Admin

• Assignment 1:
  – Marks visible on UBC Connect.

• Assignment 2:
  – Solution posted after class.

• Assignment 3:
  – Due Wednesday (at any time on Earth).
  – Solutions will be released next Wednesday after class.

• Tutorial room change: T1D (Monday @5pm) moved to DMP 101.

• Midterm on Friday October 28.
  – Practice midterm and list of topics posted (covers Assignments 1-3)
  – In class, 55 minutes, closed-book, cheat sheet: 2-pages each double-sided.
Summary of Last Lecture

1. **Error functions:**
   - Squared error is sensitive to outliers.
   - Absolute ($L_1$) error and Huber error are more robust to outliers.
   - Brittle ($L_\infty$) error is more sensitive to outliers.

2. $L_1$ and $L_\infty$ error functions are **non-differentiable:**
   - Finding ‘w’ minimizing these errors is harder.

3. We can **approximate these with differentiable functions:**
   - $L_1$ can be approximated with Huber.
   - $L_\infty$ can be approximated with log-sum-exp.

4. **Gradient descent** finds local minimum of differentiable function.

5. For **convex functions**, any local minimum is a global minimum.
Very Robust Regression

• Consider data with extreme outliers:

- Non-convex errors can be very robust:
  – Eventually ‘give up’ on trying to make large errors smaller.
• But with non-convex errors, finding global minimum is hard.
• Absolute value is the most robust convex error function.
How do we know if a function is convex?

• Some useful tricks for showing a function is convex:
  – 1-variable, twice-differentiable function is convex iff \( f''(w) \geq 0 \) for all ‘w’.

Consider \( f(w) = \frac{1}{2} aw^2 \) for \( a > 0 \).
We have \( f'(w) = aw \) and \( f''(w) = a > 0 \) by assumption.

Consider \( f(w) = e^w \).
We have \( f'(w) = e^w \) and \( f''(w) = e^w > 0 \) by definition of exponential function.
How do we know if a function is convex?

• Some useful tricks for showing a function is convex:
  – 1-variable, twice-differentiable function is convex iff \( f''(w) \geq 0 \) for all ‘\( w ' \).
  – A convex function multiplied by non-negative constant is convex.

We showed that \( f(w) = e^w \) is convex, so \( f(w) = 10e^w \) is convex.
How do we know if a function is convex?

• Some useful tricks for showing a function is convex:
  – 1-variable, twice-differentiable function is convex iff $f''(w) \geq 0$ for all ‘$w$’.
  – A convex function multiplied by non-negative constant is convex.
  – Norms and squared norms are convex.
    $$\|w\|, \|w\|^2, \|w\|_1, \|w\|_\infty, \|w\|^2_1,$$ and so on are all convex.
**How do we know if a function is convex?**

- Some useful tricks for showing a function is convex:
  - 1-variable, twice-differentiable function is convex iff $f''(w) \geq 0$ for all ‘$w$’.
  - A convex function multiplied by non-negative constant is convex.
  - Norms and squared norms are convex.
  - The sum of convex functions is a convex function.

$$f(x) = 10e^w + \frac{1}{2} \|w\|^2$$ is convex

From earlier, constant norm squared
How do we know if a function is convex?

• Some useful tricks for showing a function is convex:
  – 1-variable, twice-differentiable function is convex iff $f''(w) \geq 0$ for all ‘$w$’.
  – A convex function multiplied by non-negative constant is convex.
  – Norms and squared norms are convex.
  – The sum of convex functions is a convex function.
  – The max of convex functions is a convex function.

$$f(w) = \max \sum \ell(w) + w^2$$ is convex.
How do we know if a function is convex?

• Some useful tricks for showing a function is convex:
  – 1-variable, twice-differentiable function is convex iff $f''(w) \geq 0$ for all ‘w’.
  – A convex function multiplied by non-negative constant is convex.
  – Norms and squared norms are convex.
  – The sum of convex functions is a convex function.
  – The max of convex functions is a convex function.
  – Composition of a convex function and a linear function is convex.

  If ‘f’ is convex the $f(Xw - y)$ is convex.
How do we know if a function is convex?

• Some useful tricks for showing a function is convex:
  – 1-variable, twice-differentiable function is convex iff \( f''(w) \geq 0 \) for all ‘w’.
  – A convex function multiplied by non-negative constant is convex.
  – Norms and squared norms are convex.
  – The sum of convex functions is a convex function.
  – The max of convex functions is a convex function.
  – Composition of a convex function and a linear function is convex.

• But: not true that composition of convex with convex is convex:
  Even if ‘f’ is convex and ‘g’ is convex, \( f(g(w)) \) might not be convex.
  E.g. \( x^2 \) is convex and \( -\log(x) \) is convex but \( -\log(x^2) \) is not convex.
Example: Convexity of Linear Regression

- Consider linear regression objective with error function ‘g’:
  \[ f(w) = \sum_{i=1}^{n} g(w^T x_i - y_i) \]

- Sufficient for ‘g’ to be convex for ‘f’ to be convex:
  - Then each term is composition of convex with linear.
  - And sum of convex is convex.

- Examples:
  For squared error \( g(r_i) = \frac{1}{2} r_i^2 \) so \( g''(r_i) = 1 \) and ‘f’ is convex.
  For absolute error \( g(r_i) = |r_i| \) which is a norm so ‘f’ is convex.
Example: Convexity of Linear Regression

- Consider linear regression objective with error function ‘g’:
  \[ f(w) = \sum_{i=1}^{n} g(w^T x_i - y_i) + \frac{\lambda}{2} \|w\|^2 \]

- Sufficient for ‘g’ to be convex for ‘f’ to be convex:
  – Then each term is composition of convex with linear.
  – And sum of convex is convex.

- Same condition applies with \( L_2 \)-regularization.
Linear Models with Binary Features

• What is the effect of a binary feature on linear regression?

<table>
<thead>
<tr>
<th>Year</th>
<th>Gender</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>1</td>
<td>1.85</td>
</tr>
<tr>
<td>1975</td>
<td>0</td>
<td>2.25</td>
</tr>
<tr>
<td>1980</td>
<td>1</td>
<td>1.95</td>
</tr>
<tr>
<td>1980</td>
<td>0</td>
<td>2.30</td>
</tr>
</tbody>
</table>

• Adding a bias $w_0$, our linear model is:

$$\text{height} = w_0 + w_1 \times \text{year} + w_2 \times \text{gender}$$

• The ‘gender’ variable causes a change in $y$-intercept:

  If $\text{gender} == 0$ then $\text{height} = w_0 + w_1 \times \text{year}$
  If $\text{gender} == 1$ then $\text{height} = w_0 + w_1 \times \text{year} + w_2$

http://www.medalinframe.com/athletes/sara-simeoni/
http://www.at-a-lanta.nl/weia/Progressie.html
Linear Models with Binary Features

• What if different genders have different slopes?
  – You can use gender-specific feature.

<table>
<thead>
<tr>
<th>Year</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>1</td>
</tr>
<tr>
<td>1975</td>
<td>0</td>
</tr>
<tr>
<td>1980</td>
<td>1</td>
</tr>
<tr>
<td>1980</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\text{distance} = w_0 + w_1 \times \text{year (if gender = 1)}
\]
\[
\text{distance} = w_3 + w_4 \times \text{year (if gender = 0)}
\]
Linear Models with Binary Features

• That trick fits separate ‘local’ variable for each gender.
• To share information across genders, include a ‘global’ version.

<table>
<thead>
<tr>
<th>Year</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>1</td>
</tr>
<tr>
<td>1975</td>
<td>0</td>
</tr>
<tr>
<td>1980</td>
<td>1</td>
</tr>
<tr>
<td>1980</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year (if gender = 1)</th>
<th>Year (if gender = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1975</td>
<td>0</td>
</tr>
<tr>
<td>1975</td>
<td>1975</td>
</tr>
<tr>
<td>1980</td>
<td>0</td>
</tr>
<tr>
<td>1980</td>
<td>1980</td>
</tr>
</tbody>
</table>

• ‘Global’ year feature: influence of time on both genders.
  – E.g., improvements in technique.
• ‘Local’ year feature: gender-specific deviation from global trend.
  – E.g., different effects of performance-enhancing drugs.

\[ y_i = w_0 + w_1 \times \text{year} + w_3 \times \text{year} \]
## Linear Models with Binary Features

### Feature Matrix

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
</tr>
<tr>
<td>5</td>
<td>O</td>
</tr>
<tr>
<td>2.5</td>
<td>Δ</td>
</tr>
<tr>
<td>1.5</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Δ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ x = \text{...} \]
Linear Models with Binary Features

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
</tr>
<tr>
<td>5</td>
<td>O</td>
</tr>
<tr>
<td>2.5</td>
<td>Δ</td>
</tr>
<tr>
<td>1.5</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>Δ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ X_i = \begin{bmatrix} \text{Feature 1} \\ \text{Feature 2} \end{bmatrix} \]

Model 1: only bias
\[ y_i = w_0 \]

Graph showing data points and a line for \( y_i = w_0 \).
Linear Models with Binary Features

\[ \mathbf{X} = \begin{bmatrix} 0.5 & X \\ 3 & O \\ 5 & O \\ 2.5 & \Delta \\ 1.5 & X \\ 3 & \Delta \\ \vdots & \vdots \end{bmatrix} \]

\[ y_i = w_0 \]

Model 1: only bias

Model 2: bias + feature

\[ y_i = w_0 + w_1 x_{i1} \]
Linear Models with Binary Features

Feature 1 | Feature 2
---|---
0.5 | X
3 | O
5 | O
2.5 | Δ
1.5 | X
3 | Δ
...

\[ x = \begin{bmatrix}
0.5 & X \\
3 & O \\
5 & O \\
2.5 & Δ \\
1.5 & X \\
3 & Δ \\
... & ...
\end{bmatrix} \]

Model 1: only bias
\[ y_i = w_0 \]

Model 2: bias + feature
\[ y_i = w_0 + w_1 x_{i1} \]

Model 3: "local" bias + feature
\[ y_i = w_f + w_1 x_{i1} \]
Linear Models with Binary Features

\[ \mathbf{X} = \begin{bmatrix} \text{Feature 1} & \text{Feature 2} \\ 0.5 & X \\ 3 & O \\ 5 & O \\ 2.5 & \Delta \\ 1.5 & X \\ 3 & \Delta \\ \vdots & \vdots \end{bmatrix} \]

- Model 1: only bias
  \[ y_i = w_0 \]

- Model 2: bias + feature
  \[ y_i = w_0 + w_1 x_{i1} \]

- Model 3: "local" bias + feature
  \[ y_i = w_l + w_1 x_{i1} \]

- Model 4: "local" bias and "local" slope
  \[ y_i = w_l + w_1 x_{i1} \]

Bias for shape
Slope for shape
Linear Models with Binary Features

Could also share information across categories with global bias and slope:
\[ y_i = w_0 + w_1 x_{i1} + w_e + w_{e1} x_{i1} \]

Model 1: only bias
\[ y_i = w_0 \]

Model 2: bias + feature 1
\[ y_i = w_0 + w_1 x_{i1} \]

Model 3: "local" bias + feature 1
\[ y_i = w_e + w_{e1} x_{i1} \]

Model 4: "local" bias and "local" slope
\[ y_i = w_e + w_{e1} x_{i1} \]
Motivation: Identifying Important E-mails

• How can we automatically identify ‘important’ e-mails?
  – Mark as ‘important’ if user takes some action based on them.

• We have a big collection of e-mails:
  – Mark as ‘important’ if user takes some action based on them.

• There might be some “universally” important messages:
  – “This is your mother, something terrible happened, give me a call ASAP.”

• But your “important” message may be unimportant to others.
  – Similar for spam: “spam” for one user could be “not spam” for another.
The Big Global/Local Feature Table

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

"Global" features: shared by all users

"Local" features for user "1": set to 0 for all other users.

"Local" features for user "2"
Predicting Importance of E-mail For New User

• Consider a new user:
  – Start out with no information about them.
  – Use global features to predict what is important to generic user.
  \[ y_i = \text{sign} \left( w_g^T x_g \right) \]
  – Local features make prediction personalized.
  \[ y_i = \text{sign} \left( w_g^T x_g + w_u^T x_u \right) \]
  – What is important to this user?

• With more data, update global features and user’s local features:
  – Local features make prediction personalized.

• G-mails system: classification with logistic regression.
Classification Using Regression?

• Usual approach to do classification with regression:
  – Code $y_i$ as ‘-1’ for one class and ‘+1’ for the other class.
  – E.g., ‘+1’ means ‘important’ and ‘-1’ means ‘not important’.

• At training time, fit a linear regression model:
  \[ y_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} \]
  \[ = \mathbf{w}^T \mathbf{x}_i \]

• To predict, we take the sign (i.e., closer ‘-1’ or ‘+1’?):
  \[ y_i = \text{sign}(\mathbf{w}^T \mathbf{x}_i) \]
  \[ \text{Set } y_i = +1 \text{ if } \mathbf{w}^T \mathbf{x}_i > 0 \]
  \[ \text{Set } y_i = -1 \text{ if } \mathbf{w}^T \mathbf{x}_i < 0 \]
Classification using Regression

Linear regression model:

\[ y_i = w^T x_i \]

"important" \[ \rightarrow \] \[ x \times x \times +1 \times x \times x \times \times \times \times \times \]

"not important"
Classification using Regression

"Decision boundary is at $w^T x_i = 0$"

Our "predict" function

Linear regression model $y_i = w^T x_i$
Classification Using Regression

- Can use regression tricks (basis, regularization) for classification.
- But, **usual error functions do weird things**:
Classification Using Regression

• What went wrong?
  – “Good” errors vs. “bad” errors.

\[ f(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2 \]

This is the linear regression model we want (a perfect classifier)

What happens if \( y_i = -1 \) and \( w^T x_i = -1000 \)?

#times we see "vacation"
Classification Using Regression

- What went wrong?
  - “Good” errors vs. “bad” errors.

\[ f(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2 \]

What happens if \( y_i = -1 \) and \( w^T x_i = -1000 \)?

"Bad" errors of the perfect linear classifier are HUGE.

This is the linear regression model we want (a perfect classifier).

This is what we actually get.
Comparing Loss Functions

\[(w^Tx_i - y_i)^2\]

Big penalty for being "too right"

Error" or "loss" for predicting \(w^T x_i\) when true label \(y_i\) is \(-1\).

Prediction \(w^T x_i\)

"bad" error: you should not penalize for putting \(w^T x_i\) here.

"good" error: having \(w^T x_i\) here is bad.
Comparing Loss Functions

$(w^Tx_i - y_i)^2$

"Error" or "loss" for predicting $w^Tx_i$ when true label $y_i$ is $-1$.

Big penalty for being "too right".

Absolute error reduces but does not fix this issue.

"Bad" error: you should not penalize for putting $w^Tx_i$ here.

"Good" error: having $w^Tx_i$ here is bad.
Comparing Loss Functions

- $\left( w^T x_i - y_i \right)^2$
- "Error" or "loss" for predicting $w^T x_i$ when true label $y_i$ is $-1$. 
- What we want is the "0-1 loss".

- Big penalty for being "too right".
- Absolute error reduces but does not fix this issue.
- "bad" error: you should not penalize for putting $w^T x_i$ here.
- "good" error: having $w^T x_i$ here is bad.
0-1 Loss Function

• The **0-1 loss function** is the **number of classification errors**:
  – Unlike regression, in classification it’s reasonable that \( \text{sign}(w^T x_i) = y_i \).

• Unfortunately the **0-1 loss is non-convex** in \( 'w' \).
  – It’s easy to minimize if a perfect classifier exists.
  – Otherwise, finding the \( 'w' \) minimizing 0-1 loss is a hard problem.

• **Convex approximations to 0-1 loss**:
  – **Hinge loss** (non-smooth) and **logistic loss** (smooth).
Convex Approximations to 0-1 Loss

"hinge" loss

"Error" or "loss" for predicting $w^T x_i$ when true label $y_i$ is -1.

What we want is the "0-1 loss."

Prediction $w^T x_i$
Convex Approximations to 0-1 Loss

"Hinge" loss

"Logistic" loss (similar but smooth)

"Error" or "loss" for predicting $w^T x_i$ when true label $y_i$ is $-1$.

What we want is the "0-1 loss."

They don't penalize for being "very right."

They still penalize you for being "very wrong."
Hinge Loss and Support Vector Machines

• **Hinge loss** is given by:

\[ f(w) = \sum_{i=1}^{n} \max \{0, 1 - y_i w^T x_i\} \]

  – Convex upper bound on number of classification errors.
  – Solution will be a perfect classifier, if one exists.

• **Support vector machine (SVM)** is hinge loss with L2-regularization.

\[ f(w) = \sum_{i=1}^{n} \max \{0, 1 - y_i w^T x_i\} + \frac{\lambda}{2} ||w||^2 \]

  – Next time we’ll see that it “maximizes the margin”.
Logistic Regression

• Logistic regression minimizes logistic loss:

\[ \hat{f}(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^\top x_i)) \]

• You can/should also add regularization:

\[ \hat{f}(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^\top x_i)) + \frac{\lambda}{2} \|w\|^2 \]

• Convex and differentiable: minimize this with gradient descent.
Logistic Regression and SVMs

• Logistic regression and SVMs are used EVERYWHERE!

• Why?
  – Training and testing are both fast.
  – It is easy to understand what the weights ‘$w_j$’ mean.
  – With high-dimensional features and regularization, often good test error.
  – Otherwise, often good test error with RBF basis and regularization.
  – Smoother predictions than random forests.
Summary

- **Convex functions** can be identified using a few simple rules.
- **Global vs. local features** allows ‘personalized’ predictions.
- **Classification using regression** works if done right.
- **0-1 loss** is the ideal loss, but is non-smooth and non-convex.
- **Logistic regression** uses a convex and smooth approximation to 0-1.

- Next time:
  - One more reason to use regularization, and how to find gold.
Bonus Slide: Perceptron Algorithm

• One of the first “learning” is the perceptron algorithm.
  – Searches for a ‘w’ such that $w^T x_i > 0$ when $y_i = +1$, $w^T x_i < 0$ for $y_i = -1$.

• Perceptron Algorithm:
  – Start with $w^0 = 0$.
  – Go through examples in any order until you make a mistake predicting $y^i$.
    • Set $w^{t+1} = w^t + y_i x_i$.
  – Keep going through examples until you make no errors on training data.

• If a perfect classifier exists, this algorithm converges to one.
  – In fact, “perceptron mistaked bound” result says that number of mistakes is finite.