CPSC 340:
Machine Learning and Data Mining

Multi-Class Classification
Fall 2017
Admin

- **Assignment 3:**
  - Check “update” thread on Piazza for correct definition of trainNdx.
    - This could make your cross-validation code behave weird.
  - Due tonight, 1 late day to hand in Monday, 2 late days for Wednesday.

- **Midterm:**
  - Can view your exam after class today.

- **Assignment 4:**
  - Due in 2 weeks.
Last Time: Stochastic Gradient

• **Stochastic gradient** minimizes average of smooth functions:

\[ f(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w) \]

– Function \( f_i(w) \) is error for example ‘i’.

• Iterations perform gradient descent on one random example ‘i’:

\[ w_{t+1} = w_t - \alpha_t \nabla f_i(w_t) \]

– Cheap iterations even when ‘n’ is large, but doesn’t always decrease ‘f’.

– But solves problem if \( \alpha_t \) goes to 0 at an appropriate rate.

• Theory says use \( \alpha_t = O(1/t) \), in practice you need to experiment.
Last Time: Stochastic Gradient

• Stochastic gradient converges very slowly:
  – But if your dataset is too big, there may not be much you can do.

• Practical tricks to improve performance:
  – Constant or slowly-decreasing step-sizes and/or average the $w^t$.
  – Binary search for step, stop using validation error (bonus slides).

• You can also improve performance by reducing the variance:
  – Using “mini-batches” or random samples rather than 1 random sample.
  – New “variance-reduced” methods (SAG, SVRG) for finite training sets.
Stochastic Gradient with Infinite Data

• Amazing property of stochastic gradient:
  – The classic convergence analysis does not rely on ‘n’ being finite.

• Consider an infinite sequence of IID samples.
  – Or any dataset that is so large we cannot even go through it once.

• Approach 1 (gradient descent):
  – Stop collecting data once you have a very large ‘n’.
  – Fit a regularized model on this fixed dataset.

• Approach 2 (stochastic gradient):
  – Perform a stochastic gradient iteration on each example as we see it.
  – Never re-visit any example, always take a new one.
Stochastic Gradient with Infinite Data

- Approach 2 only looks at a data point once:
  - Each example is an unbiased approximation of test data.

- So Approach 2 is doing stochastic gradient on test error:
  - It cannot overfit.

- Up to a constant, Approach 2 achieves test error of Approach 1.
  - This is sometimes used to justify SG as the “ultimate” learning algorithm.
    - “Optimal test error by computing gradient of each example once!”
  - In practice, Approach 1 usually gives lower test error.
    - The constant factor matters!
(pause)
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...

- Last time we discussed the **effect of binary features in regression**.
- Recall we can convert **categorical feature to set of binary features**:

<table>
<thead>
<tr>
<th>Age</th>
<th>City</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Van</td>
<td>22,000.00</td>
</tr>
<tr>
<td>23</td>
<td>Bur</td>
<td>21,000.00</td>
</tr>
<tr>
<td>22</td>
<td>Van</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>Sur</td>
<td>57,000.00</td>
</tr>
<tr>
<td>19</td>
<td>Bur</td>
<td>13,500.00</td>
</tr>
<tr>
<td>22</td>
<td>Van</td>
<td>20,000.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age</th>
<th>Van</th>
<th>Bur</th>
<th>Sur</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>22,000.00</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>21,000.00</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>57,000.00</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>13,500.00</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>20,000.00</td>
</tr>
</tbody>
</table>

- 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**:

\[
X = \begin{bmatrix}
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix} \quad y = \begin{bmatrix}
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix}
\]

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

https://www.youtube.com/watch?v=tntOCGkt98
Multi-Class Linear Classification

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

\[
\mathbf{X} = \begin{pmatrix}
\end{pmatrix}
\quad \mathbf{y} = \begin{pmatrix}
27 \\
16 \\
8 \\
7 \\
21 \\
5
\end{pmatrix}
\]

• 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 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 columns:

$$W = \begin{bmatrix}
  w_1 \\
  w_2 \\
  \vdots \\
  w_K
\end{bmatrix}$$

Each column ‘c’ is a binary classifier for class ‘c’
“One vs All” Classification

• “One vs all” logistic regression for classifying as cat/dog/person.
  – Prediction on example $x_i$ given parameters ‘$W$’:
    \[ W = \begin{bmatrix}
    w_1 \\
    w_2 \\
    \vdots \\
    w_k
    \end{bmatrix} \]
    Each column ‘$c$’ is a binary classifier for class ‘$c$’
  – 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 positive").
Digression: Multi-Label Classification

A related problem is multi-label classification:

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 \( \text{sign}(w_c^T x_i) = +1 \) as the labels.

\[ X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad W = \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} \]
“One vs All” Multi-Class Classification

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

\[ X = \begin{bmatrix} \vdots \end{bmatrix}, \quad Y = \begin{bmatrix} 27 \\ 16 \\ 8 \\ 7 \\ 21 \\ 5 \end{bmatrix} \]

\[ W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix} \]

- We’ll use ‘\( w_{y_i} \)’ as column \( y_i \) of ‘\( W \)’ (column 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 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 strict inequality 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 $y_i$

  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"

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

• 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.

• 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.
Multi-Class Logistic Regression

• Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?
  
  Ignoring the "1", this is equivalent to
  
  $$w_{y_i}^T x_i \geq \max_c \frac{x_i^T w_c}{x_i^T x_i}$$

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

• We want the right side to be as small as possible.

• Let’s smooth the max with the log-sum-exp:
  
  $$-w_{y_i}^T x_i + \log \left( \sum_{c=1}^{k} \exp \left( w_c^T x_i \right) \right)$$

• This is the softmax loss, used in multi-class logistic regression.
Multi-Class Logistic Regression

• We sum the loss over examples and add regularization:

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

- 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 L2-regularizer on elements of 'W'

• This objective is convex (should be clear for 1st and 3rd terms).
• Next time we’ll see that when k=2, equivalent to binary logistic.
  – Not obvious at the moment.
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:

$$\frac{\lambda}{2} \sum_{j=1}^{d} \sum_{c=1}^{k} w_{jc}^2 = \frac{\lambda}{2} \| W \|_F^2$$
(pause)
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’:
  
  \[ \text{We want } w_{c_1}^T x_i > w_{c_2}^T x_i \text{ for these particular } (c_1, c_2) \text{ values} \]

• 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:

\[
\sum_{i=1}^{n} \sum_{(c_1, c_2)} \max_{t \in 0, 1} \left\{ 1 - w_{c_1}^T x_i + w_{c_2}^T x_i \right\}^{+} + \frac{\lambda}{2} \|W\|_F^2
\]

"Sum" version of multi-class SVM

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

Summary

- **Infinite datasets** can be used with SG and do not overfit.
- **Word features**: lexical, stem, shape.
- **One vs all** turns a binary classifier into a multi-class classifier.
- **Multi-class SVMs** measure violation of classification constraints.
- **Softmax loss** is a multi-class version of logistic loss.
- Ranking

- Next time:
  - What do regression and regularization have to do with probabilities?
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.
  - 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:

<table>
<thead>
<tr>
<th>Egg (#)</th>
<th>Milk (mL)</th>
<th>Fish (g)</th>
<th>Pasta (cups)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>250</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>250</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
<td>150</td>
<td>0</td>
</tr>
</tbody>
</table>

• 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.
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.