CPSC 340: Machine Learning and Data Mining

Multi-Class Regression

Fall 2016

Admin

- Midterm:
  - Grades/solutions will be posted later this week.
- Assignment 4:
  - Posted, due November 14.
- Extra office hours:
  - Thursdays from 4:30-5:30 in ICICS X836.

Last Time: L1-Regularization

- We discussed L1-regularization:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \lambda\|w\|_1$$

  - Also known as "LASSO" and "basis pursuit denoising".
  - Regularizes 'w' so we decrease our test error (like L2-regularization).
  - Yields sparse 'w' so it selects features (like L0-regularization).
- Properties:
  - It's convex and fast to minimize (proximal-gradient).
  - Solution is not unique.
  - Tends to yield false positives.

Extensions of L1-Regularization

- “Elastic net” uses L2-regularization plus L1-regularization.
  - Solution is still sparse but is now unique.
  - Slightly better with feature dependence: selects both “mom” and “mom2”.

- “Bolasso” runs L1-regularization on bootstrap samples.
  - Selects features that are non-zero in all samples.
  - Much less sensitive to false positives.

- There are *many* non-convex regularizers (square-root, “SCAD”):
  - Much less sensitive to false positives.
  - But computing global minimum is hard.

Last: Maximum Likelihood Estimation

- We discussed computing 'w' by maximum likelihood estimation (MLE):

$$p(y \mid X, w)$$

- This is equivalent to minimizing negative log-likelihood (NLL):

$$f(w) = -\sum_{i=1}^{n} \log(p(y_i \mid x_i, w))$$

- For logistic regression, probability is $w^T x_i$ passed through sigmoid.

$$p(y_i \mid x_i, w) = \frac{1}{1 + \exp(-y_i w^T x_i)}$$

- And MLE is minimum of:

$$f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$$

- With regularization, similar to SVMs. But gives probabilities.

Maximum Likelihood and Least Squares

- Many of our objective functions can be written as an MLE.
- For example, consider Gaussian likelihood with mean of $w^T x_i$:

$$p(y_i \mid x_i, w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)$$

- So the NLL is given by:

$$f(w) = -\sum_{i=1}^n \log(p(y_i \mid x_i, w))$$

$$= -\sum_{i=1}^n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)\right)$$

$$= -\sum_{i=1}^n \Bigg[\underbrace{\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)} + \underbrace{\log\left(\exp\right.}_{\text{operations cancel}}\left.\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)\right)\Bigg]$$

Constant in terms of 'w'

$$= (\text{constant}) - \sum_{i=1}^n \left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)$$

$$= (\text{constant}) + \frac{1}{2\sigma^2} \sum_{i=1}^n (w^T x_i - y_i)^2$$

positive constant so doesn't change solution

Maximum Likelihood and Least Squares

- Many of our objective functions can be written as an MLE.
- For example, consider Gaussian likelihood we mean of $w^T x_i$:

$$p(y_i \mid x_i, w) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)$$

- So we can minimize NLL by minimizing:

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

$$= \frac{1}{2}\|Xw - y\|^2$$

- So least squares is MLE under Gaussian likelihood.
  - With a Laplace likelihood you would absolute error.

Problem with Maximum Likelihood Estimation

- Maximum likelihood estimate maximizes:

$$p(y \mid X, w)$$

- It's is a bit weird:
  - "Find the 'w' that makes 'y' have the highest probability given 'X' and 'w'."
- A problem with MLE:
  - 'y' could be very likely for some very unlikely 'w'.
  - E.g., complex model that overfit by memorizing the data.

- What we really want:
  - "Find the 'w' that has the highest probability given 'X' and 'y'."

Maximum a Posteriori (MAP) Estimation

- Maximum a posteriori (MAP) maximizes what we want:

$$p(w \mid X, y)$$

"posterior" probability of 'w' given data $\{X, y\}$

- Using Bayes' rule, we have

$$p(w \mid X, y) = \frac{p(y \mid X, w)\, p(w \mid X)}{p(y \mid X)} \propto p(y \mid X, w)\, p(w \mid X)$$

Assume 'w' does not depend on 'X'!

$$= \underbrace{p(y \mid X, w)}_{\text{"likelihood"}}\ \underbrace{p(w)}_{\text{"prior"}}$$

Maximum a Posteriori (MAP) Estimation

- Maximum a posteriori (MAP) maximizes what we want:

$$\underbrace{p(w \mid X, y)}_{\text{"posterior"}} \propto \underbrace{p(y \mid X, w)}_{\text{"likelihood"}} \underbrace{p(w)}_{\text{"prior"}}$$

- Prior p(h) is 'belief' that 'w' is the correct before seeing data:
  - Can take into account that complex models can overfit.
- If we again minimize the negative of the logarithm, we get:

$$-\log(p(w \mid X, y)) = -\log(p(y \mid X, w)) - \log(p(w)) + (\text{constant})$$

IID data

prior over the w_j are independant

$$= -\sum_{i=1}^{n} \log(p(y_i \mid x_i, w)) - \sum_{j=1}^{d} \log(p(w_j)) + (\text{constant})$$

MAP Estimation and Regularization

- While many losses are equivalent to NLLs,
  many regularizers are equivalent to negative log-priors.
- Assume each w_j comes from a Gaussian (0-mean, 1/λ variance):

$$p(w_j) = \frac{1}{\sqrt{2\pi(1/\lambda)}} \exp\left(-\frac{(w_j - 0)^2}{2(1/\lambda)}\right)$$

- Then the log-prior is:

$$\log(p(w_j)) = -\log\left(\sqrt{2\pi(1/\lambda)}\right) + \log\left(\exp\left(-\frac{\lambda}{2} w_j^2\right)\right)$$
$$= (\text{constant}) - \frac{\lambda}{2} w_j^2$$

- And negative log-prior over all 'j' is:

$$-\log(p(w)) = -\sum_{j=1}^{d} \log(p(w_j)) = (\text{constant}) + \sum_{j=1}^{d} \frac{\lambda}{2} w_j^2 = \frac{\lambda}{2}\|w\|^2 + \text{const.} \quad (L_2\text{-regularization})$$

MAP Estimation and Regularization

- MAP estimation gives link between probabilities and loss functions.
  - Gaussian likelihood and Gaussian prior gives L2-regularized least squares.

If $p(y_i \mid x_i, w) \propto \exp\left(-\frac{(w^T x_i - y_i)^2}{2\sigma^2}\right)$ $p(w_j) \propto \exp\left(-\frac{\lambda}{2} w_j^2\right)$

then MAP estimation is equivalent to minimizing $f(w) = \frac{1}{2\sigma^2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$

for MAP estimation the constant changes solution.
But setting $\sigma^2 \neq 1$ is equivalent to changing λ so we usually use $\sigma^2 = 1$

  - Sigmoid likelihood and Gaussian prior gives L2-regularized logistic regression:

If $p(y_i \mid x_i, w) = \frac{1}{1+\exp(-y_i w^T x_i)}$ and $p(w_j) \propto \exp\left(-\frac{\lambda}{2} w_j^2\right)$

then MAP estimate is minimum of $f(w) = \underbrace{\sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))}_{} + \frac{\lambda}{2}\|w\|^2$

As $n \to \infty$ effect of prior/regularizer goes to 0

Why do we care about MLE and MAP?

- Unified way of thinking about many of our tricks?
  - Laplace smoothing in naïve Bayes can be viewed as regularization.

- Remember our two ways to reduce complexity of a model:
  - Model averaging (ensemble methods).
  - Regularization (linear models).
- "Fully"-Bayesian methods combine both of these.
  - Average over all models, weighted by posterior (which includes regularizer).
  - Very powerful class of models we'll cover in CPSC 540.

- Sometimes it's easier to define a likelihood than a loss function.
  - We'll do this for multi-class classification.

Multi-Label Classification

- We’ve been considering supervised learning with a single label:

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

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

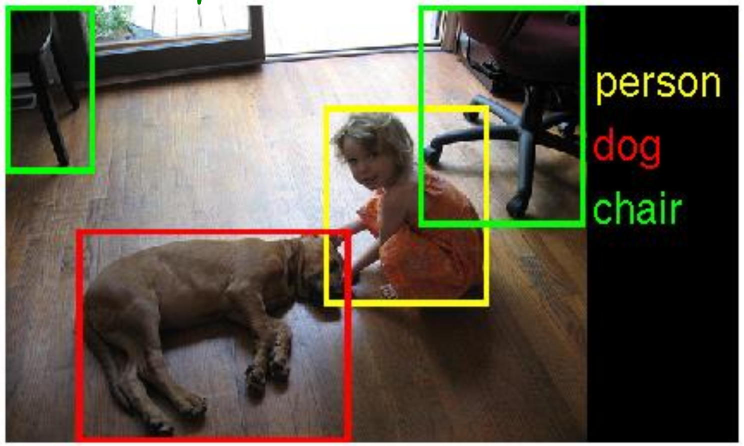
Multi-Label Classification

- In multi-label classification we want to predict 'k' binary labels:

$$X = \underbrace{\begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix}}_{d} \Bigg\} n$$

$$Y = \begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \underbrace{\begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}}_{k} \end{array} \Bigg\} n$$

- E.g., which of the 'k' objects are in this image?



http://image-net.org/challenges/LSVRC/2013/

Multi-Label Classification

$$X = \left.\underbrace{\begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix}}_{d}\right\} n$$

$$Y = \begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \left.\underbrace{\begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}}_{k}\right\} n \end{array}$$

- Approach 1:
  - Treat {1,-1,1,1,-1} as the binary label 10110.
  - Problem is that with ‘k’ labels you have 2^k classes.
    - Only useful if ‘k’ is very small.

Multi-Label Classification

$$X = \underbrace{\begin{bmatrix} \quad & \quad & \quad \\ & & \\ & & \end{bmatrix}}_{d}\Bigg\}n \qquad Y = \begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \underbrace{\begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}}_{K} \end{array}\Bigg\}n$$

- Approach 2:
  - Fit a binary classifier for each column 'c' of Y, using column as labels.
- If we use a linear model, each classifier has a weights w_c.
- Let's put the w_c together into a matrix 'W':

$$W = \underbrace{\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_K \\ | & | & & | \end{bmatrix}}_{K}\Big\}d$$

Multi-Label Classification

$$X = \begin{bmatrix} \quad & \quad & \quad \end{bmatrix}$$

$$Y = \underbrace{\begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \\ \end{array}}_{} \left.\begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}\right\} d$$

$$\underbrace{\qquad\qquad\qquad}_{K}$$

$$W = \underbrace{\left.\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_K \\ | & | & & | \end{bmatrix}\right\} d}_{K}$$

Each column 'c' is a binary classifier for class 'c'

To predict label 'c' for example 'i' use $y_{ic} = \text{sign}(w_c^T x_i)$

To predict all labels y_i for example 'i' use $y_i = \text{sign}(W^T x_i)$

To predict all labels for all examples use $Y = \text{sign}(XW)$

- Fancier methods model correlations between y_i or between the w_c.

Multi-Class Classification

$$X = \begin{bmatrix} \quad \end{bmatrix} \qquad Y = \begin{array}{c} \begin{array}{ccccc} \text{cat} & \text{dog} & \text{person} & \text{chair} & \text{mouse} \end{array} \\ \left.\underbrace{\begin{bmatrix} 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & -1 & -1 & -1 & -1 \end{bmatrix}}_{k}\right\} d \end{array}$$

$$W = \left.\underbrace{\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix}}_{k}\right\} d$$

Each column 'c' is a binary classifier for class 'c'

- Multi-class classification: special case where each y_i has 1 non-zero.

Multi-Class Classification and “One vs. All”

$$X = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 4 \\ 4 \\ 1 \end{bmatrix} \quad W = \underbrace{\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix}}_{k} \Big\} d$$

Each column 'c' is a binary classifier for class 'c'

- Multi-class classification: special case where each y_i has 1 non-zero.
  - Now we can code y_i as a discrete number {1,2,3,…} giving class ‘k’.
- One vs. all multi-class approach uses naïve multi-label approach:
  - Independently fit parameters ‘w_c’ of a linear model for each class ‘c’.
    - Each ‘w_c’ tries to predict +1 for class ‘c’ and -1 for all others.

Multi-Class Classification and "One vs. All"

$X = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$ $y = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 4 \\ 4 \\ 1 \end{bmatrix}$

$$W = \underbrace{\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix}}_{k} \Big\} d$$

Each column 'c' is a binary classifier for class 'c'

- But prediction W^Tx_i might have multiple +1 values.
- To predict the "best" label, choose 'c' with largest value of $w_c^Tx_i$.

$$y_i = \underset{c}{\text{argmax}} \{w_c^T x_i\}$$

Choose the 'c' that achieves the maximum value.

For logistic regression this choose class 'c' with the highest probability $p(y_{ic} = +1 \mid x_i, w)$

Multi-Class Classification and “One vs. All”

$$X = \begin{bmatrix} \\ \\ \\ \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 4 \\ 4 \\ 1 \end{bmatrix}$$

$$W = \underbrace{\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_k \\ | & | & & | \end{bmatrix}}_{k} \Big\} d$$

Each column 'c' is a binary classifier for class 'c'

$$y_i = \underset{c}{\operatorname{argmax}} \{ w_c^T x_i \}$$

Choose the 'c' that achieves the maximum value.

- But we only trained w_c to get the correct sign of y_{ic}:
  - We didn’t train the w_c so that the largest $w_c^T x_i$ would be y_i.

Multinomial Logistic Regression

- Can we define a loss function so that largest $w_c^T x_i$ gives y_i?
- In the multi-label logistic regression model we used:

$$p(y_{ic} = +1 \mid x_i, W) = \frac{1}{1 + \exp(-w_c^T x_i)} = \frac{\exp(w_c^T x_i)}{\exp(w_c^T x_i) + 1} \propto \exp(w_c^T x_i)$$

- The multinomial logistic regression model uses the same idea:

$$p(y_i = c \mid x_i, W) \propto \exp(w_c^T x_i)$$

- But now need to sum over 'k' classes to get a valid probability:

$$p(y_i = c \mid x_i, W) = \frac{\exp(w_c^T x_i)}{\exp(w_1^T x_i) + \exp(w_2^T x_i) + \cdots + \exp(w_k^T x_i)}$$

Multinomial Logistic Regression

- So multinomial logistic regression uses:

$$p(y_i \mid x_i, W) = \frac{\exp(w_{y_i}^T x_i)}{\sum_{c=1}^{k} \exp(w_c^T x_i)}$$

- Which is also known as the softmax function.
- Now that we have a probability, the MLE gives a loss function:

$$f(W) = -\sum_{i=1}^{n} \log(p(y_i \mid x_i, W))$$

$$= \sum_{i=1}^{n} \underbrace{-w_{y_i}^T x_i}_{\text{Tries to make this large for correct label.}} + \underbrace{\log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right)}_{\text{Log-sum-exp which is smooth approximation to } \max_c \{\exp(w_c^T x_i)\} \text{ so we're trying to make the max small.}}$$

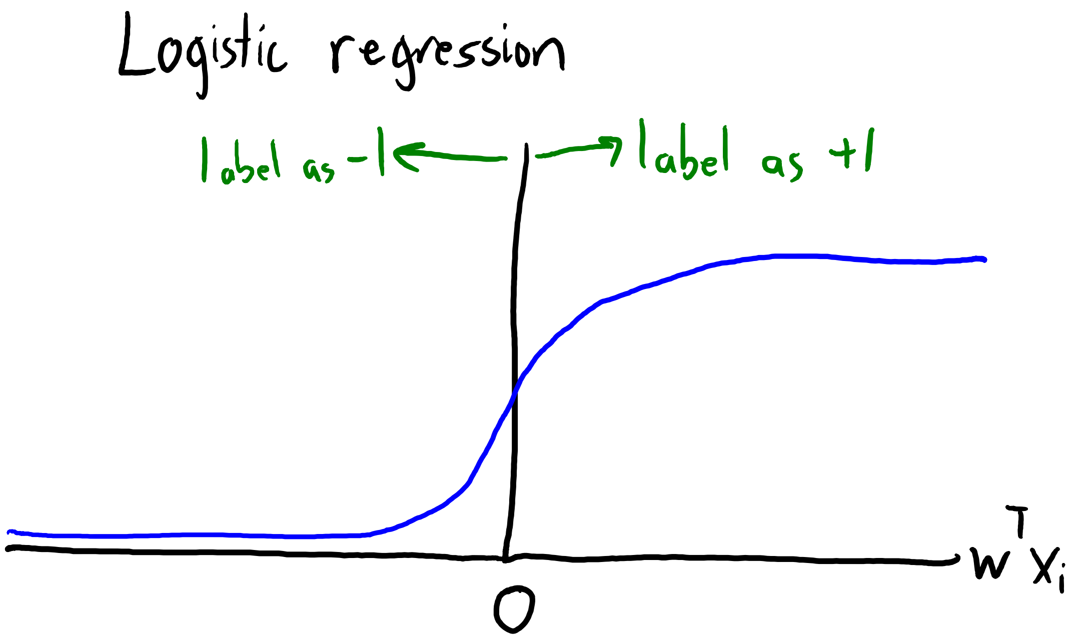
Convex so we can use gradient descent.

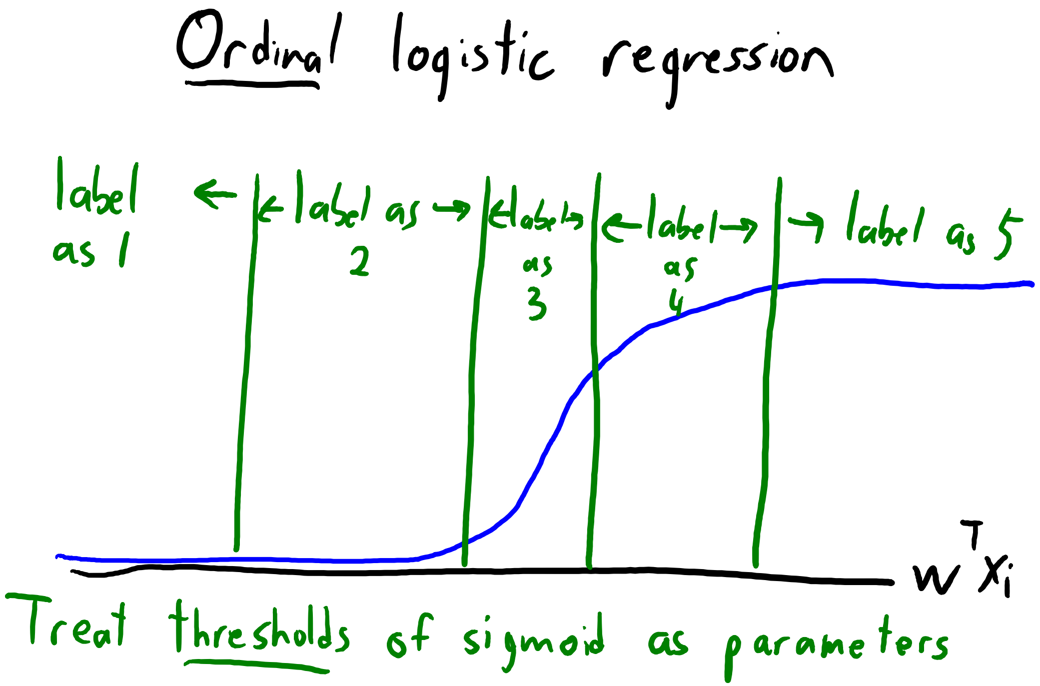
Losses for Other Discrete Labels

- MLE/MAP gives loss for classification with basic discrete labels:
  - Logistic regression for binary labels {“spam”, “not spam”}.
  - Softmax regression for multi-class {“spam”, “not spam”, “important”}.
- But MLE/MAP lead to losses with other discrete labels:
  - Ordinal: {1 star, 2 stars, 3 stars, 4 stars, 5 stars}.
  - Counts: 602 ‘likes’.
- We can also use ratios of probabilities to define more losses:
  - Multi-class SVMs (similar to softmax, but generalizes hinge loss).
  - Ranking: Difficulty(A3) > Difficulty(A4) > Difficulty (A2) > DifficultyA(1).

Ordinal Labels

- Ordinal data: categorical data where the order matters:
  - Rating hotels as {‘1 star’, ‘2 stars’, ‘3 stars’, ‘4 stars’, ‘5 stars’}.
  - Softmax would ignore order.
- Can use ‘ordinal logistic regression’.





Count Labels

- Count data: predict the number of times something happens.
  - For example, y_i = "602" Facebok likes.
- Softmax/ordinal require finite number of possible labels.
- We probably don't want separate parameter for '654' and '655'.
- Poisson regression: use probability from Poisson count distribution.
  - Many variations exist.

Summary

- -log(probability) lets us to define loss from any probability.
  - Special cases are least squares, least absolute error, and logistic regression.
- MAP estimation directly models p(w | X, y).
  - Gives probabilistic interpretation to regularization.
- Softmax loss is natural generalization of logistic regression.
- Discrete losses for weird scenarios are possible using MLE/MAP:
  - Ordinal logistic regression, Poisson regression.

- Next time:
  - What 'parts' are your personality made of?

Bonus Slide: Multi-Class SVMs

Bonus Slide: Ranking