ML and Society

Mar 1, 2022

First in-class presentation this Thursday

Slides due by 5pm TOMORROW

note @25 1 views

Autolab accepting first presentation slides

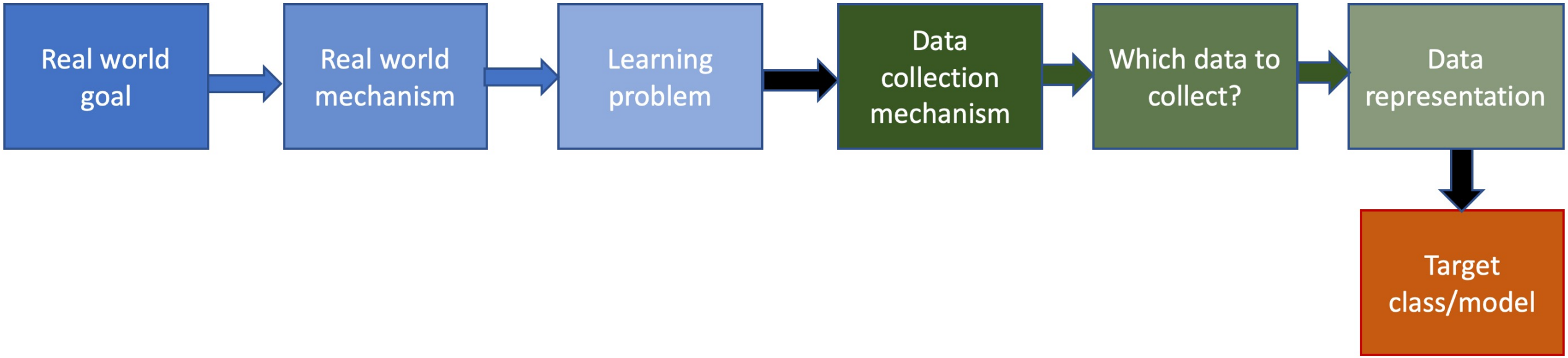
Autolab is now accepting submission for your groups presentation slides. Some reminders:

- The slides are due by **5pm on Wed, Mar 2**
- The slides should be uploaded as **PDF only**
- The presentation is Thursday in class (Mar 3)
  - The order among groups (picked randomly) will be as follows:
    - Multiple notions of fairness
    - Human Acceptance
    - Teaching tools
    - Algorithmic Auditing
- Aim for **15 minutes** of presentation
  - Followed by 5 mins of Q&A

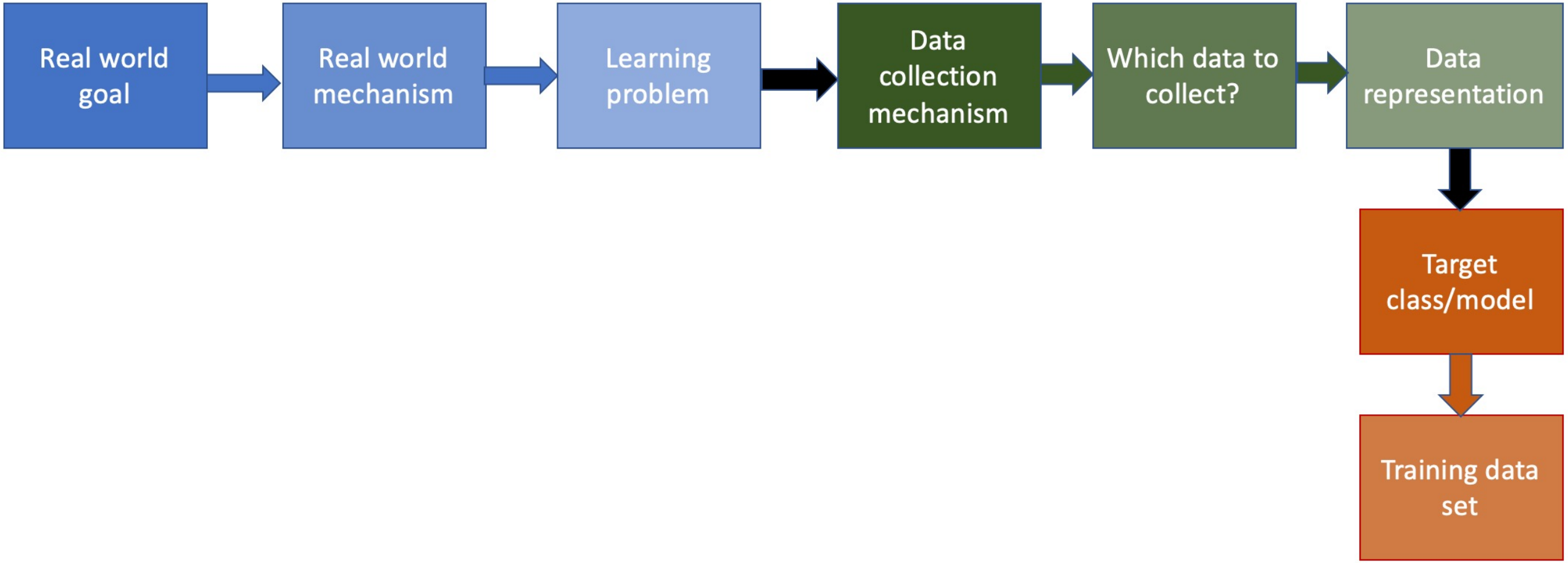
Like with the progress summary (@16), all the groups should be setup on Autolab but **please double-check the following:**

- Make sure you can see your group. To do so, follow these steps
  - Go to CSE 440/441/540 > First Presentation Slides (under Project)
  - Go to Options > Group options
  - Check to make sure your and your partners' emails are in there.
  - **DO NOT**
    - Click on invite another student to the group or the leave group buttons
- Please submit a dummy PDF well in advance of the deadline and check with ALL your group-mates to make sure they see your submitted PDF in their account.
  - I will grade the last PDF submitted by any of the group members.
  - I recommend that you designate one person in your group as the official submitter but whatever works for your group is fine with me.

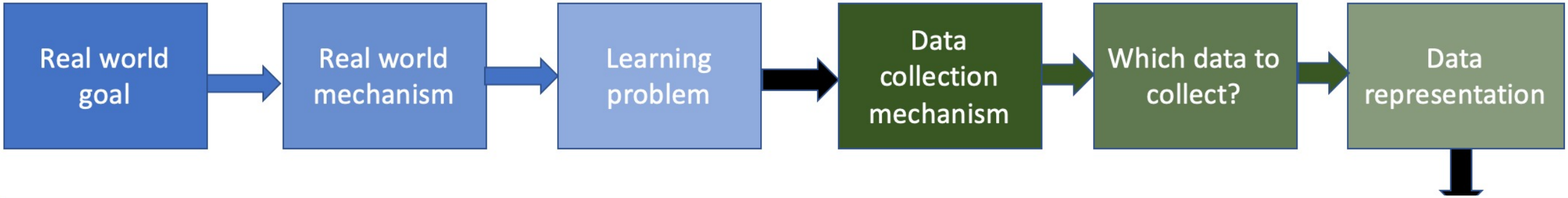
We finished the model class step



Finally to next step in the ML pipeline!



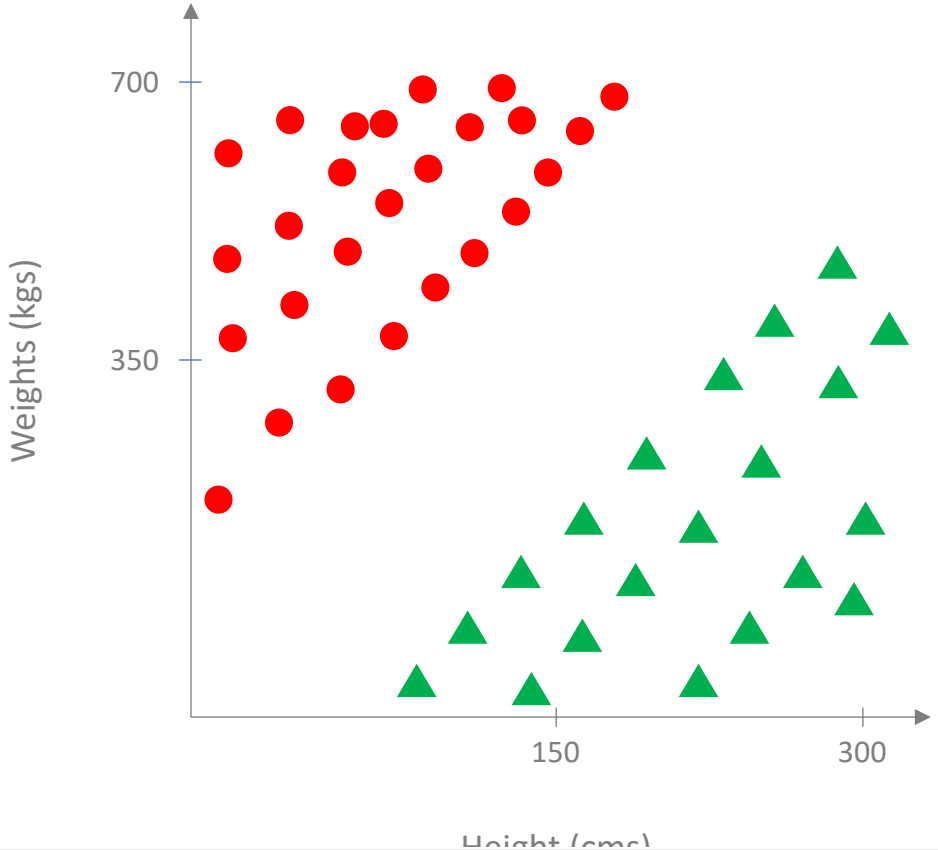
Choosing the training dataset



Choosing a training dataset

At this point in the pipeline, we already have access to a dataset. We now need to decide on which part of the dataset we should train our model that we choose in the previous step in the pipeline.

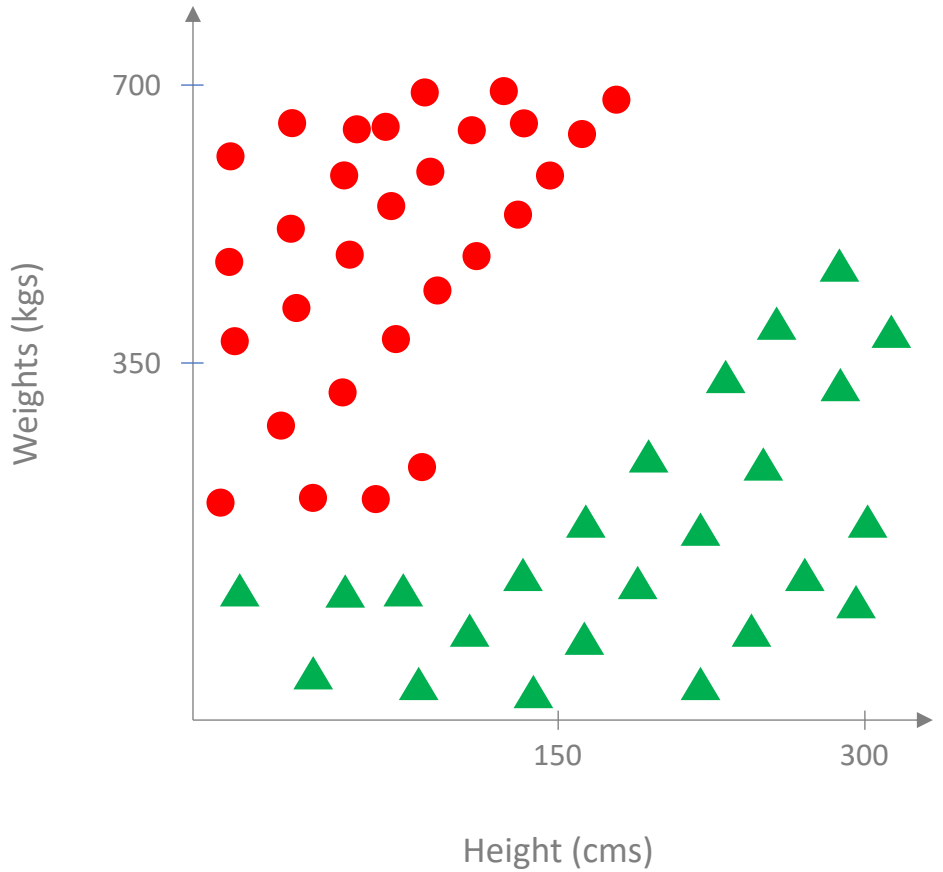
Training dataset = entire dataset?



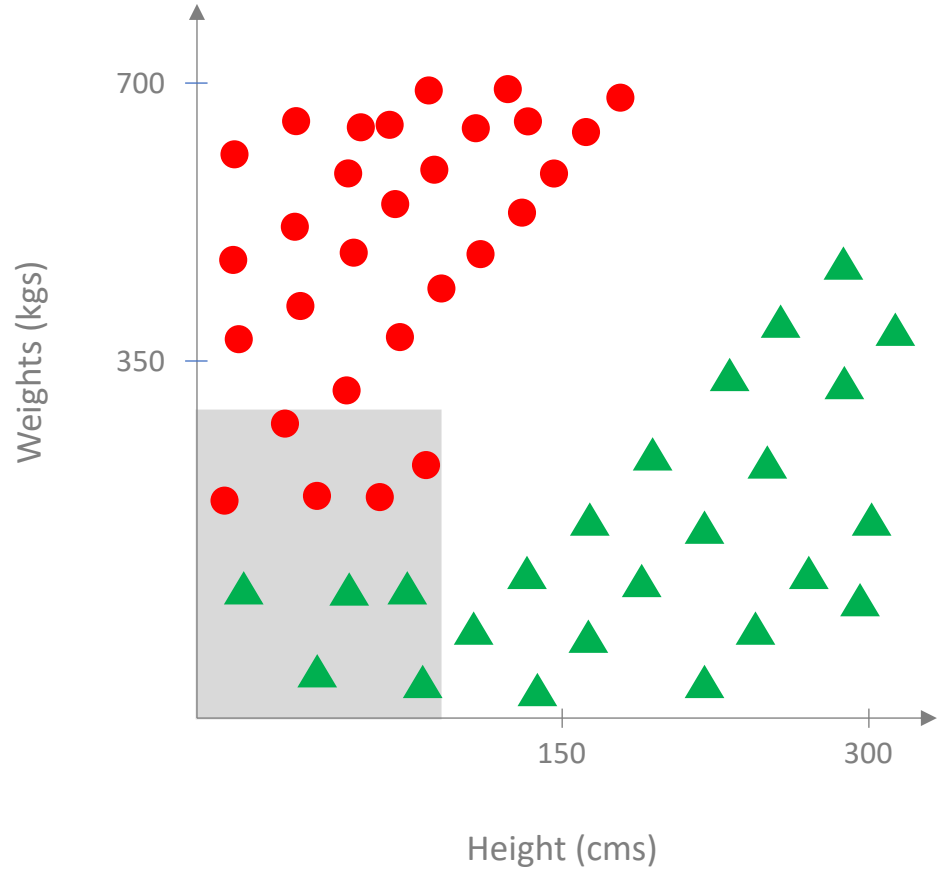
Changing representation of positively labeled points

We are switching how we represent a positively labeled point in the dataset from a green circle to a green triangle, with the hope that it is more accessible to readers. At a later point, we will update the earlier pictures as well. We will remove this note when that is done.

A more realistic dataset

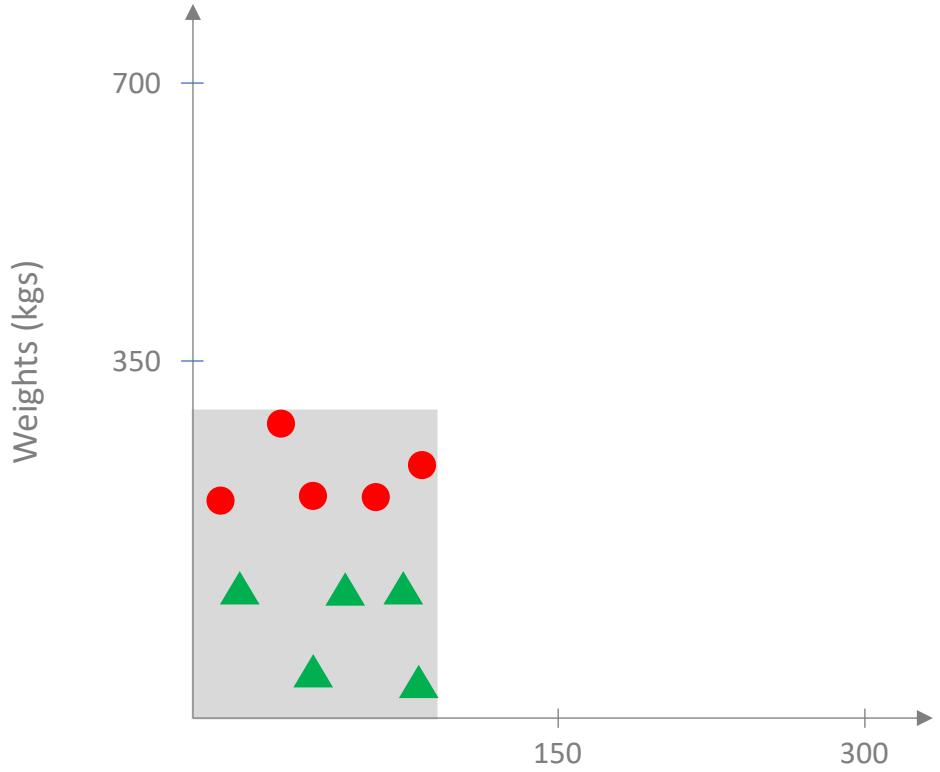


We can only collect part of the ground truth



What if we had access to the entire ground truth?

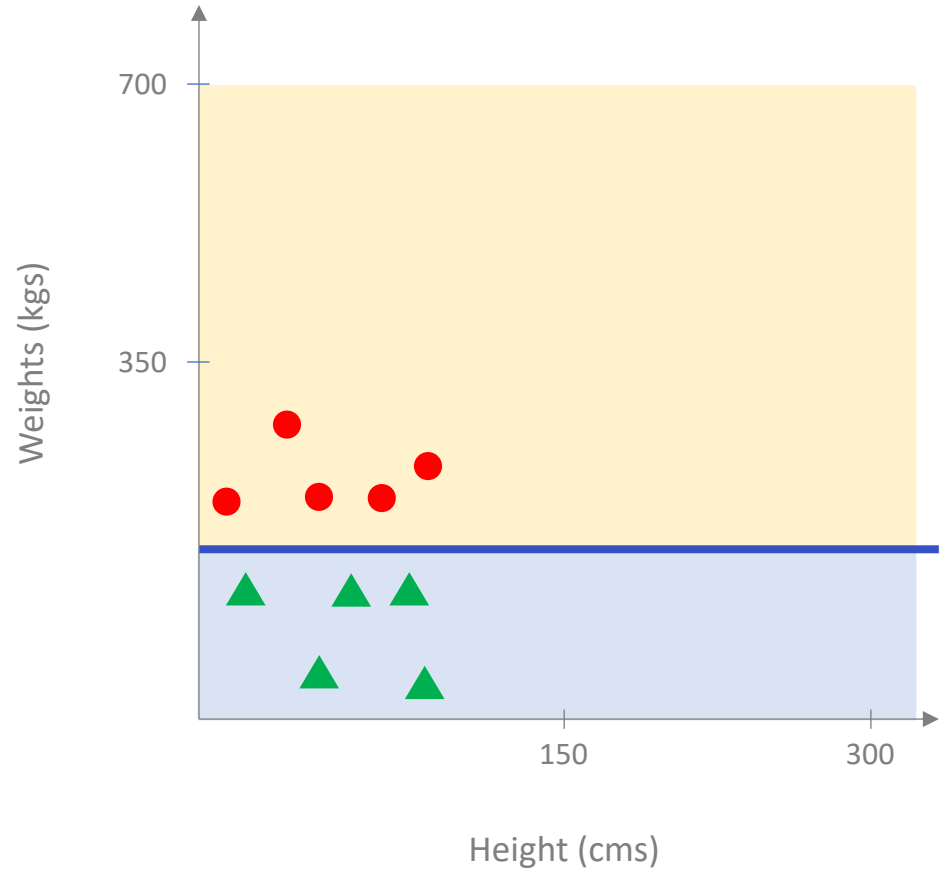
This is our dataset



Focus on linear models

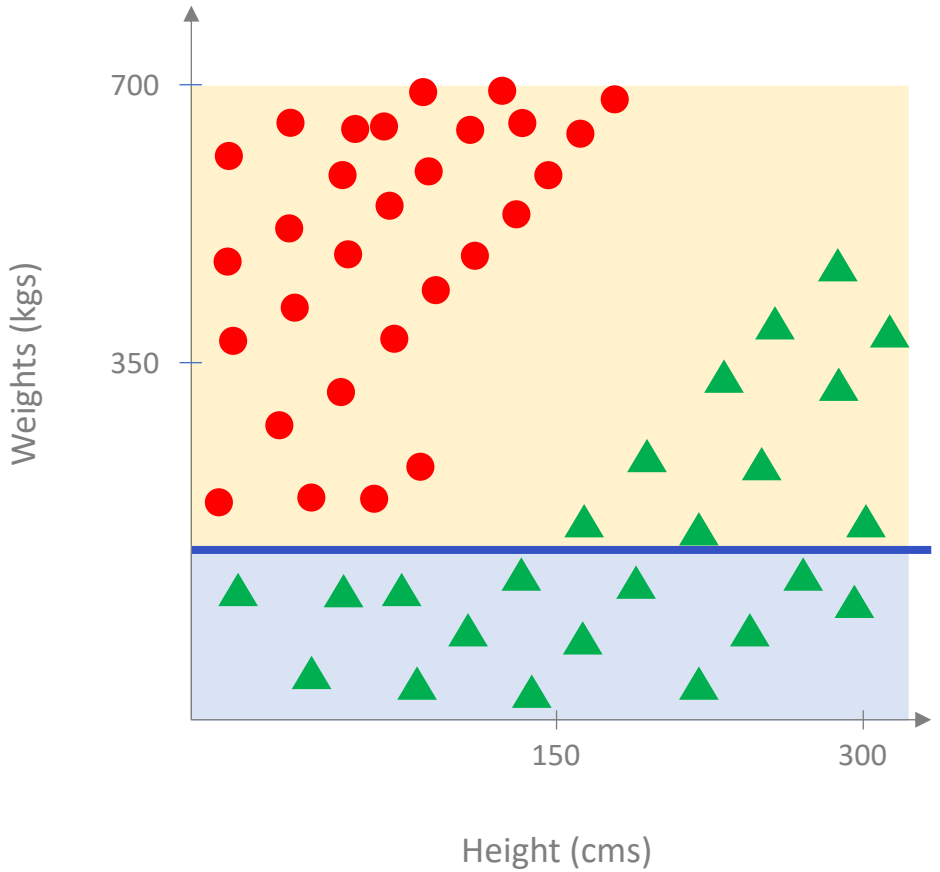
For simplicity, we will assume that we have chosen the class of linear models in our previous step. This will make the pictures easier to draw etc.

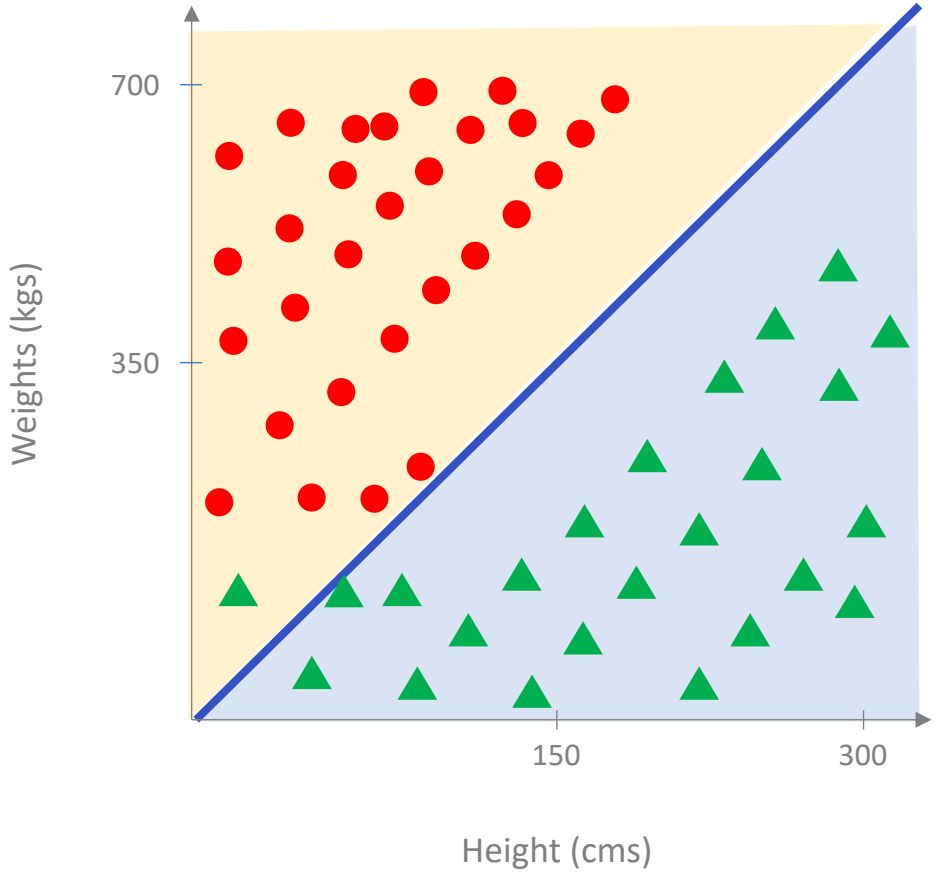
Let’s find the optimal linear model



What is an optimal linear model?

Going back to the entire ground truth



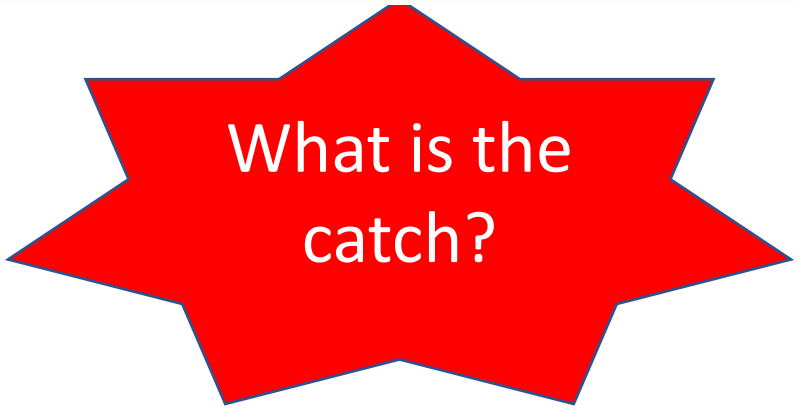


What if we had a true random sample?

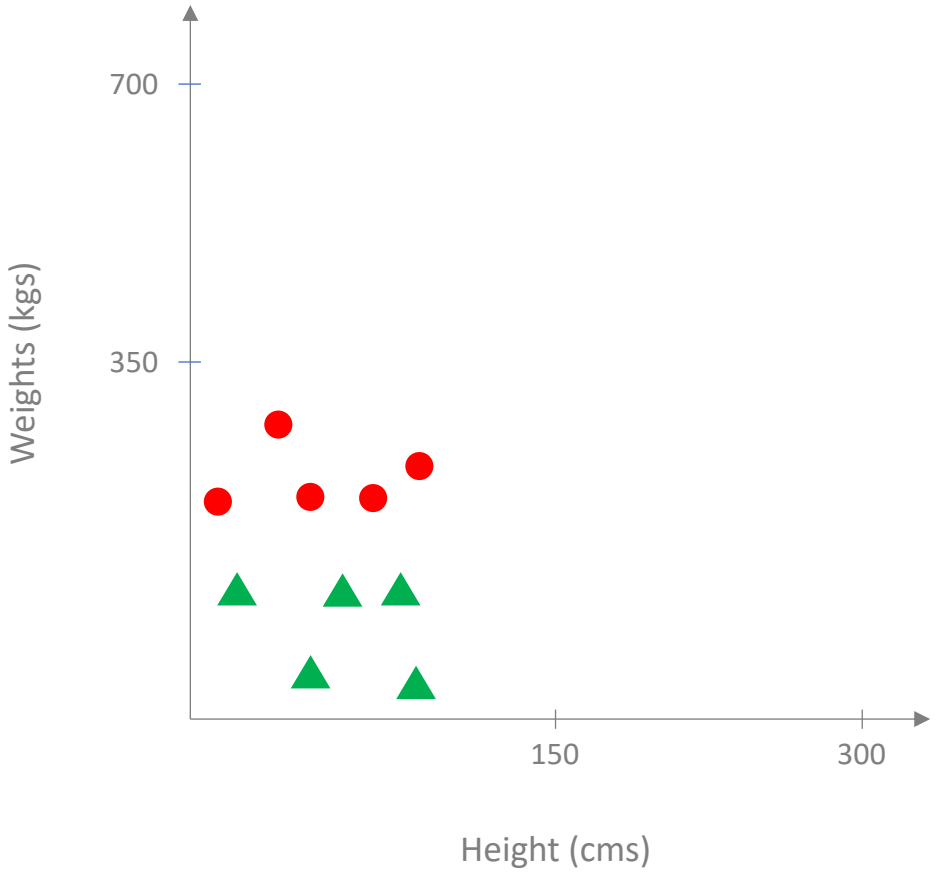
What IF our dataset was a true random sample of the ground truth

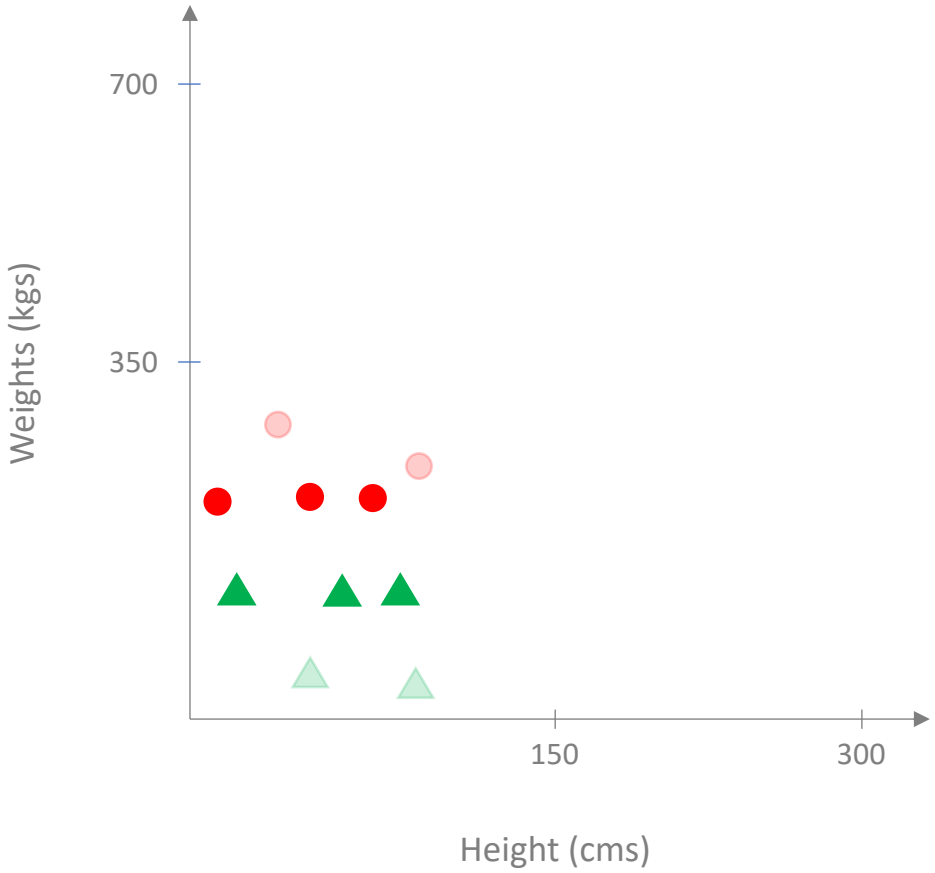
Now consider the following scenario: what if our dataset is a true random sample from the underlying distribution? In this case, we can actually **guarantee** that the optimal linear model for the random sample will be **very close** to the optimal linear model for the entire population (assuming there are **enough** random samples in the dataset). We will not go into this but the mathematical buzzword that makes this work is that the VC dimension of linear models is constant (in case you care it is 3).

*"What **if** our dataset is a true random sample from the underlying distribution?"*

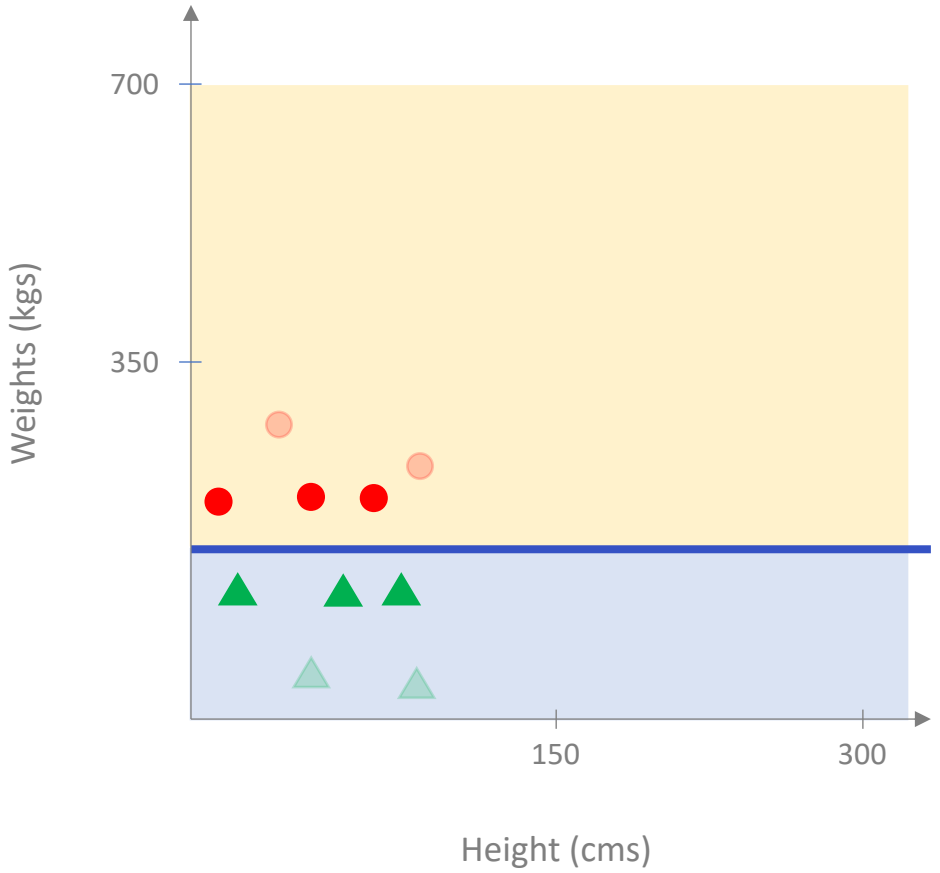


Which subset to pick? Choice 1

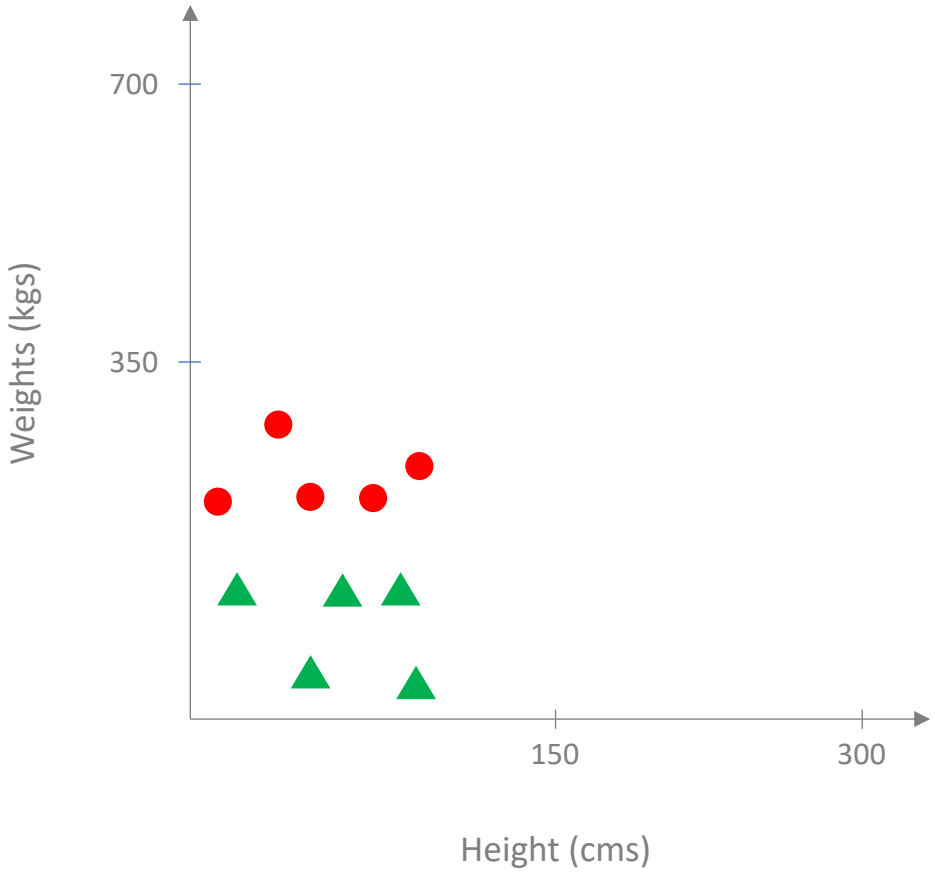


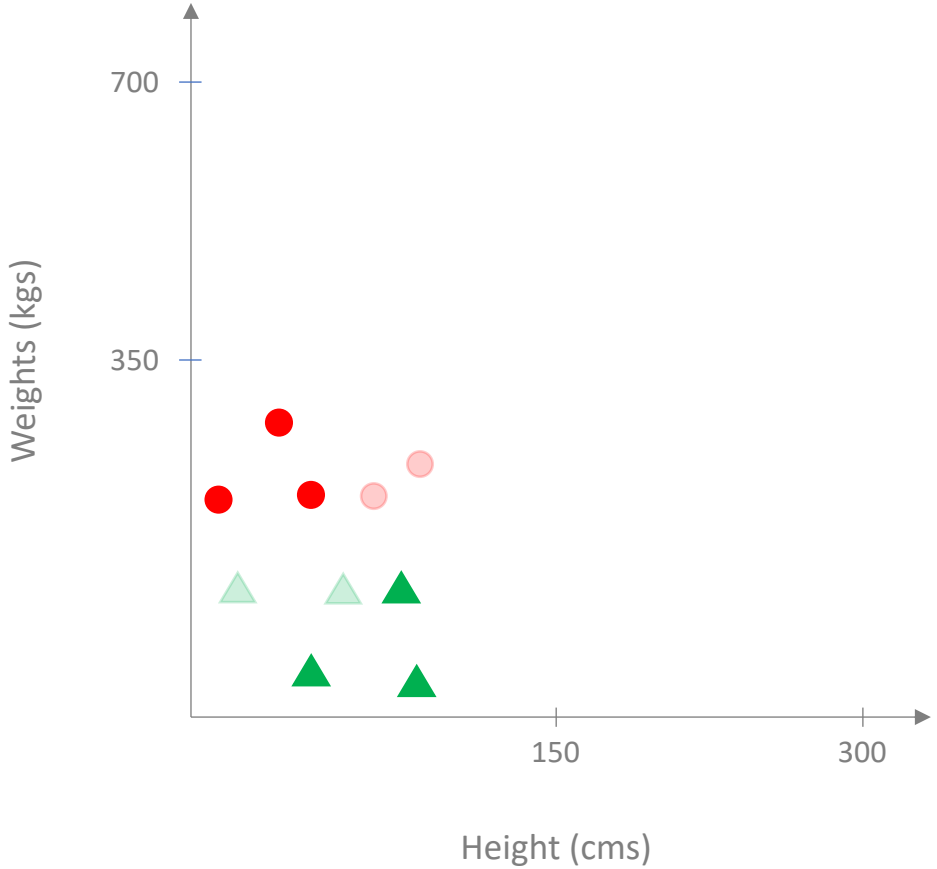


Optimal linear model for Choice 1

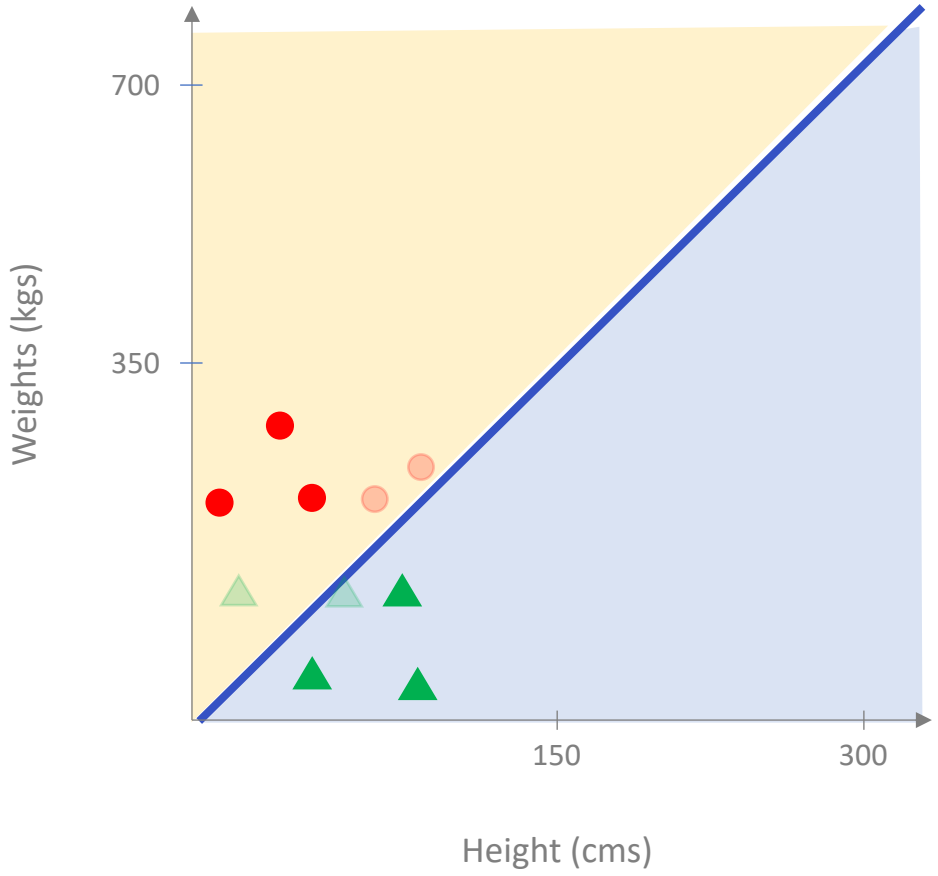


Choice 2 for subset

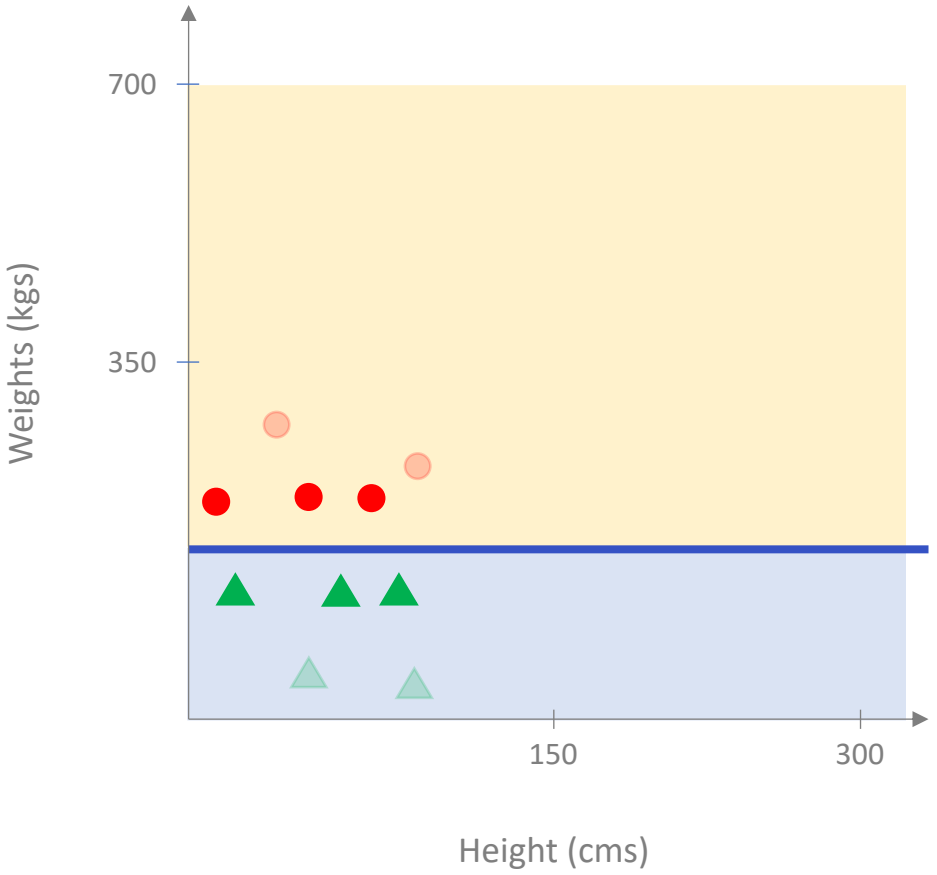


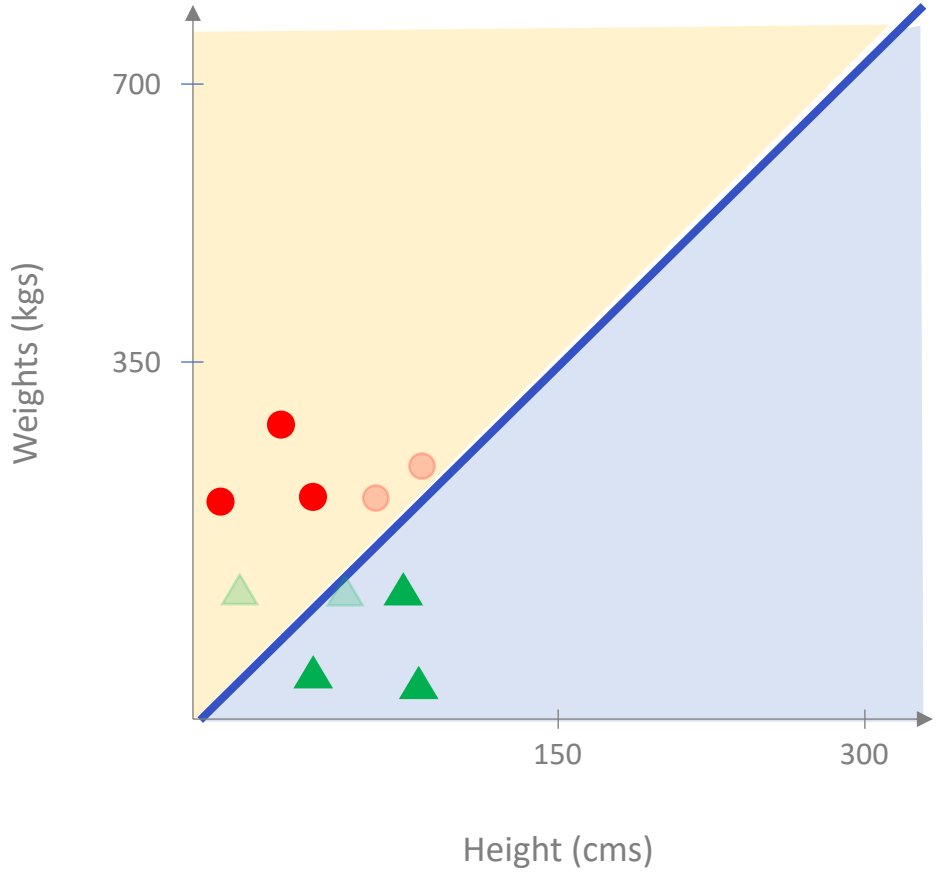


Optimal linear model for Choice #2



How do we pick the “best” subset?





Answer: we don’t (or can’t)!

How do we pick the best subset?

So the pesky question now becomes: how do we figure out the "best" subset?

The problem is that answering this question is just not possible: to figure out the best possible subset, we need to know the underlying ground truth. But the whole reason we are doing prediction is that we do **not have access** to the ground truth in thee first place. So how do we get around this?

The answer is that we punt and "guess." In other words, we do the following:

1. Figure out what fraction of dataset we want to use to obtain out training set. I.e., we first figure out the **size** of the training set relative to the dataset. In the above examples, the training sets were of size 6 while the dataset had 10 points in it-- i.e. we picked 60% of the dataset as the size of our training set.
2. We then pick a **random subset** of the size determined in the previous step and use that as the training dataset.

Two kinds of randomness

Now is a good time to recap the two random subsets of data we talked about.

The first random subset we considered was a random subset of the **ground truth**. This subset has very nice theoretical properties is something we do **not have access to in real life**.

The second random subset we considered was a random sunset of the **given dataset**. Thus, in this case we are looking at a random subset that **we can always construct**.

Beyond random training dataset

Beyond random training datasets

As it turns out picking a random training dataset from the given dataset is not the only way to construct training dataset. For example, in many situations, one might construct **multiple** training datasets and use those to get multiple models. Then one could pick the "best" model based on how well these models do on the test dataset. A well-known example of this is cross validation. However, to keep our discussion simple, we will not consider these more (though not that much) complicated methods to create training (as well as testing) datasets.

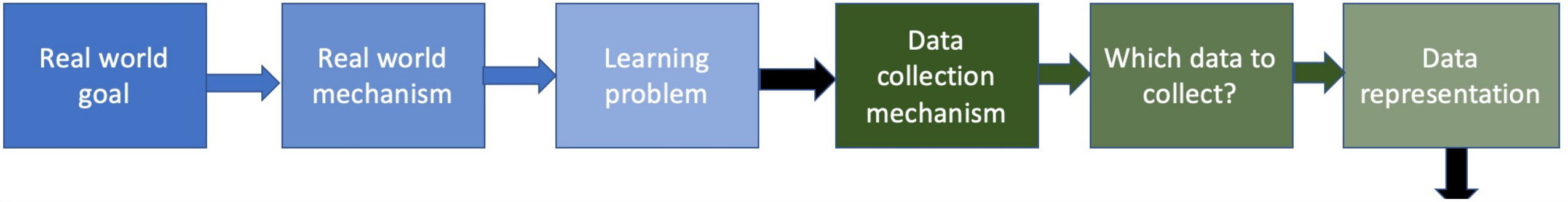
Passphrase for today: Grace Wahba

Grace Goldsmith Wahba

It is easy (and accurate) to say that Grace Wahba is this country's most eminent female statistician, but it is also misleading: Grace is in the top few among *all* American statisticians, and is in fact one of our most important applied mathematicians. She was also the first female faculty member in the Department of Statistics at UW–Madison. Supervising the theses of 39 graduate students from four continents, her career there lasted 51 years, ending with **retirement in August 2018**.

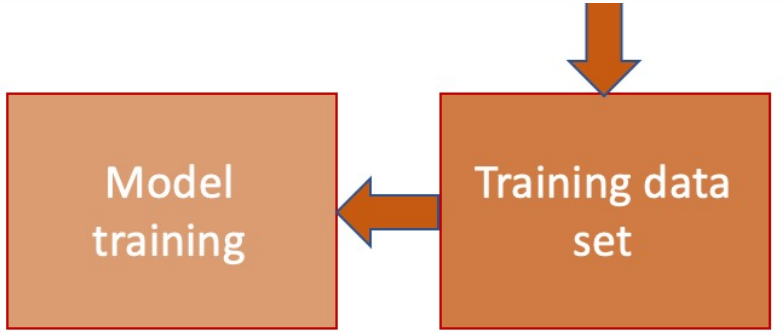
Grace almost owns the word “spline” as it is used in statistics; her 1990 monograph *Spline Models for Observational Data* has been a scientific best-seller. It is based on a series of fundamental papers written with various co-authors in the preceding years. Of these, the most influential might be 1979's “Generalized cross-validation as a method of choosing a good ridge parameter”, with Gene Golub and Michael Heath, which introduced the GCV (generalized cross validation) criterion. The hallmark of Grace Wahba's work is a combination of high-powered mathematics, often involving functional analysis ideas such as reproducing kernel Hilbert spaces, but with the practical problems of real data analyses kept firmly in view. The very best statisticians always turn out to be good scientists as well as

The next step…

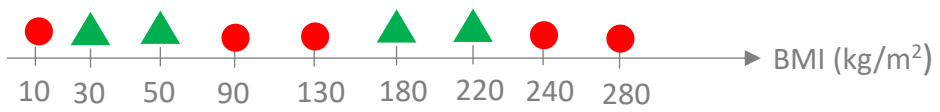


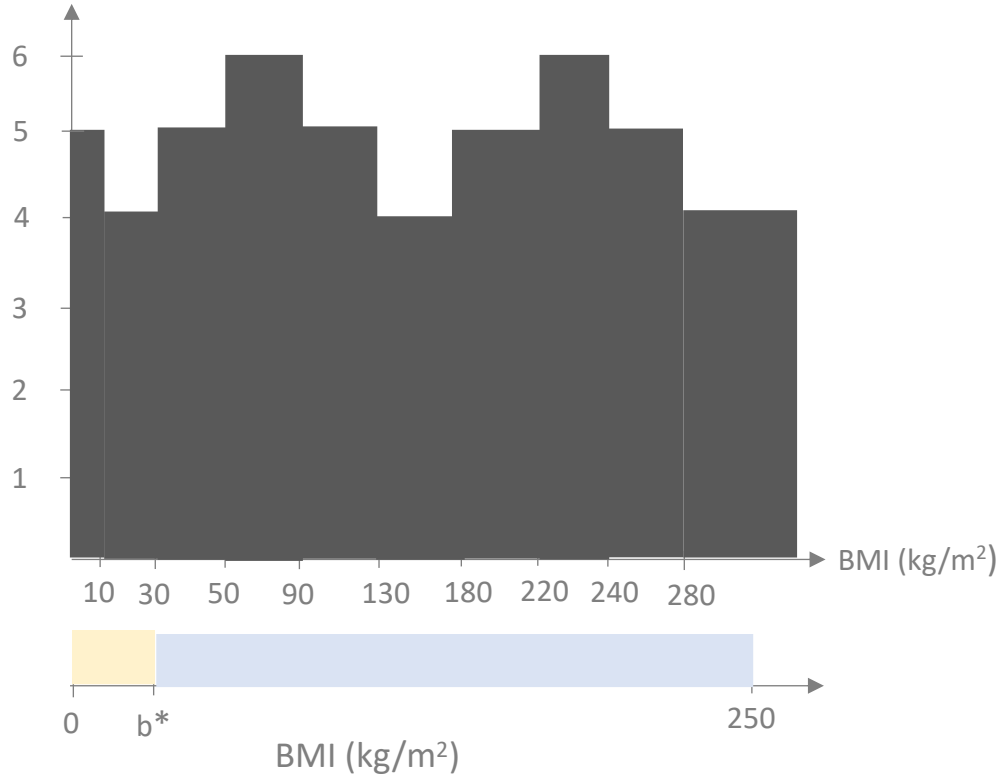
Linear model in one variables

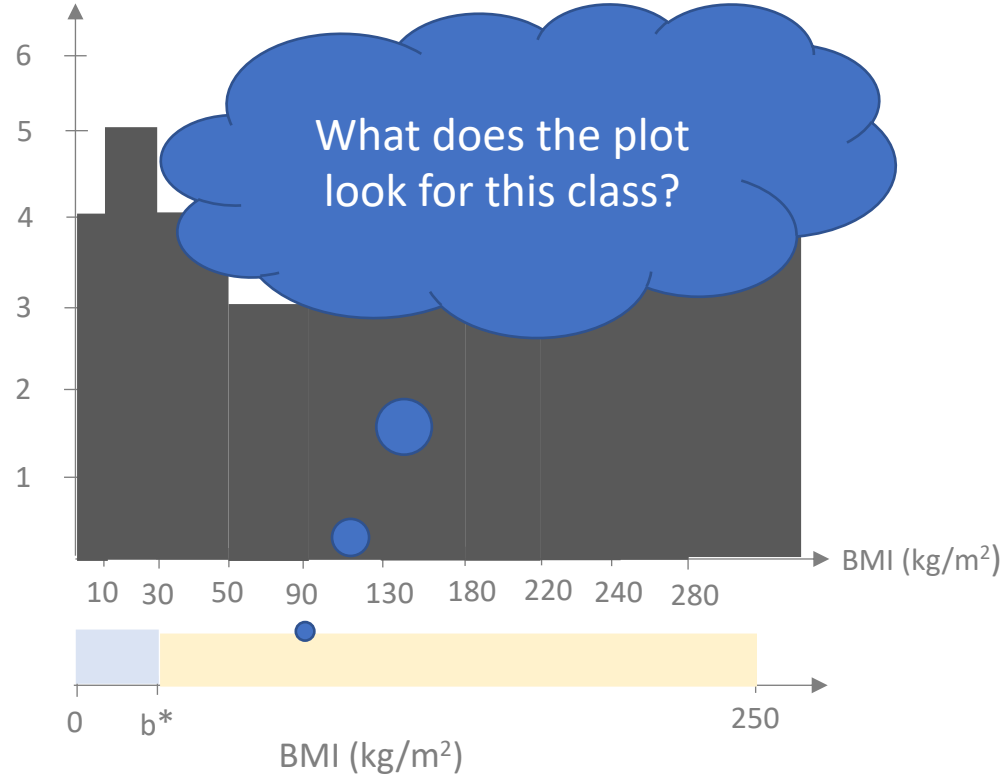
Even though we know how to efficiently solve the training problem for linear models in one variable, we will focus on this special case since it makes it easier to explain the relevant concepts.



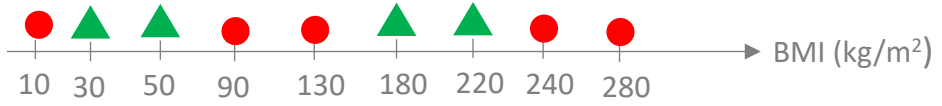
Minimize number of mis-classifications



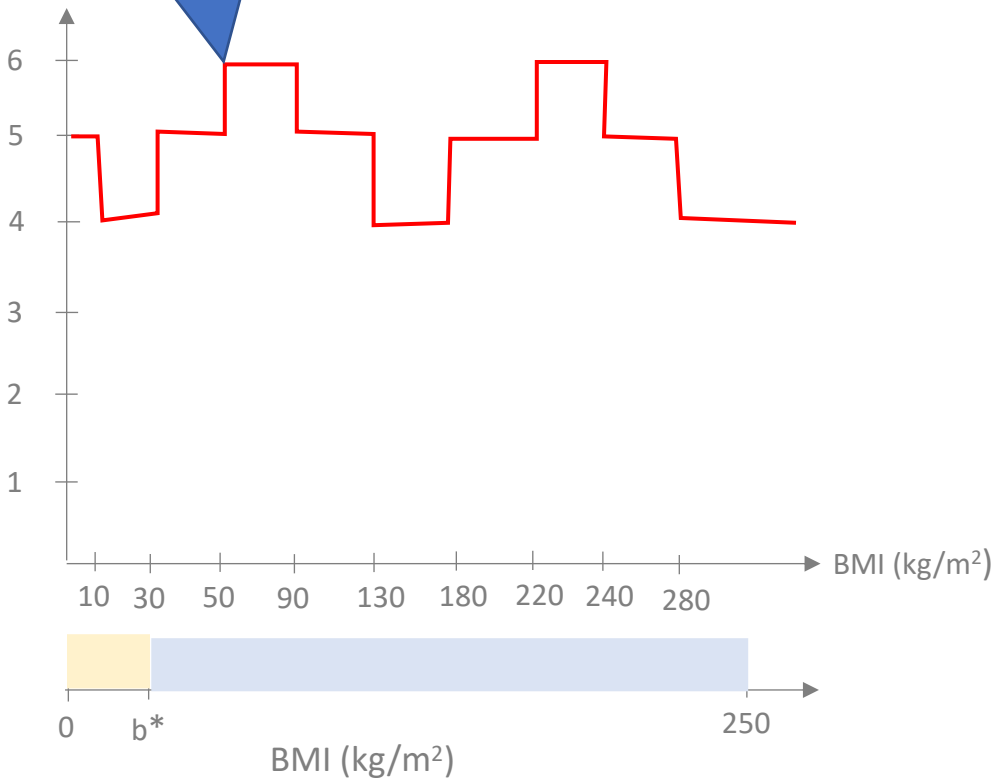




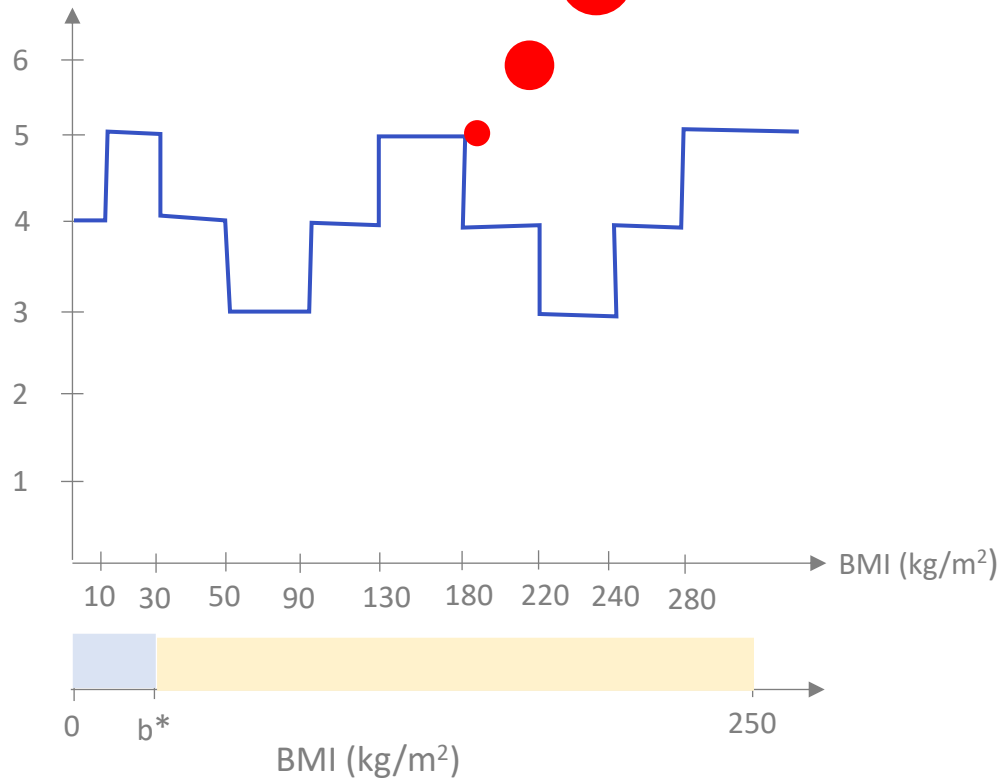
The actual “loss” function



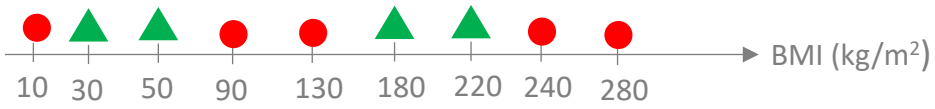
Only check the corner points!

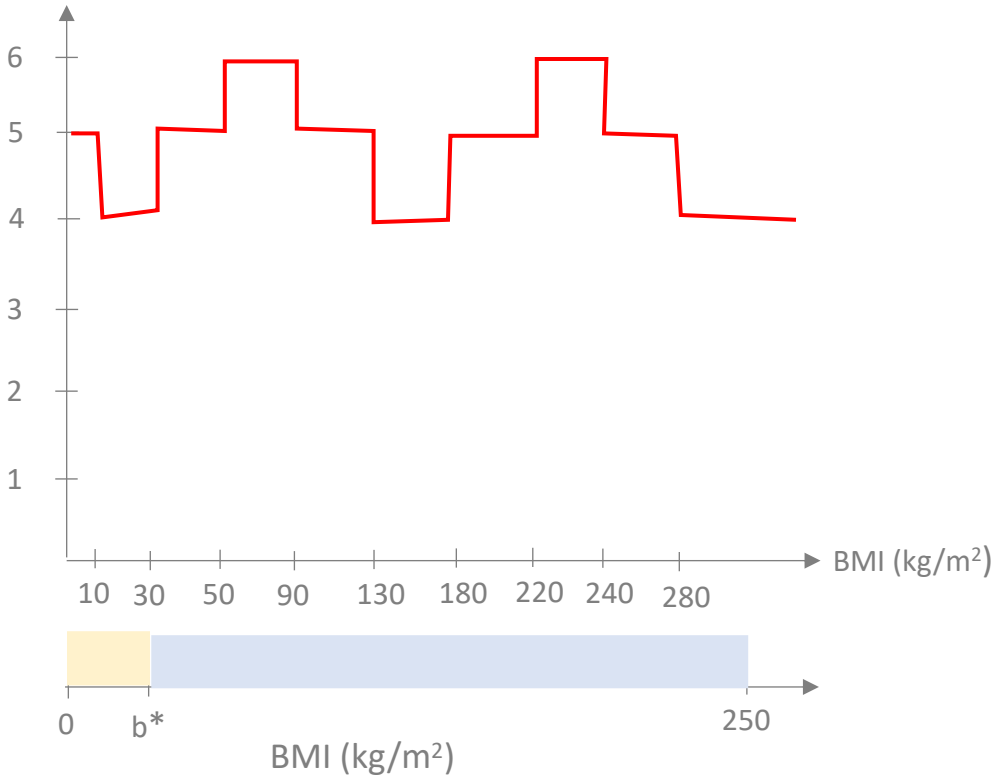


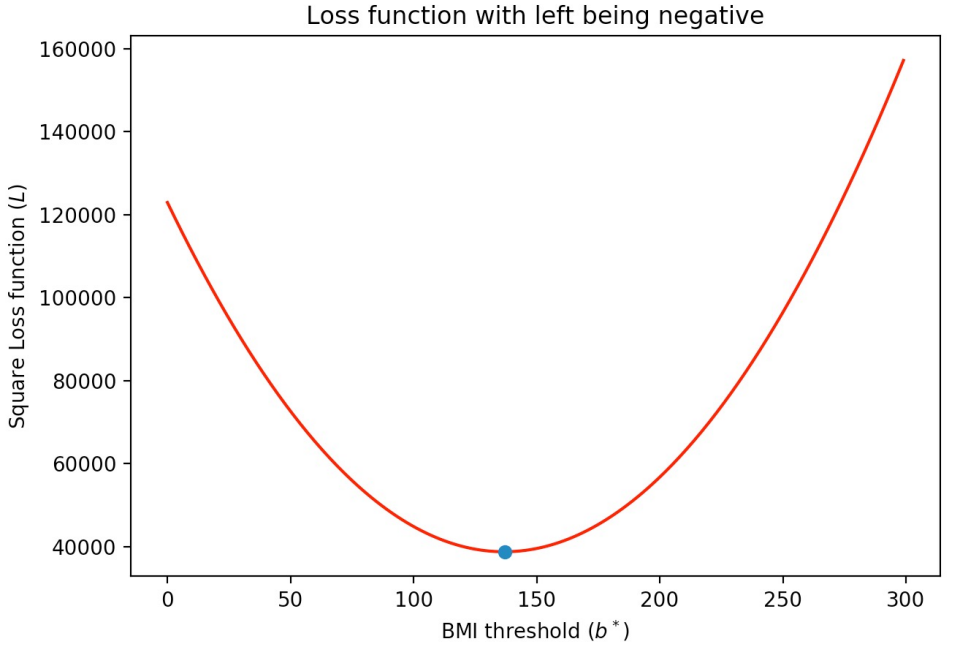
Number of corners grows like n^d



A “better” loss function







What is a loss function?

Loss function

For simplicity, we will define a loss function for linear models on one variable (though the definition is fairly easy to generalize to other settings). Recall that the training dataset is a collections of input value and label pairs: i.e. $(b_1, y_1), \ldots, (b_n, y_n)$. Then the loss function L for linear model ℓ is defined as follows. For any $1 \leq i \leq n$, $L(\ell, i) \geq 0$ is defined to be the `loss` that the model ℓ incurs for the pair (b_i, y_i). For example, for misclassification, the loss function is defined as 0 if $\ell(b_i)$ is the same as y_i and 1 otherwise. Bit more mathematically:

$$L(\ell, i) = \begin{cases} 0 & \text{if } \ell(b_i) = y_i \\ 1 & \text{otherwise} \end{cases}.$$

The `overall loss` for the model ℓ is to just sum up the losses for each of the n points. In other words,

$$L(\ell) = \sum_{i=1}^{n} L(\ell, i).$$

For the example of mis-classification the above is exactly the same as the total number of mis-classified points.

Square Loss Function

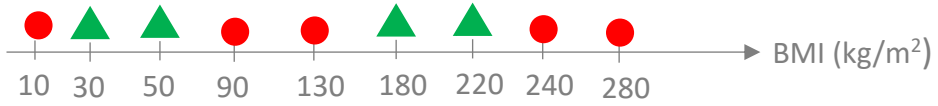
Square Loss function

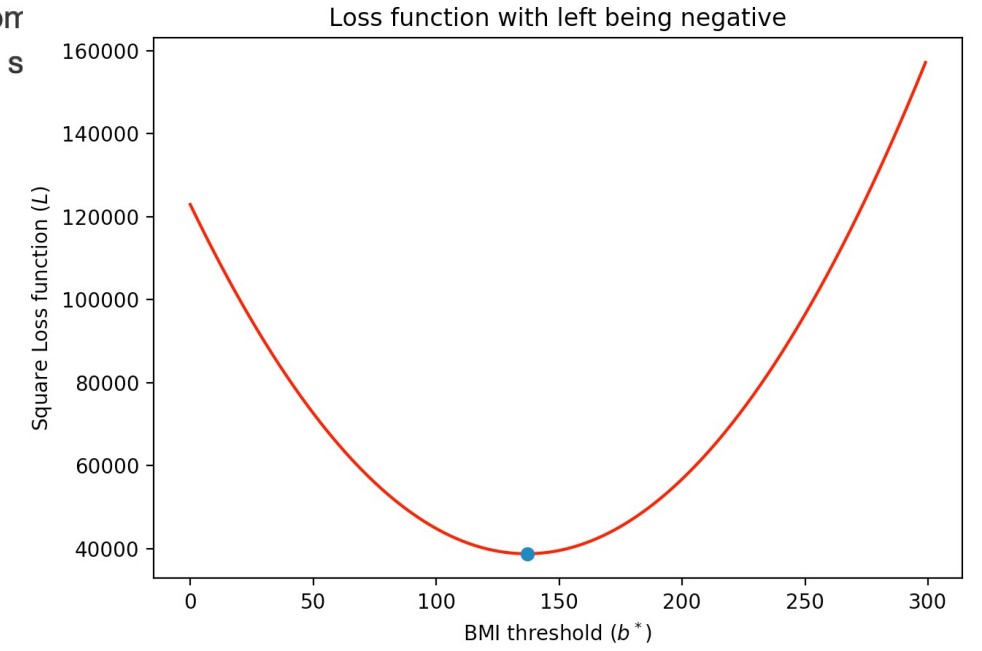
For simplicity, let us consider linear models where the values to the left of the threshold value of b^* get the yellow color. Recall that this means that the linear model is defined as

$$\ell(b) = \begin{cases} -1 & \text{if } b < b^* \\ 1 & \text{otherwise} \end{cases}.$$

The idea in the `square loss function` is that mistakes for b_i values that are farther away from counts the number of misclassified points, the loss function counts all mis-classification as the s

n that
ience):



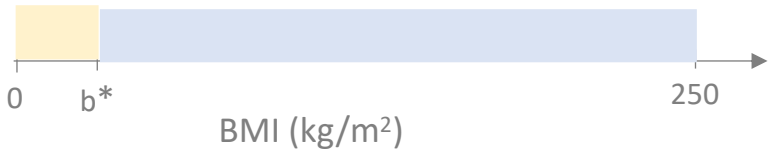


Math behind the square loss function

Linear model

$$\ell(b) = \text{sign}(b - b^*),$$

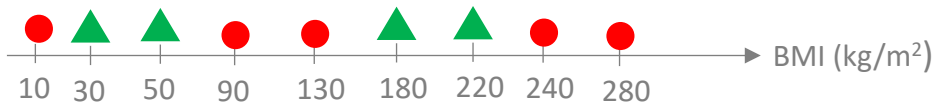
$$\text{sign}(x) = \begin{cases} -1 & \text{if } b < b^* \\ 1 & \text{otherwise.} \end{cases}$$

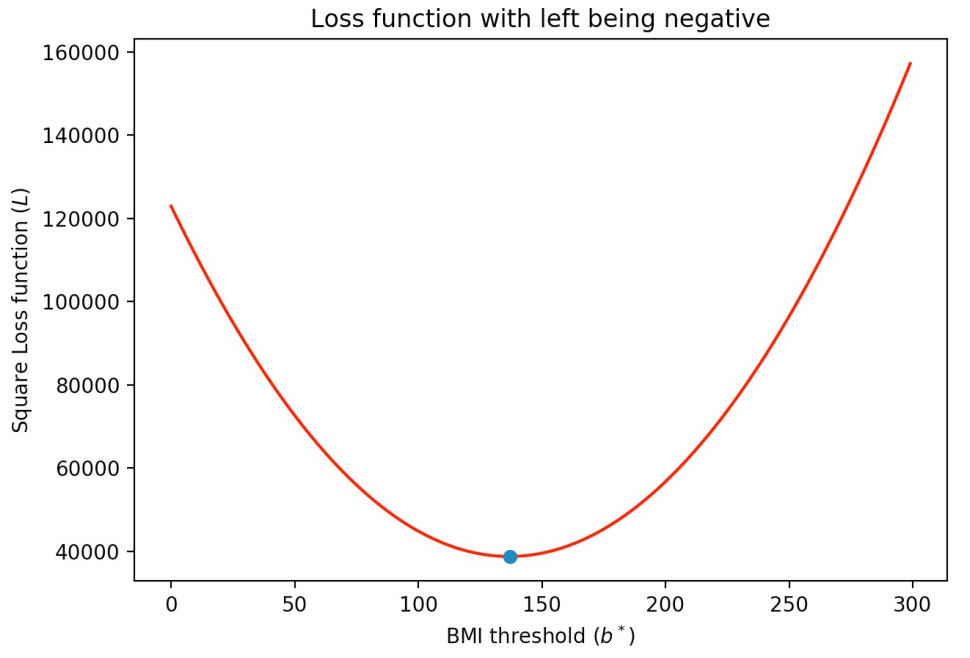


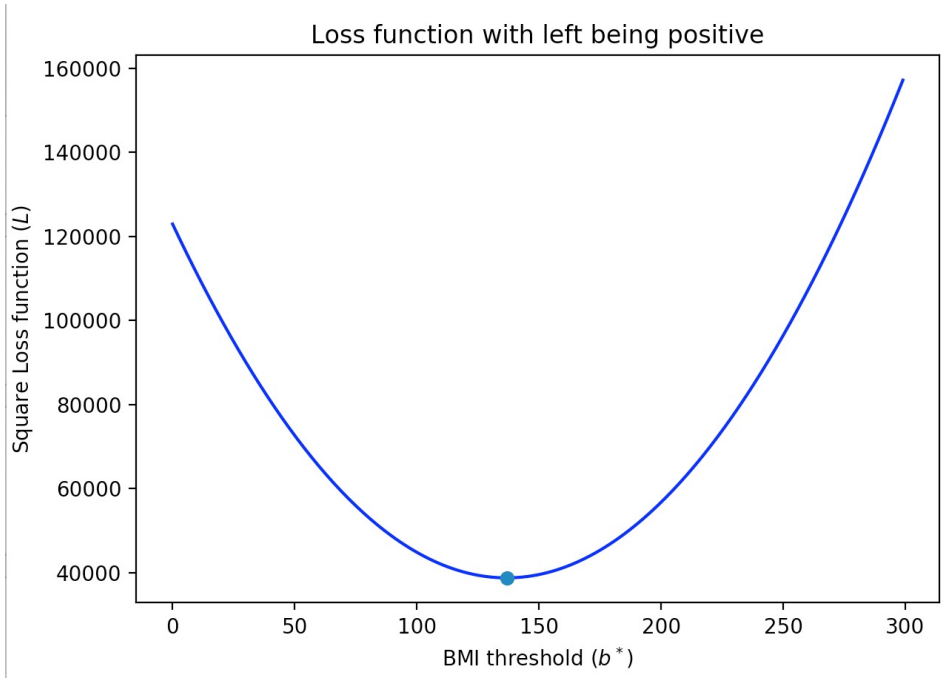
$$L_{\text{accuracy}}(\ell, i) = \left(\frac{y_i - \text{sign}(b_i - b^*)}{2} \right)^2.$$

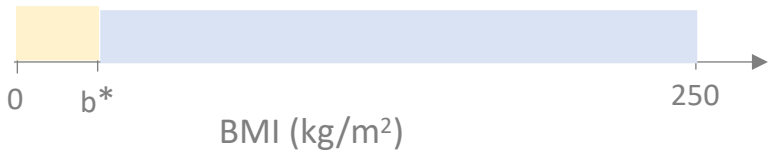
$$L_{\text{sq}}(\ell, i) = \frac{1}{4} \cdot \left(y_i - (b_i - b^*) \right)^2.$$

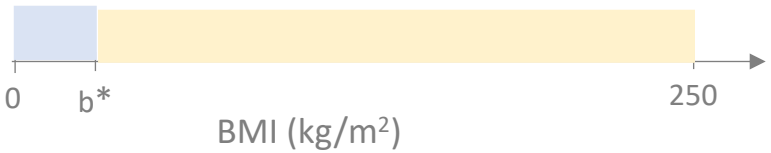
Square loss function









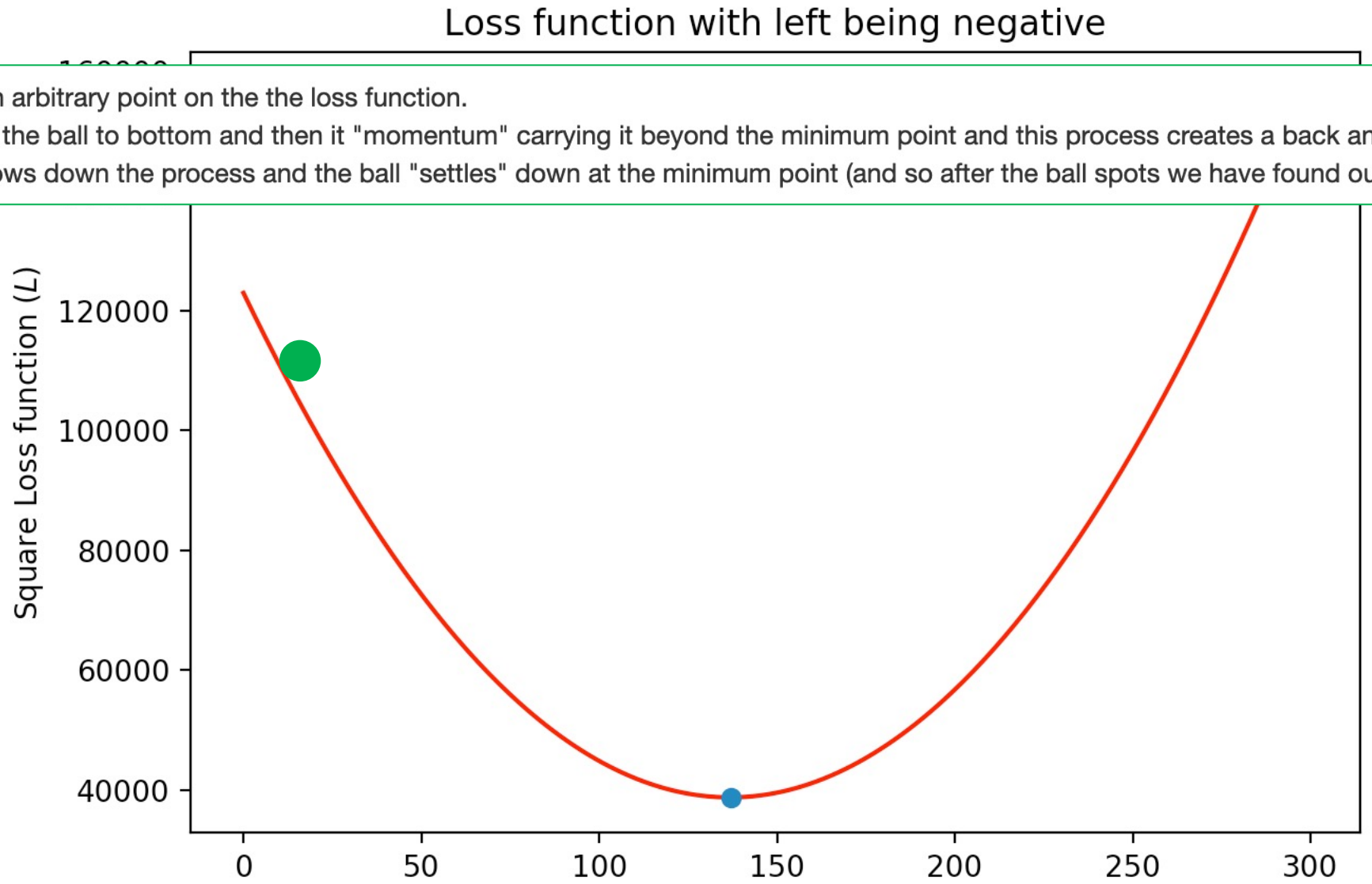


So did we gain anything here?

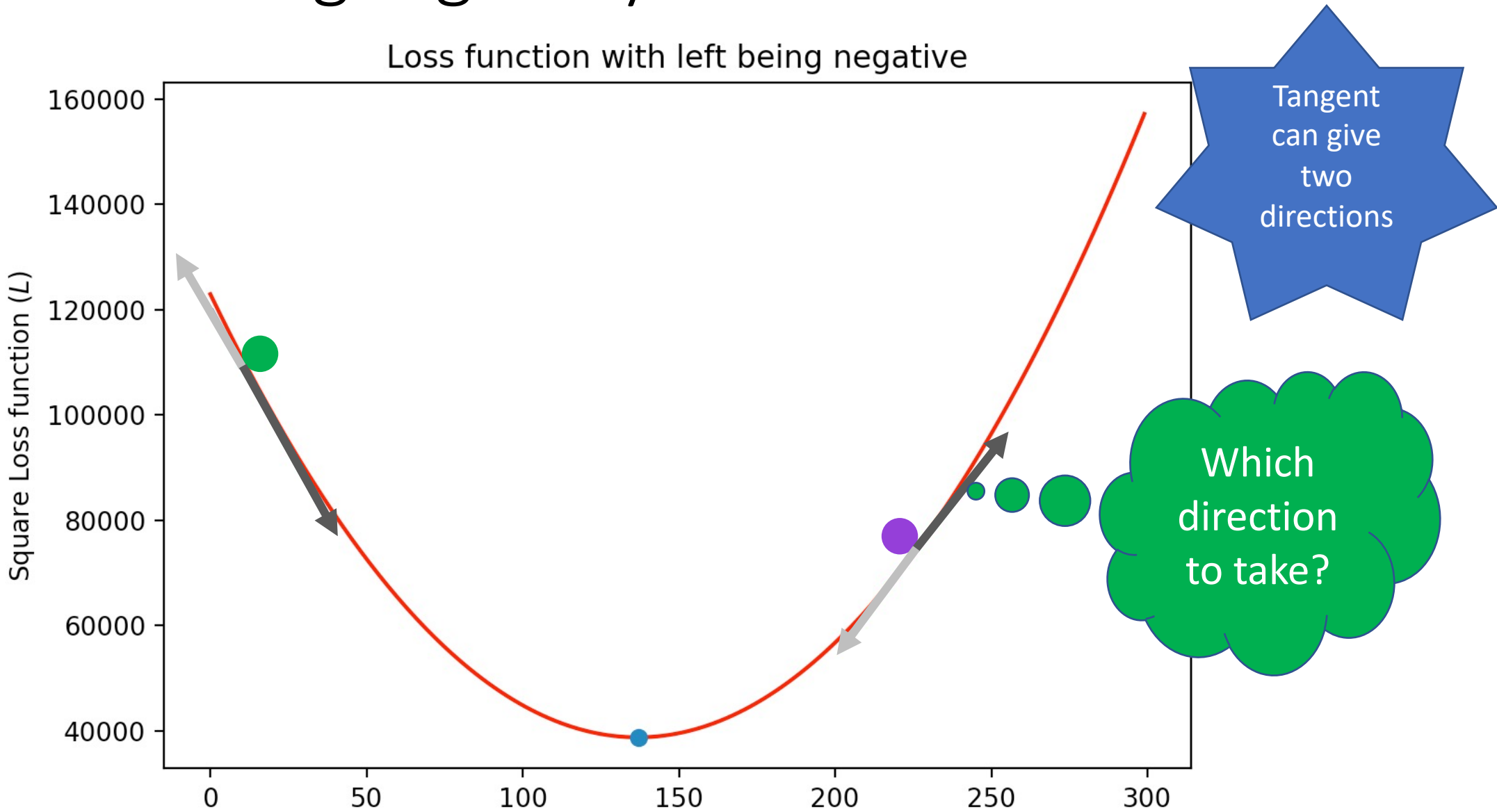
Do you know what a half-pipe means in the context of sports?

Towards gradient descent

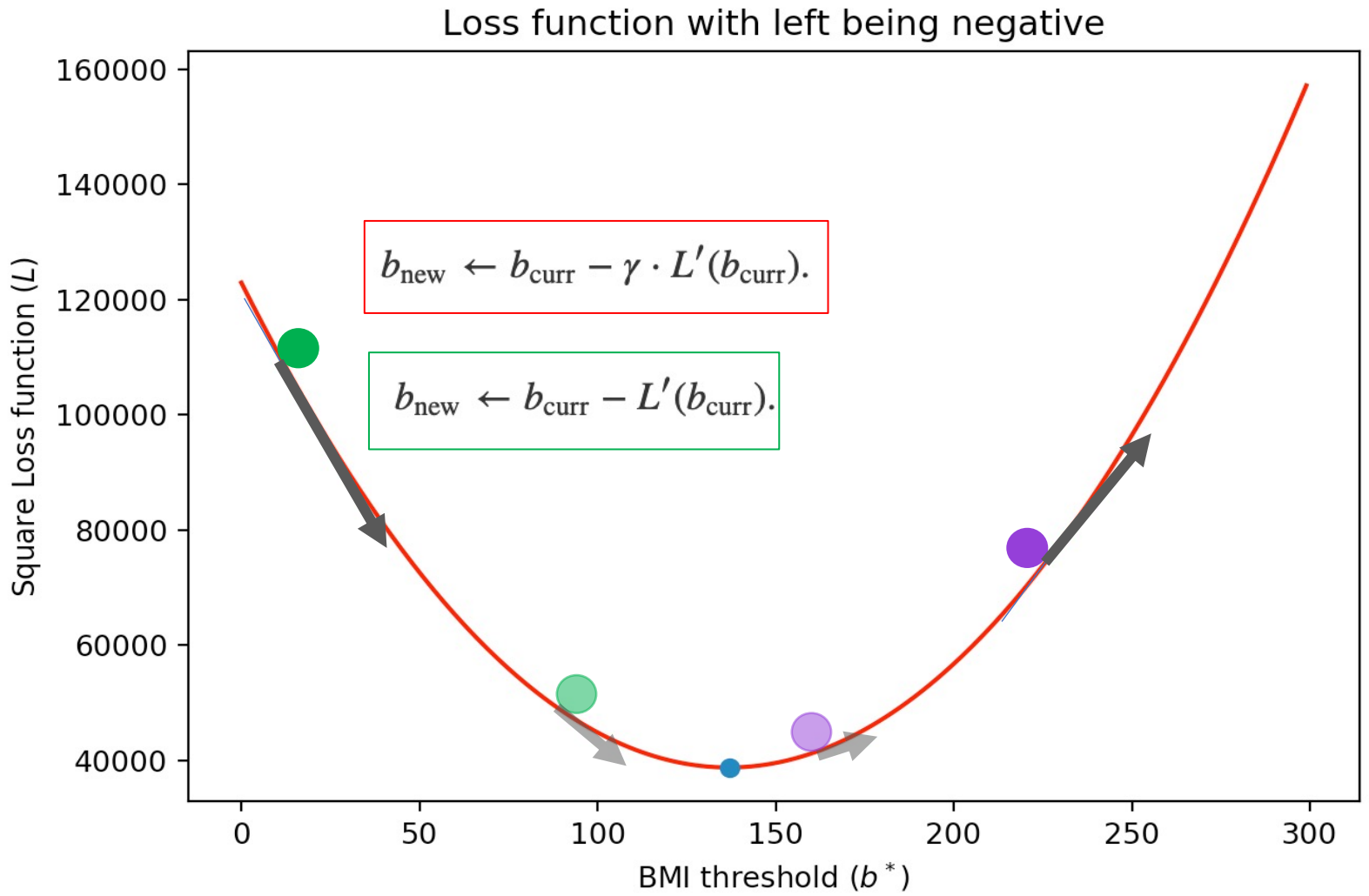
1. Put the green ball at an arbitrary point on the the loss function.
2. Then let "gravity" drag the ball to bottom and then it "momentum" carrying it beyond the minimum point and this process creates a back and forth movement.
3. Eventually "friction" slows down the process and the ball "settles" down at the minimum point (and so after the ball spots we have found our unknown minimum point).



Where do we get gravity and friction?



Derivative is all you need!

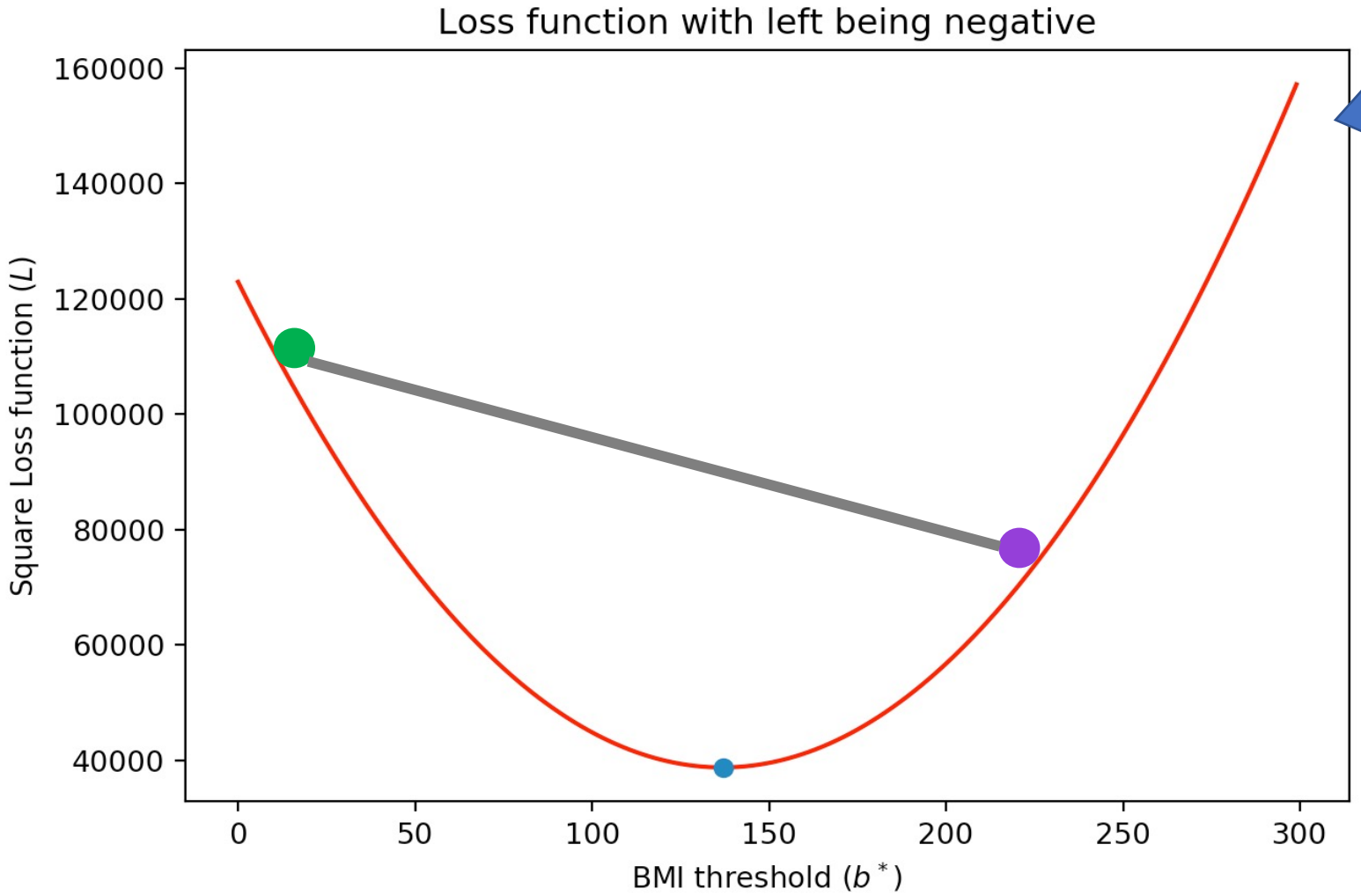


Overall Algorithm

Gradient Descent based model training algorithms

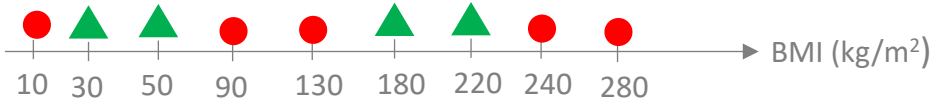
```
                                // There is only one input variable b
                                // Input: Training data set (b1, y1), ...., (bn, yn)
                                // Input parameter: learning parameter gamma

 5.                             //Compute some functions
                                Compute the loss function L from training dataset
                                Compute the derivative function derivL of L


10.                             /*Gradient Descent part starts now*/

                                Initialize b* to an arbitrary value

                                while stopping condition not satisfied //Stop if  L(b*)is small enough or too many iterations
15.
                                    delta = derivL(b*)
                                    update b* to b* - gamma*delta

                                Output the current value of b*
20.
```

When does this work?

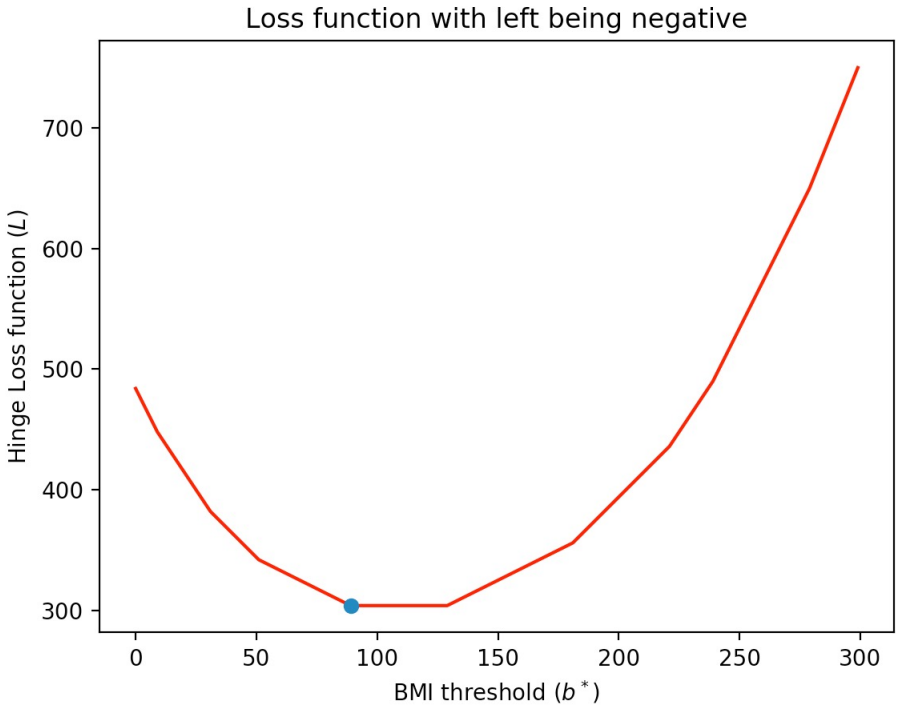


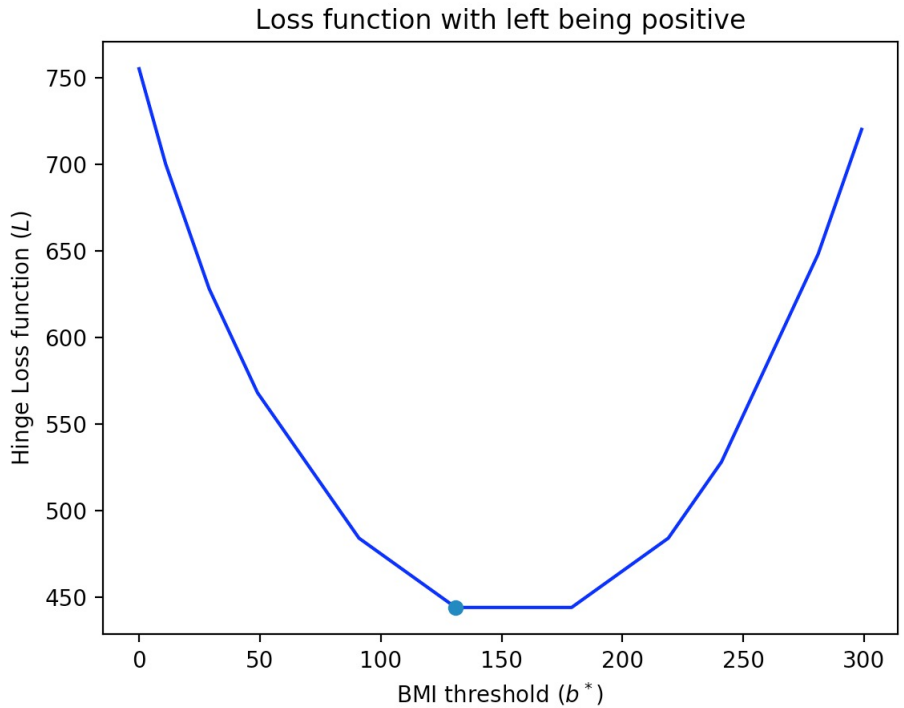
Convex loss functions

Hinge loss function

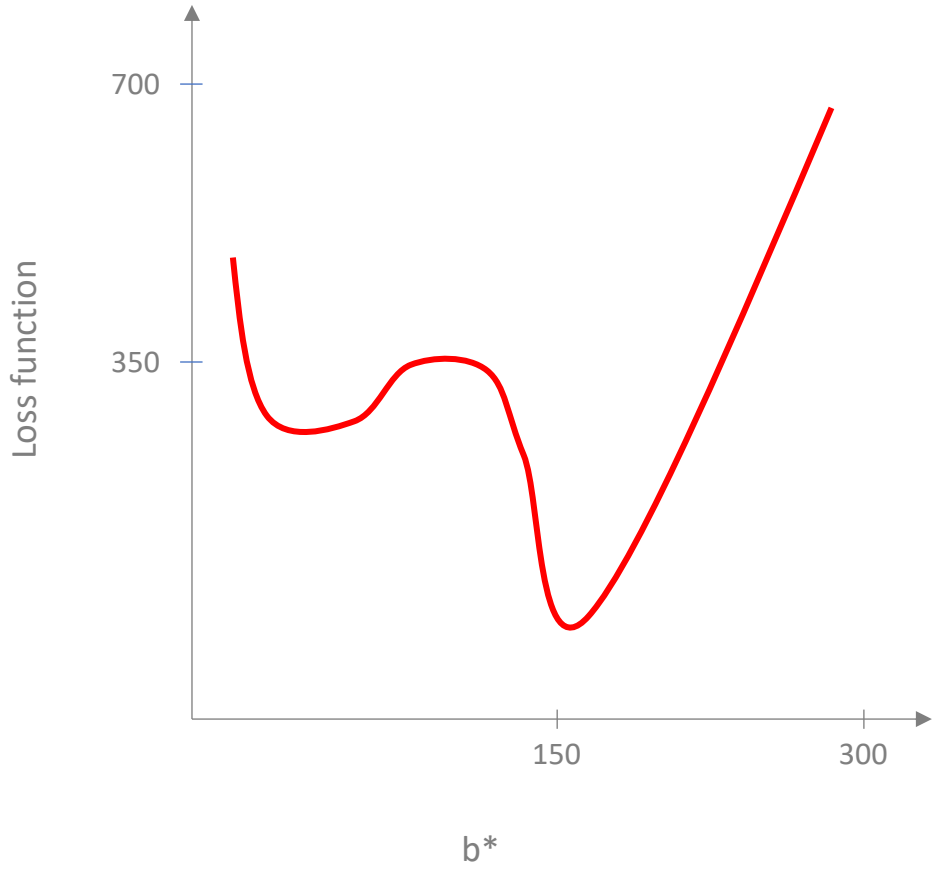


Convex functions need not be smooth!

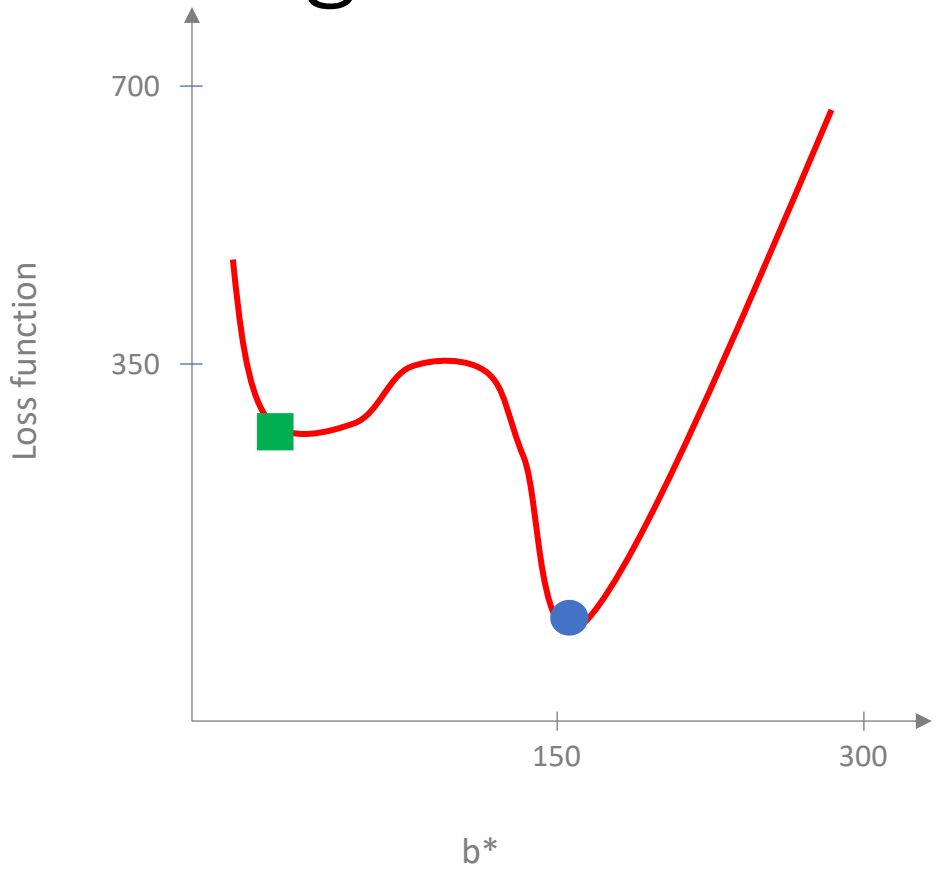




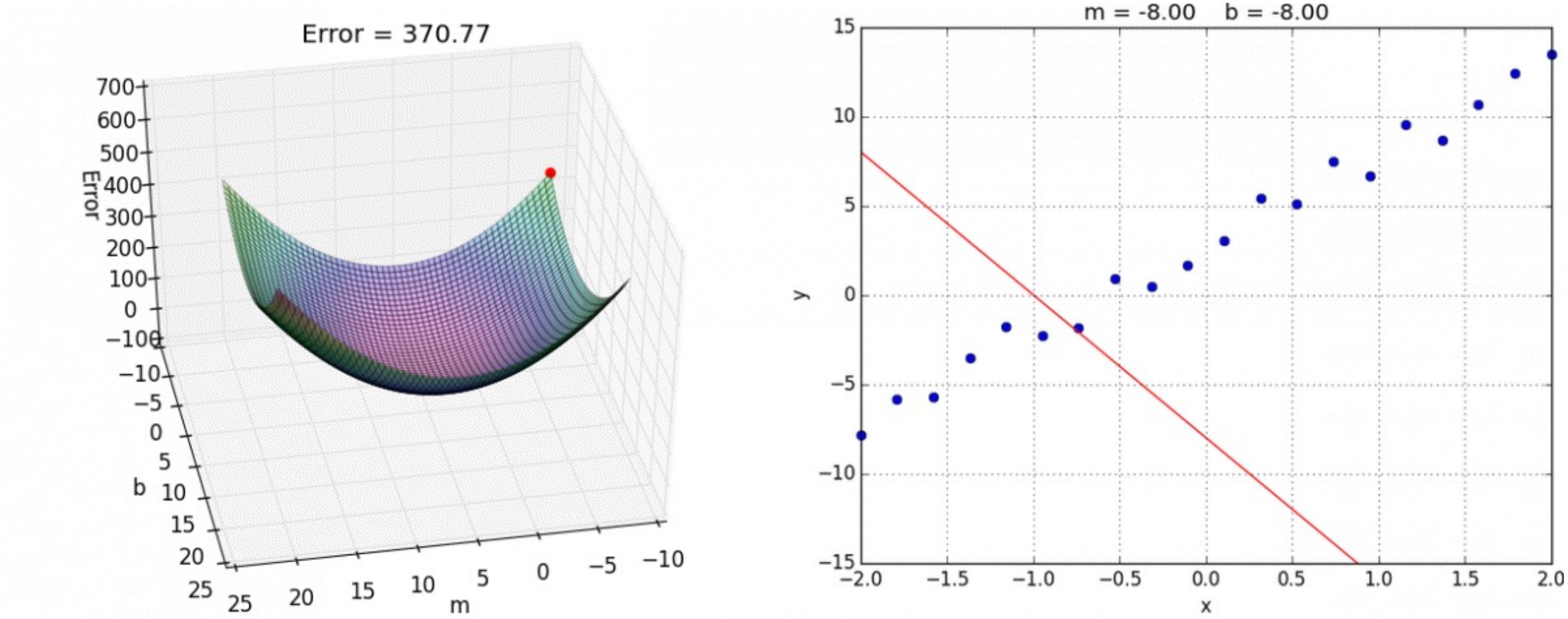
What about non-convex loss functions?



What can go wrong if we run GD?



GD on more than one variable



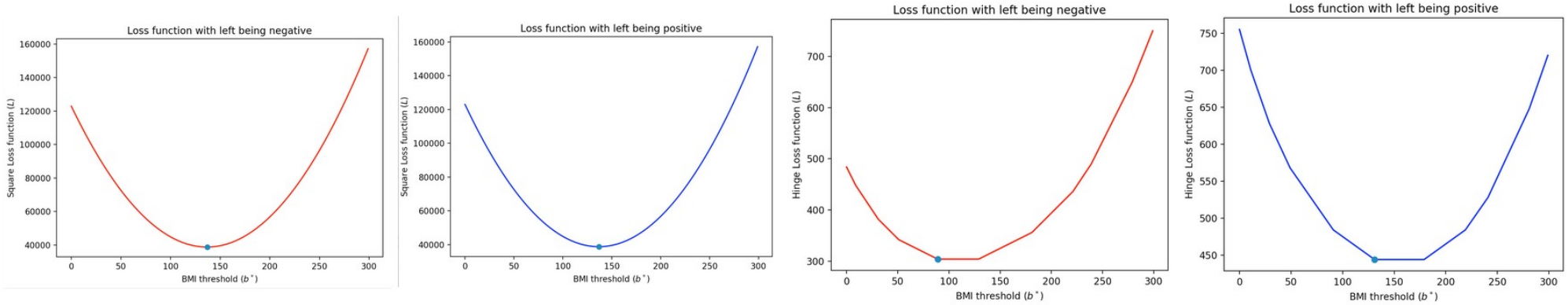
https://alykhantejani.github.io/images/gradient_descent_line_graph.gif

Did we lose something?

What did we lose?

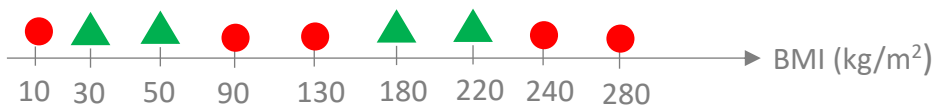
Recall that ideally we would still have liked to use the total number of misclassified points as our loss function (since that is what we will end up using as our main way to measure accuracy during the last step in the ML pipeline). In other words, we used the square/hinge loss as a **proxy** for the actual loss function we really wanted to use.

Perhaps we can get lucky and somehow get the optimal threshold value for our ideal loss function. Now would be good time to tell y'all about the exact minimum threshold value that we get from the four convex loss functions we have seen so far:



The minimum threshold values for the four loss functions (in order are): $137, 137, 89$ and 131 respectively.

Minimize number of mis-classifications



The minimum threshold values for the four loss functions (in order are): 137, 137, 89 and 131