - Suffix of length 2.
    - Suffix of length 3 (“does it end in ING?”)
    - Suffix of length 4.
  - Binary features (“shape” features):
    - Does word contain a number?
    - Does word contain a capital?
    - Does word contain a hyphen?

Multi-Class Linear Classification

- We've been considering linear models for binary classification:

$$X = \begin{bmatrix} \quad & & \quad \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

- E.g., is there a cat in this image or not?

Multi-Class Linear Classification

- Today we'll discuss linear models for multi-class classification:

$$X = \begin{bmatrix} \quad & \quad & \quad \end{bmatrix} \qquad y = \begin{bmatrix} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{bmatrix}$$

- In POS classification we have 43 possible labels instead of 2.
  - This was natural for methods of Part 1 (decision trees, naïve Bayes, KNN).
  - For linear models, we need some new notation.

“One vs All” Classification

- One vs all method for turns binary classifier into multi-class.

- Training phase:
  - For each class ‘c’, train binary classifier to predict whether example is a ‘c’.
  - So if we have ‘k’ classes, this gives ‘k’ classifiers.

- Prediction phase:
  - Apply the ‘k’ binary classifiers to get a “score” for each class ‘c’.
  - Return the ‘c’ with the highest score.

“One vs All” Classification

- “One vs all” logistic regression for classifying as cat/dog/person.
  - Train a separate classifier for each class.
    - Classifier 1 tries to predict +1 for “cat” images and -1 for “dog” and “person” images.
    - Classifier 2 tries to predict +1 for “dog” images and -1 for “cat” and “person” images.
    - Classifier 3 tries to predict +1 for “person” images and -1 for “cat” and “dog” images.
  - This gives us a weight vector w_c for each class ‘c’:
    - Weights w_c try to predict +1 for class ‘c’ and -1 for all others.
    - We’ll use ‘W’ as a matrix with the w_c as rows:

$$W = \underbrace{\begin{bmatrix} \text{———} \; w_1^T \; \text{———} \\ \text{———} \; w_2^T \; \text{———} \\ \vdots \\ \text{———} \; w_K^T \; \text{———} \end{bmatrix}}_{d} \left.\vphantom{\begin{matrix} a \\ a \\ a \\ a \end{matrix}}\right\} k \longrightarrow$$

Each row ‘c’ gives weights w_c for a binary logistic regression model to predict class ‘c’.

“One vs All” Classification

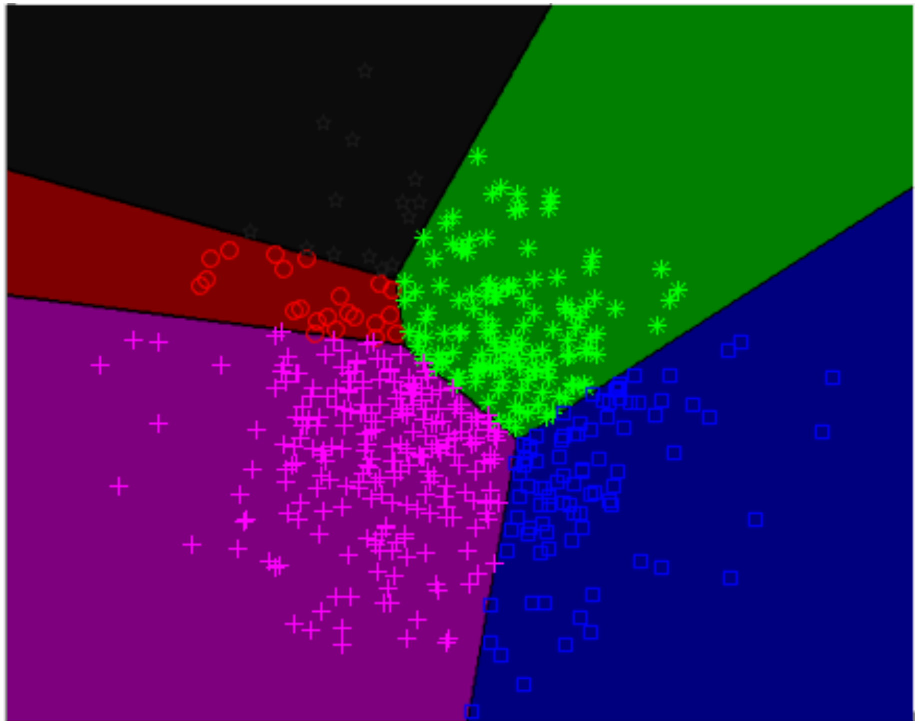
- “One vs all” logistic regression for classifying as cat/dog/person.
  - Prediction on example x_i given parameters ‘W’ :

$$W = \left.\underbrace{\begin{bmatrix} \text{—} \; w_1^T \; \text{—} \\ \text{—} \; w_2^T \; \text{—} \\ \vdots \\ \text{—} \; w_K^T \; \text{—} \end{bmatrix}}_{d}\right\} k$$

  - For each class ‘c’, compute $w_c^T x_i$.
    - Ideally, we’ll get $\text{sign}(w_c^T x_i) = +1$ for one class and $\text{sign}(w_c^T x_i) = -1$ for all others.
    - In practice, it might be +1 for multiple classes or no class.
  - To predict class, we take maximum value of $w_c^T x_i$ (“most confident”).

Shape of Decision Boundaries

- Multi-class linear classifier is intersection of these “half-spaces”:
  - This divides the space into convex regions (like k-means):



"Blue" region is region where we have:

$w_{blue}^T x_i \geq w_{green}^T x_i$

$w_{blue}^T x_i \geq w_{magenta}^T x_i$

$w_{blue}^T x_i \geq w_{red}^T x_i$

$w_{blue}^T x_i \geq w_{black}^T x_i$

  - Could be non-convex with kernels or change of basis.

Digression: Multi-Label Classification

- A related problem is multi-label classification:

$$X = \underbrace{\begin{bmatrix} \quad & & \quad \\ & & \\ & & \end{bmatrix}}_{d} \Big\} n \qquad Y = \underbrace{\begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix} \end{array}}_{k} \Big\} n \qquad W = \underbrace{\begin{bmatrix} \text{---} \; w_1^T \; \text{---} \\ \text{---} \; w_2^T \; \text{---} \\ \vdots \\ \text{---} \; w_k^T \; \text{---} \end{bmatrix}}_{d} \Big\} k$$

- Which of the ‘k’ objects are in this image?
  - There may be more than one “correct” class label.
  - Here we can also fit ‘k’ binary classifiers.
    - But we would take all sign($w_c^T x_i$)=+1 as the labels.

"One vs All" Multi-Class Classification

- Back to multi-class classification where we have 1 "correct" label:

$$X = \begin{bmatrix} \quad \end{bmatrix} \quad y = \begin{bmatrix} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{bmatrix} \Big\} \text{train 'k' classifiers} \rightarrow \underbrace{W = \begin{bmatrix} \text{---} \; w_1^T \; \text{---} \\ \text{---} \; w_2^T \; \text{---} \\ \vdots \\ \text{---} \; w_k^T \; \text{---} \end{bmatrix}}_{\text{Predict by maximizing } w_c^T x_i}$$

- We'll use 'w_{y_i}' as classifier $c=y_i$ (row w_c of correct class label).
- Problem: We didn't train the w_c so that the largest $w_c^T x_i$ would be $w_{y_i}^T x_i$.
  - Each classifier is just trying to get the sign right.

Multi-Class Linear Classifiers

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?

- Yes!
  - We'll go into detail for logistic regression.
  - See bonus slides for SVM.

Multi-Class Logistic Regression: Predictions

- How do we make predictions? Let's try to get probabilities again.
  - Compute $w_c^T x_i$ for each class 'c'
  - Make them positive: taking $\exp(w_c^T x_i)$ solves this
  - Make them add up to 1: dividing by the sum solves this

$$P(y_i = c) = \frac{\exp(w_c^T x_i)}{\sum_{c=1}^{k} \exp(w_c^T x_i)}$$

- This is the softmax function.

Multi-Class Logistic Regression: Loss function

- We want the raw model output of the true class to be largest:

$$w_{y_i}^T x_i \geq \max_c \{w_c^T x_i\}$$

$$\text{or} \quad 0 \geq -w_{y_i}^T x_i + \max_c \{w_c^T x_i\}$$

- Let's smooth the max with the log-sum-exp:

$$-w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right)$$

- We want this to be as small as possible, so let's minimize it.
- This is the softmax loss (which goes by several names)

Multi-Class Logistic Regression: Loss function

- We sum the loss over examples and add regularization:

$$f(W) = \sum_{i=1}^{n}\left[-w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k}\exp(w_c^T x_i)\right)\right] + \frac{\lambda}{2}\sum_{j=1}^{d}\sum_{c=1}^{k} w_{jc}^2$$

Tries to make $w_c^T x_i$ big for the correct label

Approximates $\max_c \{w_c^T x_i\}$ so tries to make $w_c^T x_i$ small for all labels.

Usual L_2-regularizer on elements of 'W'

- This objective is convex (should be clear for 1st and 3rd terms).
  - It's differentiable so you can use gradient descent.
- When k=2, equivalent to binary logistic.
  - Not obvious since it has twice as many parameters.

Digression: Frobenius Norm

- The Frobenius norm of a matrix 'W' is defined by:

$$\|W\|_F = \sqrt{\sum_{j=1}^{d} \sum_{c=1}^{k} w_{jc}^2}$$

(L2-norm if you "stack" columns into one big vector)

- We can write regularizer in matrix notation using:

$$\frac{\lambda}{2} \sum_{j=1}^{d} \sum_{c=1}^{k} w_{jc}^2 = \frac{\lambda}{2} \|W\|_F^2$$

Summary

- Word features: lexical, stem, shape.
- One vs all turns a binary classifier into a multi-class classifier.
- Multi-class SVMs exist but we didn't cover them.
- Softmax loss is a multi-class version of the logistic loss.

Multi-Class SVMs

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?

- Recall our derivation of the hinge loss (SVMs):
  - We wanted $y_i w^T x_i > 0$ for all 'i'.
  - We avoided non-degeneracy by aiming for $y_i w^T x_i \geq 1$.
  - We used the constraint violation as our loss: $\max\{0, 1 - y_i w^T x_i\}$.

- We can derive multi-class SVMs using the same steps…

Multi-Class SVMs

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?

We want $w_{y_i}^T x_i > w_c^T x_i$ for all 'c' that are not correct label y_i

If we penalize violation of this constraint it's degenerate.

We use $w_{y_i}^T x_i \geq w_c^T x_i + 1$ for all $c \neq y_i$ to avoid strict inequality

Equivalently: $0 \geq 1 - w_{y_i}^T x_i + w_c^T x_i$

- For here, there are two ways to measure constraint violation:

"Sum"

$$\sum_{c \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}$$

"Max"

$$\max_{c \neq y_i} \{\max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}\}$$

Multi-Class SVMs

- Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?

"Sum"

$$\sum_{c \neq y_i} \max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}$$

"Max"

$$\max_{c \neq y_i} \{\max\{0, 1 - w_{y_i}^T x_i + w_c^T x_i\}\}$$

- For each training example 'i':
  - "Sum" rule penalizes for each 'c' that violates the constraint.
  - "Max" rule penalizes for one 'c' that violates the constraint the most.
    - "Sum" gives a penalty of 'k' for W=0, "max" gives a penalty of '1'.
- If we add L2-regularization, both are called multi-class SVMs:
  - "Max" rule is more popular, "sum" rule usually works better.
  - Both are convex upper bounds on the 0-1 loss.

Softmax Loss Function

- What we want is $\arg\max_c \{w_c^T x_i\} = y_i$
  - y_i is the true class of example 'i'
- We can rewrite this as $\max\{w_1^T x_i, \ldots, w_k^T x_i\} = w_{y_i}^T x_i$
  - If these are equal then you've classified example i correctly
- So we minimize the difference between these two things:

$$f_i(W) = \max\{w_1^T x_i, \ldots, w_k^T x_i\} - w_{y_i}^T x_i$$

  - $f_i(W) = 0$ if example i is classified correctly
  - $f_i(W) > 0$ if example i is classified incorrectly
  - So minimizing f indeed pushes us toward correct classification!
- We invoke the log-sum-exp approximation of max examples

Softmax Loss Function

$$f_i(W) = \max\{w_1^T x_i, \ldots, w_k^T x_i\} - w_{y_i}^T x_i$$

- Because max is non-smooth with invoke the log-sum-exp approximation of the max function (hence smooth or "soft" max)

$$\max\{z_1, \ldots, z_n\} \approx \log\left(\sum_{i=1}^{n} \exp(z_i)\right)$$

- Applying this we get: $f_i(W) = \log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right) - w_{y_i}^T x_i$

- Finally, we sum over all examples to get the softmax loss

$$f(W) = \sum_{i=1}^{n} \log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right) - w_{y_i}^T x_i$$

Motivation: Dog Image Classification

- Suppose we're classifying images of dogs into breeds:

- What if we have images where class label isn't obvious?
  - Syberian husky vs. Inuit dog?

https://www.slideshare.net/angjoo/dog-breed-classification-using-part-localization
https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Preferences

- Do we need to throw out images where label is ambiguous?
  - We don't have the y_i.

  - We want classifier to prefer Syberian husky over bulldog, Chihuahua, etc.
    - Even though we don't know if these are Syberian huskies or Inuit dogs.
  - Can we design a loss that enforces preferences rather than "true" labels?

https://ischlag.github.io/2016/04/05/important-ILSVRC-achievements

Learning with Pairwise Preferences (Ranking)

- Instead of y_i, we're given list of (c_1,c_2) preferences for each 'i':

We want $w_{c_1}^T x_i > w_{c_2}^T x_i$ for these particular (c_1, c_2) values

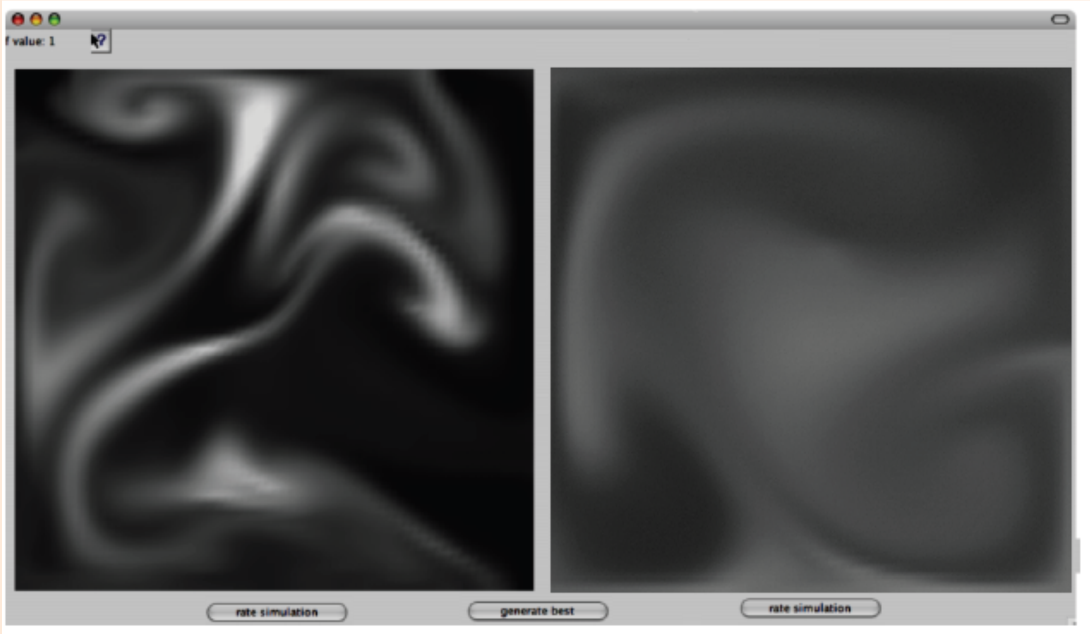
- Multi-class classification is special case of choosing (y_i,c) for all 'c'.

- By following the earlier steps, we can get objectives for this setting:

$$\sum_{i=1}^{n} \underbrace{\sum_{(c_1,c_2)} \max\{0, 1 - w_{c_1}^T x_i + w_{c_2}^T x_i\}}_{\text{"sum" version of multi-class SVM}} + \frac{\lambda}{2}\|W\|_F^2$$

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for computer graphics:
  - We have a smoke simulator, with several parameters:

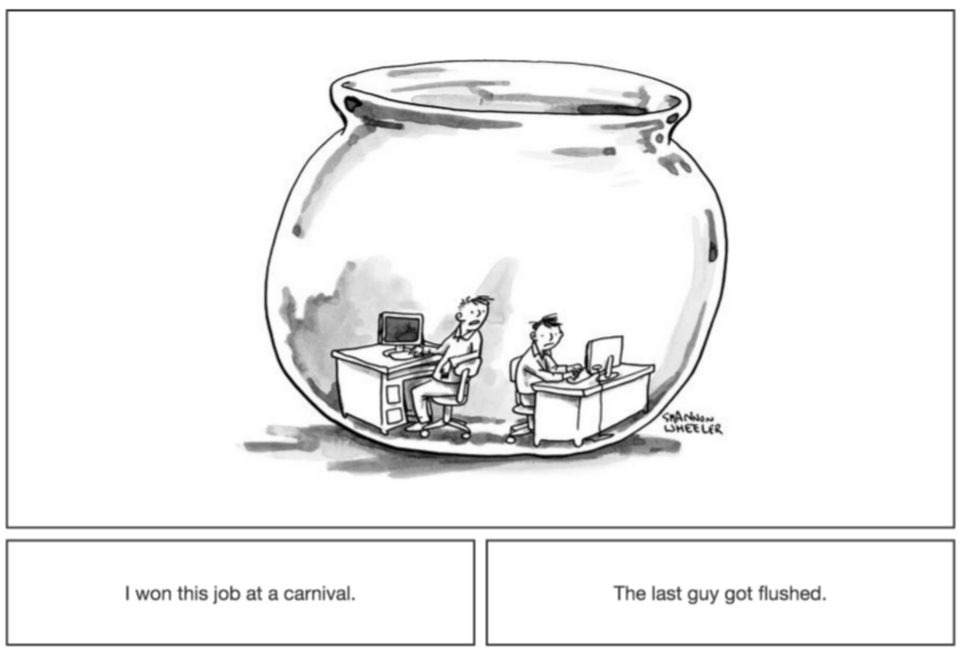


  - Don't know what the optimal parameters are, but we can ask the artist:
    - "Which one looks more like smoke"?

https://circle.ubc.ca/bitstream/handle/2429/30519/ubc_2011_spring_brochu_eric.pdf?sequence=3

Learning with Pairwise Preferences (Ranking)

- Pairwise preferences for humour:
  - New Yorker caption contest:



  - “Which one is funnier”?

https://homes.cs.washington.edu/~jamieson/resources/next.pdf

Feature Engineering

- “…some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.”
  – Pedro Domingos

- “Coming up with features is difficult, time-consuming, requires expert knowledge. "Applied machine learning" is basically feature engineering.”
  – Andrew Ng

Feature Engineering

- Better features usually help more than a better model.

- Good features would ideally:
  - Capture most important aspects of problem.
  - Generalize to new scenarios.
  - Allow learning with few examples, be hard to overfit with many examples.

- There is a trade-off between simple and expressive features:
  - With simple features overfitting risk is low, but accuracy might be low.
  - With complicated features accuracy can be high, but so is overfitting risk.

Feature Engineering

- The best features may be dependent on the model you use.

- For counting-based methods like naïve Bayes and decision trees:
  - Need to address coupon collecting, but separate relevant “groups”.

- For distance-based methods like KNN:
  - Want different class labels to be “far”.

- For regression-based methods like linear regression:
  - Want labels to have a linear dependency on features.

Discretization for Counting-Based Methods

- For counting-based methods:
  - Discretization: turn continuous into discrete.

Age
23
23
22
25
19
22

< 20	>= 20, < 25	>= 25
0	1	0
0	1	0
0	1	0
0	0	1
1	0	0
0	1	0

  - Counting age "groups" could let us learn more quickly than exact ages.
    - But we wouldn't do this for a distance-based method.

Standardization for Distance-Based Methods

- Consider features with different scales:

Egg (#)	Milk (mL)	Fish (g)	Pasta (cups)
0	250	0	1
1	250	200	1
0	0	0	0.5
2	250	150	0

- Should we convert to some standard ‘unit’?
  – It doesn’t matter for counting-based methods.
- It matters for distance-based methods:
    - KNN will focus on large values more than small values.
    - Often we “standardize” scales of different variables (e.g., convert everything to grams).

Non-Linear Transformations for Regression-Based

- Non-linear feature/label transforms can make things more linear:
  - Polynomial, exponential/logarithm, sines/cosines, RBFs.

www.google.com/finance

Discussion of Feature Engineering

- The best feature transformations are application-dependent.
  - It’s hard to give general advice.
- My advice: ask the domain experts.
  - Often have idea of right discretization/standardization/transformation.
- If no domain expert, cross-validation will help.
  - Or if you have lots of data, use deep learning methods from Part 5.

“All-Pairs” and ECOC Classification

- Alternative to “one vs. all” to convert binary classifier to multi-class is “all pairs”.
  - For each pair of labels ‘c’ and ‘d’, fit a classifier that predicts +1 for examples of class ‘c’ and -1 for examples of class ‘d’ (so each classifier only trains on examples from two classes).
  - To make prediction, take a vote of how many of the (k-1) classifiers for class ‘c’ predict +1.
  - Often works better than “one vs. all”, but not so fun for large ‘k’.
- A variation on this is using “error correcting output codes” from information theory (see Math 342).
  - Each classifier trains to predict +1 for some of the classes and -1 for others.
  - You setup the +1/-1 code so that it has an “error correcting” property.
    - It will make the right decision even if some of the classifiers are wrong.