CPSC 340:
Machine Learning and Data Mining

MAP Estimation
Fall 2018
Admin

• Assignment 4:
  – Due tonight.

• Assignment 5:
  – Out early next week.
Last Time: Maximum Likelihood Estimation (MLE)

- **Maximum likelihood estimation (MLE):**
  - Define a likelihood function, probability of data given parameters: $p(D \mid w)$.
  - Choose parameters ‘$w$’ to maximize the likelihood.

- Gives naïve Bayes “counting” estimates we used.

- Typically easier to equivalently minimize negative log-likelihood (NLL).
  - Turns product of probability over IID examples into sum over examples.

- We showed that least squares is MLE with Gaussian errors.
  - Other likelihoods give other models: Laplace noise $\rightarrow$ L1-norm loss.
Maximum Likelihood Estimation and Overfitting

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

\[ \hat{w} \in \arg \max_w \{ p(D|w) \} \]

• 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:

\[ \hat{w} \in \text{argmax}_w \{ p(w \mid D) \} \]

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

• MLE and MAP are connected by Bayes rule:

\[ \frac{p(w \mid D)}{p(D)} \propto p(D \mid w) p(w) \]

  \( \text{posterior} \) \( \text{likelihood} \) \( \text{prior} \)

• So MAP maximizes the likelihood \( p(D \mid w) \) 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:
  $$\hat{w} \in \arg\max_w \left\{ \frac{p(w|D)}{p(D)} \right\} \equiv \arg\max_w \left\{ \prod_{i=1}^n p(D_i|w)p(w) \right\}$$

• By again taking the negative of the logarithm as before we get:
  $$\hat{w} \in \arg\min_w \left\{ -\sum_{i=1}^n \log p(D_i|w) - \log p(w) \right\}$$

  - Many regularizers are equivalent to negative log-priors.

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

- This implies that:
  \[ -\log(p(w)) = -\log\left(\exp\left(-\frac{\lambda}{2} \|w\|^2\right)\right) + \text{(constant)} = \frac{\lambda}{2} \|w\|^2 + \text{(constant)} \]

- So we have that:
  \[ \hat{w} \in \arg\min_{w} \sum_{i=1}^{n} -\log(p(y_i|X,w)) - \log(p(w)) = \arg\min_{w} \left\{ \sum_{i=1}^{n} \log(p(y_i|X,w)) + \frac{\lambda}{2} \|w\|^2 \right\} \]
MAP Estimation and Regularization

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

\[
\text{If } p(y_i | x_i, w) \propto \exp\left(-\frac{(w^T x_i - y_i)^2}{2}\right) \quad p(w_j) \propto \exp\left(-\frac{\sigma^2}{2} w_j^2\right)
\]

then MAP estimation is equivalent to minimizing

\[
f(w) = \frac{1}{2} \|Xw - y\|^2 + \frac{\sigma^2}{2} \|w\|^2
\]

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

\[
\text{If } p(y_i | x_i, w) \propto \exp\left(-|w^T x_i - y_i|\right) \quad p(w) \propto \exp\left(-\frac{\sigma^2}{2} w_j^2\right)
\]

then MAP estimation is equivalent to minimizing

\[
f(w) = \|Xw - y\|^2 + \frac{\sigma^2}{2} \|w\|^2
\]

– 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: $\frac{\text{count}(\text{spam, lactase})}{\text{count}(\text{spam})}$.
  • But this caused problems if $\text{count}(\text{spam},\text{lactase}) = 0$.

• Our solution was Laplace smoothing:
  – Add “+1” to our estimates: $(\text{count}(\text{spam, lactase}) + 1)/(\text{counts}(\text{spam}) + 2)$.
  – This corresponds to a “Beta” prior so Laplace smoothing is a regularizer.
(pause)
Previously: Identifying Important E-mails

• Recall problem of identifying ‘important’ e-mails:

• We can do binary classification by taking sign of linear model:
  \[ \hat{y}_i = \text{sign}(w^T x_i) \]
  – Convex loss functions (hinge loss, logistic loss) let us find an appropriate ‘w’.

• Global/local features in linear models give personalized prediction.

• We can train on huge datasets like Gmail with stochastic gradient.

• But what if we want a probabilistic classifier?
  – Want a model of \( p(y_i = \text{“important”} \mid x_i) \) for use in decision theory.
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: \( p(y_i \mid x_i) \) might still be really complicated.
  – If \( x_i \) has ‘d’ binary features, need to estimate \( p(y_i \mid x_i) \) for \( 2^d \) input values.

• Practical solution: assume \( p(y_i \mid x_i) \) has a “parsimonious” form.
  – Model with fewer parameters so we need less “coupon collecting”.
  – Typically, we transform output of a linear model to be a probability.
    • So we only need to estimate the parameters of a linear model.

• Most common example is binary logistic regression:
  1. The linear prediction \( w^T x_i \) gives us a number in \((-\infty, \infty)\).
  2. We’ll map \( w^T x_i \) to a number in \([0,1]\), with a map acting like a probability.
How should we transform \( w^T x_i \) into a probability?

• Let \( z_i = w^T x_i \) in a binary logistic regression model:
  – If \( \text{sign}(z_i) = +1 \), we should have \( p(y_i = +1 \mid z_i) > \frac{1}{2} \).
    • The linear model thinks \( y_i = +1 \) is more likely.
  – If \( \text{sign}(z_i) = -1 \), we should have \( p(y_i = +1 \mid z_i) < \frac{1}{2} \).
    • The linear model thinks \( y_i = -1 \) is more likely.
    • Remember that \( p(y_i = -1 \mid z_i) = 1 - p(y_i = +1 \mid z_i) \).
  – If \( z_i = 0 \), we should have \( p(y_i = +1 \mid z_i) = \frac{1}{2} \).
    • Both classes are equally likely.

• And we might want size of \( w^T x_i \) to affect probabilities:
  – As \( z_i \) becomes really positive, we should have \( p(y_i = +1 \mid z_i) \) converge to 1.
  – As \( z_i \) becomes really negative, we should have \( p(y_i = +1 \mid z_i) \) converge to 0.
So we want a transformation of $z_i = w^T x_i$ that looks like this:

The most common choice is the **sigmoid function**:

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

Values of $h(z_i)$ match what we want:

<table>
<thead>
<tr>
<th>$z_i$</th>
<th>$h(z_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-\infty$</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>$\approx 0.27$</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>0.5</td>
<td>$\approx 0.62$</td>
</tr>
<tr>
<td>1</td>
<td>$\approx 0.73$</td>
</tr>
<tr>
<td>$+\infty$</td>
<td>1</td>
</tr>
</tbody>
</table>
Sigmoid: Transforming $w^T x_i$ to a Probability

- We’ll define $p(y_i = +1 \mid z_i) = h(z_i)$, where ‘$h$’ is the sigmoid function.

$$p(y_i = -1 \mid z_i) = 1 - p(y_i = +1 \mid z_i)$$

$$= 1 - h(z_i)$$

$$= h(-z_i) \quad \text{(definition of ‘$h$’)}$$

- We can write both cases as $p(y_i \mid z_i) = h(y_i z_i)$.

- So we convert $z_i = w^T x_i$ into “probability of $y_i$” using:

$$p(y_i \mid w, x) = h(y_i w^T x)$$

$$= \frac{1}{1 + e^{yp(-y_i w^T x)}}$$

- 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:

\[
- \log p(y \mid X, w) = - \log \left( \prod_{i=1}^{n} p(y_i \mid x_i, w) \right) = - \sum_{i=1}^{n} \log p(y_i \mid x_i, w)
\]

• Logistic regression assumes sigmoid(\(w^T x_i\)) conditional likelihood:

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

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

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

(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 gives us “probability that e-mail is important”:

\[
\pi(y_1 | x_1, w) = \frac{1}{1 + \exp(-y_1 w^T x_i)}
\]

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

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

  - 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.
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 (CPSC 540) methods 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:
  \[ w^T x_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id} \]

• We model non-linear effects using a **change of basis**:
  – Replace d-dimensional \( x_i \) with k-dimensional \( z_i \) and use \( v^T z_i \).
  – Examples include **polynomial basis** and (non-parametric) **RBFs**.

• **Regression** is supervised learning with continuous labels.
  – Logical error measure for regression is **squared error**:
    \[ f(w) = \frac{1}{2} \| X w - y \|_2^2 \]
  – 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

• **MAP estimation** directly models \( p(w \mid X, y) \).
  – Gives probabilistic interpretation to regularization.

• **Discriminative probabilistic models** directly model \( p(y_i \mid x_i) \).
  – Unlike naïve Bayes that models \( p(x_i \mid y_i) \).
  – 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 Guinness 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 \(z_c\):

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

  - 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:

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

  - 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

$\text{Logistic regression}$

$\text{Ordinal logistic regression}$

$\text{Treat thresholds of sigmoid as parameters}$
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:

  \[
  p(y_i, v_i | x_i, w) = \exp(v_i w^T x_i) \exp(-w^T x_i) 
  \]

- 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 really 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?
  – One way is to build a “weighted” model, like you did with weighted least squares in your assignment (put higher weight on the training examples with $y_i=+1$)
    • This is equivalent to replicating those examples in the training set.
    • You could also subsample the majority class to make things more balanced.
  – Another approach is to try to make “fake” data to fill in minority class.
  – Another option is to change to an asymmetric loss function that penalizes one type of error more than the other.
  – There is 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).
Unbalanced Data and Extreme-Value Loss

- Extreme-value distribution:

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

To make it a probability,

\[ p(y_i = -1 | \hat{y}_i) = \exp(-\exp(\hat{y}_i)) \]

Similar to logistic for majority class

Loss Function for majority class ($y = +1$)

Loss Function for minority class ($y = +1$)

Big penalty for getting minority class wrong.
Unbalanced Data and Extreme-Value Loss

- **Extreme-value distribution:**

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

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

![Logistic Regression](error=0.18)

![Logistic (blue have 5x bigger weight)](error=0.15)

![Extreme-Value Regression](error=0.13)
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.

\[
\rho(y_i \mid x_i, w) = \frac{1}{1 + \exp(-y_i w^T x_i)} = \frac{\exp(\frac{1}{2} y_i w^T x_i)}{\exp(\frac{1}{2} y_i w^T x_i) + \exp(-\frac{1}{2} y_i w^T x_i)} \propto \exp(\frac{1}{2} y_i w^T x_i)
\]

Same normalizing constant for \( y_i = +1 \) and \( x_i = -1 \)
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.

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

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

Notice that normalizing constant doesn't matter:
\[
\frac{\exp\left(\frac{1}{2} y_i w^T x_i\right)}{\exp\left(-\frac{1}{2} y_i w^T x_i\right)} \geq \beta
\]
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:

\[ \frac{\exp\left(\frac{1}{2} y_i w^T x_i \right)}{\exp\left(-\frac{1}{2} y_i w^T x_i \right)} \geq \beta \]

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:

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

\[ \begin{align*}
\text{when } & 1 - y_i w^T x_i \leq 0 \\
\text{when } & 1 - y_i w^T x_i > 0
\end{align*} \]

We get SVMs by looking at regularized average loss:

\[ f(w) = \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y_i w^T x_i\} + \frac{\alpha}{2} \|w\|^2 \]
Loss Functions from Probability Ratios

• General approach for defining losses using probability ratios:
  1. Define constraint based on probability ratios.

• Example: softmax => multi-class SVMs.

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

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

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

$w_{y_i}^T x_i - w_c^T x_i \geq 1$

for all $c' \neq y_i$

Option 1: penalize all violations:

$\sum_{c'=1}^{K} \max \{ \max_{c} \{ 0, 1 - w_{y_i}^T x_i + w_c^T x_i \} \}

Option 2: penalize only max violation:

$\max \{ \max_{c} \{ 0, 1 - w_{y_i}^T x_i + w_c^T x_i \} \}$
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 $p(y_i | X, w) \propto \exp(w^\top x_i)$ is probability that object ‘i’ has highest rank.

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

For $\beta = \exp(1)$ equivalent to \[ w^\top x_i - w^\top x_j \geq 1 \]

for preferences (i,j)  

\[ f(w) = \sum_{(i,j) \in P} \max \{ 0, 1 - w^\top x_i + w^\top x_j \} \]

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