Classification and Logistic Regression

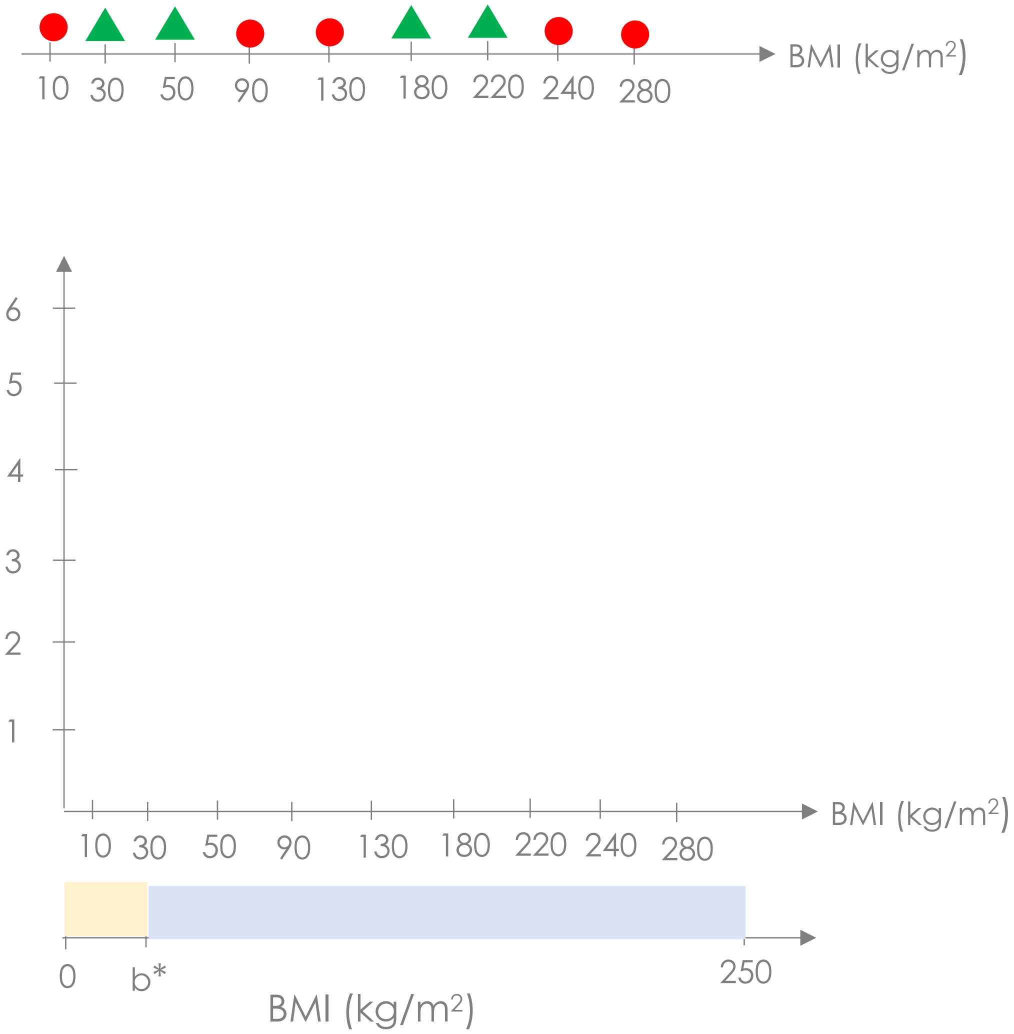
Kenneth (Kenny) Joseph

University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

Announcements

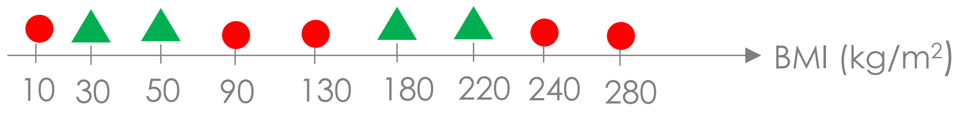
- PA2 … I know, I know.
- Quiz 5 is out
- Midterm is March 17th
  - In class
  - One page handwritten notes, front and back
  - Official Accessibility requests **due TODAY**
- Review Thursday
- Questions?

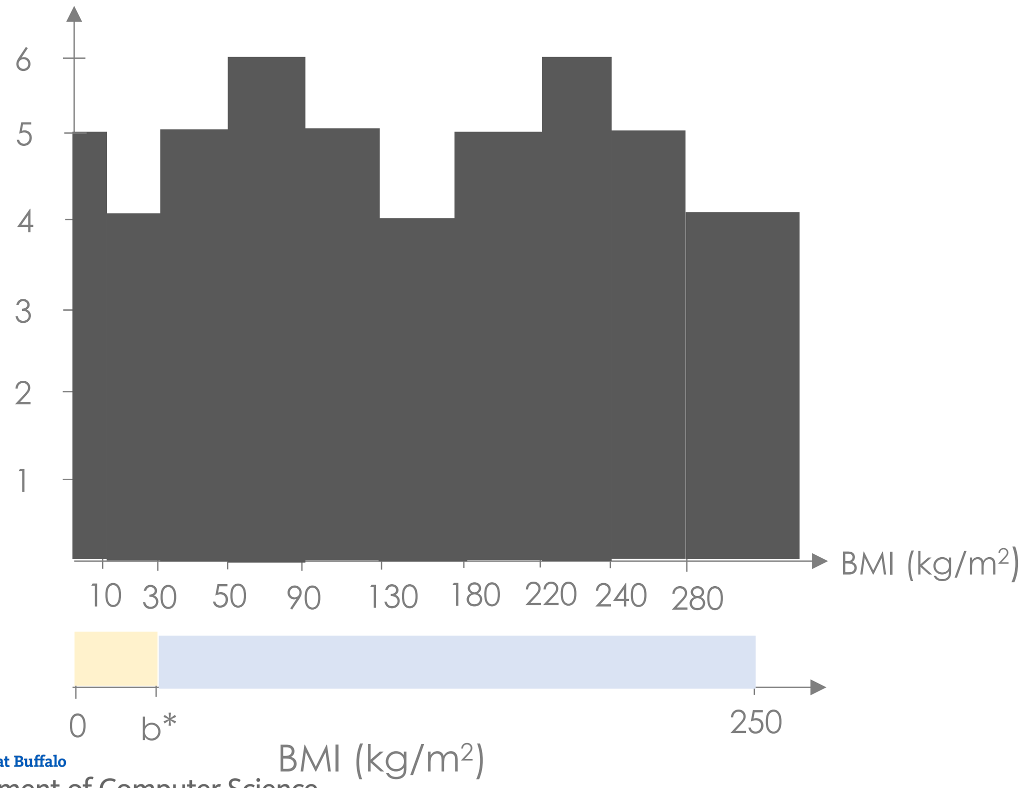
Trying to optimize 0/1 Loss in 1 Dimension

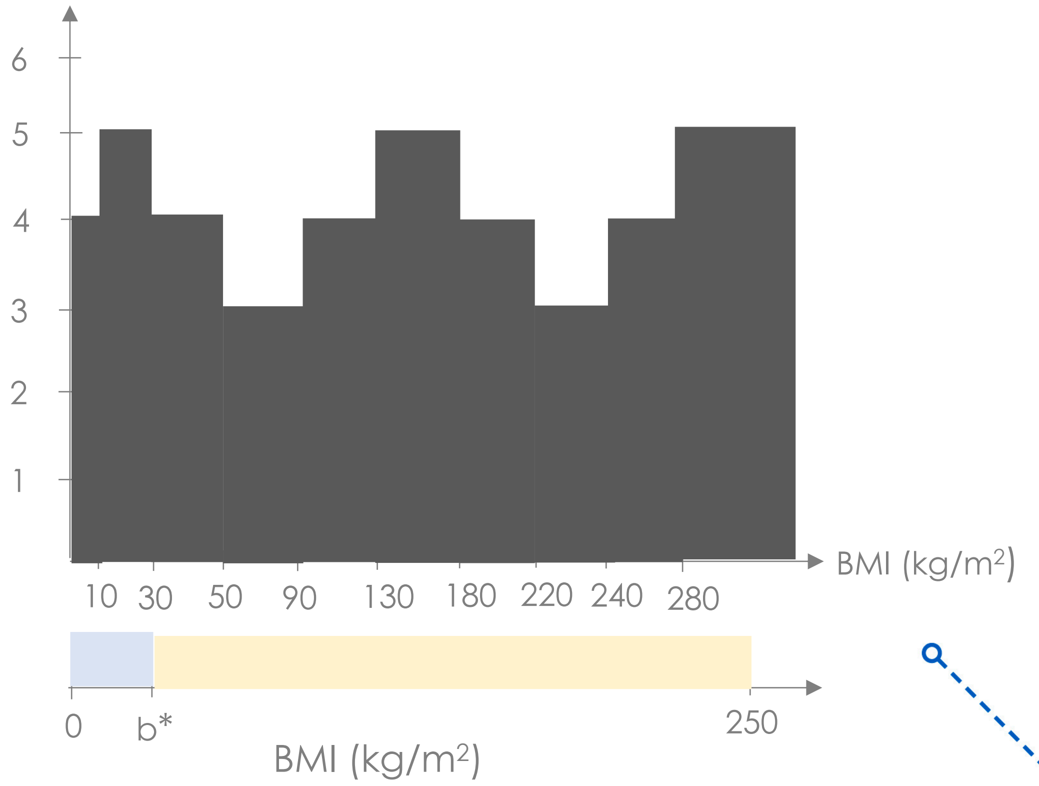


University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences
@_kenny_joseph

Trying to optimize 0/1 Loss in 1 Dimension



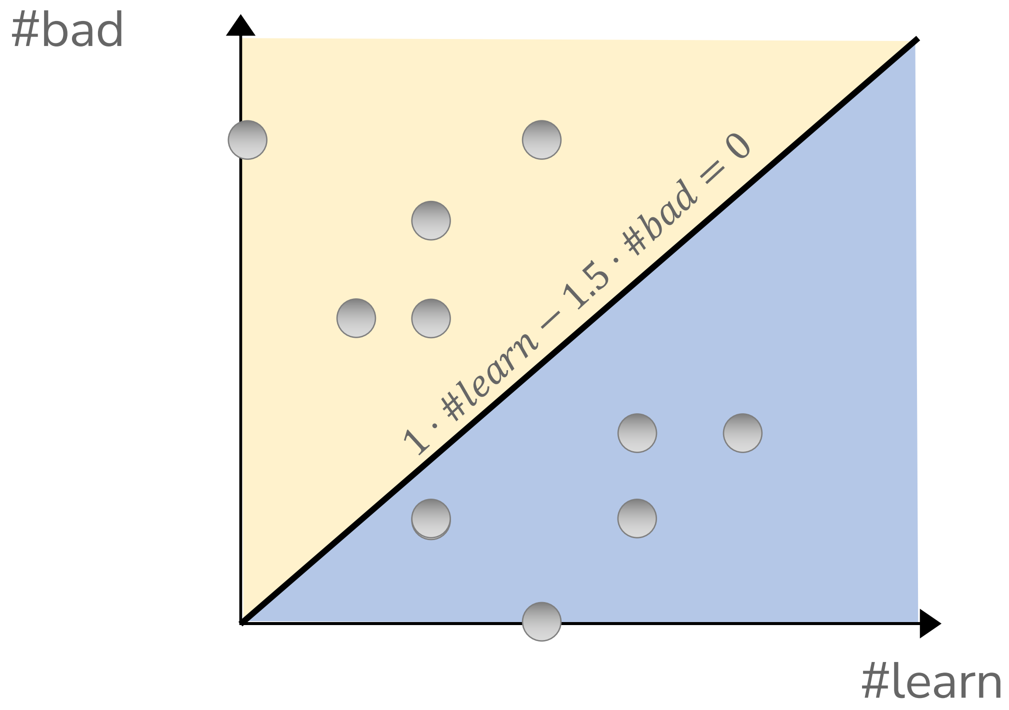


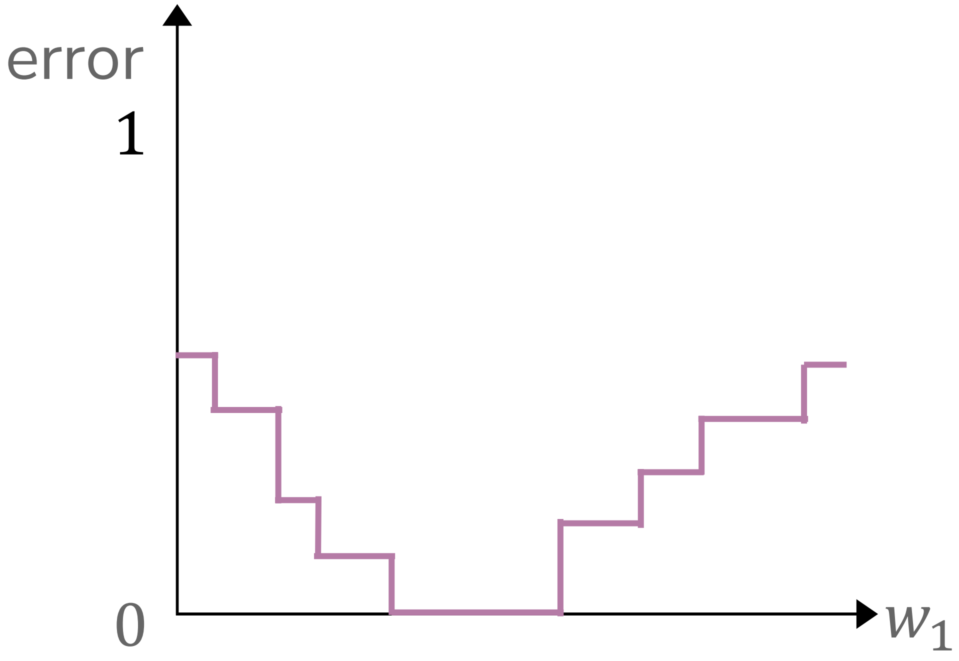


University at Buffalo
Department of Computer Science and Engineering
School of Engineering and Applied Sciences

@_kenny_joseph

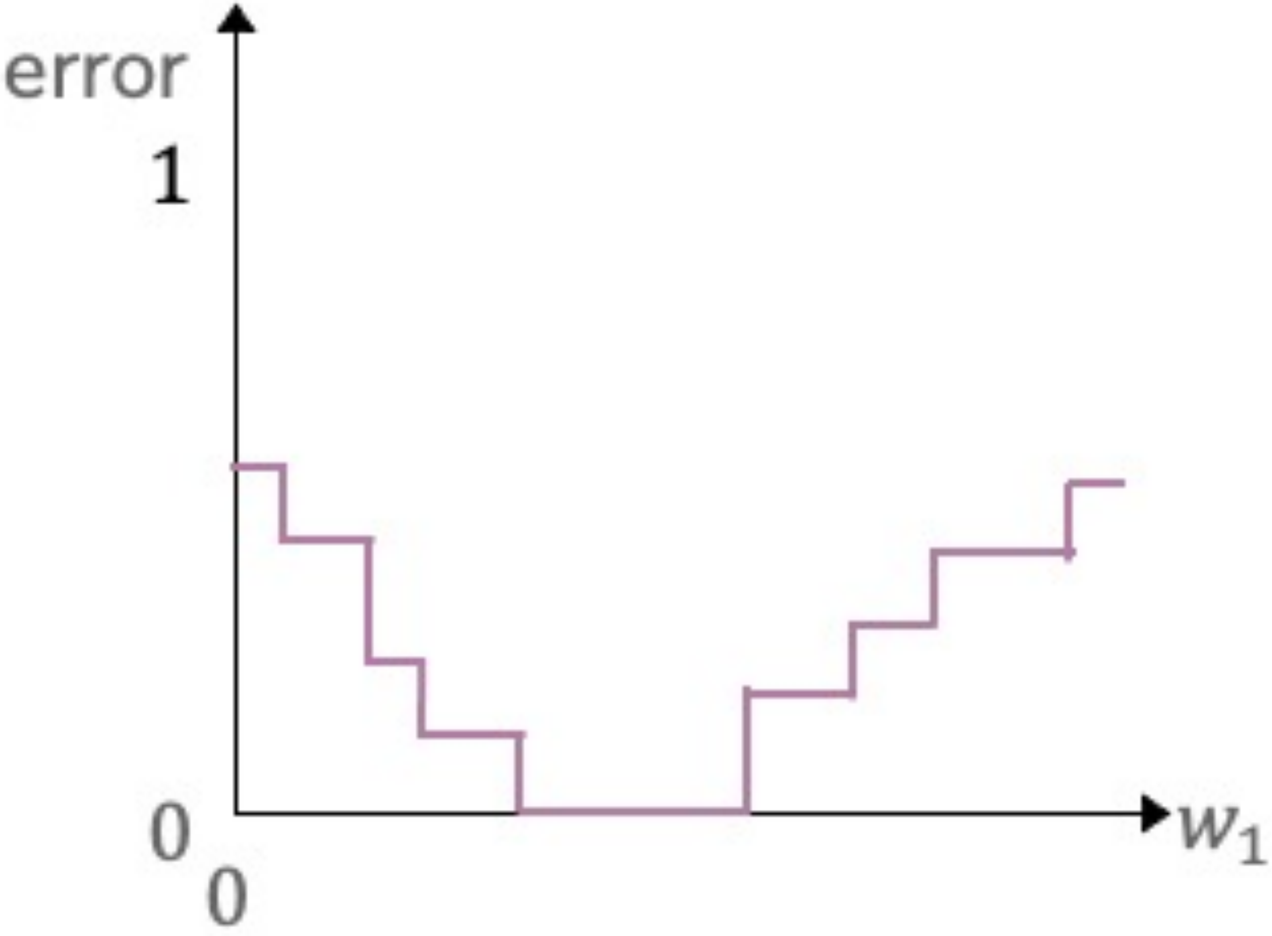
Assume w_2 is fixed, and we want to min. loss w.r.t. w_2



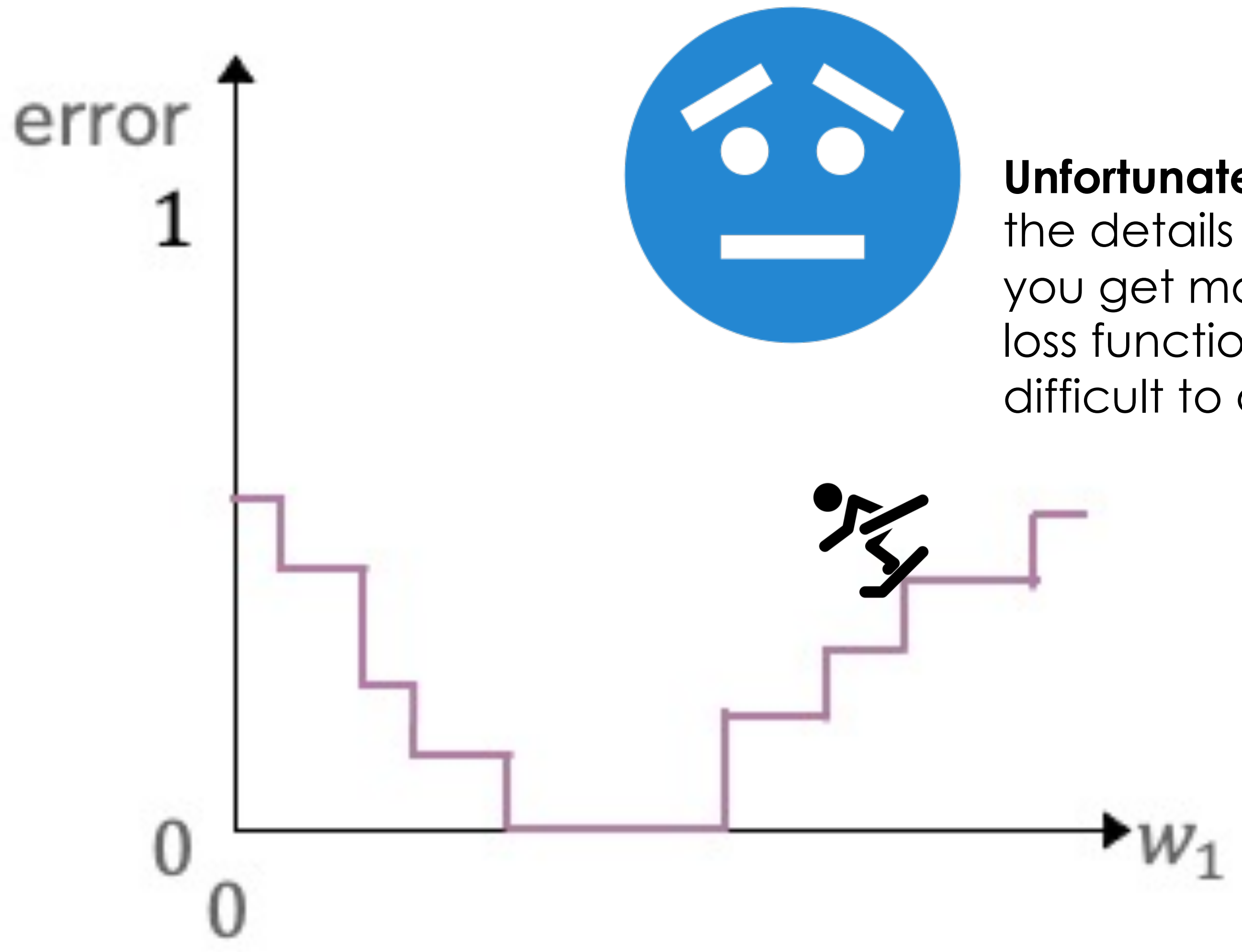


Adapted from: https://courses.cs.washington.edu/courses/cse416/21sp/

Can we just run gradient descent on this?

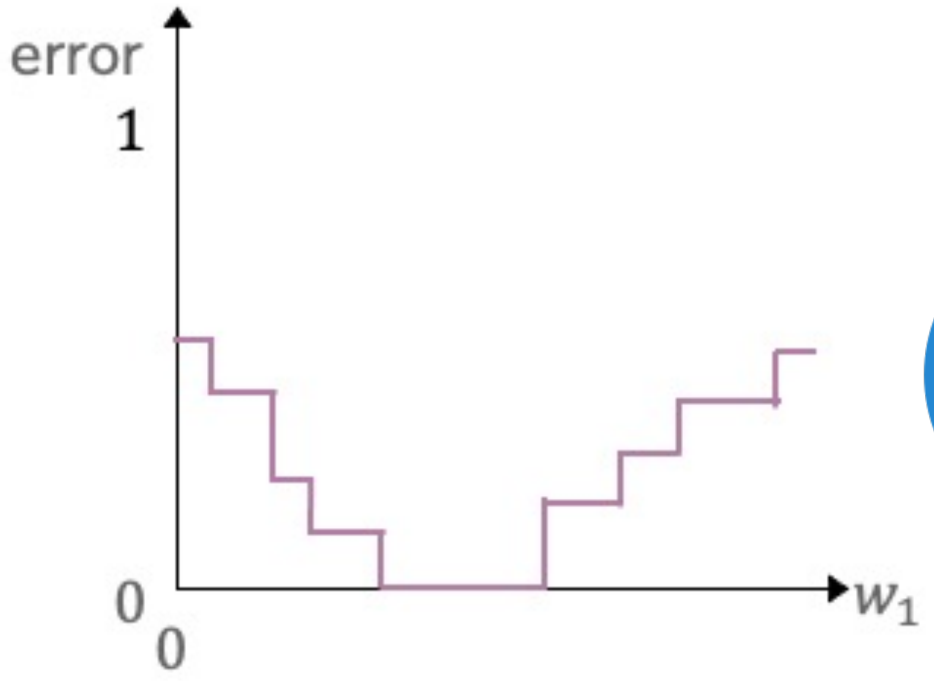


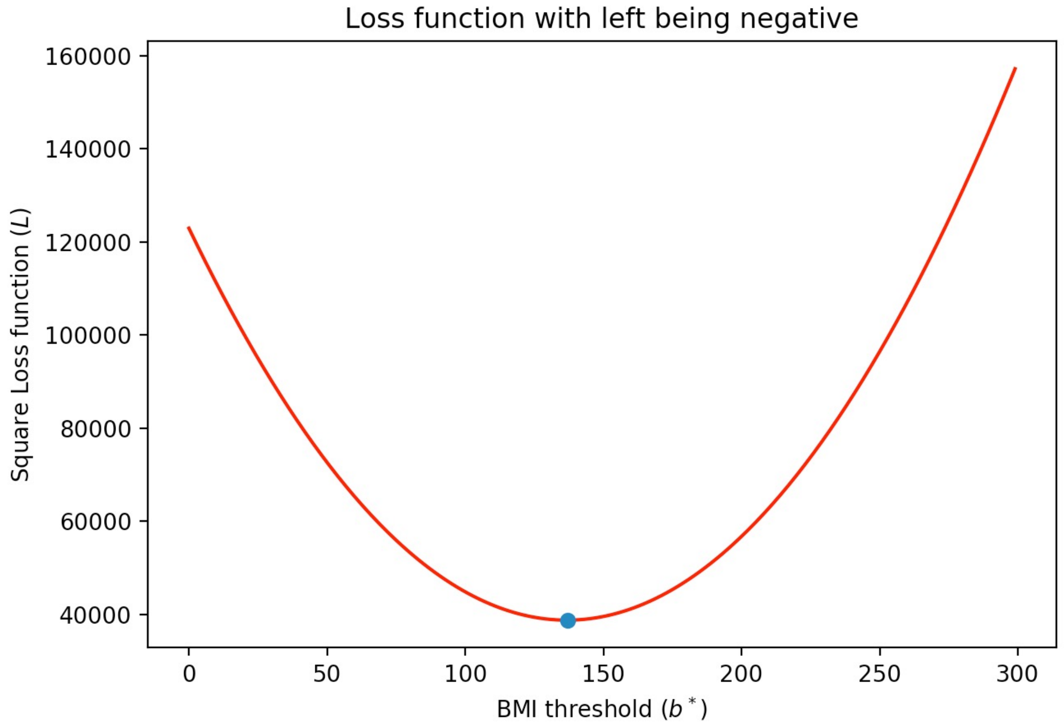
Can we just run gradient descent on this?



Unfortunately not. We will not cover the details on why, but essentially, as you get more features, these spiky loss functions become extremely difficult to optimize.

What to do? Optimization view…





Change the loss function to something we can more easily optimize!

… which is…?

Adapted from: https://courses.cs.washington.edu/courses/cse416/21sp/

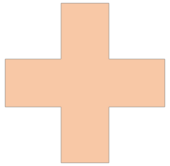
Approach 2: Bag of words + Linear classifier, Optimization view

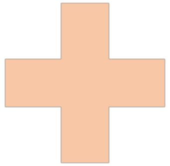
1. Convert each course evaluation statement into a “bag of words” representation
2. Specify model class:
3. Define loss function:
4. Optimize loss fn.:
5. For new test point, compute h(x) =
6. If h(x) is > 0, predict “pro”, otherwise, predict “anti”

Problem: How to interpret predictions? What does h(x)=10 mean?

What to do? Probabilistic view…

Model p(y | **x**) !

P(y =  | **This class is garbage. The professor makes bad jokes and I can't read his handwriting**) = ?

P(y =  | **The class is fine. I wish he would stop making up course evaluations though.**) = ?

Approach 3: Logistic Regression

1. Convert each course evaluation statement into a “bag of words” representation
2. Specify form of p(y | **x**)
3. Write down (log) likelihood function
4. Maximize log-likelihood fn.
5. Use trained model to estimate p(y=+ | x)
6. If p(+ | x) > .5, predict “pro-5/474”, otherwise, predict “anti”

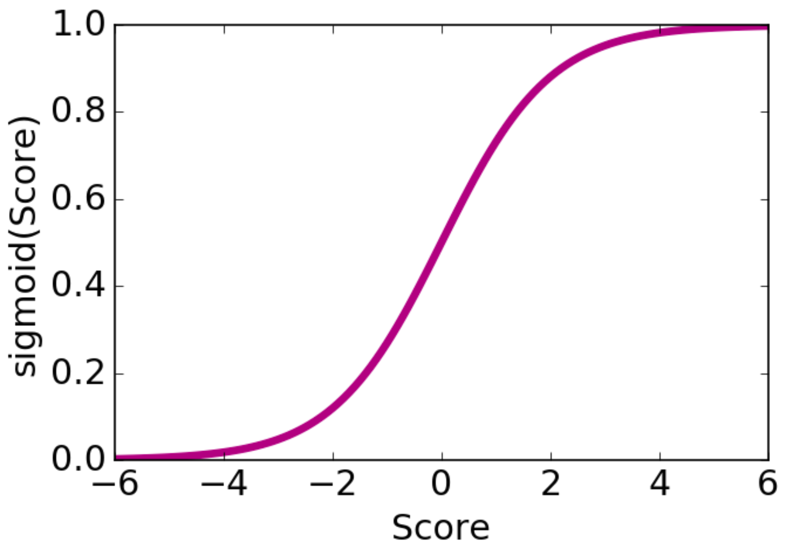
Question: How to specify p(y|x)?

Logistic Function

Use a function that takes numbers arbitrarily large/small and maps them between 0 and 1.

$$sigmoid(Score(x)) = \frac{1}{1 + e^{-Score(x)}}$$

$Score(x)$	$sigmoid(Score(x))$
$-\infty$	
-2	
0	
2	
∞	



Directly from: https://courses.cs.washington.edu/courses/cse416/21sp/

Interpreting Score

$$Score(x_i) = w^T h(x_i)$$

$-\infty$ $\quad \hat{y}_i = -1 \quad$ 0 $\quad \hat{y}_i = +1 \quad$ ∞

Very sure $\hat{y}_i = -1$

Not sure if $\hat{y}_i = -1 \; or \; \hat{y}_i = +1$

Very sure $\hat{y}_i = +1$

$\hat{P}(y_i = +1 | x_i) = 0$ $\quad \hat{P}(y_i = +1 | x_i) = 0.5$ $\quad \hat{P}(y_i = +1 | x_i) = 1$

0 $\quad 1$

$\hat{P}(y = +1 | x)$

Directly from: https://courses.cs.washington.edu/courses/cse416/21sp/

Approach 3: Logistic Regression

1. Convert each course evaluation statement into a "bag of words" representation
2. Specify form of $P(y_i = +1 | x_i, w) = sigmoid\big(score(x)\big) = \frac{1}{1+e^{-w^T x_i}}$
3. Write down (log) likelihood function
4. Maximize log-likelihood fn.
5. Use trained model to estimate p(+ | x)
6. If p(+ | x) > .5, predict "pro-5/474", otherwise, predict "anti"

Approach 3: Logistic Regression

1. Convert each course evaluation statement into a “bag of words” representation
2. Specify form of $P(y_i = +1|x_i, w) = \frac{1}{1+e^{-w^T x_i}}$
3. **Write down (log) likelihood function**

Approach 3: Logistic Regression

1. Convert each course evaluation statement into a "bag of words" representation
2. Specify form of $P(y_i = +1|x_i, w) = \frac{1}{1+e^{-w^T x_i}}$
3. Write down (log) likelihood function
4. **Maximize log-likelihood fn.**

Approach 3: Logistic Regression

1. Convert each course evaluation statement into a “bag of words” representation
2. Specify form of $P(y_i = +1 | x_i, w) = \frac{1}{1+e^{-w^T x_i}}$
3. Write down (log) likelihood function
4. **Maximize log-likelihood fn.**
   - No closed form solution!
   - Have to use gradient ascent/descent
   - Can do slightly better by using the second derivative as well to guide the movement through the space…
   - This is the **Newton-Raphson method**

Approach 3: Logistic Regression

1. Convert each course evaluation statement into a "bag of words" representation
2. Specify form of $P(y_i = +1|x_i, w) = \frac{1}{1+e^{-w^T x_i}}$
3. Write down (log) likelihood function
4. Maximize log-likelihood fn.
5. Use trained model to estimate p(+ | x)
6. If p(+ | x) > .5, predict "pro-5/474", otherwise, predict "anti"

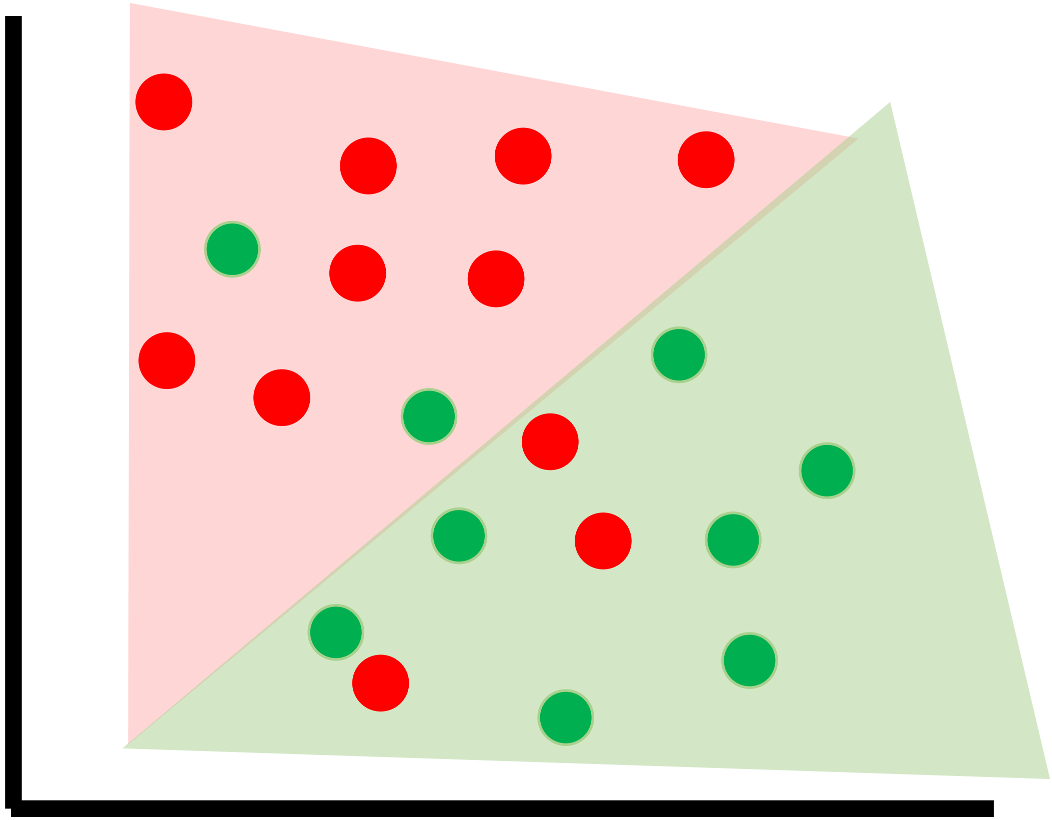
Some details we'll get to

- Do we have to use .5 as the threshold for classification?
  - No, and sometimes it's actually not a good idea
- Can we use logistic regression to learn non-linear decision boundaries?
  - Yes! **How?**
- Can we regularize logistic regression?
  - Yes! **How?**
- How do we get labels for data?
  - (Kind of discussed) Annotation! Lecture next week, PA3!
- Can we go beyond "bag of words"?
  - Yes! lectures post Spring break!
- How do we evaluate classifiers?
  - **A bit now, more next week**

OK!

What questions do you have?!

Evaluating classification models

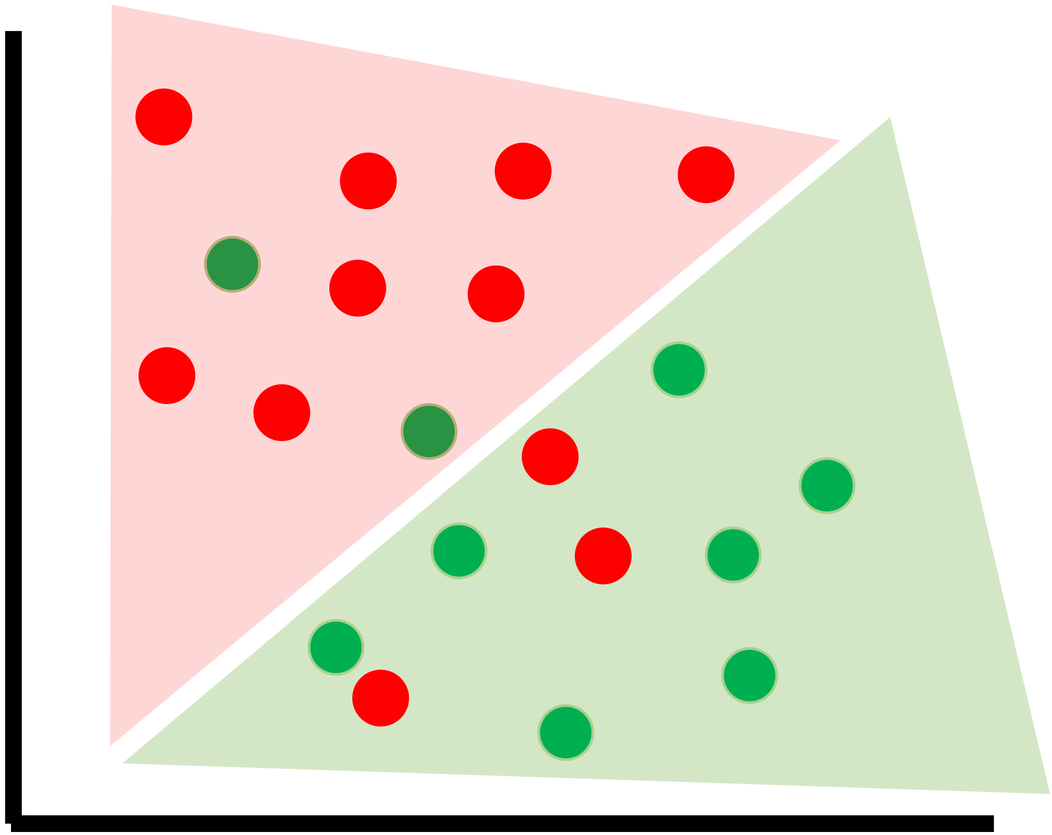


Our guess:

“Truth”

	8	3
	2	7

Accuracy – how many did we get correct?

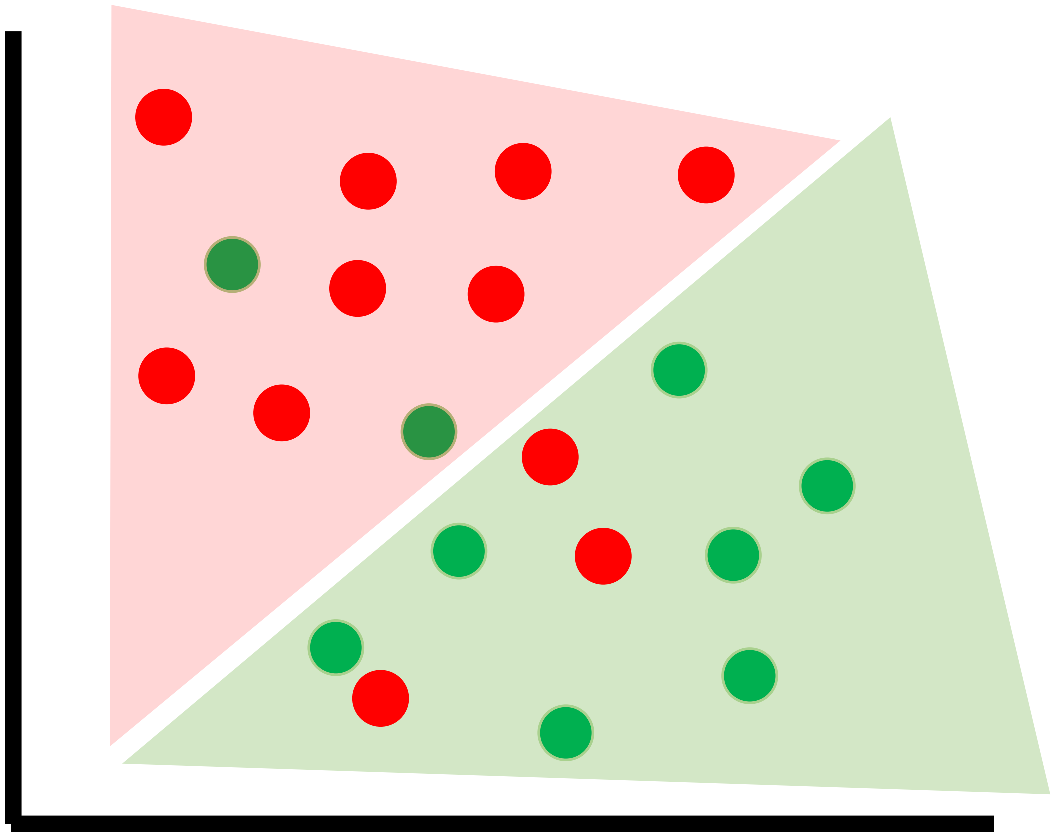


Our guess:

"Truth"		
	8	3
	2	**7**

Accuracy =
(8 + 7) / (8 + 7 + 2 + 3)
= .75

Precision - Of + guesses, how many actually +s?

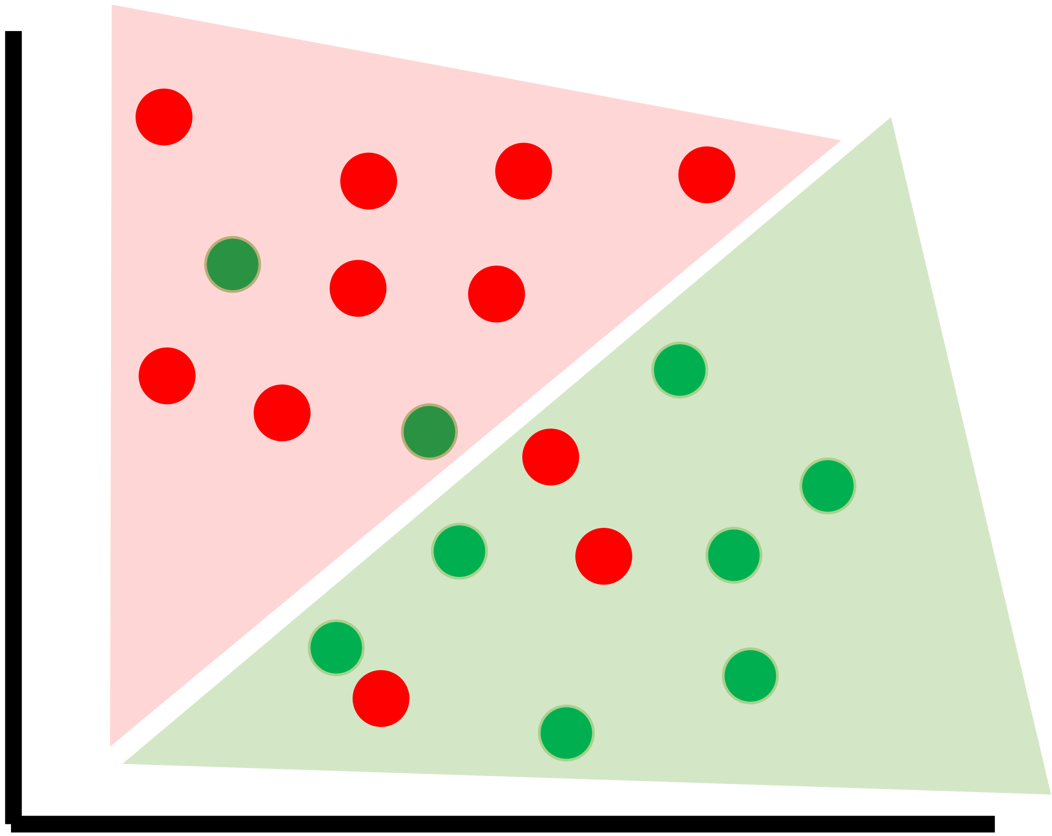


Our guess:

"Truth"		
red	8	**3**
green	2	**7**

Precision =
7 / (7 + 3) = .7

Recall - Of actual +, how many do we guess?



Our guess:

"Truth"		
red	8	3
green	2	7

Precision =
7 / (7 + 2) = .78

Which metric do we want?

- Diagnosing cancer
- Putting someone in jail
- Identifying someone for a restorative justice intervention

Evaluation Review

- Big idea:
  - Different metrics for different things
  - Evaluation metrics != loss function
- Other performance metrics (next week):
  - F1 Score
  - …
- Other considerations (next week)
  - Class imbalance (accuracy bad)
  - …

What is missing from these evaluations?

OK. Back to modeling

More soon!

Decision Trees for Classification: Main Ideas

- A **recursive** algorithm that **splits the feature space**
- Split can be based on a number of **criterion**
  - In the StatQuest: Gini, we'll go through one other (entropy)
- Prone to overfitting
  - Fix w/ early stopping (*pre-pruning)*
    - Fixed length
    - Stop if you don't get *that* much better
    - Min number of samples at leaves
  - Fix w/ pruning (*post-pruning)*
    - Grow full length trees, then *prune* back

Comparing to other models

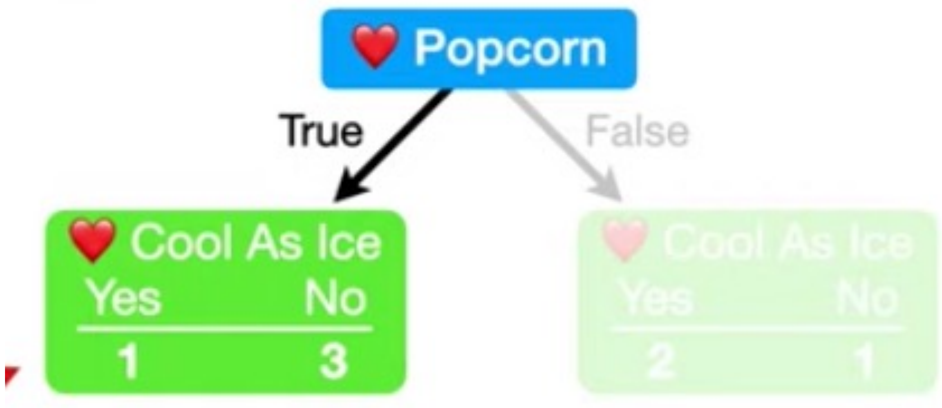
kNN

Logistic regression

Comparing to other models

Can you draw a true decision boundary for which Logistic regression is better than a tree?
What about vice versa?

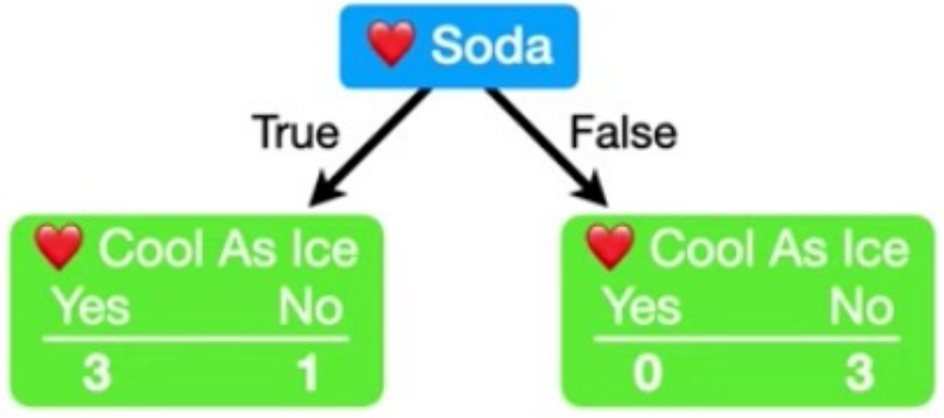
Another approach for splitting: *entropy*



$$1 - \sum_j p_j^2$$

$$-\sum_j p_j * log_2(p_j)$$

Another approach for splitting: *entropy*



$$1 - \sum_j p_j^2$$

$$-\sum_j p_j * log_2(p_j)$$

Preventing Overfitting

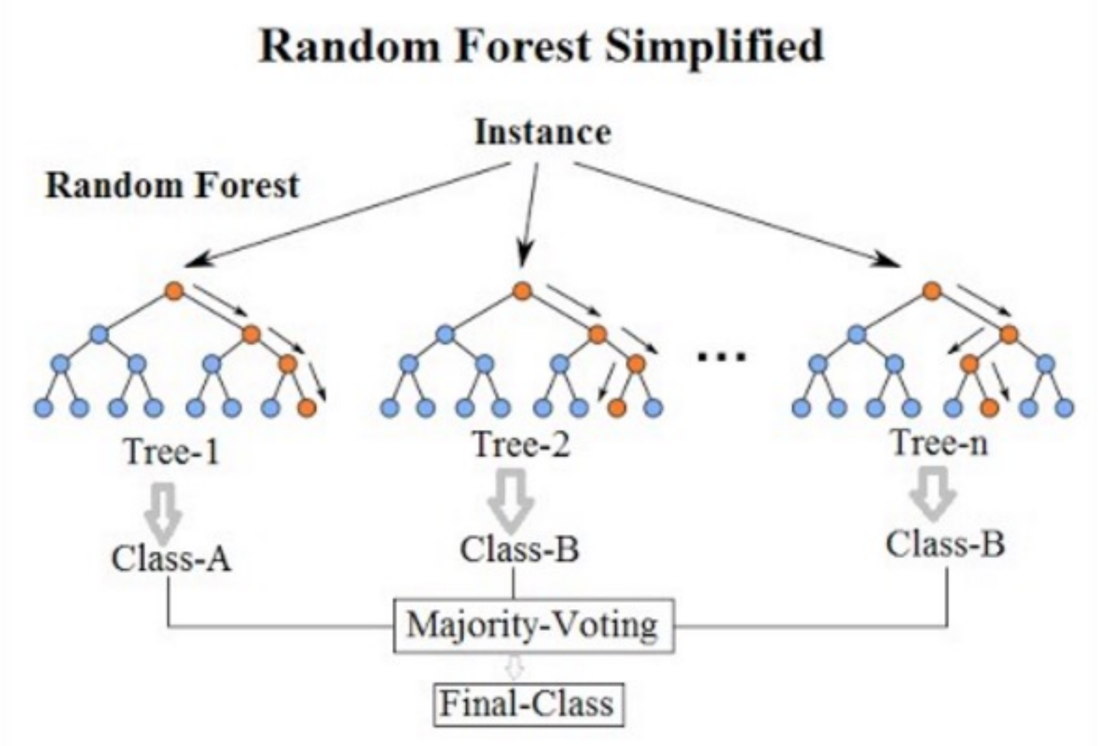
- Pre-pruning
  - Why do you think it is called **pre**-pruning?
  - Can you summarize three ways we know of to do pre-pruning?
- Post-pruning
  - What are two ways you can think of to do post-pruning?
- In theory, these make sense and should work!
- **In practice, decision trees are almost always pretty meh.**
- **Can we do better (with Trees?)**

Can we do better? Yes!

- Lots of ways to do better, but two high-level ideas:
  - **B**ootstrapped **Agg**regation (**bagging**)
    - Take a bunch of bootstrapped samples of the data, create a bunch of trees from them, and average across the trees
    - Best example: Random Forests
  - **Boosting**
    - Take a bunch of *weak learners* and apply them *sequentially*
    - Best examples:
      - Adaboost is the easiest to explain
      - Gradient boosting is very popular, but pretty difficult to explain

Random Forests

- Two key ideas:
  - Bagging
  - *Decorrelate trees* – use a random subset of features for each tree split ("feature bagging")
    - **Why (intuitively?)**



By Venkata Jagannath - https://community.tibco.com/wiki/random-forest-template-tibco-spotfirer-wiki-page, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=68995764

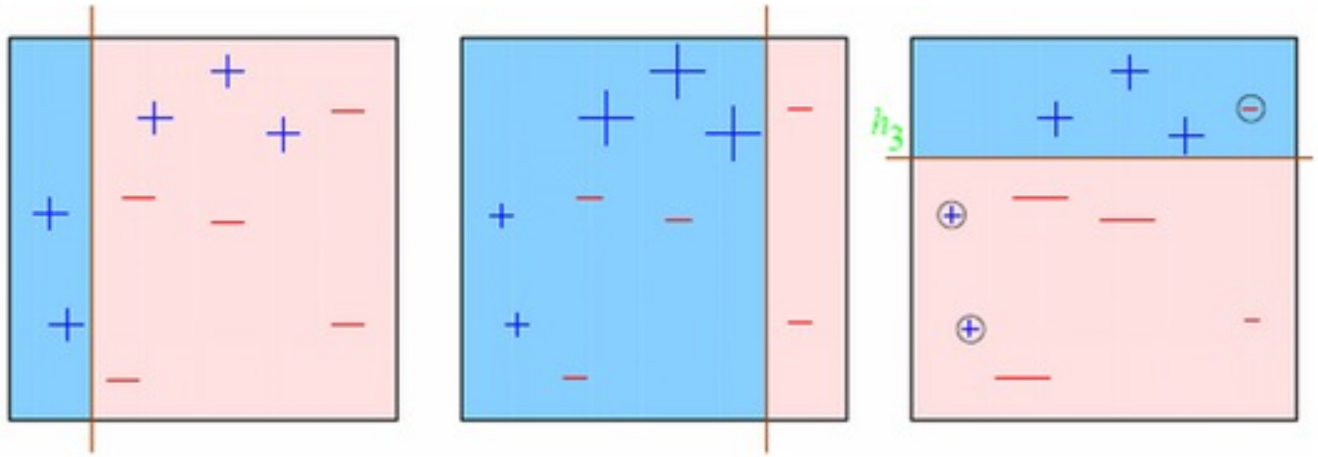
Random forests - details

- Hyperparameters (using sklearn parameters):
  - **N_estimators** – number of trees
  - **Max_features** – maximum features to consider at each split
  - **Min_sample_leaf –** minimum # leaves to split internal node
  - In practice, not super sensitive to these choices
- Out-of-bag samples
  - Anyone remember from video what these are?
- Variable importance
  - How would we compute the most important features?

Random forests – variable importance

- There are many ways to do this, we may discuss others later in the semester
- A brief intuition today: *permutation-based feature importance*
- General idea:
  - Train the full model
  - Record OOB accuracy
  - Shuffle feature values for a feature
  - Record new OOB accuracy
  - **How would you then identify important features?**

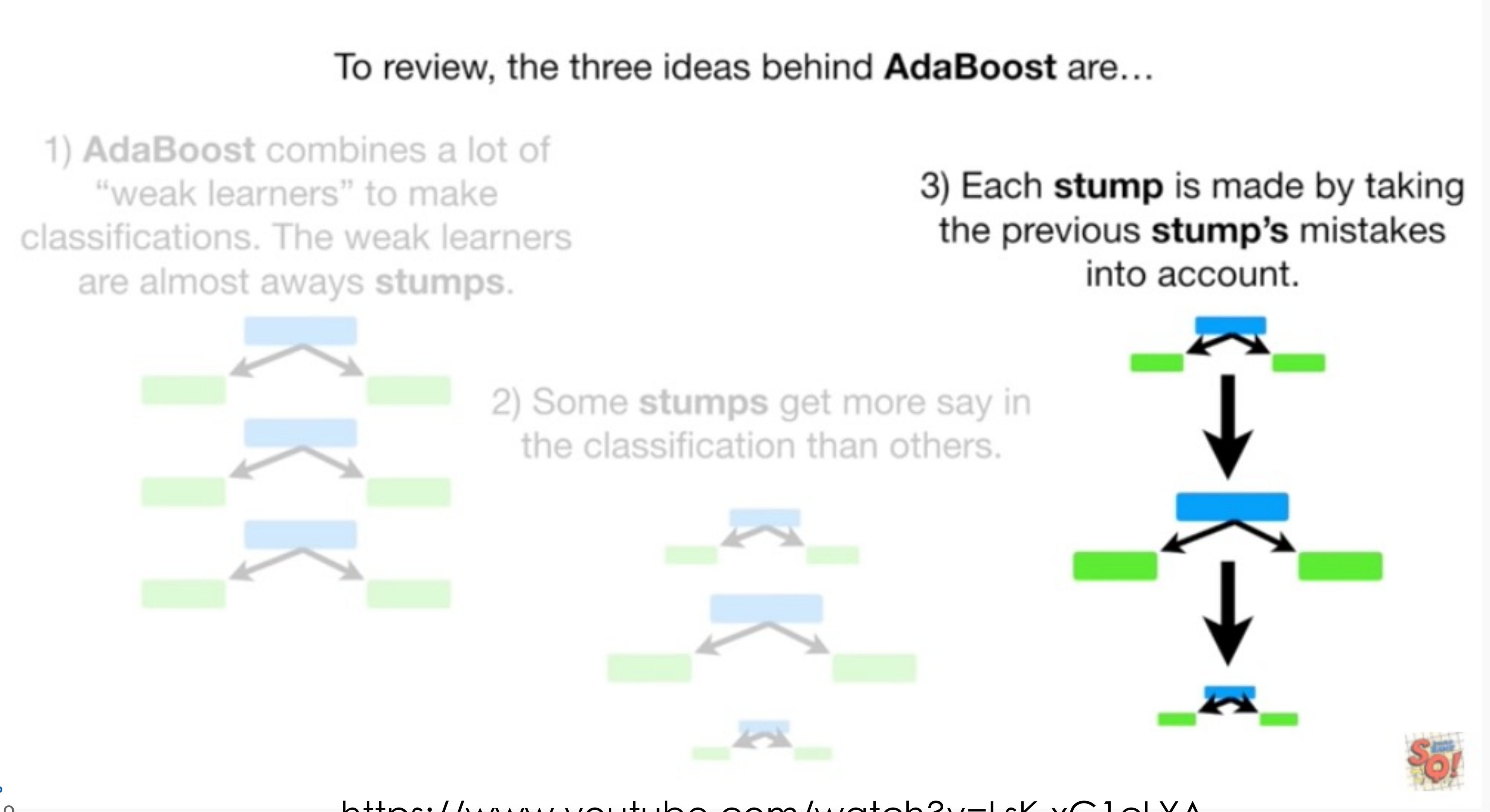
Boosting



- General idea: combine a bunch of weak learners in a smart way
- What is a **weak learner** (intuitively)?

https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

AdaBoost



https://www.youtube.com/watch?v=LsK-xG1cLYA

Adaboost

With:

- Samples $x_1 \ldots x_n$
- Desired outputs $y_1 \ldots y_n, y \in \{-1, 1\}$
- Initial weights $w_{1,1} \ldots w_{n,1}$ set to $\frac{1}{n}$
- Error function $E(f(x), y, i) = e^{-y_i f(x_i)}$
- Weak learners $h: x \to \{-1, 1\}$

For t in $1 \ldots T$:

- Choose $h_t(x)$:
  - Find weak learner $h_t(x)$ that minimizes ϵ_t, the weighted sum error for misclassified points $\epsilon_t = \sum_{\substack{i=1 \\ h_t(x_i) \neq y_i}}^{n} w_{i,t}$
  - Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)$
- Add to ensemble:
  - $F_t(x) = F_{t-1}(x) + \alpha_t h_t(x)$
- Update weights:
  - $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t h_t(x_i)}$ for i in $1 \ldots n$
  - Renormalize $w_{i,t+1}$ such that $\sum_i w_{i,t+1} = 1$
  - (Note: It can be shown that $\frac{\sum_{h_{t+1}(x_i) = y_i} w_{i,t+1}}{\sum_{h_{t+1}(x_i) \neq y_i} w_{i,t+1}} = \frac{\sum_{h_t(x_i) = y_i} w_{i,t}}{\sum_{h_t(x_i) \neq y_i} w_{i,t}}$ at every step, which can simplify the calculation of the new weights.)

Gradient Boosting

- Gradient boosting, and in particular, the implementation of gradient boosting (+ regularization) seem to be quite popular in the real world **(Thoughts?)**
- These models have an enormous number of hyperparameters.
- They essentially do "gradient descent with trees"
- Will not cover in detail.

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

$$\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \omega(f_t) + \text{constant}
\end{aligned}$$

https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Can we combine boosting and bagging? Yes!

Algorithm 8.3 *Bayesian Additive Regression Trees*

1. Let $\hat{f}_1^1(x) = \hat{f}_2^1(x) = \cdots = \hat{f}_K^1(x) = \frac{1}{nK}\sum_{i=1}^n y_i$.
2. Compute $\hat{f}^1(x) = \sum_{k=1}^K \hat{f}_k^1(x) = \frac{1}{n}\sum_{i=1}^n y_i$.
3. For $b = 2, \ldots, B$:
   (a) For $k = 1, 2, \ldots, K$:
      i. For $i = 1, \ldots, n$, compute the current partial residual
$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i).$$
      ii. Fit a new tree, $\hat{f}_k^b(x)$, to r_i, by randomly perturbing the kth tree from the previous iteration, $\hat{f}_k^{b-1}(x)$. Perturbations that improve the fit are favored.
   (b) Compute $\hat{f}^b(x) = \sum_{k=1}^K \hat{f}_k^b(x)$.
4. Compute the mean after L burn-in samples,
$$\hat{f}(x) = \frac{1}{B-L}\sum_{b=L+1}^{B} \hat{f}^b(x).$$

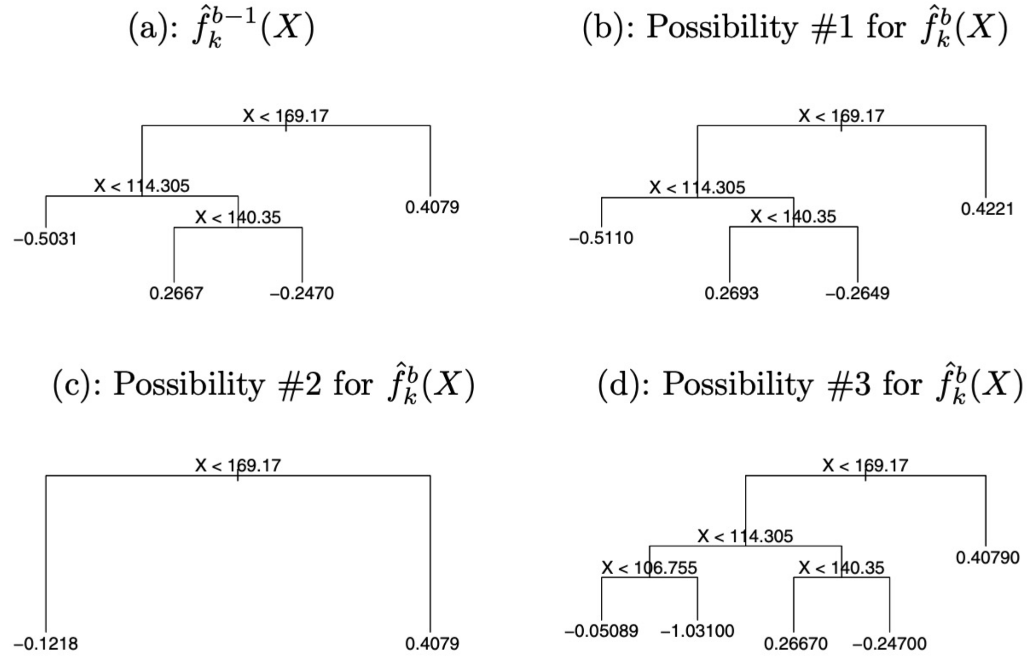


FIGURE 8.12. *A schematic of perturbed trees from the BART algorithm.* (a): *The kth tree at the $(b-1)$st iteration, $\hat{f}_k^{b-1}(X)$, is displayed. Panels* (b)–(d) *display three of many possibilities for $\hat{f}_k^b(X)$, given the form of $\hat{f}_k^{b-1}(X)$.* (b): *One possibility is that $\hat{f}_k^b(X)$ has the same structure as $\hat{f}_k^{b-1}(X)$, but with different predictions at the terminal nodes.* (c): *Another possibility is that $\hat{f}_k^b(X)$ results from pruning $\hat{f}_k^{b-1}(X)$.* (d): *Alternatively, $\hat{f}_k^b(X)$ may have more terminal nodes than $\hat{f}_k^{b-1}(X)$.*

https://hastie.su.domains/ISLR2/ISLRv2_website.pdf

Code examples