Machine Learning

Pawel Wocjan

University of Central Florida

Fall 2020

Sources for Slides

- I have extensively used the machine learning materials that have been prepared by Google.

  `https://developers.google.com/machine-learning/crash-course/`

- Google has licensed these materials under the Creative Commons Attribution 3.0 License.

  `https://creativecommons.org/licenses/by/3.0/`

Outline

Framing

- ML Terminology

Descending into ML

- Linear Regression
- Training and Loss

Reducing Loss

- An Iterative Approach
- Gradient Descent
- Learning Rate
- Optimizing Learning Rate
- Stochastic Gradient Descent

ML Terminology

- What is (supervised) machine learning? Concisely put, it is the following:

  ML systems learn how to combine input to produce useful predictions on never-before-seen data.

- Let's explore fundamental machine learning terminology.

ML Terminology

- A **label** is the thing we're predicting – the y variable in simple linear regression. The label could be the future price of a stock, the kind of animal shown in a picture, the meaning of an audio clip, or just about anything.
- A **feature** is an input variable – the x variable in simple linear regression.
- A simple machine learning project might use a single feature, while a more sophisticated machine learning project could use millions of features, specified as:

$$x_1, x_2, \ldots, x_n$$

ML Terminology

- In the spam detector example, the features could include the following:
  - words in the email text
  - sender's address
  - time of day the email was sent
  - email contains the phrase "one weird trick."
- An **example** is a particular instance of data, x. We break examples into two categories:
  - labeled examples
  - unlabeled examples

ML Terminology

- A **labeled example** includes both feature(s) and the label. That is:

  `labeled example:` {`features, label`}: (x, y)

- We use labeled examples to train the model. In our spam detector example, the labeled examples would be individual emails that users have explicitly marked as "spam" or "not spam."
- An **unlabeled example** contains features but not the label. That is:

  `unlabeled example:` {`features, ?`}: $(x, ?)$

- Once we've trained our model with labeled examples, we use that model to predict the label on unlabeled examples. In the spam detector, unlabeled examples are new emails that humans haven't yet labeled.

ML Terminology

- A model defines the relationship between features and label. For example, a spam detection model might associate certain features strongly with "spam". Let's highlight two phases of a model's life:
  - **Training** means creating or learning the model. That is, you show the model labeled examples and enable the model to gradually learn the relationships between features and label.
  - **Inference** means applying the trained model to unlabeled examples. That is, you use the trained model to make useful predictions y'.

ML Terminology

- A **regression model** predicts continuous values. For example, regression models make predictions that answer questions like the following:
  - What is the value of a house in California?
  - What is the probability that a user will click on this ad?
- A **classification model** predicts discrete values. For example, classification models make predictions that answer questions like the following:
  - Is a given email message spam or not spam?
  - Is this an image of a dog, a cat, or a hamster?

Key Terms

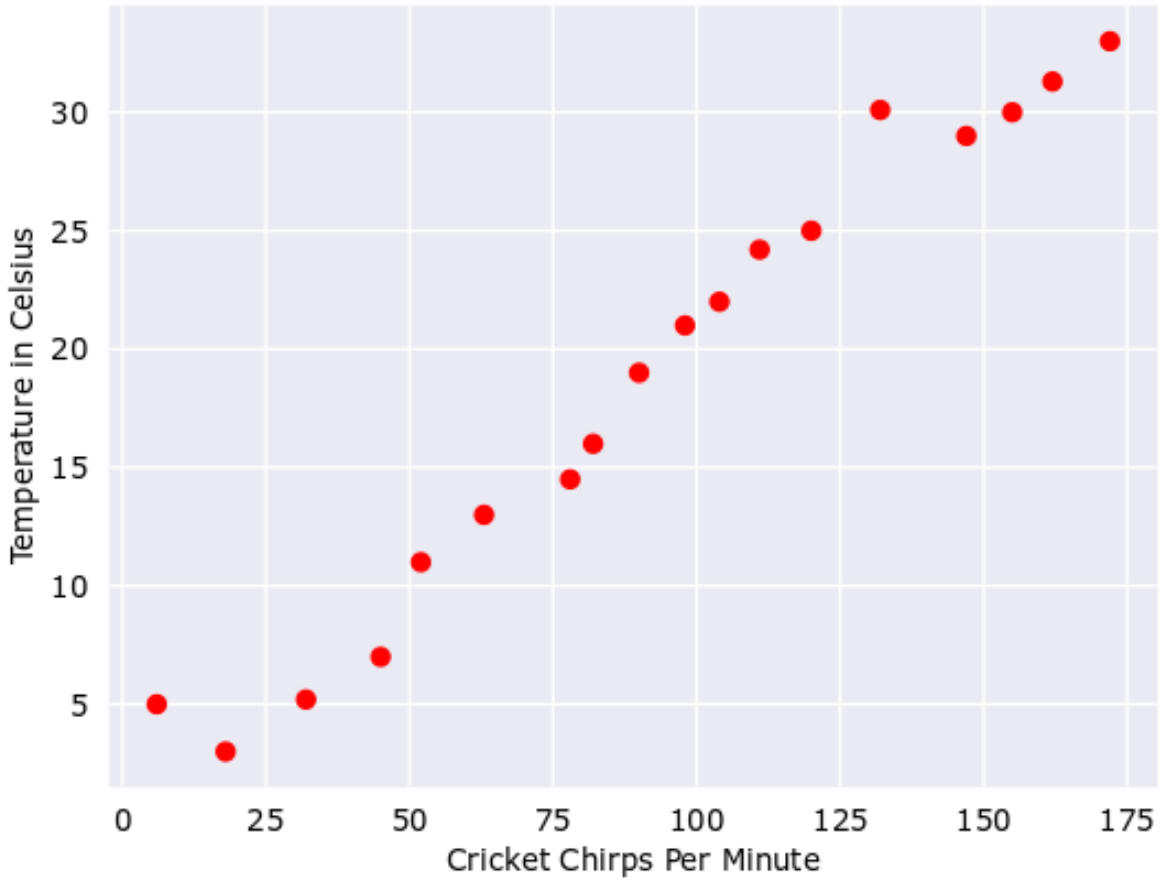
- classification model
- example
- feature
- inference
- label
- model
- regression model
- training

Linear Regression

- It has long been known that crickets (an insect species) chirp more frequently on hotter days than on cooler days. For decades, professional and amateur scientists have cataloged data on chirps-per-minute and temperature.
- Using this data, you want to explore this relationship.

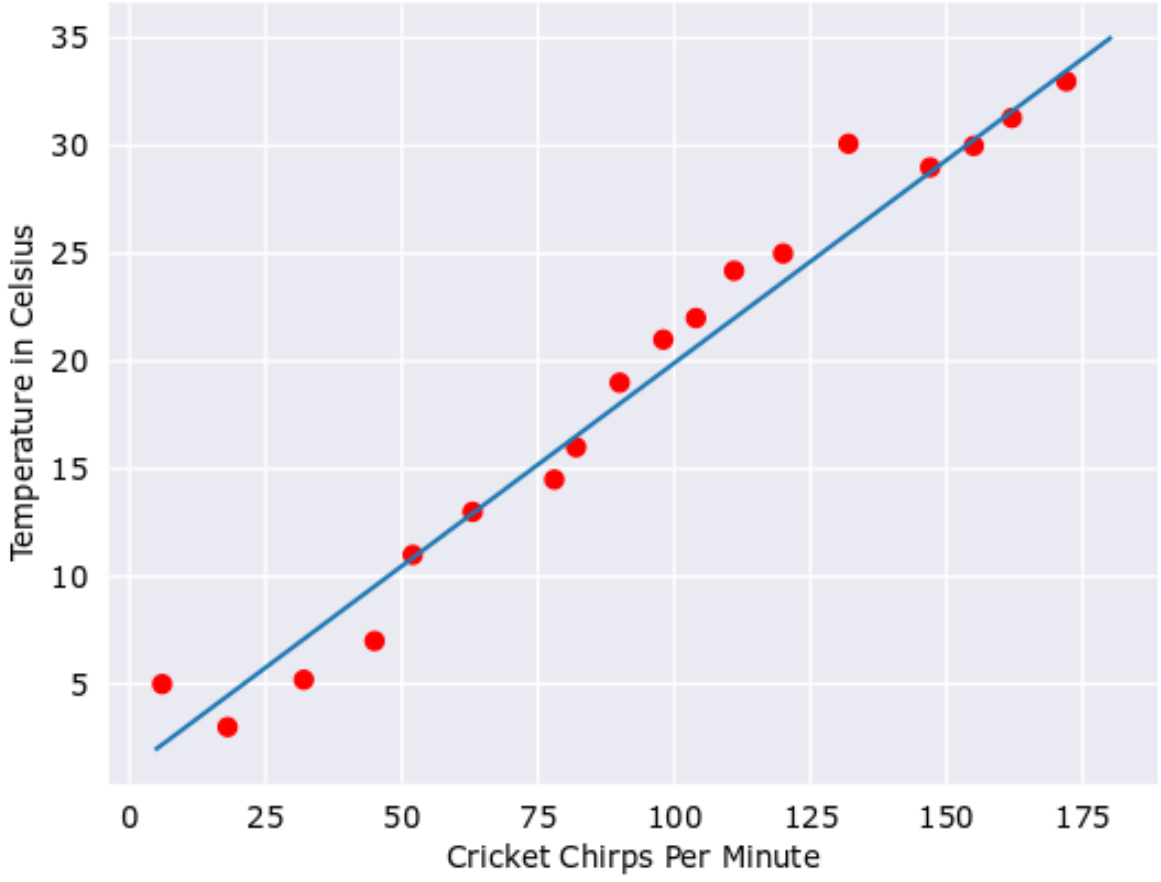
Linear Regression

- First, examine your data by plotting it:



Linear Regression

- You could draw a single straight line like the following to approximate this relationship between chirps and temperature.



Linear Regression

- The line doesn't pass through every dot, but the line does clearly show the relationship between chirps and temperature.
- Using the equation for a line, you could write down this relationship as follows:

$$y = mx + b$$

where:

- y is the temperature in Celsius—the value we're trying to predict.
- m is the slope of the line.
- x is the number of chirps per minute—the value of our input feature.
- b is the y-intercept.

Linear Regression

- By convention in ML, you'll write the equation for a model slightly differently:

$$y' = b + w_1 x_1$$

  where:

  - y is the predicted label (a desired output).
  - b is the bias (the y-intercept), sometimes referred to as w_0.
  - w_1 is the weight of feature 1. Weight is the same concept as the "slope" m in the traditional equation of a line.
  - x_1 is a feature (a known input).

Linear Regression

- To **infer** (predict) the temperature y' for a new chirps-per-minute value x_1, just substitute the x_1 value into this model.
- Although this model uses only one feature, a more sophisticated model might rely on multiple features, each having a separate weight w_1, w_2, etc.
- For example, a model that relies on three features might look as follows:

$$y' = b + w_1 x_1 + w_2 x_2 + w_3 x_3$$

Key Terms

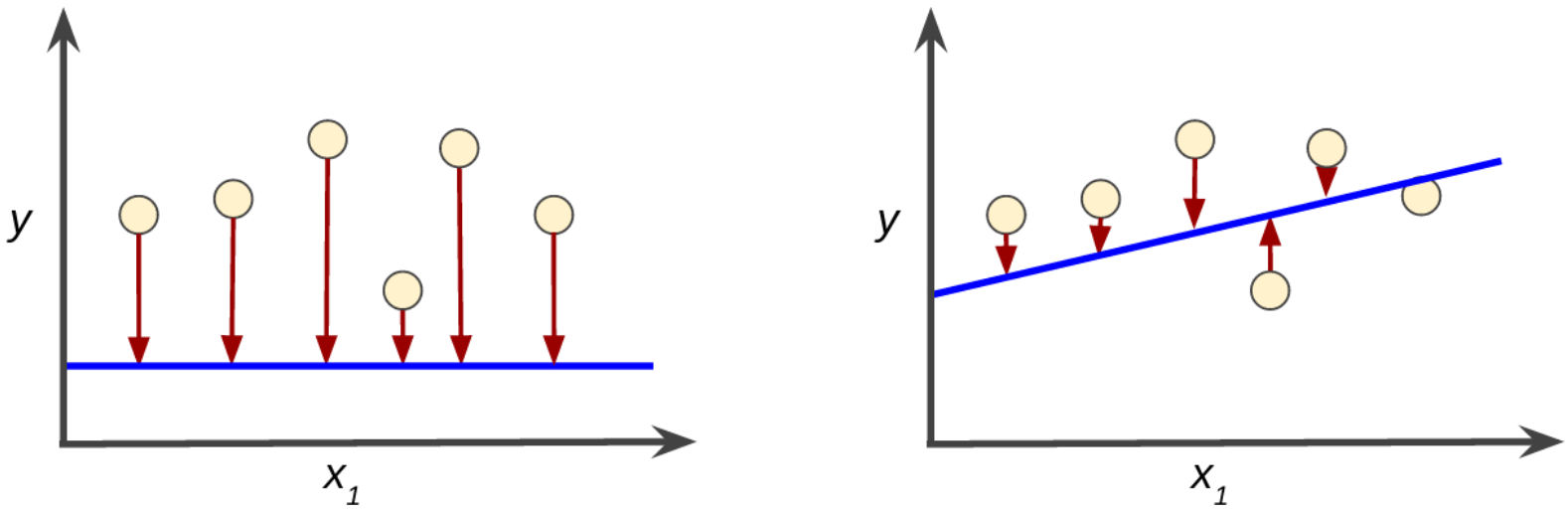
- bias
- inference
- linear regression
- weight

Training and Loss

- **Training** a model simply means learning (determining) good values for all the weights and the bias from labeled examples.
- In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called **empirical risk minimization**.
- Loss is the penalty for a bad prediction. That is, **loss** is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.
- The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.

Training and Loss

- For example, the figure below shows a high loss model on the left and a low loss model on the right.
  - The arrows represent loss.
  - The lines represent predictions.



Training and Loss

- Notice that the arrows in the left plot are much longer than their counterparts in the right plot. Clearly, the line in the right plot is a much better predictive model than the line in the left plot.
- You might be wondering whether you could create a mathematical function – a loss function – that would aggregate the individual losses in a meaningful fashion.
- The linear regression models we'll examine here use a loss function called **squared loss** (also known as L_2 loss).

Training and Loss

- The squared loss for a single example is as follows is the difference between the observation (label) y and the prediction $\hat{y}$. That is:

$$(y - \hat{y})^2$$

Training and Loss

- **Mean square error (MSE)** is the average squared loss per example over the whole dataset. To calculate MSE, sum up all the squared losses for individual examples and then divide by the number of examples:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - y^{(i)})^2$$

  - $(\boldsymbol{x}^{(i)}, y^{(i)})$ is the example in which
    - $\boldsymbol{x}^{(i)}$ is the set of features (for instance, chirps/minute, age, gender)
    - $y^{(i)}$ is the label (for instance, temperature).
  - $y^{(i)}$ is a function of the weights and bias in combination with the set of features $\boldsymbol{x}$.
  - m is the number of examples

Training and Loss

- Although MSE is commonly-used in machine learning, it is neither the only practical loss function nor the best loss function for all circumstances.

Key Terms

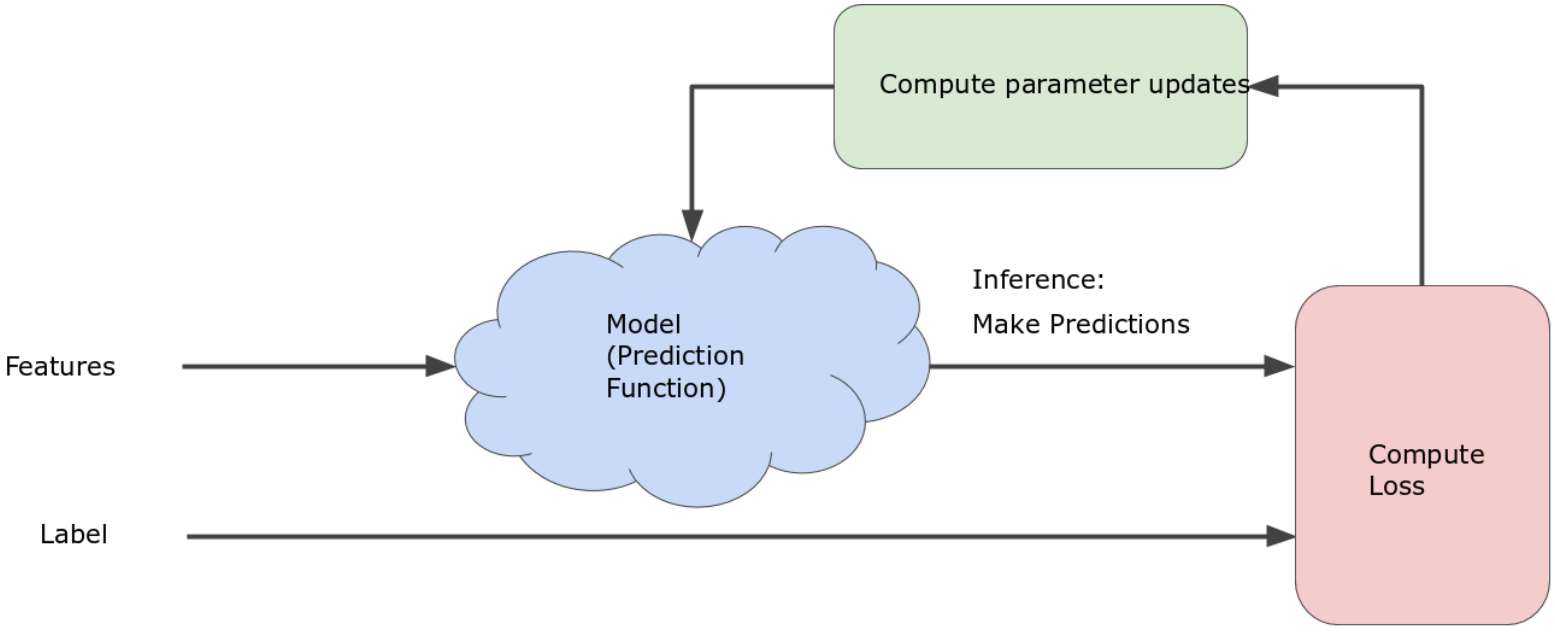
- empirical risk minimization
- loss
- mean squared error
- squared loss
- training

An Iterative Approach

- You'll learn now how a machine learning model iteratively reduces loss.
- Iterative learning might remind you of the "Hot and Cold" kid's game for finding a hidden object.
- In this game, the "hidden object" is the best possible model. You'll start with a wild guess ("The value of w_1 is 0.") and wait for the system to tell you what the loss is. Then, you'll try another guess ("The value of w_1 is 0.5.") and see what the loss is.
- If you play this game right, you'll usually be getting warmer. The real trick to the game is trying to find the best possible model as efficiently as possible.

An Iterative Approach

- The following figure suggests the iterative trial-and-error process that machine learning algorithms use to train a model:



An Iterative Approach

- Iterative strategies are prevalent in machine learning, primarily because they scale so well to large data sets.
- The "model" takes one or more features as input and returns one prediction y' as output.

An Iterative Approach

- To simplify, consider a model that takes one feature and returns one prediction:

$$\hat{y} = b + w_1 x_1$$

- What initial values should we set for b and w_1? For linear regression problems, it turns out that the starting values aren't important. We could pick random values, but we'll just take the following trivial values instead:

$$b = 0\,, \quad w_1 = 0$$

An Iterative Approach

- Suppose that the first feature value is 10. Plugging that feature value into the prediction function yields:

$$\hat{y} = 0 + 0 \cdot 10 = 0$$

- The "Compute Loss" part of the diagram is the loss function that the model will use. Suppose we use the squared loss function. The loss function takes in two input values:
  - $\hat{y}$: The model's prediction for features $\boldsymbol{x}$
  - y: The correct label corresponding to features $\boldsymbol{x}$.

An Iterative Approach

- At last, we've reached the "Compute parameter updates" part of the diagram.
- It is here that the machine learning system examines the value of the loss function and generates new values for b and w_1.
- For now, just assume that this mysterious box devises new values and then the machine learning system re-evaluates all those features against all those labels, yielding a new value for the loss function, which yields new parameter values.
- The learning continues iterating until the algorithm discovers the model parameters with the lowest possible loss.
- Usually, you iterate until overall loss stops changing or at least changes extremely slowly. When that happens, we say that the model has **converged**.

Key Point

- A Machine Learning model is trained by starting with an initial guess for the weights and bias and iteratively adjusting those guesses until learning the weights and bias with the lowest possible loss.

Key Terms

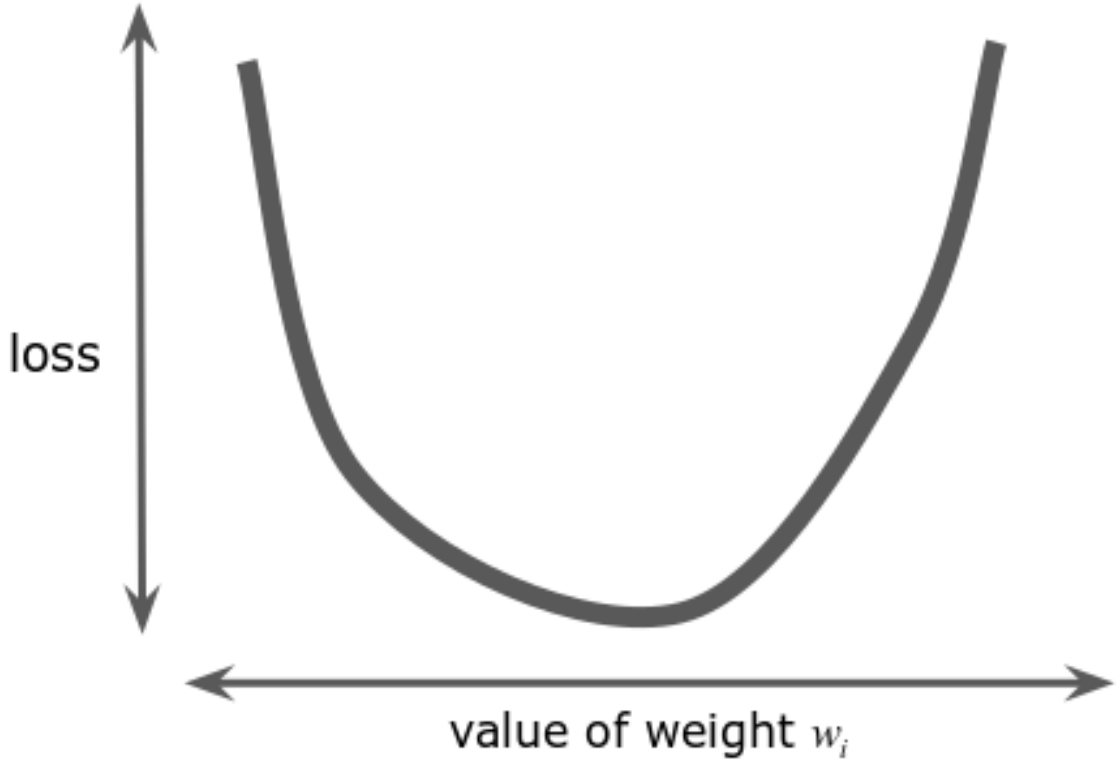
- convergence
- loss
- training

Gradient Descent

- The iterative approach diagram contained a green hand-wavy box entitled "Compute parameter updates." We'll now discuss the underlying algorithms in detail.
- Suppose we had the time and the computing resources to calculate the loss for all possible values of w_1.

Gradient Descent

- For the kind of regression problems we've been examining, the resulting plot of loss vs. w_1 will always be convex. In other words, the plot will always be bowl-shaped, kind of like this:

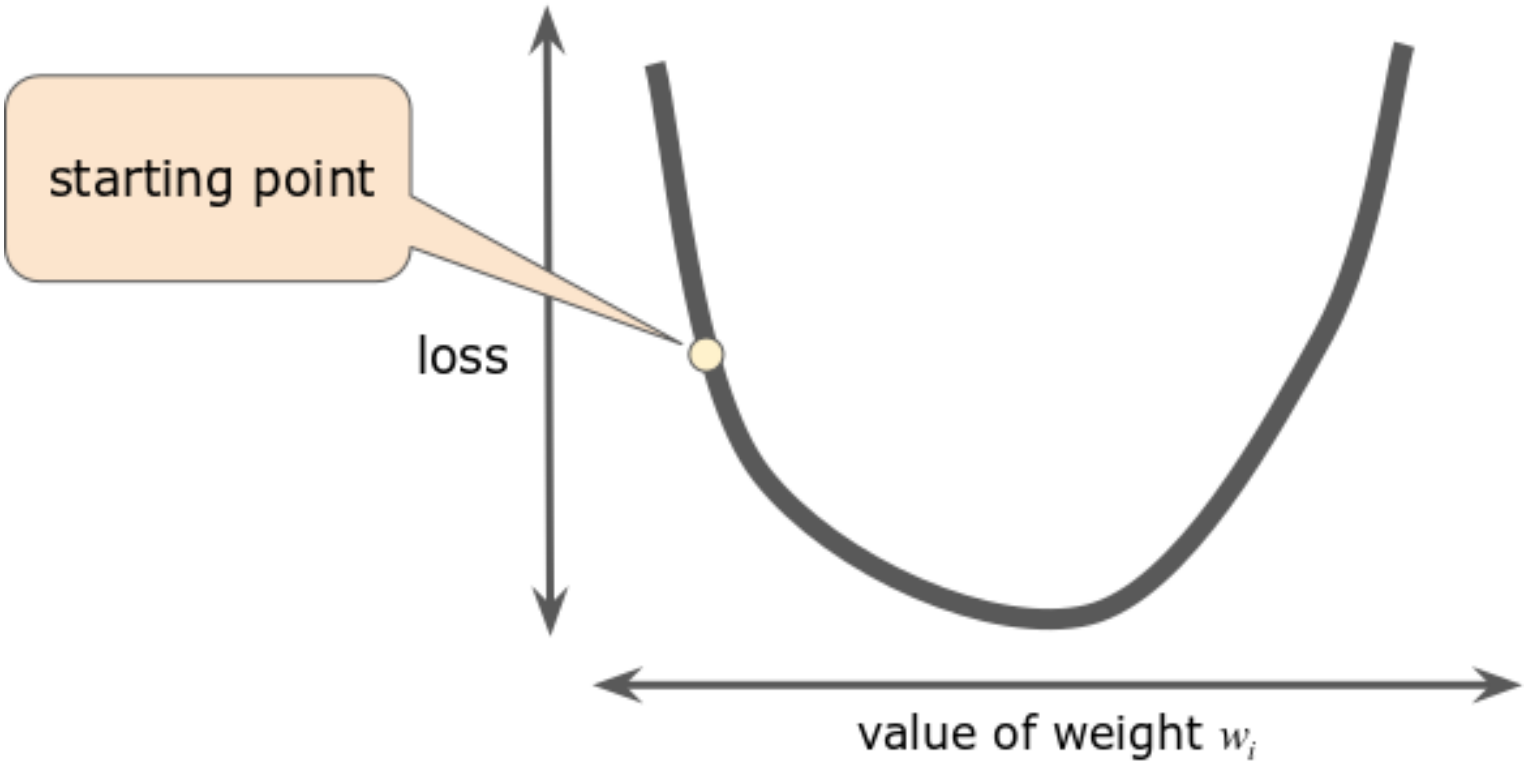


Gradient Descent

- Convex problems have only one minimum; that is, only one place where the slope is exactly 0. That minimum is where the loss function converges.
- Calculating the loss function for every conceivable value of over the entire data set would be an inefficient way of finding the convergence point.
- Let's examine a better mechanism—very popular in machine learning—called **gradient descent**.

Gradient Descent

- The first stage in gradient descent is to pick a starting value for w_1. The starting point doesn't matter much; therefore, many algorithms simply set w_1 to 0 or pick a random value.

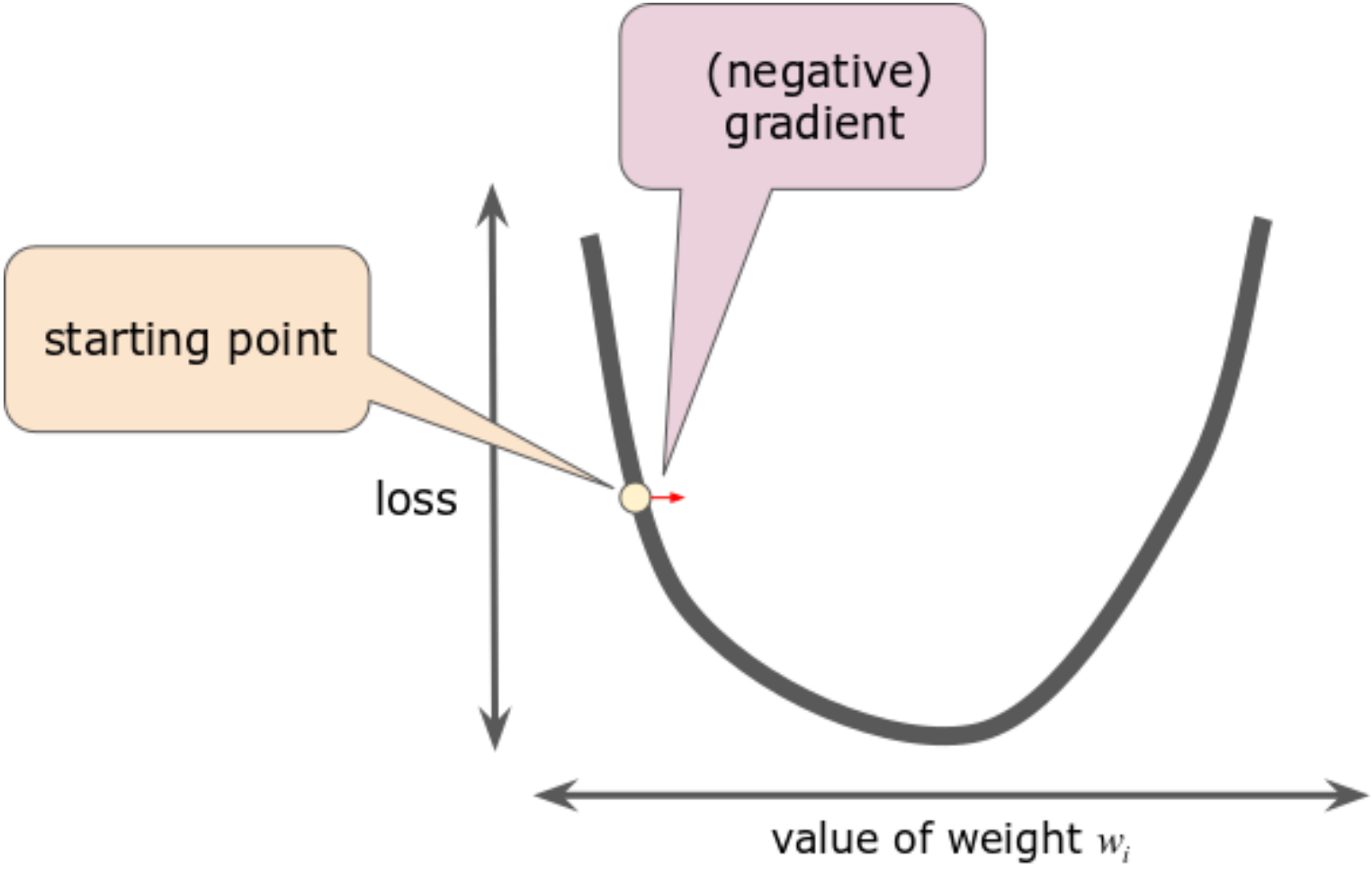


Gradient Descent

- The gradient descent algorithm then calculates the gradient of the loss curve at the starting point.
- In the previous figure, the gradient of loss is equal to the derivative (slope) of the curve, and tells you which way is "warmer" or "colder."
- When there are multiple weights, the gradient is a vector of partial derivatives with respect to the weights.
- Note that a gradient is a vector, so it has both of the following characteristics:
  - a direction
  - a magnitude

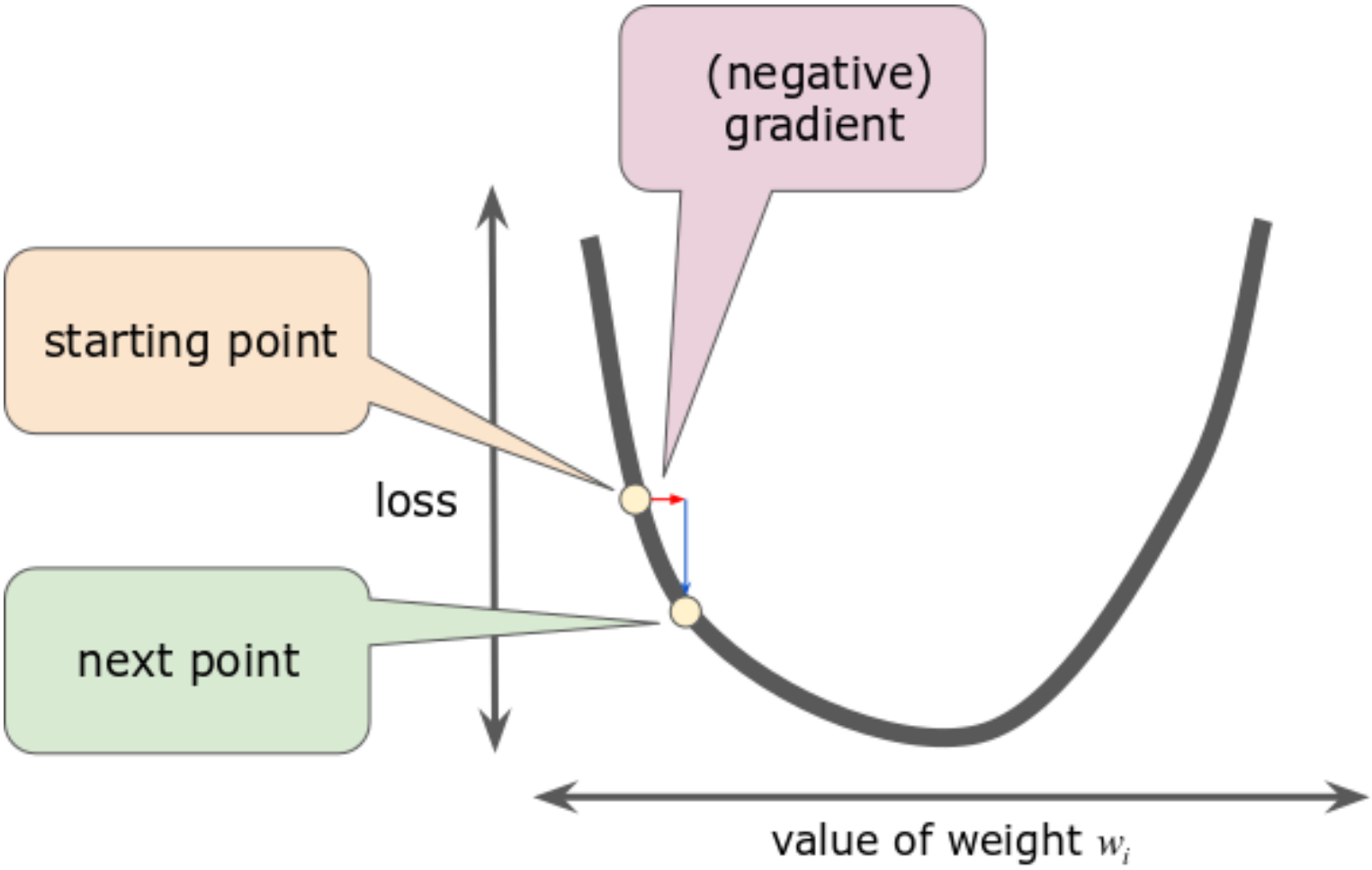
Gradient Descent

- The gradient always points in the direction of steepest increase in the loss function. The gradient descent algorithm takes a step in the direction of the negative gradient in order to reduce loss as quickly as possible.



Gradient Descent

- To determine the next point along the loss function curve, the gradient descent algorithm adds some fraction of the gradient's magnitude to the starting point as shown in the following figure:



Gradient Descent

- When performing gradient descent, we generalize the above process to tune all the model parameters simultaneously.
- For example, to find the optimal values of both w_1 and the bias b, we calculate the gradients with respect to both w_1 and b.
- Next, we modify the values of w_1 and b based on their respective gradients.
- Then we repeat these steps until we reach minimum loss.

Key Terms

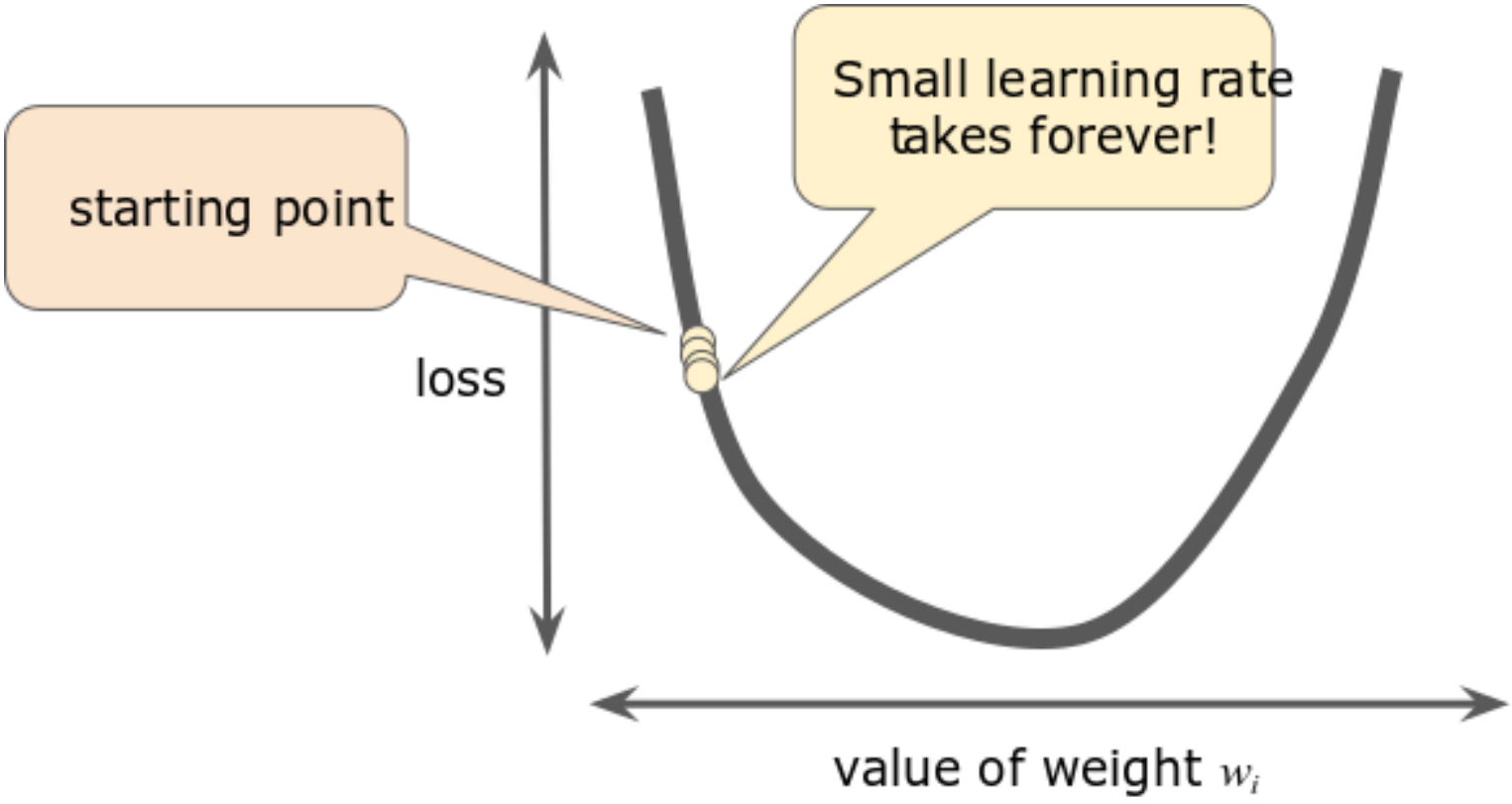
- gradient
- gradient descent
- step

Learning Rate

- The gradient vector has both a direction and a magnitude.
- Gradient descent algorithms multiply the gradient by a scalar known as the learning rate (also sometimes called step size) to determine the next point.
- For example, if the gradient magnitude is 2.5 and the learning rate is 0.01, then the gradient descent algorithm will pick the next point 0.025 away from the previous point.

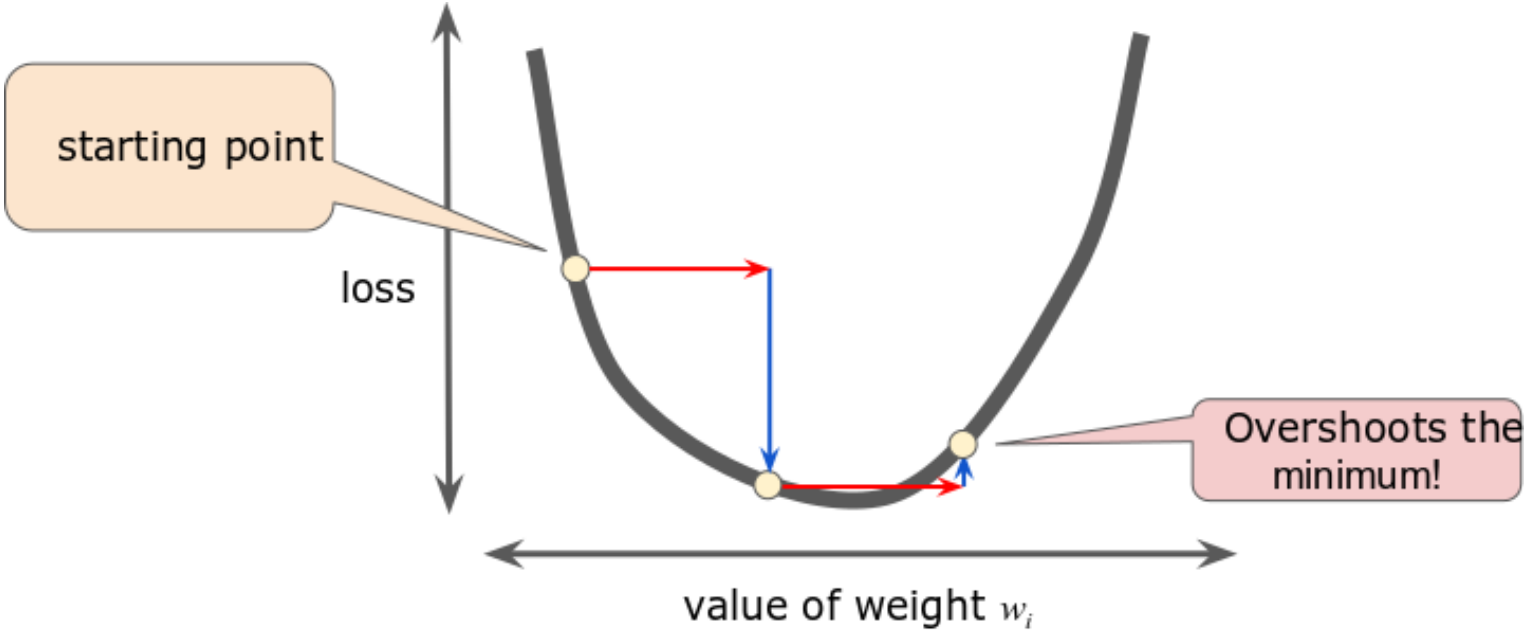
Learning Rate

- **Hyperparameters** are the knobs that programmers tweak in machine learning algorithms. Most machine learning programmers spend a fair amount of time tuning the learning rate. If you pick a learning rate that is too small, learning will take too long:



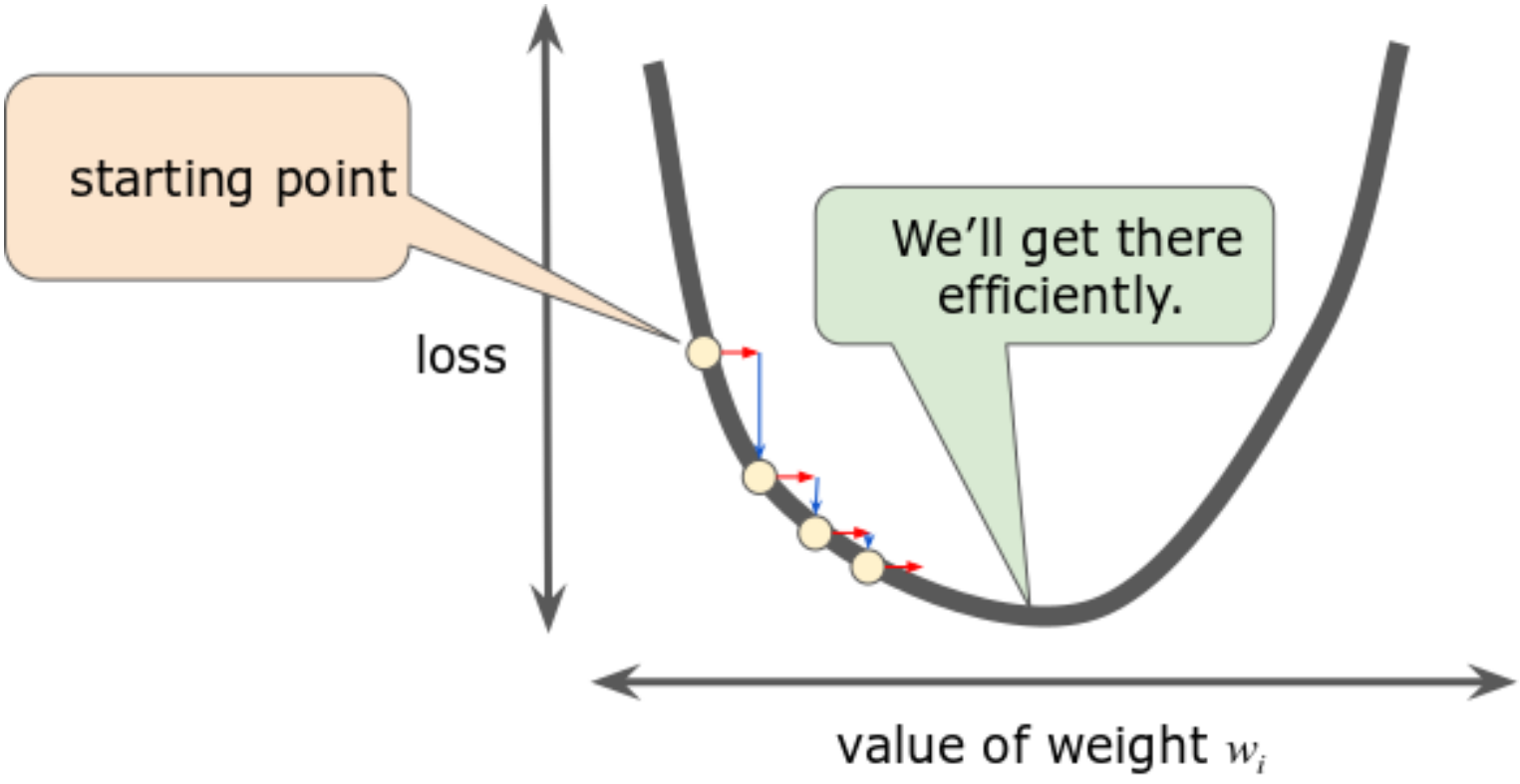
Learning Rate

- Conversely, if you specify a learning rate that is too large, the next point will perpetually bounce haphazardly across the bottom of the well:



Learning Rate

- There's a Goldilocks learning rate for every regression problem. The Goldilocks value is related to how flat the loss function is. If you know the gradient of the loss function is small then you can safely try a larger learning rate, which compensates for the small gradient and results in a larger step size.



Key Terms

- hyperparameter
- learning rate
- step size

Optimizing Learning Rate

- Check out jupyter notebook.

Stochastic Gradient Descent

- In gradient descent, a **batch** is the total number of examples you use to calculate the gradient in a single iteration.
- So far, we've assumed that the batch has been the entire data set.
- But often data sets contain huge numbers of examples with huge numbers of features.
- Consequently, a batch can be enormous. A very large batch may cause even a single iteration to take a very long time to compute.
- A large data set with randomly sampled examples probably contains redundant data. In fact, redundancy becomes more likely as the batch size grows.
- Some redundancy can be useful to smooth out noisy gradients, but enormous batches tend not to carry much more predictive value than large batches.

Stochastic Gradient Descent

- What if we could get the right gradient on average for much less computation?
- By choosing examples at random from our data set, we could estimate (albeit, noisily) a big average from a much smaller one.
- **Stochastic gradient descent (SGD)** takes this idea to the extreme–it uses only a single example (a batch size of 1) per iteration.
- Given enough iterations, SGD works but is very noisy. The term "stochastic" indicates that the one example comprising each batch is chosen at random.

Reducing Loss

- **Mini-batch stochastic gradient descent (mini-batch SGD)** is a compromise between full-batch iteration and SGD. A mini-batch is typically between 10 and 1,000 examples, chosen at random.
- Mini-batch SGD reduces the amount of noise in SGD but is still more efficient than full-batch.
- To simplify the explanation, we focused on gradient descent for a single feature. Rest assured that gradient descent also works on feature sets that contain multiple features.

Key Terms

- batch
- batch size
- mini-batch
- stochastic gradient descent (SGD)