

# CPSC 340: Machine Learning and Data Mining

MLE and MAP

Fall 2019

# Last Time: Maximum Likelihood Estimation (MLE)

- Maximum likelihood estimation (MLE):
  - Define a likelihood function, probability of data given parameters:  $p(D | w)$ .
  - Choose parameters ‘w’ to maximize the likelihood.
- Gives naïve Bayes “counting” estimates we used.
- Typically easier to equivalently minimize negative log-likelihood (NLL).

$$\hat{w} \in \arg\max_w \{ p(D|w) \} \equiv \arg\min_w \{ -\log(p(D|w)) \}$$

↑  
"equivalent"

- This will turns product of probability over IID examples into sum over examples.

# But first: “argmin” and “argmax”

- We've repeatedly used the **min** and **max** functions:

$$\min_w \{w^2\} = 0 \quad \max_w \{\cos(w)\} = 1$$

- Minimum (or maximum) value achieved by a function.
- A related set of functions are the **argmin** and **argmax**:
  - The **set of parameter values** achieving the minimum (or maximum).

$$\min_w \{(w-1)^2\} = 0$$

$$\operatorname{argmin}_w \left\{ \frac{1}{2} \|X_w - y\|^2 + \frac{\lambda}{2} \|w\|^2 \right\} = (X^\top X + \lambda I)^{-1} (X^\top y)$$

$$\operatorname{argmin}_w \{(w-1)^2\} = 1$$

$$\operatorname{argmax}_w \{\cos(w)\} = 0, 2\pi, 4\pi, \dots$$

'1' is the 'w' value that gives the min.

# But first: “argmin” and “argmax”

- The last slide is a little sloppy for the following reason:
  - There **may be multiple values** achieving the min and/or max.
  - So the **argmin** and **argmax** return sets.

$$\underset{w}{\operatorname{argmin}} \{(w-1)^2\} \equiv \{1\}$$

"sets are equivalent"

*"set containing the element '1'"*

$$\underset{w}{\operatorname{argmax}} \{\cos(w)\} \equiv \{\dots, -4\pi, -2\pi, 0, 2\pi, 4\pi, \dots\}$$

$$\underset{w}{\operatorname{argmax}} \left\{ \frac{1}{2} \|X_w - y\|^2 \right\} \equiv \{w \mid X^T X_w = X^T y\}$$

- And we don't say a variable "is" the argmax, but that it "is in" the argmax.

$$2\pi \in \underset{w}{\operatorname{argmax}} \{\cos(w)\}$$

$$(X^T X + \lambda I)^{-1}(X^T y) \in \underset{w}{\operatorname{argmax}} \left\{ \frac{1}{2} \|X_w - y\|^2 + \frac{\lambda}{2} \|w\|^2 \right\}$$

# Minimizing the Negative Log-Likelihood (NLL)

- We use **log-likelihood** because it **turns multiplication into addition**:

$$\log(\alpha \beta) = \log(\alpha) + \log(\beta)$$

- More generally:  $\log\left(\prod_{i=1}^n a_i\right) = \sum_{i=1}^n \log(a_i)$

- If data is ' $n$ ' IID samples then  $p(D|w) = \prod_{i=1}^n p(D_i|w)$

likelihood of  
example  $i$

and our MLE is  $\hat{w} \in \operatorname{argmax}_w \left\{ \prod_{i=1}^n p(D_i|w) \right\} \equiv \operatorname{argmin}_w \left\{ - \sum_{i=1}^n \log(p(D_i|w)) \right\}$

# Least Squares is Gaussian MLE (Gory Details)

- Let's assume that  $y_i = w^T x_i + \varepsilon_i$ , with  $\varepsilon_i$  following standard normal:

$$p(\varepsilon_i) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\varepsilon_i^2}{2}\right)$$

also known  
as "Gaussian"  
distribution

- This leads to a Gaussian likelihood for example 'i' of the form:

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

- Finding MLE (minimizing NLL) is least squares:

$$\begin{aligned} f(w) &= -\sum_{i=1}^n \log(p(y_i | w, x_i)) \\ &= -\sum_{i=1}^n \log\left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(w^T x_i - y_i)^2}{2}\right)\right) \\ &= -\sum_{i=1}^n \left[ \log\left(\frac{1}{\sqrt{2\pi}}\right) + \log\left(\exp\left(-\frac{(w^T x_i - y_i)^2}{2}\right)\right) \right] \end{aligned}$$

constant in 'w'

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

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

$= (\text{constant}) + \frac{1}{2} \|Xw - y\|^2$

operations cancel

# Loss Functions and Maximum Likelihood Estimation

- So least squares is MLE under Gaussian likelihood.

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

then MLE of ' $w$ ' is minimum of  $f(w) = \frac{1}{2} \|Xw - y\|^2$

- With a Laplace likelihood you would get absolute error.

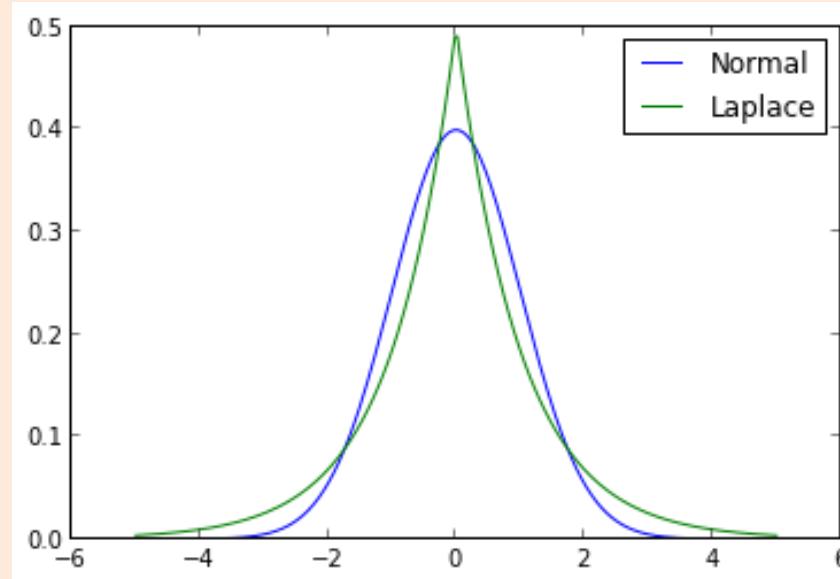
If  $p(y_i | x_i, w) = \frac{1}{2} \exp(-|w^T x_i - y_i|)$

then MLE is minimum of  $f(w) = \|Xw - y\|_1$

- Other likelihoods lead to different errors ("sigmoid" -> logistic loss).

# “Heavy” Tails vs. “Light” Tails

- We know that L1-norm is more robust than L2-norm.
  - What does this mean in terms of probabilities?



Here “tail” means  
“mass of the  
distribution away  
from the mean.”

- Gaussian has “light tails”: assumes everything is close to mean.
- Laplace has “heavy tails”: assumes some data is far from mean.
- Student ‘t’ is even more heavy-tailed/robust, but NLL is non-convex.

(pause)

# Maximum Likelihood Estimation and Overfitting

- In our abstract setting with data D the MLE is:

$$\hat{w} \in \underset{w}{\operatorname{argmax}} \{ p(D|w) \}$$

- But conceptually MLE is a bit weird:
  - “Find the ‘w’ that makes ‘D’ have the highest probability given ‘w’.”
- And MLE often leads to **overfitting**:
  - Data could be very likely for some **very unlikely ‘w’**.
  - For example, a complex model that overfits by memorizing the data.
- What we really want:
  - “Find the ‘w’ that has the highest probability given the data D.”

# Maximum a Posteriori (MAP) Estimation

- Maximum a posteriori (MAP) estimate maximizes the reverse probability:

$$\hat{w} \in \underset{w}{\operatorname{argmax}} \{ p(w|D) \}$$

- This is what we want: the probability of ‘w’ given our data.

- MLE and MAP are connected by Bayes rule:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)} \propto p(D|w)p(w)$$

posterior                          likelihood    prior

- So MAP maximizes the likelihood  $p(D|w)$  times the prior  $p(w)$ :

- Prior is our “belief” that ‘w’ is correct before seeing data.
- Prior can reflect that complex models are likely to overfit.

# MAP Estimation and Regularization

- From Bayes rule, the MAP estimate with IID examples  $D_i$  is:

$$\hat{w} \in \arg\max_w \{ p(w | D) \} \equiv \arg\max_w \left\{ \prod_{i=1}^n [p(D_i | w)] p(w) \right\}$$

- By again taking the negative of the logarithm as before we get:

$$\hat{w} \in \arg\min_w \left\{ - \sum_{i=1}^n \underbrace{[\log(p(D_i | w))]_i}_{\text{loss}} - \underbrace{\log(p(w))}_{\text{regularizer}} \right\}$$

- So we can view the negative log-prior as a regularizer:
  - Many regularizers are equivalent to negative log-priors.

# L2-Regularization and MAP Estimation

- We obtain L2-regularization under an independent Gaussian assumption:

Assume each  $w_j$  comes from a Gaussian with mean 0 and variance  $\frac{1}{\lambda}$

- This implies that:

$$p(w) = \prod_{j=1}^d p(w_j) \propto \prod_{j=1}^d \exp\left(-\frac{\lambda}{2} w_j^2\right) = \exp\left(-\frac{\lambda}{2} \sum_{j=1}^d w_j^2\right)$$

Independence Gaussian assumption  $e^\alpha e^\beta = e^{\alpha + \beta}$

- So we have that:

$$-\log(p(w)) = -\log\left(\exp\left(-\frac{\lambda}{2}\|w\|^2\right)\right) + (\text{constant}) = \frac{\lambda}{2}\|w\|^2 + (\text{constant})$$

- With this prior, the MAP estimate with IID training examples would be

$$\hat{w} \in \arg\min \left\{ -\log(p(y|x, w)) - \log(p(w)) \right\} \equiv \arg\min_w \left\{ - \sum_{i=1}^n [\log(p(y_i|x_i, w))] + \frac{\lambda}{2}\|w\|^2 \right\}$$

# MAP Estimation and Regularization

- MAP estimation gives link between probabilities and loss functions.
  - Gaussian likelihood ( $\sigma = 1$ ) + Gaussian prior gives L2-regularized least squares.

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

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

- Laplace likelihood ( $\sigma = 1$ ) + Gaussian prior give L2-regularized robust regression:

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

then MAP estimation is equivalent to minimizing  $f(w) = \|Xw - y\|_1 + \frac{\lambda}{2} \|w\|^2$

- As 'n' goes to infinity, effect of prior/regularizer goes to zero.
- Unlike with MLE, the choice of  $\sigma$  changes the MAP solution for these models.

# Summarizing the past few slides

- Many of our **loss functions and regularizers have probabilistic interpretations.**
  - Laplace likelihood leads to absolute error.
  - Laplace prior leads to L1-regularization.
- The choice of **likelihood** corresponds to the choice of **loss**.
  - Our assumptions about how the  $y_i$ -values can come from the  $x_i$  and ‘w’.
- The choice of **prior** corresponds to the choice of **regularizer**.
  - Our assumptions about which ‘w’ values are plausible.

# Regularizing Other Models

- We can view priors in other models as regularizers.
- Remember the problem with MLE for naïve Bayes:
  - The MLE of  $p(\text{'lactase'} = 1 \mid \text{'spam'})$  is:  $\text{count(spam,lactase)}/\text{count(spam)}$ .
  - But this caused problems if  $\text{count(spam,lactase)} = 0$ .
- Our solution was Laplace smoothing:
  - Add “+1” to our estimates:  $(\text{count(spam,lactase)}+1)/(\text{counts(spam)}+2)$ .
  - This corresponds to a “Beta” prior so Laplace smoothing is a regularizer.

(pause)

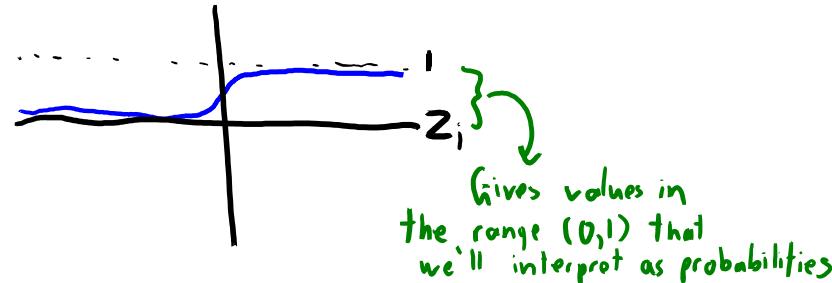
# Generative vs. Discriminative Models

- Naïve Bayes is called a generative model.
  - It models  $p(y_i, x_i)$ , we model “how the features ‘X’ are generated”.
    - You can get  $p(y_i | x_i)$  using the rules of probability to make predictions.
    - This type of approach often works well with lots of features but small ‘n’.
    - A “generative” version of linear regression is “linear discriminant analysis”.
- Linear (and logistic) regression are called discriminative models.
  - Treat features ‘X’ as fixed, and directly model  $p(y_i | x_i)$ .
    - No need to model  $x_i$ , so we can use complicated features.
    - Tends to work better with large ‘n’ or when naïve assumptions aren’t satisfied.
- MLE for generative models maximizes  $p(y, X | w)$ .
- MLE for discriminative models maximizes  $p(y | X, w)$ .
  - So they really do “conditional” MLE.

# “Parsimonious” Parameterization and Linear Models

- Challenge:  $p(y_i | x_i)$  might still be **really complicated**.
  - If  $x_i$  has ‘d’ binary features, need to **estimate  $p(y_i | x_i)$  for  $2^d$  input values**.
- Practical solution: assume  $p(y_i | x_i)$  has a “parsimonious” form.
  - Model with fewer parameters so we need less “coupon collecting”.
- For binary classification, transform linear model with sigmoid:
  - The **linear model  $w^T x_i$**  gives us a number in  $(-\infty, \infty)$ .
  - We’ll **map  $w^T x_i$  to a probability** with the **sigmoid function**.

$$h(z_i) = \frac{1}{1 + \exp(-z_i)}$$



# Sigmoid: Transforming $w^T x_i$ to a Probability

- We'll define  $p(y_i = +1 | z_i) = h(z_i)$ , where 'h' is the sigmoid function.

$$\begin{aligned} \text{So } p(y_i = -1 | z_i) &= 1 - p(y_i = +1 | z_i) \\ &= 1 - h(z_i) \quad \text{can show from} \\ &= h(-z_i) \quad \text{definition of 'h'} \end{aligned}$$

- With  $y_i$  in  $\{-1, +1\}$ , we can write both cases as  $p(y_i | z_i) = h(y_i z_i)$ .
- So we convert  $z_i = w^T x_i$  into "probability of  $y_i$ " using:

$$\begin{aligned} p(y_i | w, x_i) &= h(y_i \underbrace{w^T x_i}_{z_i}) \\ &= \frac{1}{1 + e^{-y_i w^T x_i}} \end{aligned}$$

- MLE with this likelihood is equivalent to minimizing logistic loss.

# MLE Interpretation of Logistic Regression

- For IID regression problems the conditional NLL can be written:

$$\underbrace{-\log(p(y|X, w))}_{NLL} = -\log(\underbrace{\prod_{i=1}^n p(y_i|x_i, w)}_{IID \text{ assumption}}) = -\sum_{i=1}^n \log(p(y_i|x_i, w))$$

*log turns product into sum*

- Logistic regression assumes sigmoid( $w^T x_i$ ) conditional likelihood:

$$p(y_i|x_i, w) = h(y_i w^T x_i) \quad \text{where} \quad h(z_i) = \frac{1}{1 + \exp(-z_i)}$$

- Plugging in the sigmoid likelihood, the NLL is the logistic loss:

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

*(since  $\log(1) = 0$ )*

# MLE Interpretation of Logistic Regression

- We just derived the logistic loss from the perspective of MLE.
  - Instead of “smooth convex approximation of 0-1 loss”, we now have that logistic regression is doing MLE in a probabilistic model.
  - The training and prediction would be the same as before.
    - We still minimize the logistic loss in terms of ‘w’.
  - But MLE viewpoint justifies sigmoid for “probability that e-mail is important”:

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

- And L2-regularized logistic loss would correspond to MAP with a Gaussian prior.

# Multi-Class Logistic Regression

- Previously we talked about **multi-class classification**:
  - We want  $w_{y_i}^T x_i$  to be the most positive among ‘k’ real numbers  $w_c^T x_i$ .
- We have ‘k’ real numbers  $z_c = w_c^T x_i$ , want to map  $z_c$  to probabilities.
- Most common way to do this is with **softmax** function:

$$p(y \mid z_1, z_2, \dots, z_k) = \frac{\exp(z_y)}{\sum_{c=1}^k \exp(z_c)}$$

- Taking  $\exp(z_c)$  makes it non-negative, denominator makes it sum to 1.
- So this gives a probability for each of the ‘k’ possible values of ‘c’.
- The NLL under this likelihood is the softmax loss.

(pause)

# Why do we care about MLE and MAP?

- Unified way of thinking about many of our tricks?
  - Probabilistic interpretation of logistic loss.
  - Laplace smoothing and L2-regularization are doing the same thing.
- Remember our two ways to reduce overfitting in complicated models:
  - Model averaging (ensemble methods).
  - Regularization (linear models).
- “Fully”-Bayesian methods (CPSC 540) combine both of these.
  - Average over all models, weighted by posterior (including regularizer).
  - Can use extremely-complicated models without overfitting.

# Losses for Other Discrete Labels

- MLE/MAP gives loss for classification with basic labels:
  - Least squares and absolute loss for regression.
  - 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 (bonus):
  - Ordinal: {1 star, 2 stars, 3 stars, 4 stars, 5 stars}.
  - Counts: 602 'likes'.
  - Survival rate: 60% of patients were still alive after 3 years.
- Define likelihood of labels, and use NLL as the loss function.
- We can also use ratios of probabilities to define more losses (bonus):
  - Binary SVMs, multi-class SVMs, and “pairwise preferences” (ranking) models.

# End of Part 3: Key Concepts

- Linear models predict based on linear combination(s) of features:

$$w^\top x_i = w_1 x_{i1} + w_2 x_{i2} + \dots + w_d x_{id}$$

- We model non-linear effects using a change of basis:

- Replace d-dimensional  $x_i$  with k-dimensional  $z_i$  and use  $v^\top z_i$ .
  - Examples include polynomial basis and (non-parametric) RBFs.

- Regression is supervised learning with continuous labels.

- Logical error measure for regression is squared error:

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

- Can be solved as a system of linear equations.

# End of Part 3: Key Concepts

- Gradient descent finds local minimum of smooth objectives.
  - Converges to a global optimum for convex functions.
  - Can use smooth approximations (Huber, log-sum-exp)
- Stochastic gradient methods allow huge/infinite ‘n’.
  - Though very sensitive to the step-size.
- Kernels let us use similarity between examples, instead of features.
  - Lets us use some exponential- or infinite-dimensional features.
- Feature selection is a messy topic.
  - Classic method is forward selection based on L0-norm.
  - L1-regularization simultaneously regularizes and selects features.

# End of Part 3: Key Concepts

- We can reduce over-fitting by using regularization:

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

- Squared error is not always right measure:
  - Absolute error is less sensitive to outliers.
  - Logistic loss and hinge loss are better for binary  $y_i$ .
  - Softmax loss is better for multi-class  $y_i$ .
- MLE/MAP perspective:
  - We can view loss as log-likelihood and regularizer as log-prior.
  - Allows us to define losses based on probabilities.

# The Story So Far...

- Part 1: Supervised Learning.
  - Methods based on **counting and distances**.
- Part 2: Unsupervised Learning.
  - Methods based on **counting and distances**.
- Part 3: Supervised Learning (just finished).
  - Methods based on **linear models and gradient descent**.
- Part 4: Unsupervised Learning (next time).
  - Methods based on **linear models and gradient descent**.

# Summary

- Maximum likelihood estimate viewpoint of common models.
  - Objective functions are equivalent to maximizing  $p(y, X | w)$  or  $p(y | X, w)$ .
- MAP estimation directly models  $p(w | X, y)$ .
  - Gives probabilistic interpretation to regularization.
- Discriminative probabilistic models directly model  $p(y_i | x_i)$ .
  - Unlike naïve Bayes that models  $p(x_i | y_i)$ .
  - Usually, we use linear models and define “likelihood” of  $y_i$  given  $w^T x_i$ .
- Discrete losses for weird scenarios are possible using MLE/MAP:
  - Ordinal logistic regression, Poisson regression.
- Next time:
  - What ‘parts’ are your personality made of?

# Discussion: Least Squares and Gaussian Assumption

- Classic **justifications for the Gaussian assumption** underlying least squares:
  - Your **noise might really be Gaussian**. (It probably isn't, but maybe it's a good enough approximation.)
  - The **central limit theorem** (CLT) from probability theory. (If you add up enough IID random variables, the estimate of their mean converges to a Gaussian distribution.)
- I think the CLT justification is wrong as we've never assumed that the  $x_{ij}$  are IID across 'j' values. We only assumed that the examples  $x_i$  are IID across 'i' values, so the CLT implies that our estimate of 'w' would be a Gaussian distribution under different samplings of the data, but this says nothing about the distribution of  $y_i$  given  $w^T x_i$ .
- On the other hand, there are reasons **\*not\*** to use a Gaussian assumption, like it's sensitivity to outliers. This was (apparently) what lead Laplace to propose the Laplace distribution as a more robust model of the noise.
- The "student t" distribution (published anonymously by Gosset while working at the Guiness beer company) is even more robust, but doesn't lead to a convex objective.

# Binary vs. Multi-Class Logistic

- How does multi-class logistic generalize the binary logistic model?
- We can re-parameterize softmax in terms of  $(k-1)$  values of  $z_c$ :

$$p(y|z_1, z_2, \dots, z_{k-1}) = \frac{\exp(z_y)}{1 + \sum_{c=1}^{k-1} \exp(z_c)} \quad \text{if } y \neq k \quad \text{and} \quad p(y|z_1, z_2, \dots, z_{k-1}) = \frac{1}{1 + \sum_{c=1}^{k-1} \exp(z_c)} \quad \text{if } y = k$$

- This is due to the “sum to 1” property (one of the  $z_c$  values is redundant).
- So if  $k=2$ , we don’t need a  $z_2$  and only need a single ‘ $z$ ’.
- Further, when  $k=2$  the probabilities can be written as:

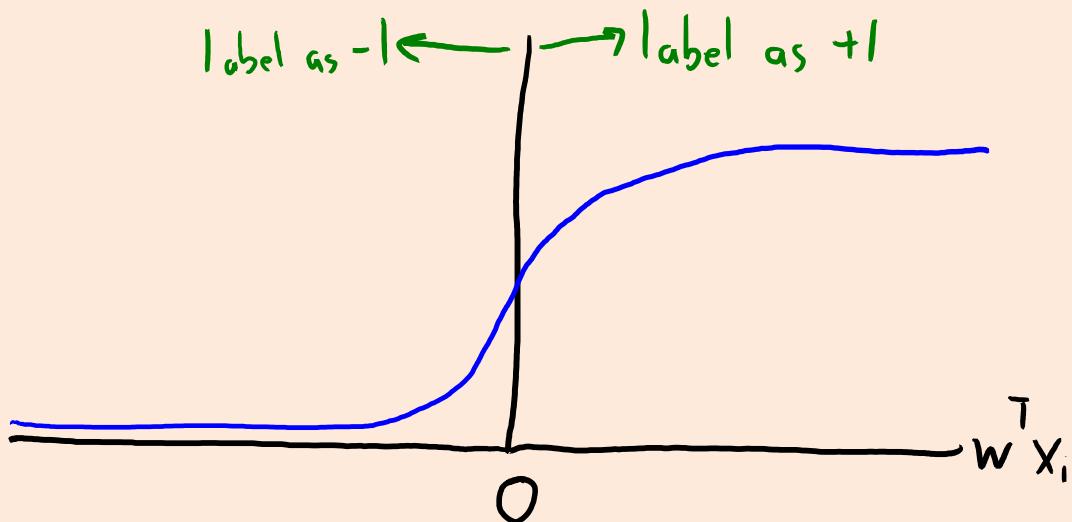
$$p(y=1|z) = \frac{\exp(z)}{1 + \exp(z)} = \frac{1}{1 + \exp(-z)} \quad p(y=2|z) = \frac{1}{1 + \exp(z)}$$

- Renaming ‘2’ as ‘-1’, we get the **binary logistic regression** probabilities.

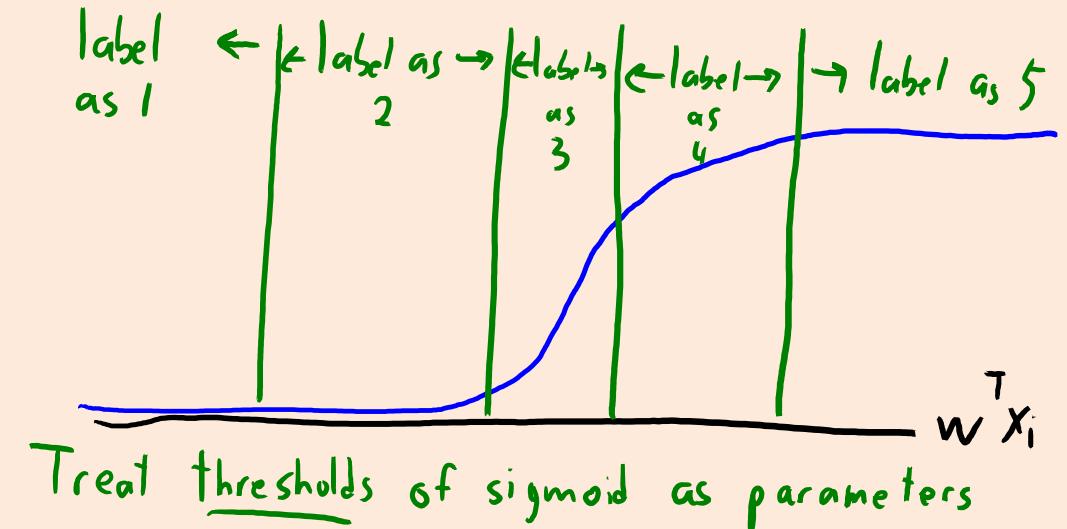
# Ordinal Labels

- **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’.

Logistic regression



Ordinal logistic regression



# Count Labels

- Count data: predict the number of times something happens.
  - For example,  $y_i = "602"$  Facebook likes.
- Softmax requires 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, a lot of people think this isn't the best likelihood.

# Censored Survival Analysis (Cox Partial Likelihood)

- Censored survival analysis:
  - Target  $y_i$  is last time at which we know person is alive.
    - But some people are still alive (so they have the same  $y_i$  values).
    - The  $y_i$  values (time at which they die) are “censored”.
  - We use  $v_i=0$  if they are still alive and otherwise we set  $v_i = 1$ .
- Cox partial likelihood assumes “instantaneous” rate of dying depends on  $x_i$  but not on total time they’ve been alive (not that realistic). Leads to likelihood of the “censored” data of the form:

$$P(y_i, v_i | x_i, w) = \exp(v_i w^\top x_i) \exp(-\sum_j v_j \exp(w^\top x_j))$$

- There are many extensions and alternative likelihoods.

# Other Parsimonious Parameterizations

- Sigmoid isn't the only parsimonious  $p(y_i \mid x_i, w)$ :
  - Probit (uses CDF of normal distribution, very similar to logistic).
  - Noisy-Or (simpler to specify probabilities by hand).
  - Extreme-value loss (good with class imbalance).
  - Cauchit, Gosset, and many others exist...

# Unbalanced Training Sets

- Consider the case of binary classification where your training set has 99% class -1 and **only 1% class +1**.
  - This is called an “**unbalanced**” training set
- Question: is this a problem?
- Answer: it depends!
  - If these **proportions are representative of the test set proportions**, and you care about both types of errors equally, then “no” it’s not a problem.
    - You can get 99% accuracy by just always predicting -1, so ML can only help with the 1%.
  - But it’s a **problem if the test set is not like the training set** (e.g. your data collection process was biased because it was easier to get -1’s)
  - It’s also a **problem if you care more about one type of error**, e.g. if mislabeling a +1 as a -1 is much more of a problem than the opposite
    - For example if +1 represents “tumor” and -1 is “no tumor”

# Unbalanced Training Sets

- This issue comes up a lot in practice!
- How to fix the problem of unbalanced training sets?
  - Common strategy is to build a “**weighted**” model:
    - Put higher weight on the training examples with  $y_i = +1$ .

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

*Make this weight bigger  
for under-represented class*

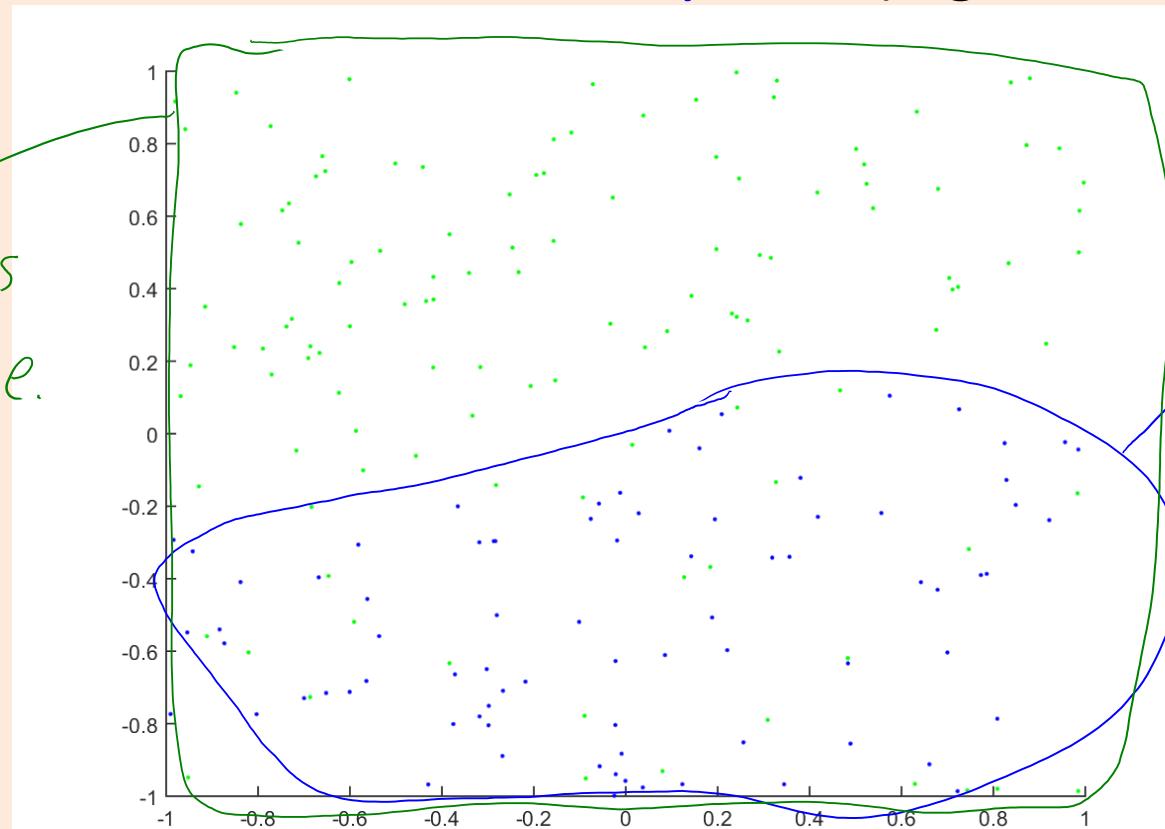
- This is equivalent to replicating those examples in the training set.
- You could also subsample the majority class to make things more balanced.
- Bootstrap: create a dataset of size ‘n’ where  $n/2$  are sampled from +1,  $n/2$  from -1.
- Another approach is to try to make “fake” data to fill in minority class.
- Another option is to change to an **asymmetric loss function** (next slides) that penalizes one type of error more than the other.
- Some discussion of different methods [here](#).

# Unbalanced Data and Extreme-Value Loss

- Consider binary case where:
  - One class overwhelms the other class ('unbalanced' data).
  - Really important to find the minority class (e.g., minority class is tumor).

"majority" class  
is everywhere.

important "minority" class

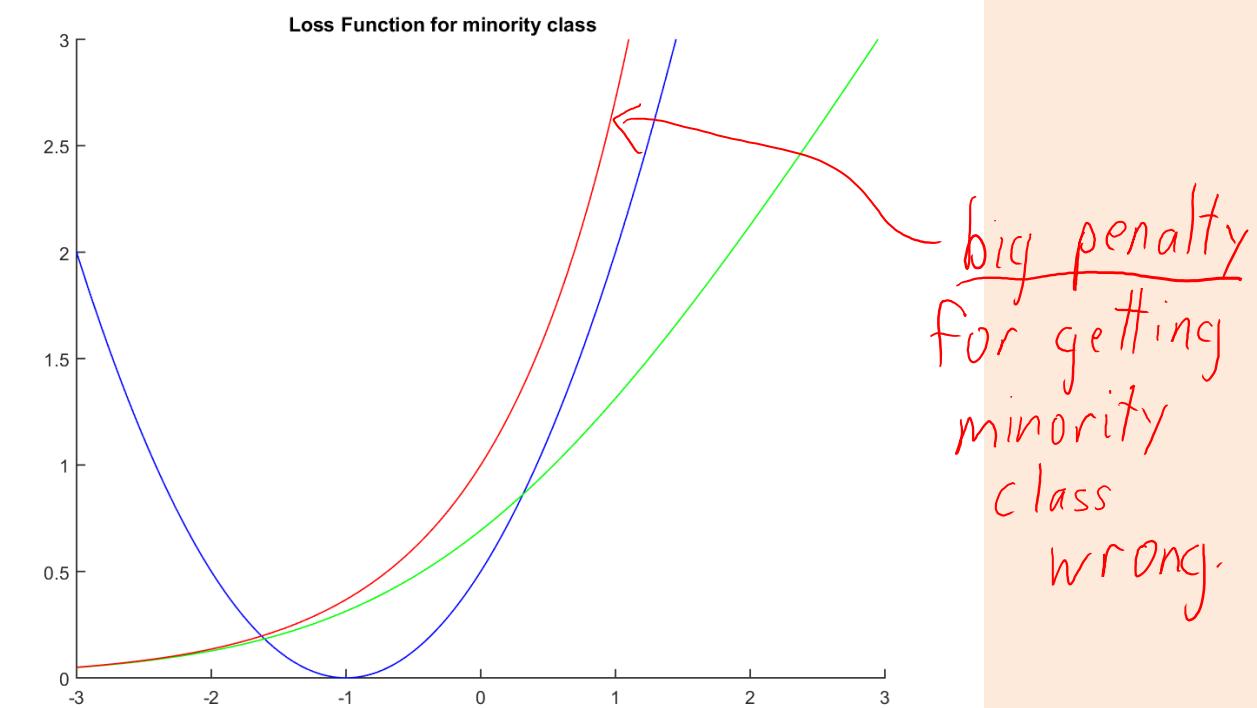
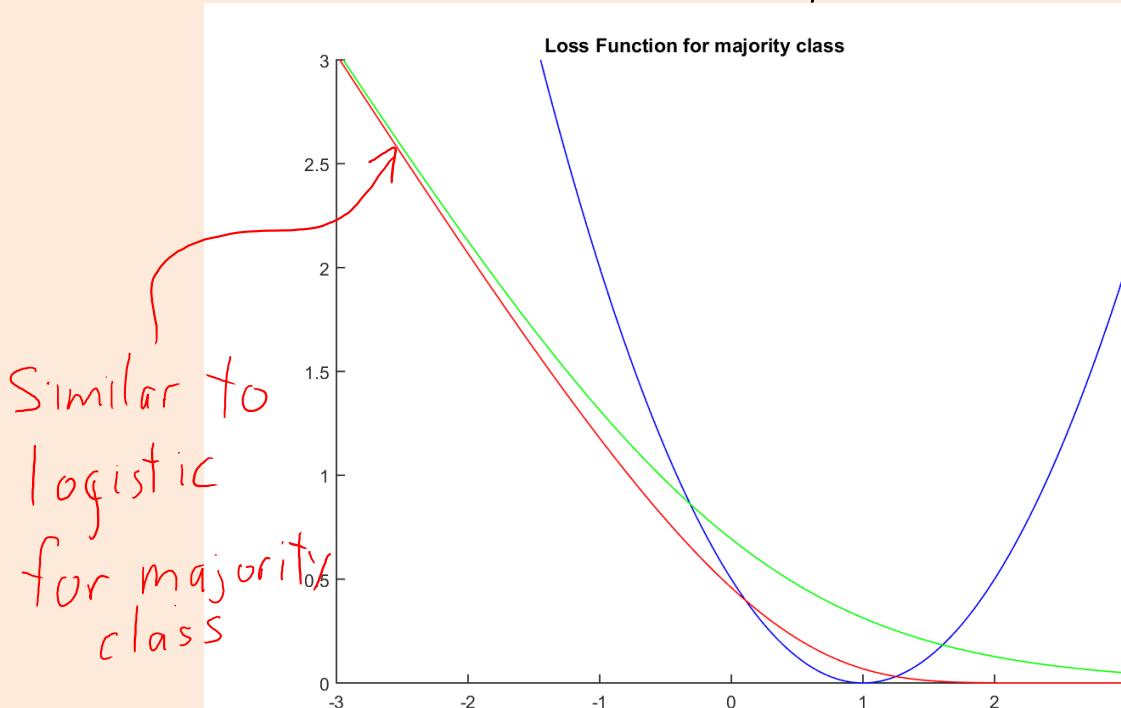


# Unbalanced Data and Extreme-Value Loss

- Extreme-value distribution:

$$p(y_i = +1 | \hat{y}_i) = 1 - \exp(-\exp(\hat{y}_i)) \quad [+1 \text{ is majority class}] \quad \text{asymmetric}$$

To make it a probability,  $p(y_i = -1 | \hat{y}_i) = \exp(-\exp(\hat{y}_i))$

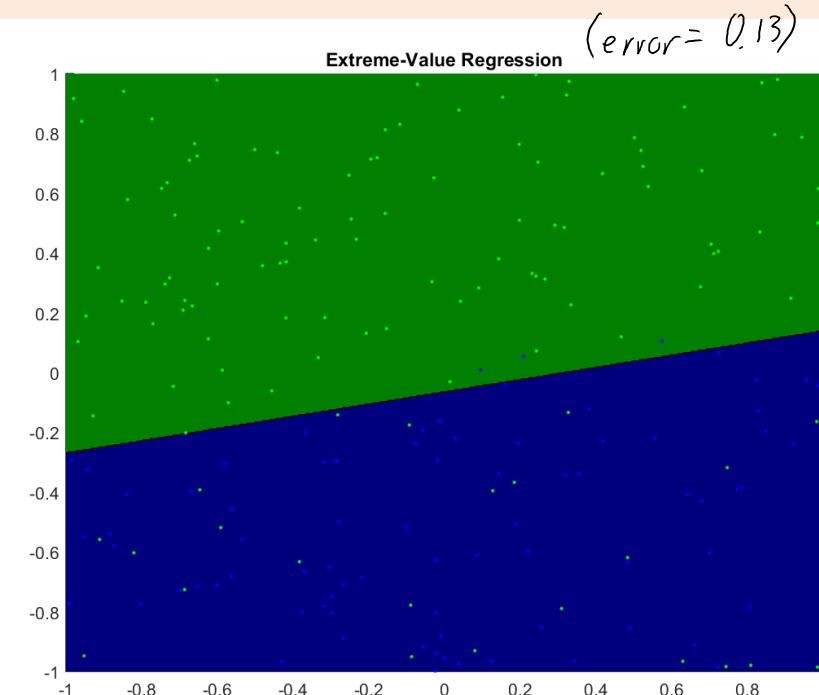
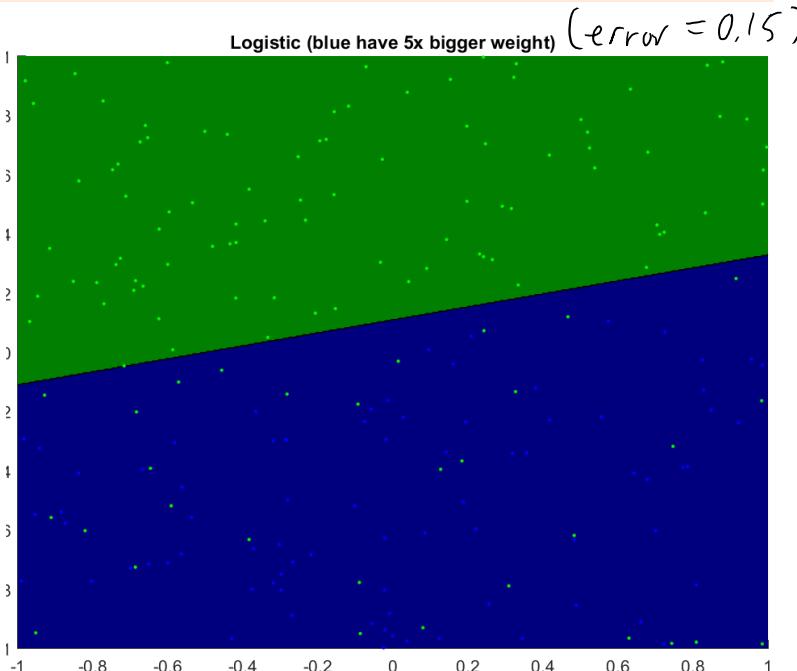
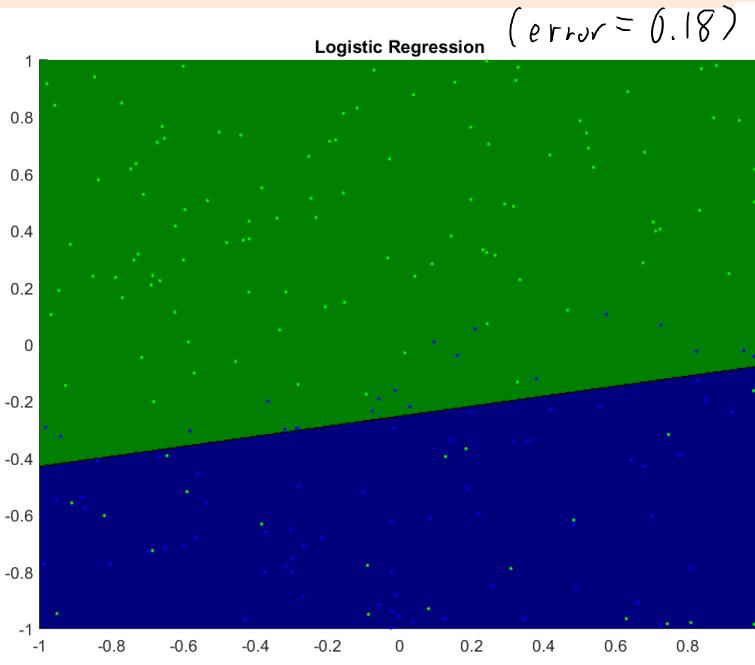


# Unbalanced Data and Extreme-Value Loss

- **Extreme-value distribution:**

$$p(y_i = +1 | \hat{y}_i) = 1 - \exp(-\exp(\hat{y}_i)) \quad [+1 \text{ is majority class}] \quad \xrightarrow{\text{asymmetric}}$$

To make it a probability,  $p(y_i = -1 | \hat{y}_i) = \exp(-\exp(\hat{y}_i))$



# Loss Functions from Probability Ratios

- We've seen that loss functions can come from probabilities:
  - Gaussian => squared loss, Laplace => absolute loss, sigmoid => logistic.
- Most other loss functions can be derived from probability ratios.
  - Example: sigmoid => hinge.

$$p(y_i | x_i, w) = \frac{1}{1 + \exp(-y_i w^T x_i)} = \frac{\exp(\frac{1}{2} y_i w^T y_i)}{\underbrace{\exp(\frac{1}{2} y_i w^T x_i) + \exp(-\frac{1}{2} y_i w^T x_i)}_{\text{Same normalizing constant}} \propto \exp(\frac{1}{2} y_i w^T x_i)}$$

# Loss Functions from Probability Ratios

- We've seen that loss functions can come from probabilities:
  - Gaussian => squared loss, Laplace => absolute loss, sigmoid => logistic.
- Most other loss functions can be derived from probability ratios.
  - Example: sigmoid => hinge.

$$p(y_i | x_i, w) \propto \exp\left(\frac{1}{2} y_i w^T x_i\right)$$

To classify  $y_i$  correctly, it's sufficient to have  $\frac{p(y_i | x_i, w)}{p(-y_i | x_i, w)} > \beta$  for some ' $\beta$ ' > 1

Notice that normalizing constant doesn't matter:

$$\frac{\exp\left(\frac{1}{2} y_i w^T x_i\right)}{\exp\left(-\frac{1}{2} y_i w^T x_i\right)} > \beta$$

# Loss Functions from Probability Ratios

- We've seen that loss functions can come from probabilities:
  - Gaussian => squared loss, Laplace => absolute loss, sigmoid => logistic.
- Most other loss functions can be derived from probability ratios.
  - Example: sigmoid => hinge.

$$p(y_i | x_i, w) \propto \exp\left(\frac{1}{2} y_i w^T x_i\right)$$

We need:  $\frac{\exp\left(\frac{1}{2} y_i w^T x_i\right)}{\exp\left(-\frac{1}{2} y_i w^T x_i\right)} \geq \beta$

Take  $\log$ :

$$\log\left(\frac{\exp\left(\frac{1}{2} y_i w^T x_i\right)}{\exp\left(-\frac{1}{2} y_i w^T x_i\right)}\right) \geq \log(\beta) \iff \frac{1}{2} y_i w^T x_i + \frac{1}{2} y_i w^T x_i \geq \log(\beta)$$

$$y_i w^T x_i \geq 1 \quad (\text{if we choose } \log(\beta) = 1)$$

# Loss Functions from Probability Ratios

- We've seen that loss functions can come from probabilities:
  - Gaussian => squared loss, Laplace => absolute loss, sigmoid => logistic.
- Most other loss functions can be derived from probability ratios.
  - Example: sigmoid => hinge.

$$p(y_i | x_i, w) \propto \exp\left(\frac{1}{2} y_i w^T x_i\right)$$

We need:  $\frac{\exp(\frac{1}{2} y_i w^T x_i)}{\exp(-\frac{1}{2} y_i w^T x_i)} \geq \beta$

Or equivalently:

$$y_i w^T x_i \geq 1 \quad (\text{for } \beta = \exp(1))$$

Define a loss function by amount of constraint violation:

$$\max \{ 0, 1 - y_i w^T x_i \}$$

when  $1 - y_i w^T x_i \leq 0$  when  $1 - y_i w^T x_i \geq 0$

We get SVMs by looking at regularized average loss:

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

# Loss Functions from Probability Ratios

- General approach for defining losses using probability ratios:
  - Define constraint based on probability ratios.
  - Minimize violation of logarithm of constraint.
- Example: softmax => multi-class SVMs.

Assume:  $p(y_i = c | x_i, w) \propto \exp(w_c^\top x_i)$

Want:  $\frac{p(y_i = c | x_i, w)}{p(y_i = c' | x_i, w)} \geq \beta$  for all  $c'$  and some  $\beta > 1$

For  $\beta = \exp(1)$  equivalent to

$$w_{y_i}^\top x_i - w_c^\top x_i \geq 1 \quad \text{for all } c' \neq y_i$$

Option 1: penalize all violations:

$$\sum_{c'=1}^K \max\{0, 1 - w_{y_i}^\top x_i + w_c^\top x_i\}$$

Option 2: penalize only max violation:

$$\max_{c' \neq c} \left\{ \max\{0, 1 - w_{y_i}^\top x_i + w_c^\top x_i\} \right\}$$

# Supervised Ranking with Pairwise Preferences

- Ranking with pairwise preferences:

- We aren't given any explicit  $y_i$  values.

- Instead we're given list of objects  $(i,j)$  where  $y_i > y_j$ .

Assume  $p(y_i | X, w) \propto \exp(w^T x_i)$  is probability that object ' $i$ ' has highest rank.

Want:  $\frac{p(y_i | X, w)}{p(y_j | X, w)} \geq \beta$  for all preferences  $(i,j)$

For  $\beta = \exp(1)$  equivalent to

$$w^T x_i - w^T x_j \geq 1$$

for preferences  $(i,j)$

→ We can use  $f(w) = \sum_{(i,j) \in R} \max\{0, 1 - w^T x_i + w^T x_j\}$

This approach can also be used to define losses for total/partial orderings. (but this information is hard to get)