CPSC 340: Machine Learning and Data Mining

L1-Regularization

Fall 2016

Admin

- Midterm:
  - Grades/solutions will be posted later this week.
- Assignment 4:
  - Coming soon.

Last Time: Feature Selection

- Before midterm we discussed feature selection:
  - Choosing set of “relevant” features.

$$X = [\quad\quad] \qquad y = [\ ]$$

“relevant”

- There a numerous challenges:
  - Conditional independence and variable dependence (can do these wrong).
  - Tiny effects and context-specific relevance (depends on application).
  - Causality and confounding (can’t resolve these with “observational” data).

Last Time: Feature Selection

- Before midterm we discussed feature selection:
  - Choosing set of "relevant" features.

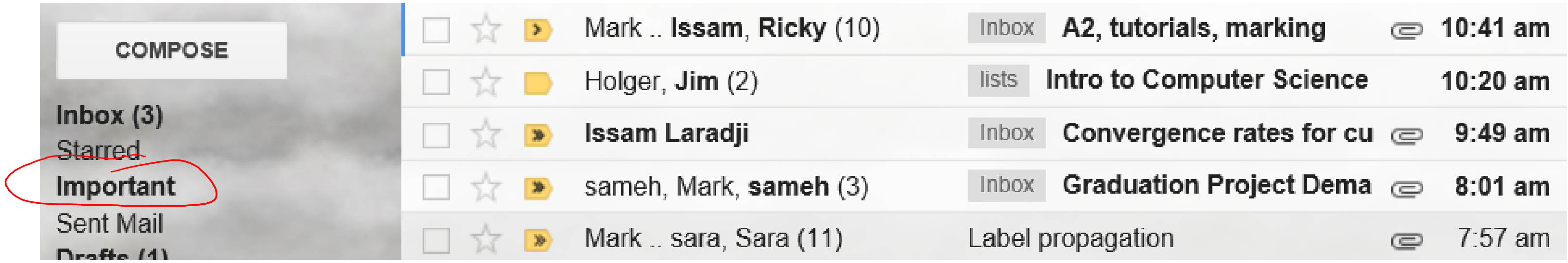
$X = \begin{bmatrix} \quad \end{bmatrix} \qquad y = \begin{bmatrix} \, \end{bmatrix}$

"relevant"

- Two common approaches:
  - Hypothesis testing: sequence of conditional independence tests.
    - Often better for identifying relevant factors, worse for regression.
    - Variable dependence problems.
  - Search and score: define a "score" and search for the best score.
    - Often better for regression, worse for identifying factors.
    - Hard to define "score" and search for the best score.

Motivation: 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 use 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 talked generative probabilistic models:
  - These use Bayes rule and models $p(x_i|y_i)$ to predict $p(y_i \mid x_i)$.

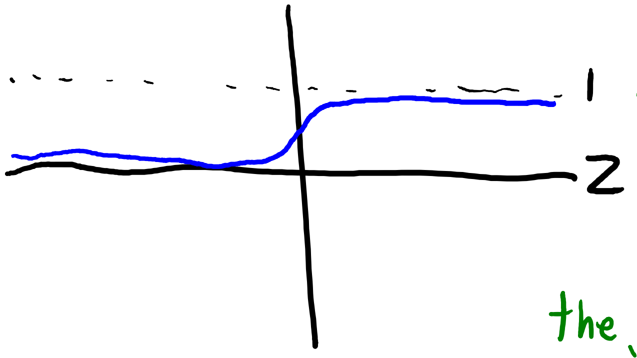
$$p(y_i|x_i) \propto p(x_i|y_i)p(y_i)$$

  - Classic example is naïve Bayes.
- Alternative is discriminative probabilistic 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.
    - Tend to work better when we have lots of data.
  - Classic example is logistic regression.

Logistic Regression

- 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.
  - E.g., $p(y_i \mid x_i)$ has a form with only 'd' parameters.
- Most common choice is linear model passed through sigmoid:

$$\underbrace{h(z) = \frac{1}{1+e^{-z}}}_{\text{"sigmoid" function}}$$



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

Linear passed through sigmoid:

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

Takes $w^T x_i \in (-\infty, \infty)$ returns probability $\in (0,1)$

Logistic Regression

- Linear passed through sigmoid is called logistic regression.

$$p(y_i = +1 \mid x_i, w) = \underbrace{h}_{\text{sigmoid}}(w^T x_i) = \frac{1}{1 + \exp(-\underbrace{w^T}_{\text{only 'd' parameters 'w'}} x_i)}$$

Given this, we have

$$p(y_i = -1 \mid x_i, w) = 1 - p(y_i = +1 \mid x_i, w)$$

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

$$= \frac{1 + \exp(-w^T x_i) - 1}{1 + \exp(-w^T x_i)} = \frac{\exp(-w^T x_i)}{1 + \exp(-w^T x_i)} = \frac{1}{\exp(w^T x_i) + 1} = h(-w^T x_i)$$

Logistic Regression

- Linear passed through sigmoid is called logistic regression.

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

only 'd' parameters 'w'

Given this, we have

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

We can write both cases as

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

Maximum Likelihood Estimation

- We can find 'w' using maximum likelihood estimation (MLE).
  - Given data {X,y} and parameters 'w', MLE is given by maximum of:

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

"likelihood"

  - Appealing "consistency" properties as n goes to infinity (take STAT 4XX).
  - If our data {X,y} contains 'n' IID samples $\{x_i, y_i\}$, then likelihood simplifies:

$$p(y \mid X, w) = \prod_{i=1}^{n} p(y_i \mid x_i, w)$$

for IID data

  - We've used this before: our naïve Bayes "counting" estimate was the MLE.

Maximum Likelihood Estimation

- We can find 'w' using maximum likelihood estimation (MLE).
  - Given IID data {X,y} and parameters 'w', MLE is given by maximum of:

$$\underbrace{p(y \mid X, w)}_{\text{"likelihood"}} = \prod_{i=1}^{n} \underbrace{p(y_i \mid x_i, w)}_{\text{"likelihood" of example 'i'}}$$

  - We usually maximize the "log-likelihood":

$$\log\left(\prod_{i=1}^{n} p(y_i \mid x_i, w)\right) = \sum_{i=1}^{n} \log(p(y_i \mid x_i, w))$$

Does this give the same 'w'?

Yes because log is monotonic:

If $\alpha \geq \beta$ then $\log(\alpha) \geq \log(\beta)$ (preserves order)

Turns multiplication into addition.

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

and $\log(a_1 a_2 a_3) = \log(a_1) + \log(a_2) + \log(a_3)$

Maximum Likelihood Estimation

- We can find 'w' using maximum likelihood estimation (MLE).
  - Given IID data {X,y} and parameters 'w', MLE is given by maximum of:

$$\log\left(\prod_{i=1}^{n} p(y_i \mid w, x_i)\right) = \sum_{i=1}^{n} \log\left(p(y_i \mid x_i, w)\right)$$

  - But we like to minimize things, so let's minimize negative log-likelihood:

NLL

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

Is this ok? If 'w' minimizes $f(w)$
then 'w' maximizes $-f(w)$.

MLE for Logistic Regression

- We can find 'w' using maximum likelihood estimation (MLE).
  - Given IID data {X,y} and parameters 'w', MLE is given by minimum of:

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

  - For logistic regression we had:

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

  - So the MLE minimizes:

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

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

$$= \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i)) \quad \text{"logistic loss"}$$

MLE for Logistic Regression

- The loss function used by logistic regression:

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

- This 'f' is convex and differentiable.
  - We can minimize it using gradient descent.
- If we multiply by (1/n), this 'f' is an average.
  - We can use stochastic gradient on huge datasets.
- Can get probabilities from sigmoid: $p(y_i = +1 \mid x_i, w) = \frac{1}{1 + \exp(-y_i w^T x_i)}$
- We can/should add a regularizer: $f(w) = \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2}\|w\|^2$

(pause)

Greedy Search and Score: Forward Selection

- Forward selection algorithm for feature selection:
  1. Start with an empty set of relevant features, S = [ ].
  2. For each possible feature 'j':
     - Fit model with 'j' added to set of relevant features 'S'.

       For least squares in Matlab: w = (X(:,S)'*X(:,S))\(X(:,S)'*y)

     - Compute score of adding this feature.

       For L_0-norm in Matlab: score = sum((X(:,S)*w - y).^2) + λ*length(S)

  3. Find 'j' that improves the score the most.
     - If this 'j' improves the score, add it to 'S' and go back to 2.
     - If no feature 'j' improves the score, then stop.

Greedy Search and Score: Forward Selection

- Forward selection algorithm for feature selection:
  1. Start with an empty set of relevant features, S = [ ].
  2. For each possible feature 'j':
     - Fit model with 'j' added to set of relevant features 'S'.

       For logistic regression run gradient descent on $f(w) = \sum_{i=1}^{n} \log(1+\exp(-y_i w^T x_i))$

     - Compute score of adding this feature.

       For logistic regression compute $\text{score} = \sum_{i=1}^{\hat{n}} \log(1+\exp(-y_i w^T x_i)) + \lambda \|w\|_0$

  3. Find 'j' that improves the score the most.
     - If this 'j' improves the score, add it to 'S' and go back to 2.
     - If no feature 'j' improves the score, then stop.

Expensive to do this $O(d^2)$ times!

And shouldn't we regularize?

Another loop to select λ?

Feature Selection Approach 3: L1-Regularization

- Consider regularizing by the L1-norm:

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

- Like L2-norm, it's convex and improves our test error.
- Like L0-norm, it encourages elements of 'w' to be exactly zero.

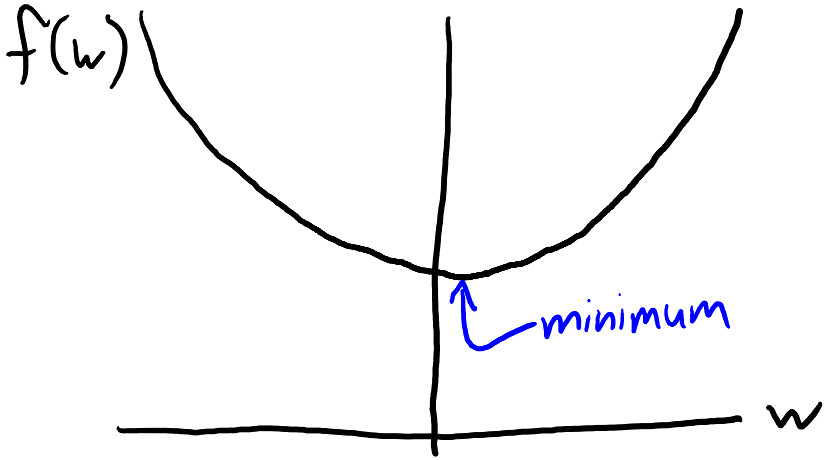
- L1-regularization simultaneously regularizes and select features.
  - Very fast alternative to search and score.

Sparsity and Least Squares

- Consider 1D least squares objective:

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

- This is a convex 1D quadratic function of 'w' (i.e., a parabola):



- This variable does not look relevant (minimum is close to 0).
  - But for finite 'n' the minimum is unlikely to be exactly zero.

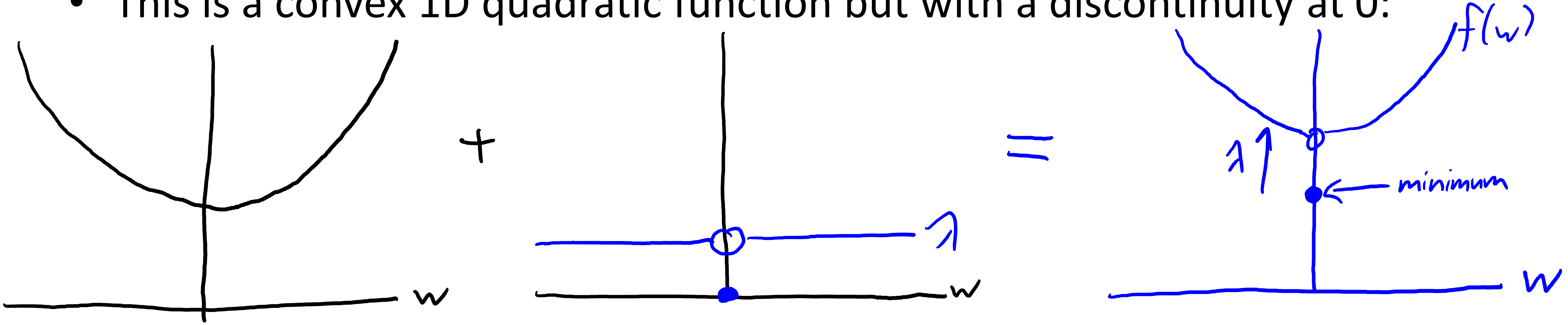
You would need to have $\sum_{i=1}^{n} y_i x_i = 0$

Sparsity and L0-Regularization

- Consider 1D L0-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(w x_i - y_i)^2 + \lambda \underbrace{\|w\|_0}_{\begin{cases}\lambda & \text{if } w \neq 0 \\ 0 & \text{if } w = 0\end{cases}}$$

- This is a convex 1D quadratic function but with a discontinuity at 0:



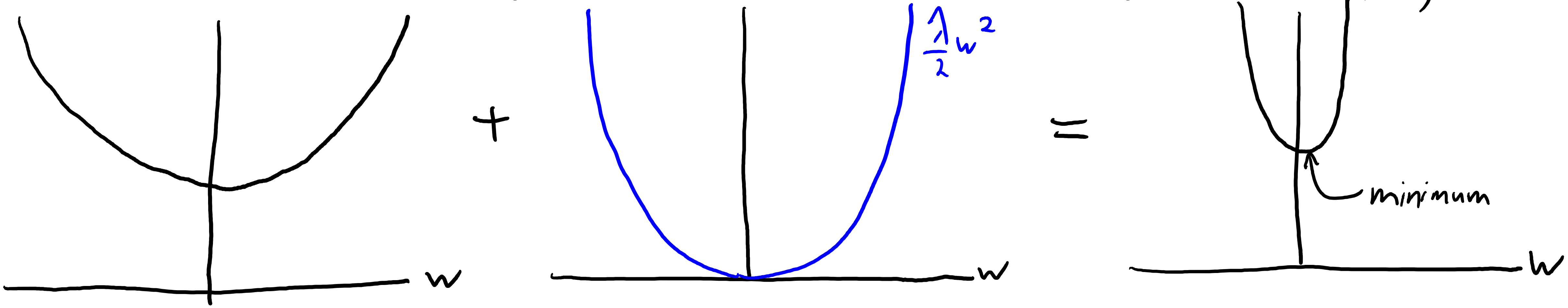
- L0-regularized minimum is often exactly at the 'discontinuity' at 0:
  - Sets the feature to exactly 0 (does feature selection), but not convex.

Sparsity and L2-Regularization

- Consider 1D L2-regularized least squares objective:

$$f(w) = \frac{1}{2}\sum_{i=1}^{n}(wx_i - y_i)^2 + \frac{\lambda}{2}w^2$$

- This is a convex 1D quadratic function of 'w' (i.e., a parabola): $f(w)$



- L2-regularization moves it a bit closer to zero.
  - But doesn't do feature selection (nothing special about being 'exactly' zero).

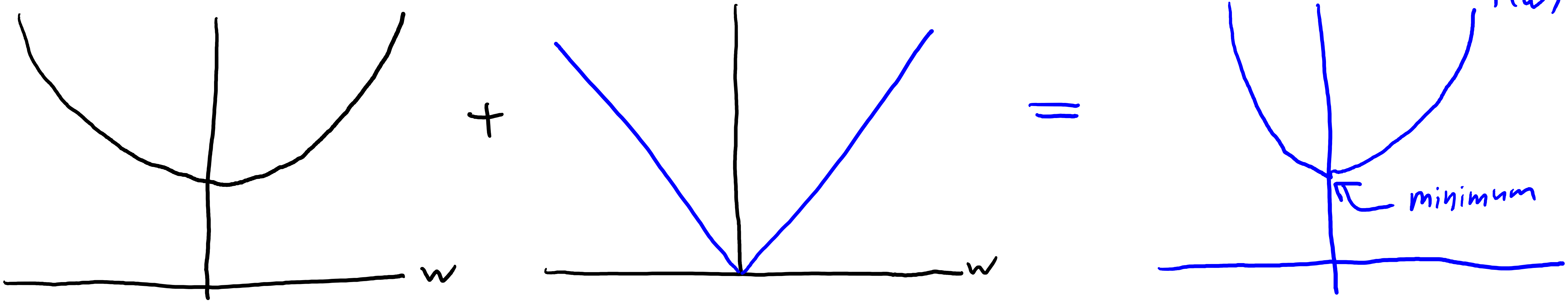
You would need $f'(0) = \sum_{i=1}^{n} y_i x_i = 0$

Sparsity and L1-Regularization

- Consider 1D L1-regularized least squares objective:

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

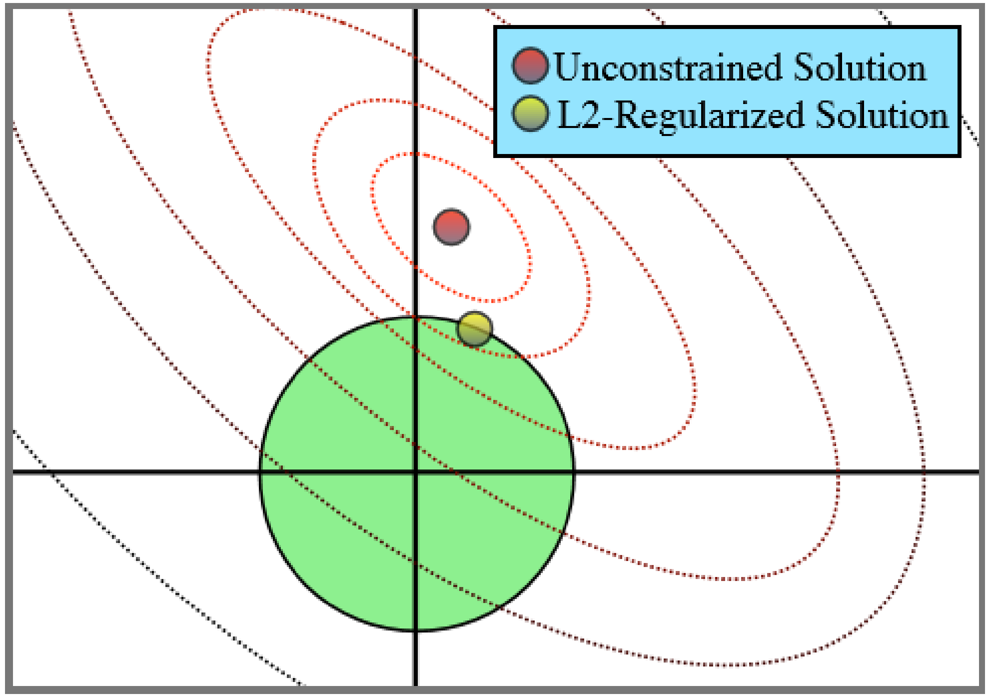
- This is a convex piecwise-quadratic function of ‘w’ with ‘kink’ at 0:



- L1-regularization minimum is often exactly at the ‘kink’ at 0:
  - Sets the feature to exactly 0 (does feature selection), → Solution is $w=0$ when $\underbrace{\left|\sum_{i=1}^{n} y_i x_i\right|}_{\text{derivative of loss at 0.}} \leq \lambda$
  - Big λ leads to very sparse solutions, small λ give dense solutions.

Sparsity and L2-Regularization

- L2-regularization conceptually restricts 'w' to a ball.



Minimizing $\frac{1}{2}\|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$

is equivalent to minimizing

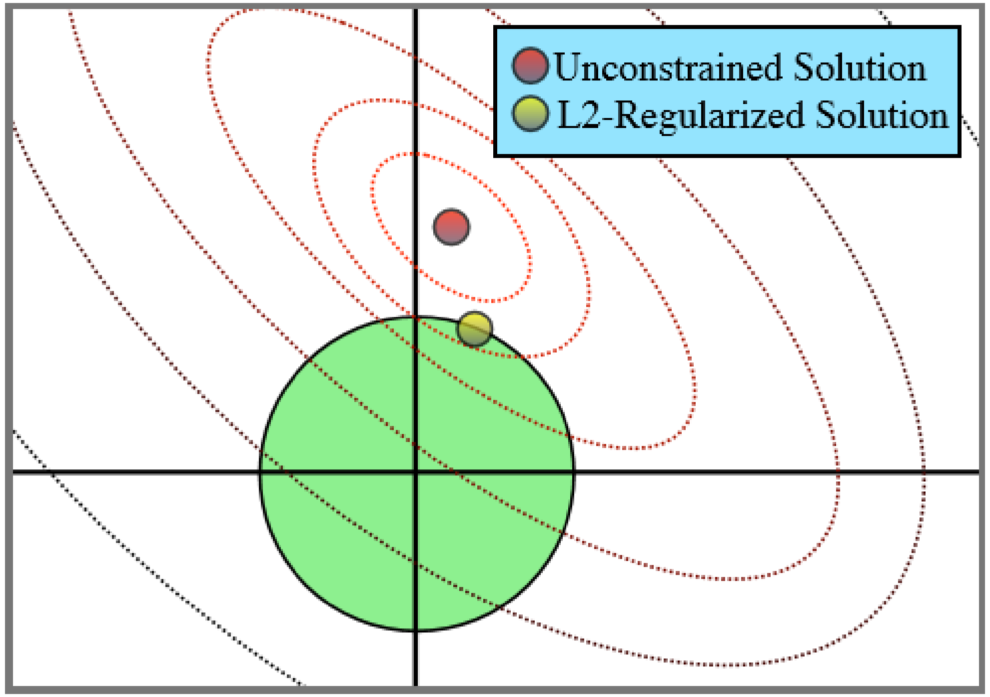
$\frac{1}{2}\|Xw - y\|^2$ subject to

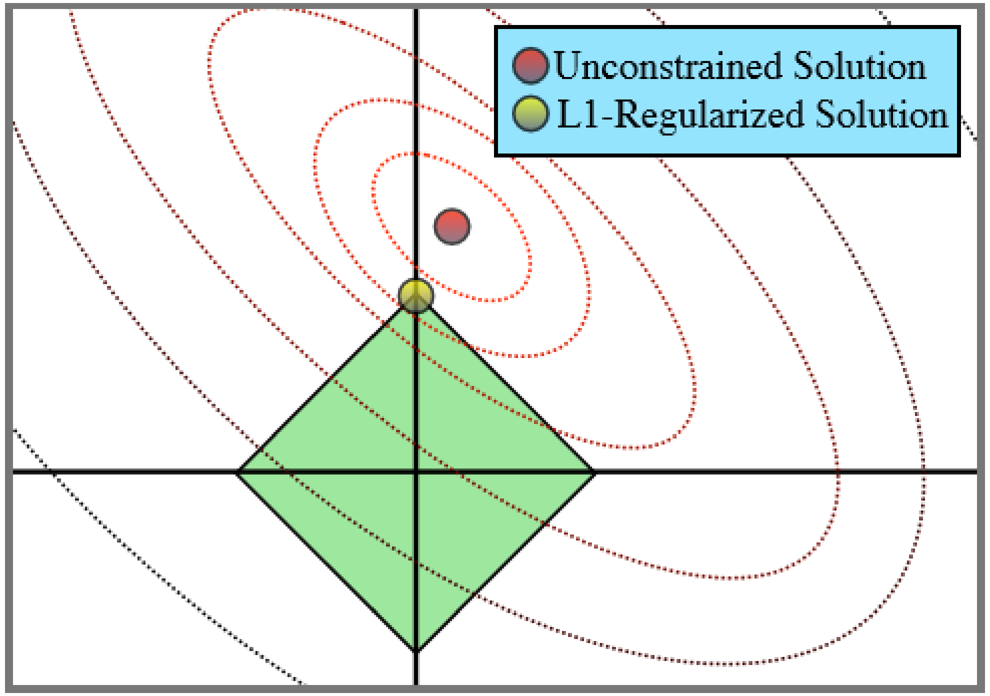
the constraint that $\|w\| \leq \tau$

for some value 'τ'

Sparsity and L2-Regularization

- L2-regularization conceptually restricts 'w' to a ball.





- L1-regularization restricts to the L1 "ball":
  - Solutions tend to be at corners where w_j are zero.

L2-Regularization vs. L1-Regularization

- L2-Regularization:
  - Insensitive to changes in data.
  - Significantly-decreased variance:
    - Lower test error.
  - Closed-form solution.
  - Solution is unique.
  - All ‘w’ tend to be non-zero.
  - Can learn with *linear* number of irrelevant features.
    - E.g., only O(d) relevant features.

- L1-Regularization:
  - Insensitive to changes in data.
  - Significantly-decreased variance:
    - Lower test error.
  - Requires iterative solver.
  - Solution is not unique.
  - Many ‘w’ tend to be zero.
  - Can learn with **exponential** number of irrelevant features.
    - E.g., only O(log(d)) relevant features.

L1-Regularization Issues

- Advantages:
  - Deals with conditional independence (if linear).
  - Sort of deals with collinearity:
    - Picks at least one of “mom” and “mom2”.
  - Very fast: we’ll talk about proximal-gradient methods next week.
- Disadvantages:
  - Tends to give false positives (selects too many variables).
- Neither good nor bad:
  - Does not take small effects.
  - Says “gender” is relevant if we know “baby”.
  - Good for prediction if we want fast training and don’t care about having some irrelevant variables.

Summary

- Discriminative models directly model $p(y_i \mid x_i)$.
- Logistic regression uses $p(y_i \mid x_i, w) = 1/(1+\exp(-y_i w^T x_i))$.
- Maximum likelihood estimation:
  - Maximizes $p(y|X,w)$, which for IID is equivalent to minimizing $-\sum_{i=1}^{n} \log p(y_i \mid x_i, w)$
- L1-regularization: simultaneous regularization / feature selection.

- Next time: what if y_i is not numerical/binary?

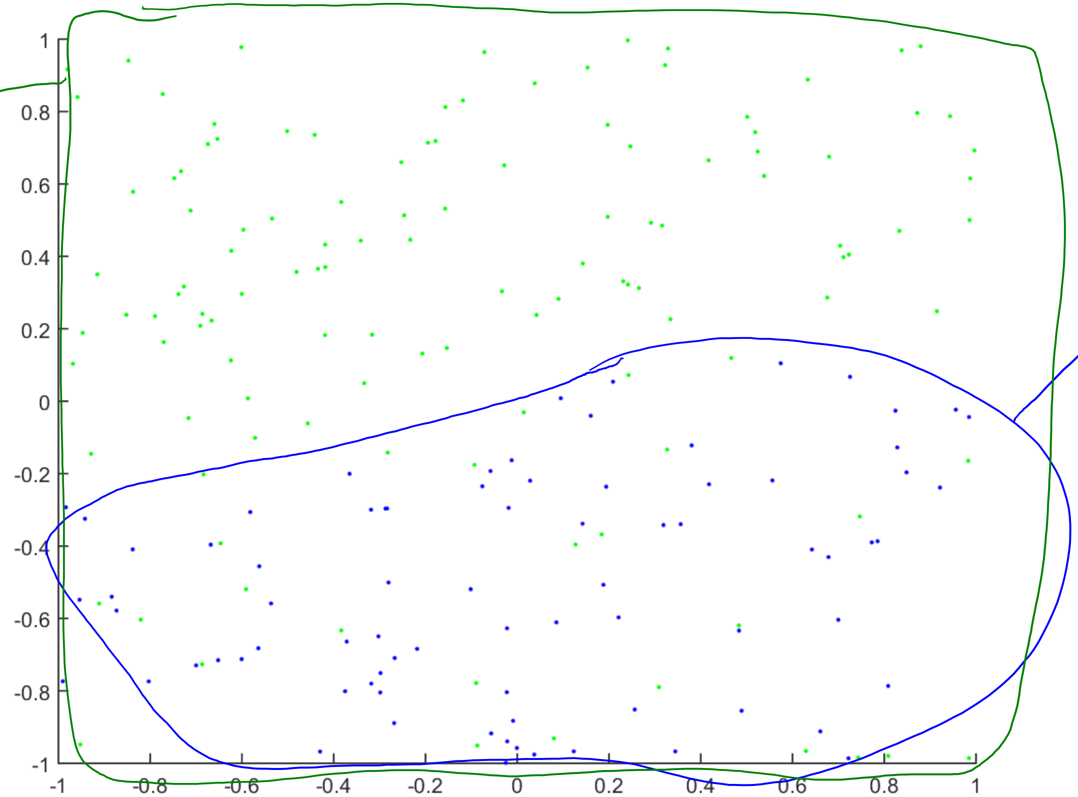
Bonus Slide: Other Parsimonious Parameterizations

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

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

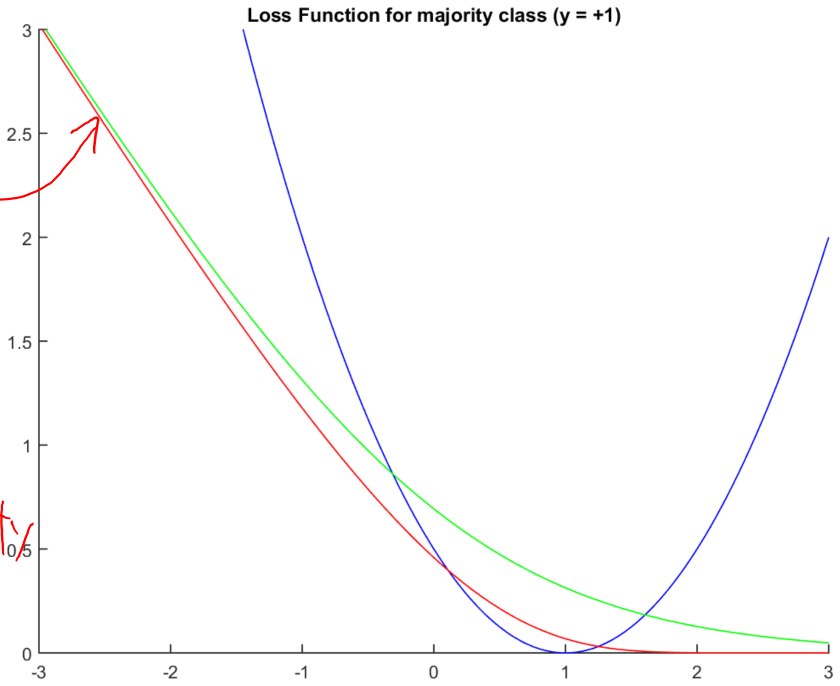
- Extreme-value distribution:

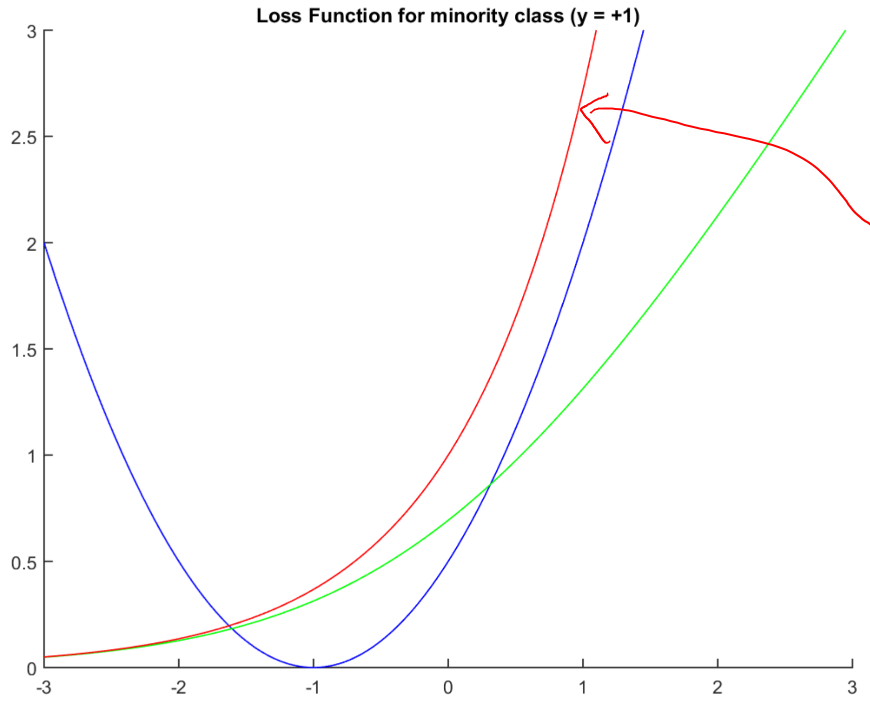
$$p(y_i = +1 \mid \hat{y}_i) = 1 - \exp(-\exp(\hat{y}_i)) \quad [+1 \text{ is majority class}]$$

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

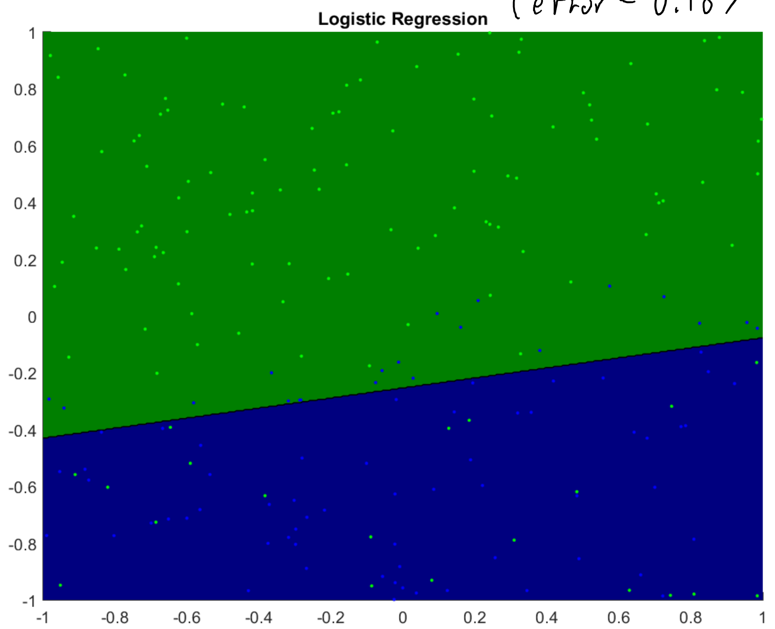
- Extreme-value distribution:

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

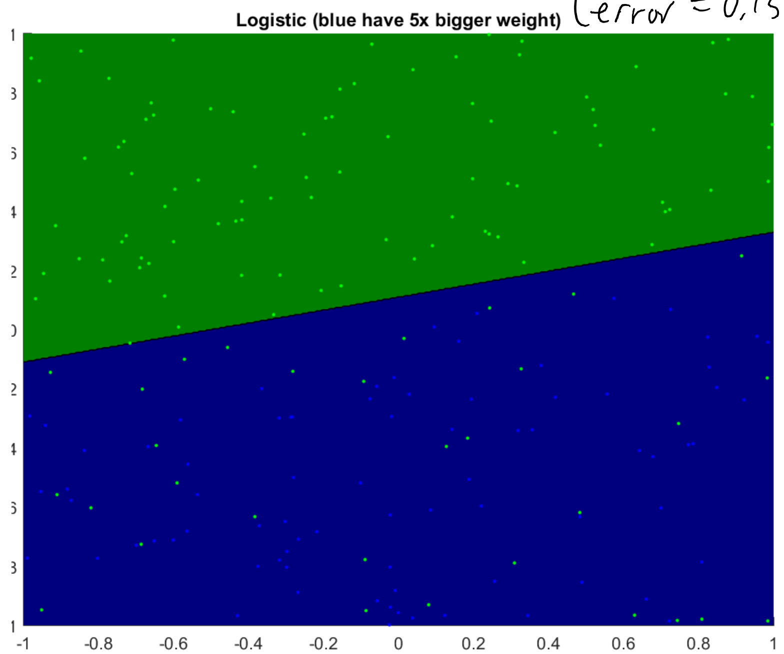
asymmetric

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)