CPSC 340: Machine Learning and Data Mining

MLE and MAP

Fall 2017

Admin

- Assignment 3:
  – 1 late day to hand in tonight, 2 late days for Wednesday.

- Assignment 4:
  – Due Friday of next week.

Last Time: Multi-Class Linear Classifiers

- We discussed multi-class linear classification: y_i in {1,2,…,k}.
- One vs. all with +1/-1 binary classifier:
  - Train weights w_c to predict +1 for class 'c', -1 otherwise.

$$W = \underbrace{\begin{bmatrix} \text{—— } w_1^T \text{ ——} \\ \text{—— } w_2^T \text{ ——} \\ \vdots \\ \text{—— } w_k^T \text{ ——} \end{bmatrix}}_{d} \Big\} k$$

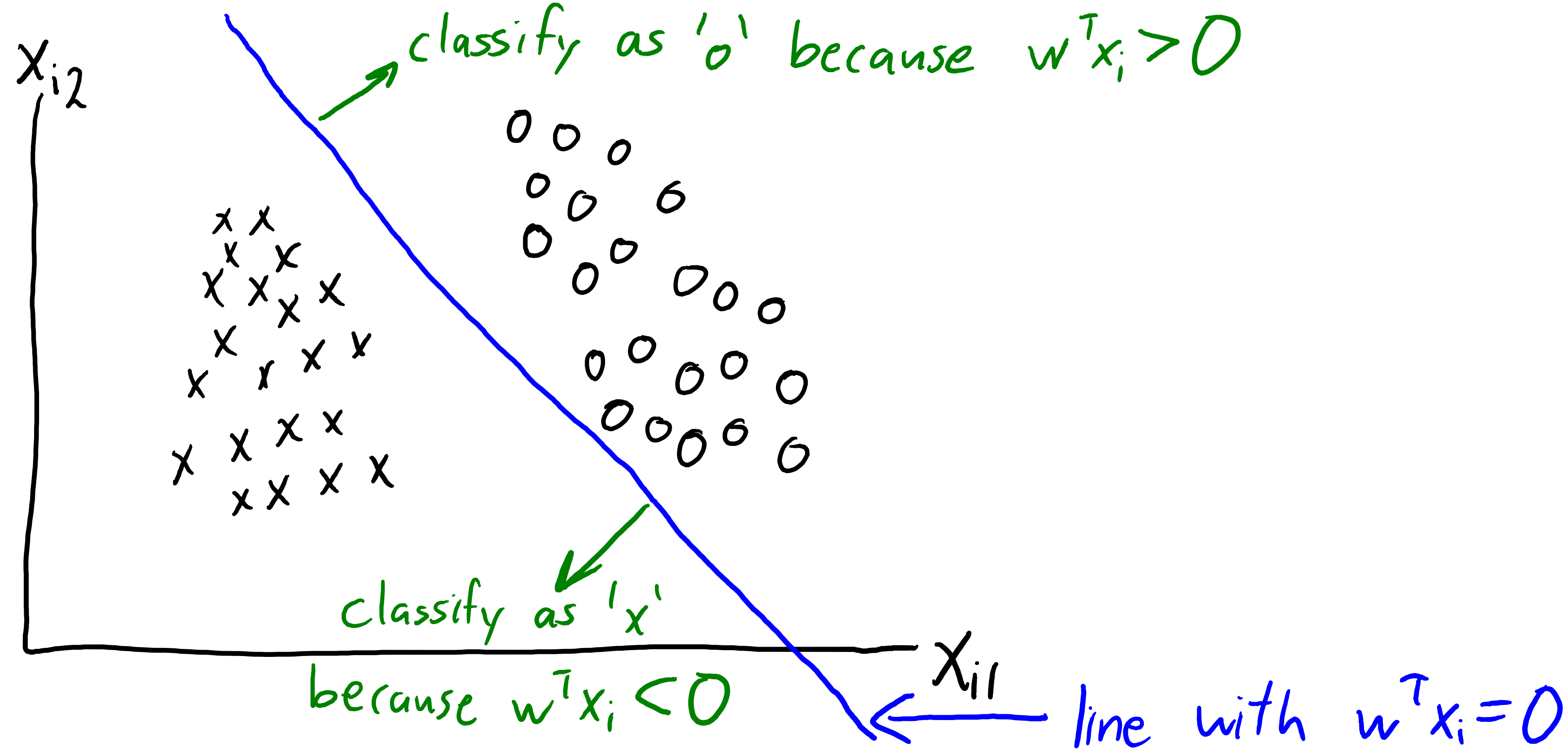
  - Predict by taking 'c' maximizing $w_c^T x_i$.
- Multi-class SVMs and multi-class logistic regression:
  - Train the w_c jointly to encourage maximum $w_c^T x_i$ to be $w_{y_i}^T x_i$.

$$f(W) = \sum_{i=1}^{n}\left[-w_{y_i}^T x_i + \log\left(\sum_{c=1}^{k} \exp(w_c^T x_i)\right)\right] + \frac{\lambda}{2}\|W\|_F^2$$

"Frobenius" norm

Shape of Decision Boundaries

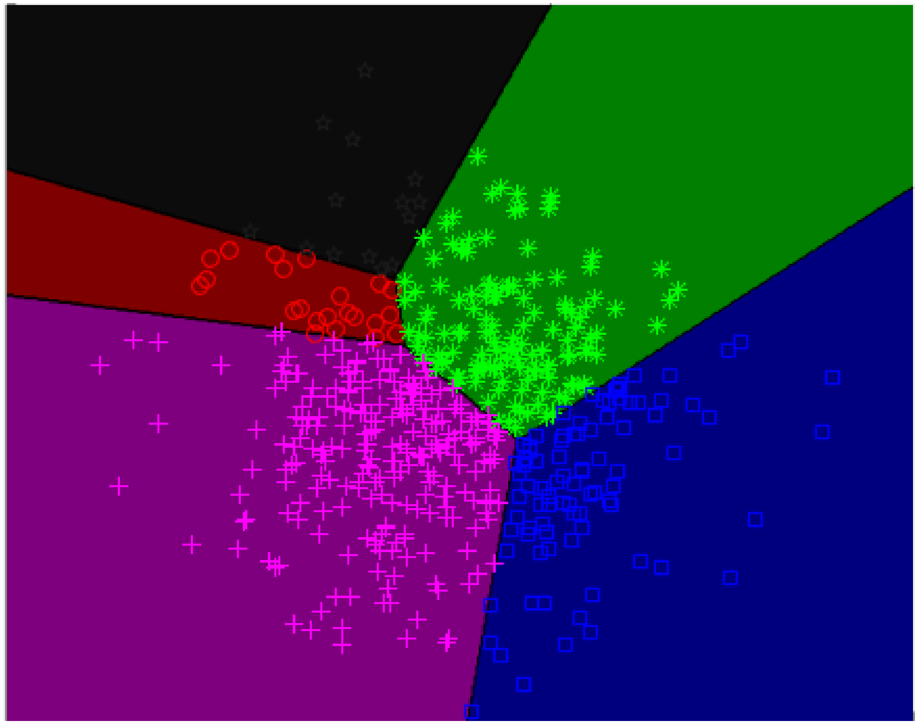
- Recall that a binary linear classifier splits space using a hyper-plane:



- Divides x_i space into 2 "half-spaces".

Shape of Decision Boundaries

- Multi-class linear classifier is intersection of these "half-spaces":
  - This divides the space into convex regions (like k-means):



"Blue" region is region where we have:

$w_{blue}^T x_i \geq w_{green}^T x_i$

$w_{blue}^T x_i \geq w_{magenta}^T x_i$

$w_{blue}^T x_i \geq w_{red}^T x_i$

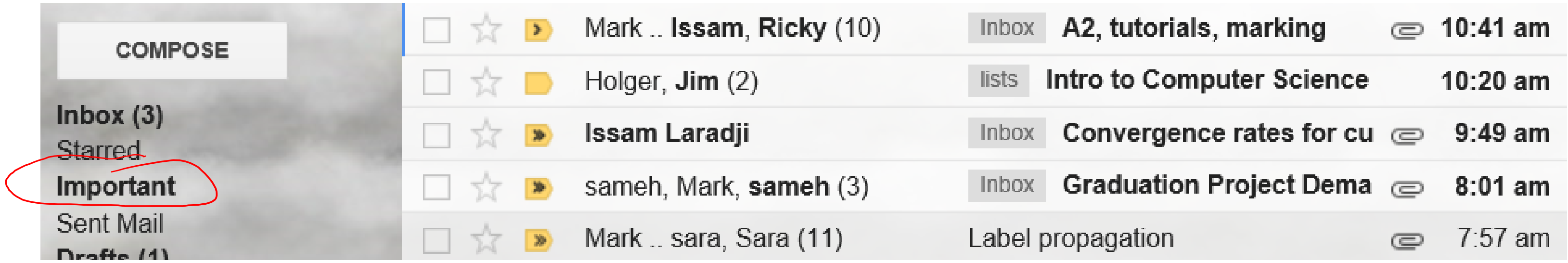
$w_{blue}^T x_i \geq w_{black}^T x_i$

  - Could be non-convex with kernels or change of basis.

(pause)

Previously: Identifying Important E-mails

- Recall problem of identifying 'important' e-mails:



- Global/local features in linear models give personalized prediction.
- We can do binary classification by taking sign of linear model:

$$y_i = \text{sign}(w^T x_i)$$

  - Convex loss functions (hinge loss, logistic loss) let us find an appropriate 'w'.
- We can train on huge datasets like Gmail with stochastic gradient.
- But what if we want a probabilistic classifier?
  - Want a model of $p(y_i = \text{"important"} \mid x_i)$.

Generative vs. Discriminative Models

- Previously we saw naïve Bayes:
  - Uses Bayes rule and model $p(x_i|y_i)$ to predict $p(y_i \mid x_i)$.

$$p(y_i \mid x_i) \propto p(x_i \mid y_i) p(y_i)$$

  - This strategy is called a generative model.
    - It "models how the features are generated".
    - Often works well with lots of features but small 'n'.
- Alternative is discriminative models:
  - Directly model $p(y_i \mid x_i)$ to predict $p(y_i \mid x_i)$.
    - 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.
  - Classic example is logistic regression.

"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 "parsimonious" form.
  - For example, we convert output of linear model to be a probability.
    - Only need to estimate the parameters of a linear model.

- In binary logistic regression, we'll do the following:
  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.

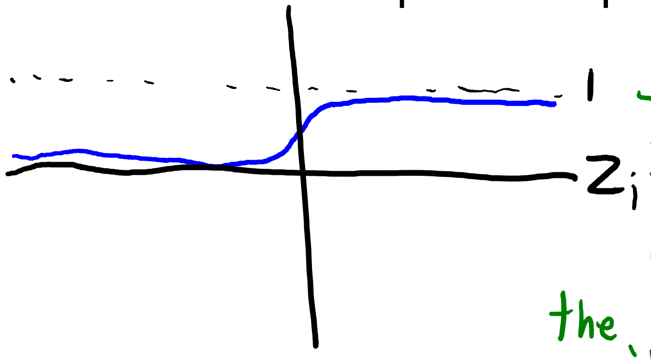
How should we transform w^Tx_i into a probability?

- Let $z_i = w^Tx_i$ in a binary logistic regression model:
  - If $\text{sign}(z_i) = +1$, we should have $p(y_i = +1 \mid z_i) > ½$.
    - The linear model thinks $y_i = +1$ is more likely.
  - If $\text{sign}(z_i) = -1$, we should have $p(y_i = +1 \mid z_i) < ½$.
    - The linear model thinks $y_i = -1$ is more likely, and $p(y_i = -1 \mid z_i) = 1 - p(y_i = +1 \mid z_i)$.
  - If $z_i = 0$, we should have $p(y_i = +1 \mid z_i) = ½$.
    - Both classes are equally likely.

- And we might want size of w^Tx_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.

Sigmoid Function

- So we want a transformation of $z_i = w^T x_i$ that looks like this:



Gives values in the range (0,1) that we'll interpret as probabilities

- 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:

$h(-\infty) = 0$ $h(-1) \simeq 0.27$ $h(0) = 0.5$ $h(0.5) \simeq 0.62$ $h(+1) \simeq 0.73$ $h(+\infty) = 1$

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.

$$\text{So } p(y_i = -1 \mid z_i) = 1 - p(y_i = +1 \mid z_i)$$
$$= 1 - h(z_i)$$
$$= h(-z_i)$$

can show from definition of 'h'

- We can write both cases as $p(y_i \mid z_i) = h(y_i z_i)$, so we convert $z = w^T x_i$ into "probability of y_i" using:

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

- Given this probabilistic perspective, how should we find best 'w'?

Maximum Likelihood Estimation (MLE)

- Maximum likelihood estimation (MLE) for fitting probabilistic models.
  - We have a dataset D.
  - We want to pick parameters ‘w’.
  - We define the likelihood as a probability mass/density function p(D | w).
  - We choose the model $\hat{w}$ that maximizes the likelihood:

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

- Appealing “consistency” properties as n goes to infinity (take STAT 4XX).

Minimizing the Negative Log-Likelihood (NLL)

- To maximize likelihood,

  usually we minimize the negative "log-likelihood" (NLL):
  - "Log-likelihood" is short for "logarithm of the likelihood".

$$\hat{w} \in \underset{w}{\operatorname{argmax}} \{ p(D \mid w) \} \equiv \underset{w}{\operatorname{argmin}} \{ -\log(p(D \mid w)) \}$$

"equivalent"

- Why are these equivalent?
  – Logarithm is monotonic: if $\alpha > \beta$, then $\log(\alpha) > \log(\beta)$.
  – Changing sign flips max to min.

- See "Max and Argmax" notes on webpage if this seems strange.

Minimizing the Negative Log-Likelihood (NLL)

- We use logarithm 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 $p(D \mid w) = \prod_{i=1}^{n} \underbrace{p(D_i, w)}_{\text{likelihood of example 'i'}}$

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

MLE for Naïve Bayes

- A long time ago, I mentioned that we used MLE in naïve Bayes.
- We estimated that $p(y_i = \text{"spam"})$ as count(spam)/count(e-mails).
  - You derive this by minimizing the NLL under a "Bernoulli" likelihood.
  - Set derivative of NLL to 0, and solve for Bernoulli parameter.
- MLE of $p(x_{ij} \mid y_i = \text{"spam"})$ gives count(spam,x_{ij})/count(spam).
  - Also derived under a conditional "Bernoulli" likelihood.
- The derivation is tedious, but if you're interested I put it [here](#).

MLE for Supervised Learning

- The MLE in generative models (like naïve Bayes) maximizes:

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

- But discriminative models directly model p(y | X, w).
  - We treat features X as fixed don’t care about their distribution.
  - So the MLE maximizes the conditional likelihood:

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

of the targets ‘y’ given the features ‘X’ and parameters ‘w’.

MLE Interpretation of Logistic Regression

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

$$\underbrace{-\log(p(y \mid X, w))}_{\text{NLL}} = -\log\Big(\underbrace{\prod_{i=1}^{n} p(y_i \mid x_i, w)}_{\text{IID assumption}}\Big) = -\underbrace{\sum_{i=1}^{n}}_{\substack{\text{log turns} \\ \text{product into sum}}} \log(p(y_i \mid x_i, w))$$

- Logistic regression assumes sigmoid(w^Tx_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:

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

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

MLE Interpretation of Logistic Regression

- We just derived the logistic loss from the perspective of MLE.
  - Instead of "smooth 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":

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

Least Squares is Gaussian MLE

- It turns out that most objectives have an MLE interpretation:
  - For example, consider minimizing the squared error:

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

  - This is MLE of a linear model under the assumption of IID Gaussian noise:

$$y_i = w^T x_i + \varepsilon_i$$

where each ε_i is sampled independently from standard normal

    - "Gaussian" is another name for the "normal" distribution.
  - Remember that least squares solution is called the "normal equations".

Least Squares is Gaussian MLE (Gory Details)

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

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

- Finding MLE is equivalent to minimizing NLL:

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

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

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

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

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

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

Loss Functions and Maximum Likelihood Estimation

- So least squares is MLE under Gaussian likelihood.

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

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

- With a Laplace likelihood you would get absolute error.

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

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

- With sigmoid likelihood we got the binary logistic loss.
- You can derive softmax loss from the softmax likelihood (bonus).

(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:

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

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

- MLE and MAP are connected by Bayes rule:

$$\underbrace{p(w \mid D)}_{\text{posterior}} = \frac{p(D \mid w)\,p(w)}{p(D)} \propto \underbrace{p(D \mid w)}_{\text{likelihood}}\,\underbrace{p(w)}_{\text{prior}}$$

- 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:

$$\hat{w} \in \underset{w}{\text{argmax}} \left\{ p(w \mid D) \right\} \equiv \underset{w}{\text{argmax}} \left\{ \prod_{i=1}^{n} \left[p(D_i \mid w) \right] p(w) \right\}$$

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

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

- 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 variance $1/\lambda$

- This implies that:

$$p(w) \underset{\text{independence}}{=} \prod_{j=1}^{d} p(w_j) \underset{\text{Gaussian assumption}}{\propto} \prod_{j=1}^{d} \exp\left(-\frac{\lambda}{2} w_j^2\right) \underset{e^{\alpha}e^{\beta} = e^{\alpha+\beta}}{=} \exp\left(-\frac{\lambda}{2}\sum_{j=1}^{d} w_j^2\right)$$

- 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

$$\hat{w} \in \operatorname{argmin}\left\{-\log(p(y \mid X, w)) - \log(p(w))\right\} \equiv \underset{w}{\operatorname{argmin}}\left\{-\sum_{i=1}^{n}\left[\log(p(y_i \mid x_i, w))\right] + \frac{\lambda}{2}\|w\|^2\right\}$$

MAP Estimation and Regularization

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

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

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

  - Sigmoid likelihood and Gaussian prior give 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 $f(w) = \underbrace{\sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))}_{\text{As } n \to \infty \text{ effect of prior/regularizer goes to } 0} + \frac{\lambda}{2}\|w\|^2$

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('lactase' = 1| 'spam') is: count(spam,lactase)/count(spam).
  - But this caused problems if count(spam,lactase) = 0.

- Our solution was Laplace smoothing:
  - Add "+1" to our estimates: (count(spam,lactase)+1)/(counts(spam)+2).
  - This corresponds to a "Beta" prior so Laplace smoothing is a regularizer.

Why do we care about MLE and MAP?

- Unified way of thinking about many of our tricks?
  - Laplace smoothing and L2-regularization are doing the same thing.

- Remember our two ways to reduce complexity of a model:
  - Model averaging (ensemble methods).
  - Regularization (linear models).
- "Fully"-Bayesian methods combine both of these (CPSC 540).
  - Average over all models, weighted by posterior (including regularizer).
  - Can use extremely-complicated models without overfitting.

- Sometimes it's easier to define a likelihood than a loss function.

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:
  - 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.

Summary

- 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^Tx_i.
- Maximum likelihood estimate viewpoint of common models.
  - Objective functions are equivalent to maximizing $p(y \mid X, w)$.
- MAP estimation directly models $p(w \mid X, y)$.
  - Gives probabilistic interpretation to regularization.
- 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

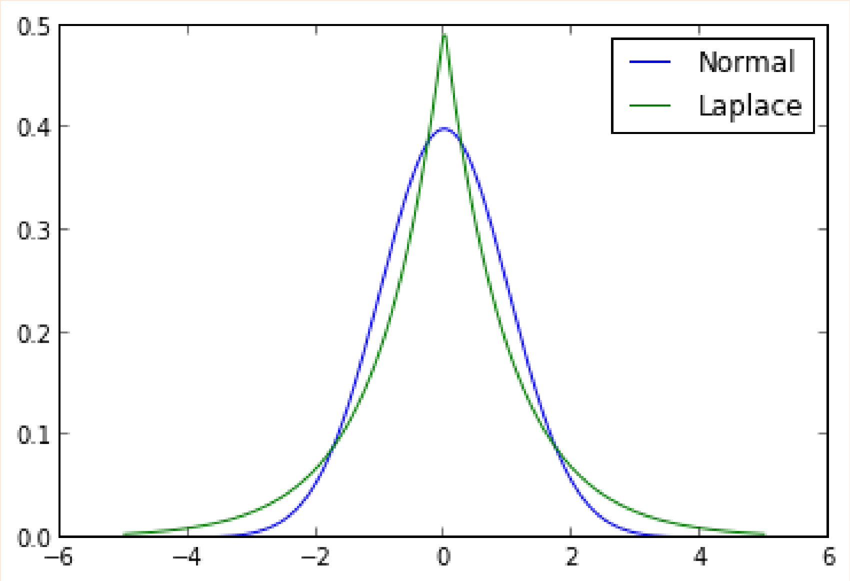
- Classic justifications for the Gaussian assumption underlying least squares:
  - Your noise might really be Gaussian. (It probably isn't, but maybe it's a good enough approximation.)
  - The central limit theorem (CLT) from probability theory. (If you add up enough IID random variables, the estimate 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^Tx_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 from (published anonymously by Gosset while working at Guiness) is even more robust, but doesn't lead to a convex objective.

“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.

http://austinrochford.com/posts/2013-09-02-prior-distributions-for-bayesian-regression-using-pymc.html

Multi-Class Logistic Regression

- Last time 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:

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

  - 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.

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:

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

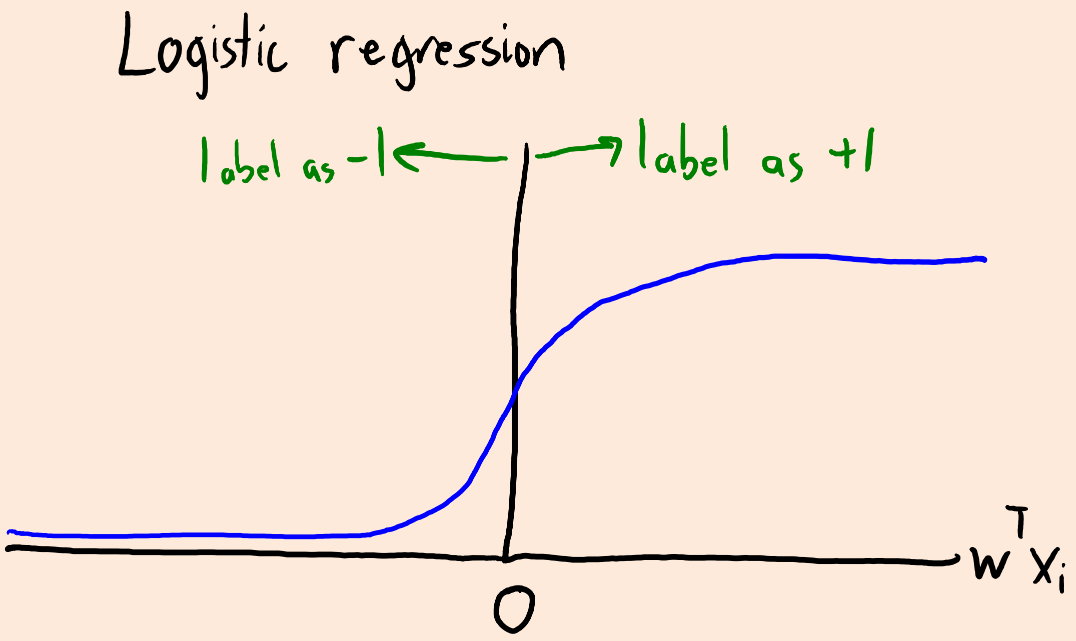
  - This is due to the "sum to 1" property (one of the z_c values is redundant).
  - So if k=2, we don't need a z_2 and only need a single 'z'.
  - Further, when k=2 the probabilities can be written as:

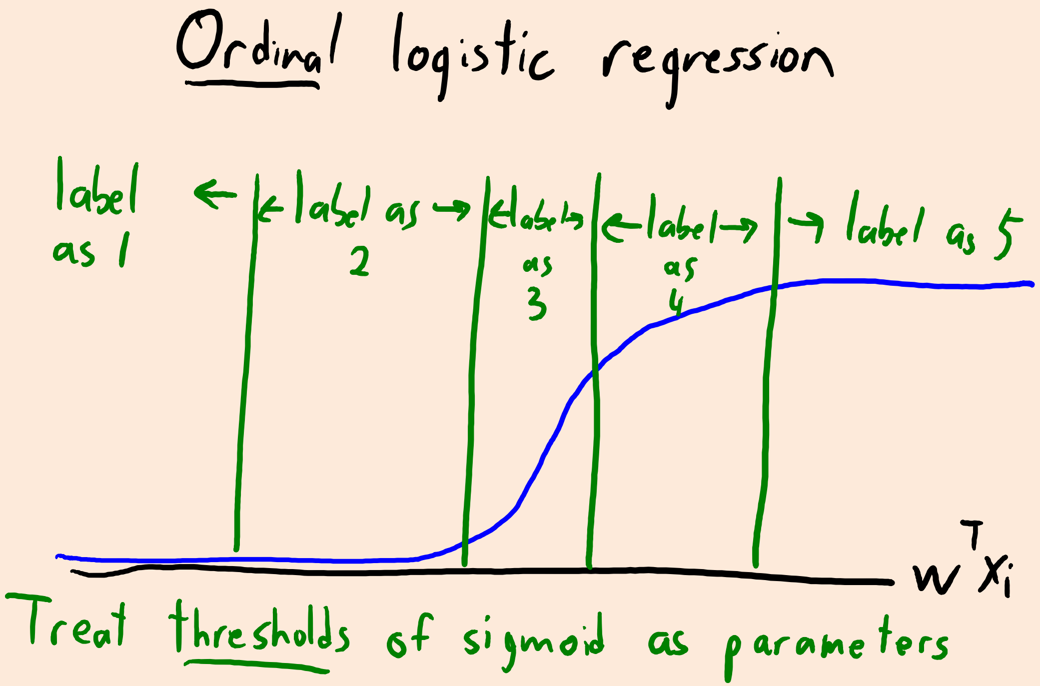
$$p(y=1 \mid z) = \frac{\exp(z)}{1+\exp(z)} = \frac{1}{1+\exp(-z)} \qquad p(y=2 \mid z) = \frac{1}{1+\exp(z)}$$

  - 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'.





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.

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

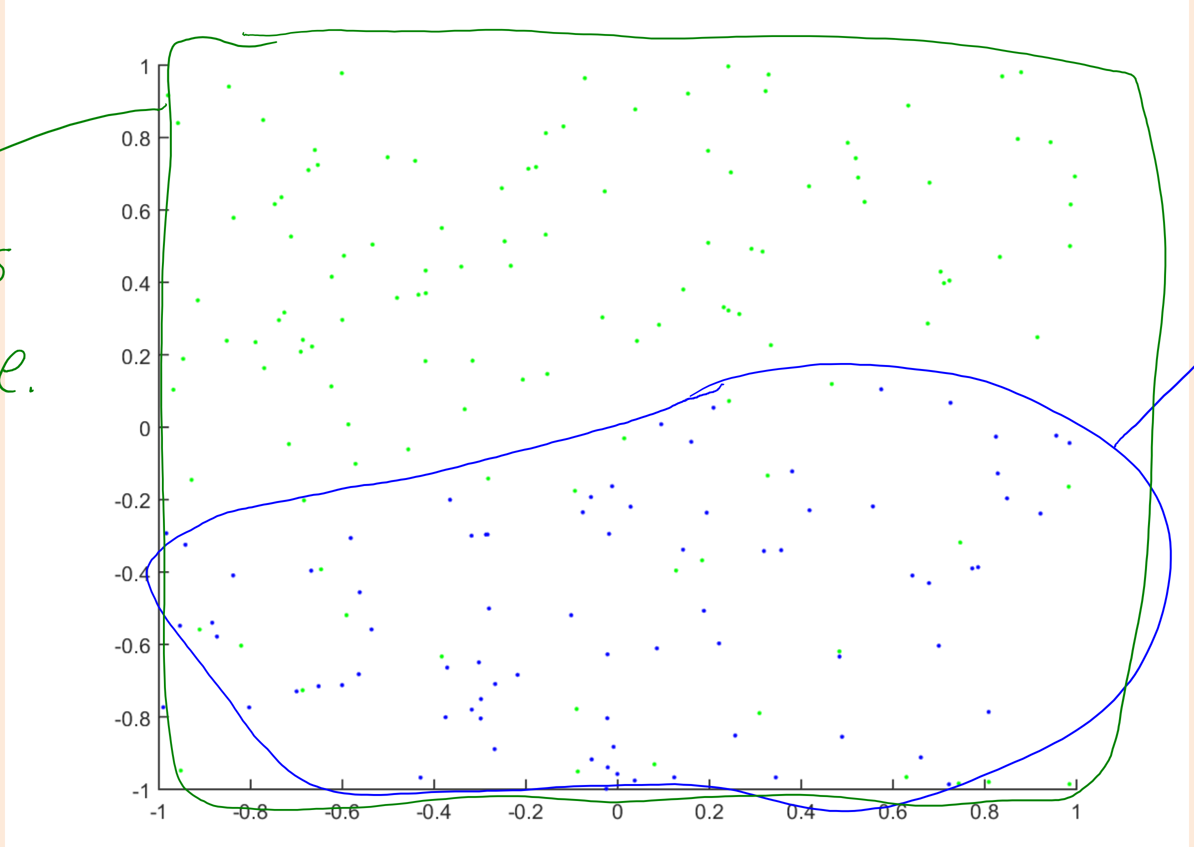
- This issue comes up a lot in practice!

- 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 option is to change to an asymmetric loss function that penalizes one type of error more than the other.

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

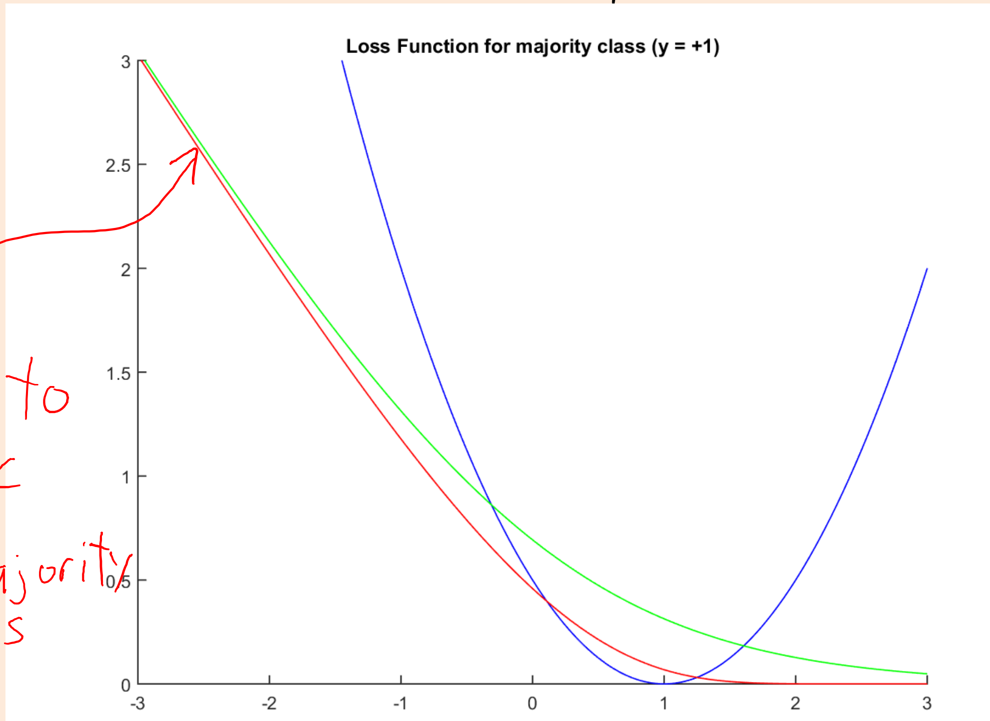
Unbalanced Data and Extreme-Value Loss

- Extreme-value distribution:

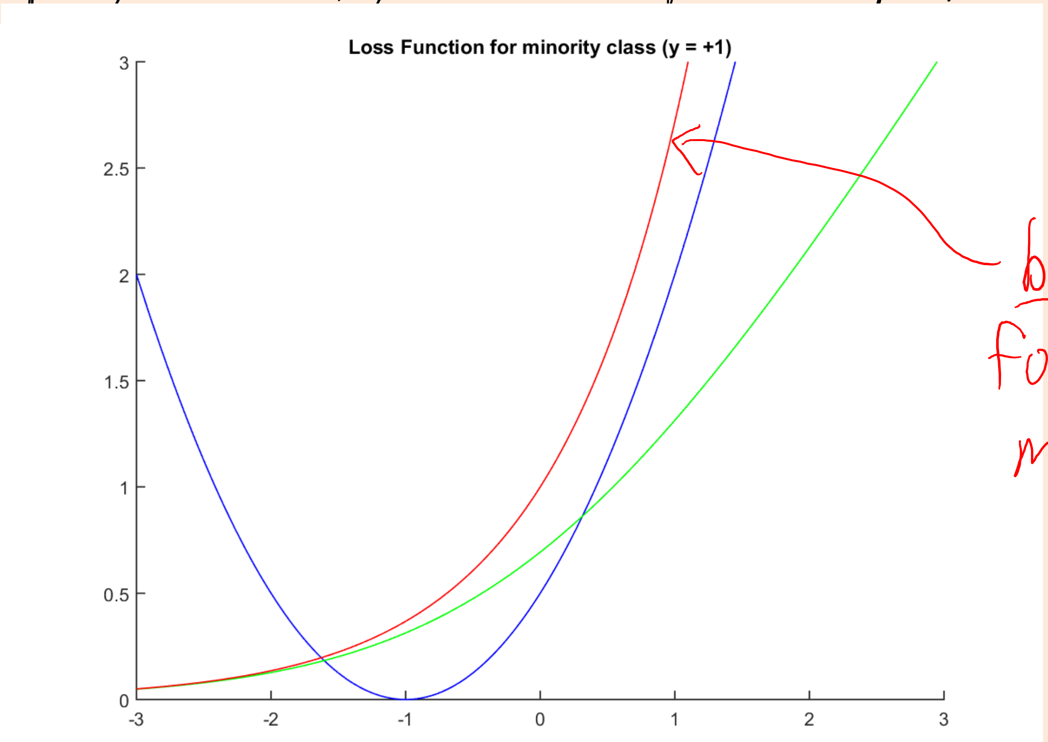
$$p(y_i = +1 \mid \hat{y}_i) = 1 - \exp(-\exp(\hat{y}_i)) \quad [+1 \text{ is majority class}]$$

asymmetric

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



Similar to logistic for majority class



big penalty for getting minority class wrong.

Unbalanced Data and Extreme-Value Loss

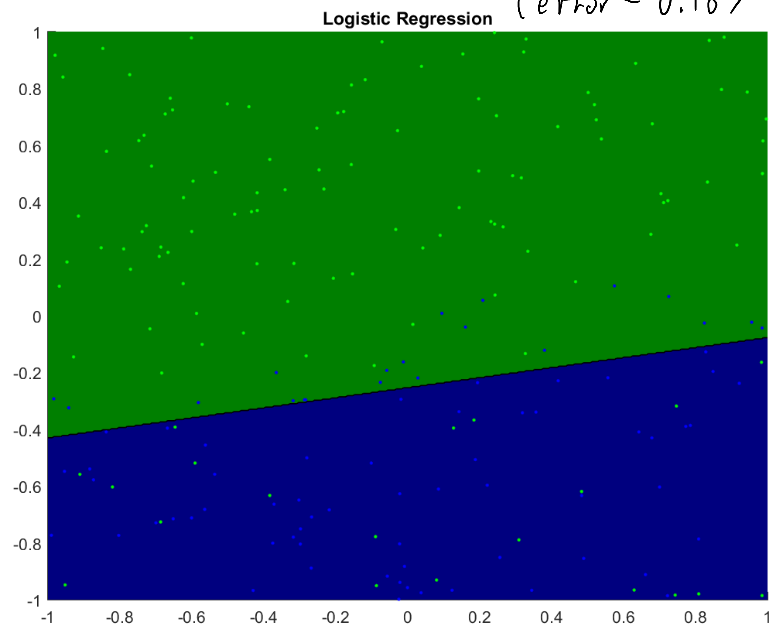
- Extreme-value distribution:

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

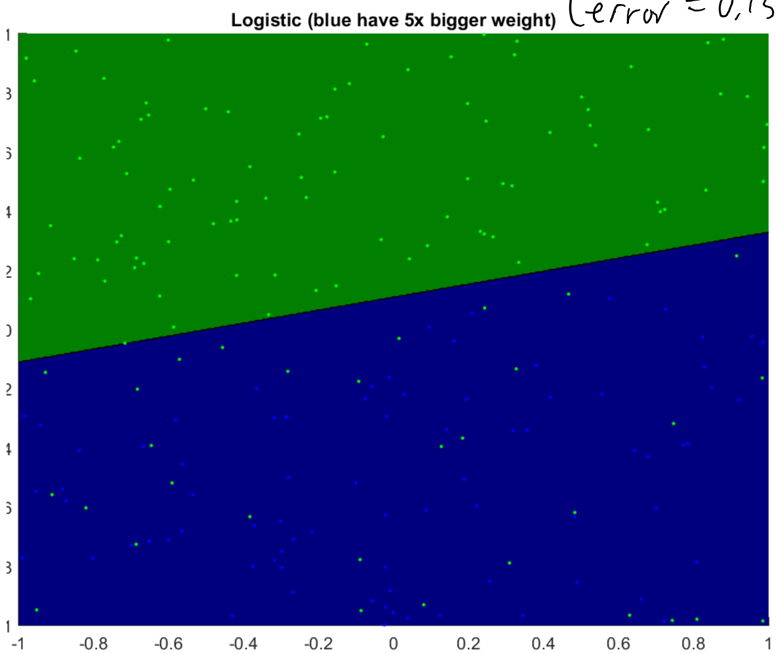
asymmetric

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

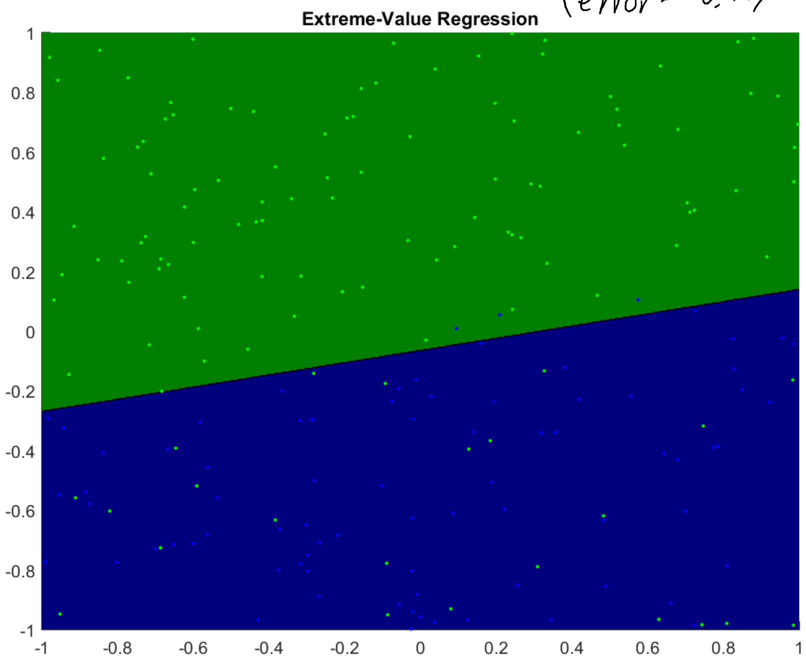
(error = 0.18)



(error = 0.15)



(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.

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

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.

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

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

Notice that normalizing constant doesn't matter:

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

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.

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

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

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

$\Updownarrow$

Take log:

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

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.

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

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

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:

$$\max\{\underbrace{0}_{\text{when } 1 - y_i w^T x_i \leq 0},\ 1 - \underbrace{y_i w^T x_i}_{\text{when } 1 - y_i w^T x_i \geq 0}\}$$

We get SVMs by looking at regularized average loss:

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

Loss Functions from Probability Ratios

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

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

Want: $\frac{p(y_i \mid x_i, w)}{p(y_i = c' \mid x_i, w)} \geq \beta$ 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:

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

Option 2: penalize only max violation:

$$\max_{c' \neq c} \{\max\{0, 1 - w_{y_i}^T x_i + w_{c'}^T x_i\}\}$$

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

Want: $\frac{p(y_i \mid X, w)}{p(y_j \mid X, w)} \geq \beta$ 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)