CPSC 340:
Machine Learning and Data Mining

Multi-Class Classification
Fall 2018
Assignment 4:
  - Due Friday of next week.

Midterm:
  - Grades posted.
  - Can view exam during Mike or my office hours this week and next week.
Last Time: SVMs, Logistic Regression, One vs. All

• We discussed **hinge loss** and **logistic loss** for binary classification.
  – Convex approximation to number of classification errors in linear models.
  – Leads to **SVMs** (hinge + L2-regularization) and **logistic regression** (logistic).

• We discussed **multi-class classification**: \( y_i \) in \( \{1, 2, \ldots, k\} \).

• **One vs. all** with +1/-1 binary classifier:
  – Train weights \( w_c \) to predict +1 for class ‘c’, -1 otherwise.
    \[
    W = \begin{bmatrix}
    w_{1}^{T} \\
    \vdots \\
    w_{k}^{T}
    \end{bmatrix}_{k 	imes d}
    \]
    – Predict by taking ‘c’ maximizing \( w_c^{T}x_i \).
    • Problem: each \( w_c \) is only “trying to get sign right” during training.
      – Didn’t train the \( w_c \) so that the largest \( w_c^{T}x_i \) would be \( w_k^{T}x_i \).
Shape of Decision Boundaries

• Recall that a binary linear classifier splits space using a hyper-plane:
  - Divides $x_i$ space into 2 “half-spaces”.

- Line with $w^T x_i = 0$
  - Classify as '0' because $w^T x_i > 0$
  - Classify as '1' because $w^T x_i < 0$
Shape of Decision Boundaries

• **Multi-class linear classifier** is intersection of these “half-spaces”:
  – This divides the space into **convex regions** (like k-means):
    – Could be non-convex with change of basis.
Multi-Class SVMs

• Can we define a loss that encourages largest $w_c^T x_i$ to be $w_{y_i}^T x_i$?
  – So when we maximizing over $w_c^T x_i$, we choose correct label $y_i$.

• Recall our derivation of the hinge loss (SVMs):
  – We wanted $y_i w^T x_i > 0$ for all ‘i’ to classify correctly.
  – 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"

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

  "Max"

  $\max_{C \neq y_i} \left\{ \max \left\{ 0, 1 - w_{y_i}^T x_i + w_c^T x_i \right\} \right\}$
Multi-Class SVMs

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

  "Sum"
  \[
  \sum_{c \neq y_i} \max \{ 0, 1 - w_{y_i}^T x_i + w_c^T x_i \}^2
  \]

  "Max"
  \[
  \max \sum_{c \neq y_i} \max \{ 0, 1 - w_{y_i}^T x_i + w_c^T x_i \}^2
  \]

• 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-1’ 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.
Multi-Class Logistic Regression

• We derived **binary logistic loss** by *smoothing* a degenerate ‘max’.
  – A degenerate constraint in the multi-class case can be written as:
    \[ w_x y_i x_i \geq \max_c \{ w_c x_i \} \]
    or
    \[ 0 \geq -w_x y_i x_i + \max_c \{ w_c x_i \} \]
• We want the right side to be as small as possible.
• Let’s *smooth* the max with the log-sum-exp:
  \[ -w_x y_i x_i + \log (\sum_c \exp (w_c x_i)) \]
  – With \( W=0 \) this gives a loss of \( \log(k) \).
• This is the **softmax loss**, the loss for multi-class logistic regression.
Multi-Class Logistic Regression

- We sum the loss over examples and add regularization:

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

- This objective is convex (should be clear for 1\textsuperscript{st} and 3\textsuperscript{rd} terms).
  - It’s differentiable so you can use gradient descent.
- When \( k=2 \), equivalent to binary logistic.
  - Not obvious at the moment.
Digression: Frobenius Norm

• We can write regularizer in matrix notation using:

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

• 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)
(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’:

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

• 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:

$$\sum_{i=1}^{n} \sum_{(c_1,c_2)} \max_{\xi} \{ 0, 1 - w_{c_1}^\top x_i + w_{c_2}^\top x_i \} + \frac{\lambda}{2} \| W \|_F^2$$

"sum" version of multi-class SVM
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”?
Learning with Pairwise Preferences (Ranking)

• Pairwise preferences for humour:
  – New Yorker caption contest:
    – “Which one is funnier”? 
    ![Cartoon image](image-url)
Support Vector Machines for Non-Separable

• What about data that is not even close to separable?

Support Vector Machines for Non-Separable

- What about data that is **not even close to separable**?
  - It may be **separable under change of basis** (or closer to separable).

Support Vector Machines for Non-Separable

- What about data that is not even close to separable?
  - It may be separable under change of basis (or closer to separable).

Support Vector Machines for Non-Separable

• What about data that is **not even close to separable**?
  – It may be **separable under change of basis** (or closer to separable).

$$y_i = w_1 x_{i1}^2 + w_2 \sqrt{2} x_{i1} x_{i2} + w_3 x_{i2}^2$$

Multi-Dimensional Polynomial Basis

• Recall fitting polynomials when we only have 1 feature:

\[
\hat{y}_i = w_0 + w_1 x_i + w_2 x_i^2
\]

• We can fit these models using a change of basis:

\[
\begin{bmatrix}
0.2 \\
-0.5 \\
1 \\
4
\end{bmatrix}
\begin{bmatrix}
1 & 0.2 & (0.2)^2 \\
1 & -0.5 & (-0.5)^2 \\
1 & 1 & (1)^2 \\
1 & 4 & (4)^2
\end{bmatrix}
\]

• How can we do this when we have a lot of features?
Multi-Dimensional Polynomial Basis

• Polynomial basis for $d=2$ and $p=2$:

$$X = \begin{bmatrix} 0.2 & 0.3 \\ 1 & 0.5 \\ -0.5 & -0.1 \end{bmatrix}$$

$$Z = \begin{bmatrix} 1 & 0.2 & 0.3 & (0.2)^2 & (0.3)^2 & (0.1)(0.3) \\ 1 & 1 & 0.5 & (1)^2 & (0.5)^2 & (1)(0.5) \\ 1 & 0.5 & -0.1 & (0.5)^2 & (-0.1)^2 & (-0.5)(-0.1) \end{bmatrix}$$

• With $d=4$ and $p=3$, the polynomial basis would include:

  – Bias variable and the $x_{ij}$: $1$, $x_{i1}$, $x_{i2}$, $x_{i3}$, $x_{i4}$.
  – The $x_{ij}$ squared and cubed: $(x_{i1})^2$, $(x_{i2})^2$, $(x_{i3})^2$, $(x_{i4})^2$, $(x_{i1})^3$, $(x_{i2})^3$, $(x_{i3})^3$, $(x_{i4})^3$.
  – Two-term interactions: $x_{i1}x_{i2}$, $x_{i1}x_{i3}$, $x_{i1}x_{i4}$, $x_{i2}x_{i3}$, $x_{i2}x_{i4}$, $x_{i3}x_{i4}$.
  – Cubic interactions: $x_{i1}^2x_{i2}$, $x_{i1}^2x_{i3}$, $x_{i1}^2x_{i4}$, $x_{i2}^2x_{i3}$, $x_{i2}^2x_{i4}$, $x_{i1}^2x_{i3}^2$, $x_{i2}^2x_{i4}^2$, $x_{i1}x_{i2}^2x_{i3}$, $x_{i1}x_{i2}x_{i3}^2$, $x_{i2}x_{i3}x_{i4}^2$, $x_{i1}^2x_{i3}^2x_{i4}$, $x_{i1}x_{i2}x_{i3}x_{i4}$, $x_{i2}x_{i3}x_{i4}$, $x_{i3}x_{i4}$.
Kernel Trick

• If we go to degree p=5, we’ll have $O(d^5)$ quintic terms:

$$X_{i_1}^5 X_{i_2}^4 X_{i_3}^4 \cdot \cdot \cdot X_{i_l}^4 X_{id}^3 X_{id}^3 \cdot \cdot \cdot X_{id}^3 X_{id}^2 \cdot \cdot \cdot X_{id}^2 X_{id} X_{id} \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot \cdot X_{id}^5$$

• For large ‘d’ and ‘p’, storing a polynomial basis is intractable!
  – ‘Z’ has $k=O(d^p)$ columns, so it does not fit in memory.

• Today: efficient polynomial basis for L2-regularized least squares.
  – Main tools: the “other” normal equations and the “kernel trick”.
The “Other” Normal Equations

• Recall the **L2-regularized least squares** objective with basis ‘Z’:

\[
\ell(v) = \frac{1}{2} \| Zv - y \|^2 + \frac{\lambda}{2} \| v \|^2
\]

• We showed that the minimum is given by

\[
v = (Z^T Z + \lambda I)^{-1} Z^T y
\]

(\( k \times k \))

(in practice you still solve the linear system, since inverse can be numerically unstable – see CPSC 302)

• With some work (bonus), this can equivalently be written as:

\[
v = Z^T (ZZ^T + \lambda I)^{-1} y
\]

(\( n \times n \))

• This is faster if \( n << k \):
  – Cost is \( O(n^2k + n^3) \) instead of \( O(nk^2 + k^3) \).
  – But for the polynomial basis, this is still too slow since \( k = O(d^p) \).
The “Other” Normal Equations

• With the “other” normal equations we have \( \nu = Z^T(ZZ^T + \lambda I)^{-1}y \)

• Given test data \( \tilde{X} \), predict \( \hat{y} \) by forming \( \tilde{Z} \) and then using:

\[
\hat{y} = \tilde{Z} \nu \\
= \tilde{Z} Z^T(ZZ^T + \lambda I)^{-1}y \\
= \tilde{K}(\tilde{K} + \lambda I)^{-1}y \\
\]

\( t \times 1 \quad t \times n \quad n \times n \quad n \times 1 \)

• Notice that if you have \( K \) and \( \tilde{K} \) then you do not need \( Z \) and \( \tilde{Z} \).

• Key idea behind “kernel trick” for certain bases (like polynomials):
  – We can efficiently compute \( K \) and \( \tilde{K} \) even though forming \( Z \) and \( \tilde{Z} \) is intractable.
The matrix $K = ZZ^T$ is called the Gram matrix $K$.

$K = ZZ^T = \begin{bmatrix} Z_1^T & Z_2^T & \cdots & Z_n^T \end{bmatrix} \begin{bmatrix} \vdots \\ Z_1 \\ \vdots \\ Z_n \end{bmatrix}$

$= \begin{bmatrix} Z_1^T Z_1 & Z_1^T Z_2 & \cdots & Z_1^T Z_n \\ Z_2^T Z_1 & Z_2^T Z_2 & \cdots & Z_2^T Z_n \\ \vdots & \vdots & \ddots & \vdots \\ Z_n^T Z_1 & Z_n^T Z_2 & \cdots & Z_n^T Z_n \end{bmatrix}$

$K$ contains the dot products between all training examples.

- Similar to ‘$Z$’ in RBFs, but using dot product as “similarity” instead of distance.
Gram Matrix

- The matrix $\tilde{K} = \tilde{Z}Z^T$ has dot products between train and test examples:

\[
\tilde{K} = \tilde{Z}Z^T = \begin{bmatrix}
\tilde{z}_1^T \\
\vdots \\
\tilde{z}_L^T \\
\end{bmatrix}
\begin{bmatrix}
z_1 \\
z_2 \\
\vdots \\
z_n \\
\end{bmatrix}
\]

- Kernel function: $k(x_i, x_j) = z_i^Tz_j$.
  - Computes dot product between in basis $(z_i^Tz_j)$ using original features $x_i$ and $x_j$. 

Kernel Trick

To apply linear regression, I only need to know $K$ and $\tilde{K}$

Use $x_i$ to form $z_i$

Use $x_j$ to form $z_j$

Compute $z_i^T z_j$

$$K = \begin{bmatrix} \end{bmatrix}$$

Set $k_{ij} = z_i^T z_j$

Final result is $n \times n$ (no matter how large $z_i$ is)
 Kernel Trick

To apply linear regression, I only need to know $K$ and $\tilde{K}$.

Use $x_i$ to form $z_i$. Use $x_j$ to form $z_j$. Compute $z_i^T z_j$. Set $K_{ij} = z_i^T z_j$.

Directly compute $K_{ij}$ from $x_i$ and $x_j$.

Final result is $n \times n$ (no matter how large $z_i$ is).
Linear Regression vs. Kernel Regression

**Linear Regression**

**Training**
1. Form basis $\tilde{Z}$ from $X$
2. Compute $\mathbf{v} = \left( Z^T \tilde{Z} + \lambda I \right)^{-1} Z^T \mathbf{y}$

**Testing**
1. Form basis $\tilde{Z}$ from $\tilde{X}$
2. Compute $\hat{\mathbf{y}} = \tilde{Z} \mathbf{v}$

**Kernel Regression**

**Training**
1. Form inner products $K$ from $X$
2. Compute $\mathbf{u} = \left( K + \lambda I \right)^{-1} \mathbf{y}$

**Testing**
1. Form inner products $\tilde{K}$ from $X$ and $\tilde{X}$
2. Compute $\hat{\mathbf{y}} = \tilde{K} \mathbf{u}$

(Everything you need to know about $Z$ and $\tilde{Z}$ is contained within $K$ and $\tilde{K}$)

Non-parametric
Summary

• **Multi-class SVMs** measure violation of classification constraints.
• **Softmax loss** is a multi-class version of logistic loss.
• **High-dimensional bases** allows us to separate non-separable data.
• “**Other” normal equations** are faster when n < d.

• Next time: how do we train on all of Gmail?
Bonus Slide: Equivalent Form of Ridge Regression

Note that \( \hat{X} \) and \( Y \) are the same on the left and right side, so we only need to show that

\[
(X^T X + \lambda I)^{-1}X^T = X^T(XX^T + \lambda I)^{-1}.
\] (1)

A version of the matrix inversion lemma (Equation 4.107 in MLAPP) is

\[
(E - FH^{-1}G)^{-1}FH^{-1} = E^{-1}F(H - GE^{-1}F)^{-1}.
\]

Since matrix addition is commutative and multiplying by the identity matrix does nothing, we can re-write the left side of (1) as

\[
(X^T X + \lambda I)^{-1}X^T = (\lambda I + X^T X)^{-1}X^T = (\lambda I + X^T I X)^{-1}X^T = (\lambda I - X^T (-I) X)^{-1}X^T = -(\lambda I - X^T (-I) X)^{-1}X^T (-I)
\]

Now apply the matrix inversion with \( E = \lambda I \) (so \( E^{-1} = \frac{1}{\lambda} I \)), \( F = X^T \), \( H = -I \) (so \( H^{-1} = -I \) too), and \( G = X \):

\[
-(\lambda I - X^T (-I) X)^{-1}X^T (-I) = -(\frac{1}{\lambda})IX^T(-I - X \left( \frac{1}{\lambda} \right) X^T)^{-1}.
\]

Now use that \((1/\alpha)A^{-1} = (\alpha A)^{-1}\), to push the \((-1/\lambda)\) inside the sum as \(-\lambda\),

\[
-(\frac{1}{\lambda})IX^T(-I - X \left( \frac{1}{\lambda} \right) X^T)^{-1} = X^T(\lambda I + XX^T)^{-1} = X^T(XX^T + \lambda I)^{-1}.
\]