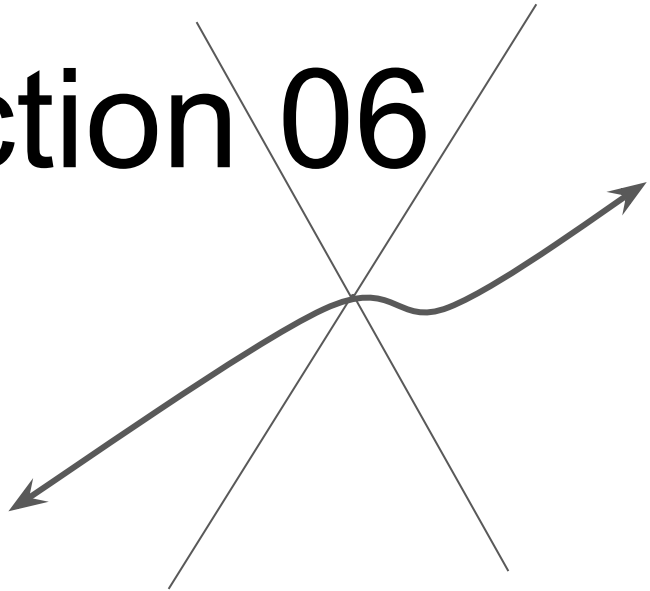
446 Section 06



Plans for today!

1. This
2. Reminders
3. Classification
    a. Binary Logistic Regression
    b. Multiclass Logistic Regression
4. PyTorch Tutorial
5. GD Review

Reminders

- HW2 due today! (1 day extension)
  - Late deadline is Saturday
  - Are you keeping track of late days? Use them!
- HW3 released! Due Wed, May 20th
- How was the midterm?
  - Q6 and Q15 are extra credit questions now

Classification

Classification

Binary → Sigmoid

$$P(y=1 \mid x) = \frac{1}{1+e^{-w^\top x}} = \frac{e^{w^\top x}}{1+e^{w^\top x}}$$

$$P(y=-1 \mid x) = \frac{1}{1+e^{w^\top x}}$$

$$\operatorname{argmax}_w \sum_{i=1}^{n} \log\left(\frac{1}{1+\mathrm{e}^{-y_i w^\top x_i}}\right)$$

Logistic loss

Multiclass → Softmax

$$P\left(y=c_j \,\middle|\, x\right) = \frac{e^{w_j^\top x}}{\sum_{j'} e^{w_{j'}^\top x}}$$

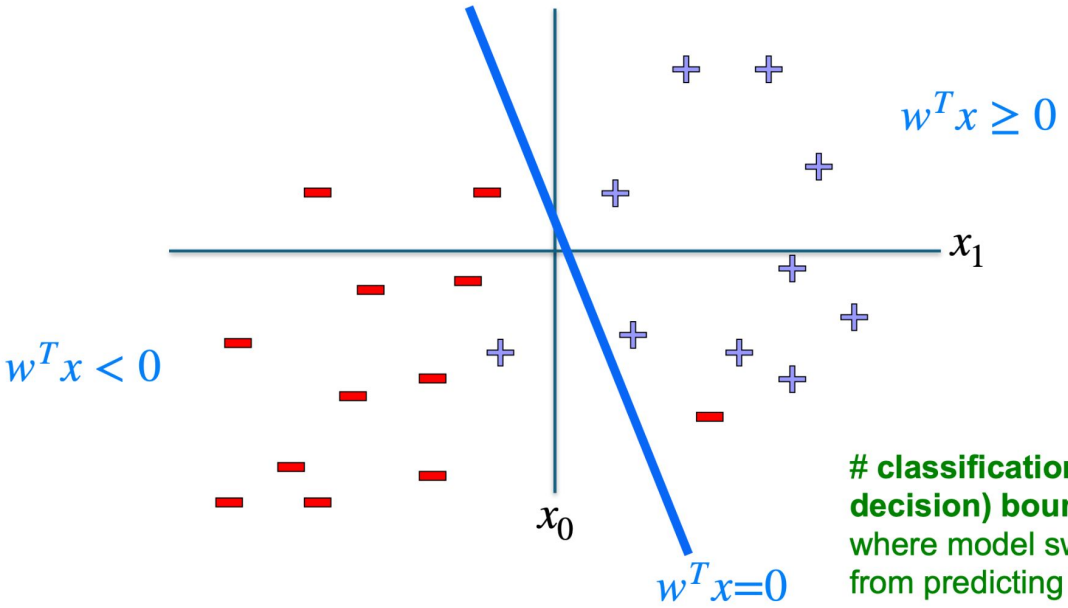
$$\operatorname{argmax}_w \sum_{i=1}^{n} \sum_{j=1}^{k} 1\left\{y_i = c_j\right\} \log\left(\frac{\mathrm{e}^{w_j^\top x_i}}{\sum_{j'=1}^{k} e^{w_{j'}^\top x_i}}\right)$$

Cross entropy loss

Binary Logistic Regression

Linear Classifier

- Logistic regression is a linear classifier
- Can classify between 2 classes



 # classification (or decision) boundary: where model switches from predicting 0 → 1

Logistic Loss

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$

Prediction

Label

Goal is to minimize errors

Very difficult to optimize…

Expected Loss:

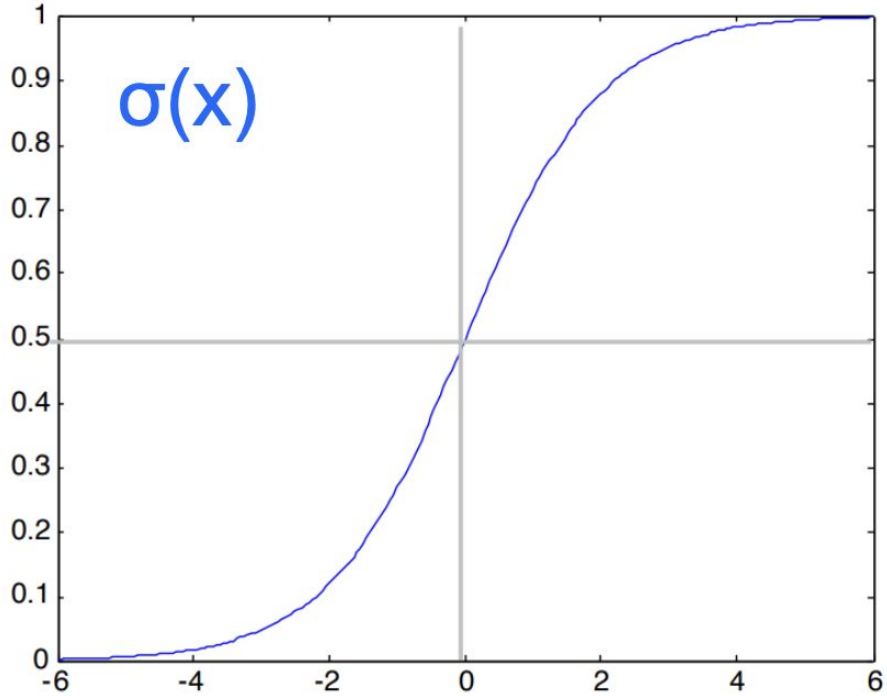
$$\mathbb{E}_{XY}[1\{f(x) \neq y\}] = 1 - P(Y = f(x) \mid X = x)$$

Sigmoid Function

Trying to learn weights w

$$P(Y=1 \mid X=x) = \frac{1}{1+e^{-w^\top x}} = \sigma(w^\top x)$$

$$\sigma\left(w_0 + \sum_k w_k x_k\right) = \frac{1}{1+e^{-w_0 + \sum_k w_k x_k}}$$



Decision Rule:

$$P(Y=1 \mid X) \geq P(Y=0 \mid X)$$

Logistic Loss:

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

Practice

Question 1a

In classification problems, we predict a discrete label y given features x. The standard approach is to model the conditional probability $P(Y = y \mid x)$ and then predict the most likely class. For binary classification, **logistic regression** models this probability for binary labels $y \in \{0, 1\}$ as:

$$P(Y = 1|x) = \sigma(w^T x) = \frac{1}{1 + e^{-w^T x}}$$

and

$$P(Y = 0|x) = 1 - \sigma(w^T x) = 1 - \frac{1}{1 + e^{-w^T x}} = \frac{e^{-w^T x}}{1 + e^{-w^T x}}$$

where $w \in \mathbb{R}^d$ is the weight vector.

(a) As we saw in lecture, it is often easier to use labels $y \in \{-1, +1\}$ for optimization. Using this technique, show that $P_w(y|x) = \frac{1}{1+\exp(-yw^T x)}$ correctly represents both cases from above (i.e., $P(Y = 1|x)$ and $P(Y = 0|x)$).

Question 1a

Solution:

First, we swap out the labels:

$$P(Y=+1|x)=\frac{1}{1+e^{-w^Tx}} \quad \text{and} \quad P(Y=-1|x)=\frac{e^{-w^Tx}}{1+e^{-w^Tx}}$$

Then, note that:

$$P(Y=-1|x)=\frac{e^{-w^Tx}}{1+e^{-w^Tx}}=\frac{e^{-w^Tx}}{1+e^{-w^Tx}}\cdot\frac{e^{w^Tx}}{e^{w^Tx}}=\frac{1}{1+e^{w^Tx}}$$

In other words, we've shown the useful identity $1-\sigma(z)=\sigma(-z)$.

Now we can verify the unified form $P_w(y|x)=\frac{1}{1+\exp(-yw^Tx)}$ for both cases:

- For $y=+1$: $\quad P_w(+1|x)=\frac{1}{1+e^{(-yw^Tx)}}=\frac{1}{1+e^{(-(+1)w^Tx)}}=\frac{1}{1+e^{-w^Tx}}=P(Y=+1|x)$
- For $y=-1$: $\quad P_w(-1|x)=\frac{1}{1+e^{(-yw^Tx)}}=\frac{1}{1+e^{(-(-1)w^Tx)}}=\frac{1}{1+e^{w^Tx}}=P(Y=-1|x)$

Both match the original expressions, thus $P_w(y|x)=\frac{1}{1+e^{(-yw^Tx)}}$ correctly represents both cases.

Question 1b

(b) Given a dataset of n i.i.d. observations $\{(x_i, y_i)\}_{i=1}^{n}$ where $y_i \in \{-1, +1\}$, use maximum liklihood estimation to derive the formula for logistic loss, which is defined as:

$$\hat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} \log(1 + \exp(-y_i w^T x_i))$$

Question 1b

Solution:

We have n i.i.d. observations $\{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. Then:

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

$$\downarrow$$

$$P_w(y|x) = \prod_{i=1}^n \sigma(-y_i w^T x_i) = \prod_{i=1}^n \frac{1}{1 + \exp(-y_i w^T x_i)}$$

$$\downarrow$$

$$\ell_w = -\log P_w(y|x) = -\sum_{i=1}^n \log(\sigma(-y_i w^T x_i)) = -\sum_{i=1}^n \log\left(\frac{1}{1 + \exp(-y_i w^T x_i)}\right)$$

$$= \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i)) \qquad \text{[Note: } \log(\frac{1}{a}) = \log(a^{-1}) = -\log(a)\text{]}$$

$$\downarrow$$

$$\hat{w}_{MLE} = \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i w^T x_i))$$

Question 1c

(c) Suppose we have a dataset that is linearly separable. If we use the objective function derived in part (b) and run gradient descent, will it converge to a specific, finite weight vector w? What happens to $||w||$?

Solution:

No; it will not converge to a specific, finite weight vector. Because the data is linearly separable, there exists some w^* such that $y_i(w^*)^T x_i > 0$ for all i, and scaling w^* by any $c > 0$ still separates the data. And due to the exponential term in the logistic loss function, the larger the weights get, the closer the loss will be to 0 (i.e., as $w \to \infty$, $\exp(-y_i w^T x_i) \to 0$ and thus $\log(1 + \exp(-y_i w^T x_i)) \to 0$). As we take more and more steps, $||w|| \to \infty$.

Multi-class Logistic Regression

Binary vs Multiclass Logistic Regression

2 classes → k classes

Sigmoid

Conditional probabilities

Softmax

$$P(y=1 \mid x) = \frac{1}{1+e^{-w^\top x}} = \frac{e^{w^\top x}}{1+e^{w^\top x}}$$

$$P(y=-1 \mid x) = \frac{1}{1+e^{w^\top x}}$$

$$P\left(y=c_j \mid x\right) = \frac{e^{w_j^\top x}}{\sum_{j'} e^{w_{j'}^\top x}}$$

MLE

$$\text{argmax}_w \sum_{i=1}^{n} \log\left(\frac{1}{1+\text{e}^{-y_i w^\top x_i}}\right)$$

Logistic loss

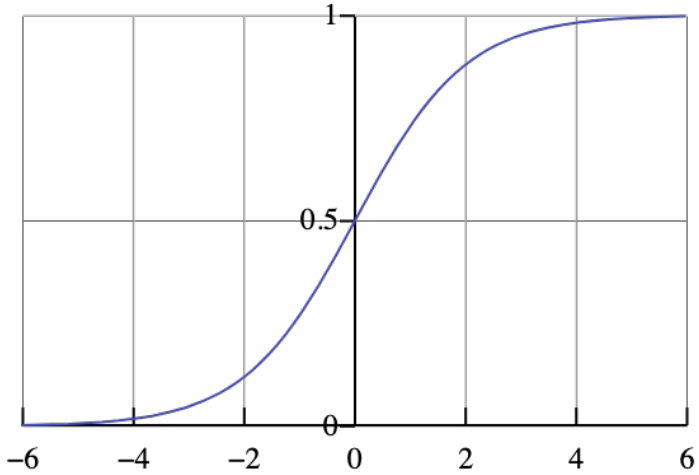
$$\text{argmax}_w \sum_{i=1}^{n} \sum_{j=1}^{k} 1\left\{y_i = c_j\right\} \log\left(\frac{\text{e}^{w_j^\top x_i}}{\sum_{j'=1}^{k} e^{w_{j'}^\top x_i}}\right)$$

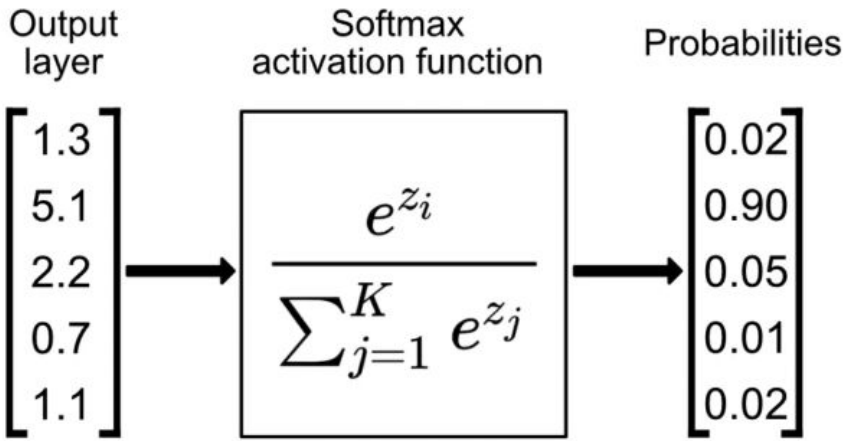
Binary cross entropy

Softmax

$$\text{Sigmoid } S(x) = \frac{1}{1 + e^{-x}}$$

$$\text{Softmax } \sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$





Practice

Question 2a

Note that inputs can be negative; exp() makes them positive!

When we move beyond binary classification to $k > 2$ classes, we use the **softmax function** to represent the conditional probability $P(Y = c_j|x)$ for each class c_j:

$$P(Y = c_j|x) = \frac{e^{w_j^T x}}{\sum_{j'=1}^{k} e^{w_{j'}^T x}}$$

The probability that a given input x belongs to class c_j is proportional to the exponential of its logit $w_j^T x$, normalized by the logits of all other possible classes. This ensures that the predicted probabilities are all non-negative and sum to 1 across all k classes.

(a) Show that when $k = 2$, this reduces to the binary logistic regression model. Specifically, treat class c_1 as $y = 1$ and show that $P(Y = c_1|x) = \sigma(w^T x)$ for some weight vector w. What is w in terms of w_1 and w_2?

Question 2a

Solution:

Plugging in $k = 2$ into the equation, we get:

$$P(Y = c_1|x) = \frac{e^{w_1^T x}}{e^{w_1^T x} + e^{w_2^T x}} \cdot \frac{e^{-w_1^T x}}{e^{-w_1^T x}} = \frac{1}{1 + (e^{w_2^T x} \cdot e^{-w_1^T x})} = \frac{1}{1 + e^{(w_2 - w_1)^T x}} = \frac{1}{1 + e^{-(w_1 - w_2)^T x}}$$

Let $w = w_1 - w_2$. We get:

$$\frac{1}{1 + e^{-(w_1 - w_2)^T x}} = \frac{1}{1 + e^{-w^T x}} = \sigma(w^T x)$$

Therefore, $w = w_1 - w_2$.
This shows that softmax with $k = 2$ is equivalent to binary logistic regression; they're the same model, just written with different parameterizations.

Question 2b

(b) Suppose a 3-class classifier produces logits $w_1^T x = \log 3$, $w_2^T x = \log 5$, and $w_3^T x = \log 1$ on some input x. Compute the softmax probabilities. Which class is predicted?

Solution:

We plug in the given values into the softmax function for each class:

$$P(Y = c_1|x) = \frac{e^{w_1^T x}}{\sum_{j'=1}^{3} e^{w_{j'}^T x}} = \frac{e^{\log 3}}{e^{\log 3} + e^{\log 5} + e^{\log 1}} = \frac{3}{3+5+1} = \frac{1}{3}$$

$$P(Y = c_2|x) = \frac{e^{w_2^T x}}{\sum_{j'=1}^{3} e^{w_{j'}^T x}} = \frac{e^{\log 5}}{e^{\log 3} + e^{\log 5} + e^{\log 1}} = \frac{5}{3+5+1} = \frac{5}{9}$$

$$P(Y = c_3|x) = \frac{e^{w_3^T x}}{\sum_{j'=1}^{3} e^{w_{j'}^T x}} = \frac{e^{\log 1}}{e^{\log 3} + e^{\log 5} + e^{\log 1}} = \frac{1}{3+5+1} = \frac{1}{9}$$

Class 2 is predicted, since it has the highest probability.

Gradescope Participation Question

What is the generalization of the sigmoid function for multiclass logistic regression?

Softmax

Pytorch Tutorial

Review:
Gradient Descent
vs.
Stochastic Gradient Descent
vs.
Mini-Batch Gradient Descent

GD to SGD

$$w^{(t+1)} = w^{(t)} - \eta \cdot \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i \left(w^{(t)} \right).$$

The computational cost of a single step here is $\mathcal{O}(dn)$. To reduce cost, one idea is to just use a subset of all samples to approximate the full gradient. Specifically, consider revising the gradient descent step as follows:

$$w^{(t+1)} = w^{(t)} - \eta \cdot \nabla \ell_{I_t}(w^{(t)}),$$

where I_t is chosen randomly within $\{1, 2, ..., n\}$ with equal probability. This is called **stochastic gradient descent (SGD)**, and the computational cost of a single step now reduces to $\mathcal{O}(d)$.

Is this estimator unbiased?

Yes!

- $\mathbb{E}_{I_t} \left[\nabla \ell_{I_t}(w^{(t)}) \right] = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i \left(w^{(t)} \right)$, which is the full gradient. Hence the estimate of gradient is unbiased.

3a. Stochastic Gradient Descent

(b) What disadvantages can SGD have? How can we balance between the noise in updates and computational cost? **Solution:**

By treating SGD as noise-injected gradient descent:

$$\nabla \ell_{I_t}(w^{(t)}) = \mathbb{E}_{I_t}\left[\nabla \ell_{I_t}(w^{(t)}\right] + e_t = \frac{1}{n}\sum_{i=1}^{n} \ell_i(w^{(t)}) + e_t,$$

where e_t represents the noise term and is random, we know that the steps taken towards a minimum can be very noisy because the gradient used in updating involves noise. One way to balance the noise in updates and computational cost is to consider a technique called **mini-batching**, which is employed with SGD.

Issues with SGD → MiniBatch GD

- Although the estimator is unbiased overall, a single point gradient estimate is very noisy
  - **Pros:**
    - Noise within the optimization process is not inherently a bad thing. It can help you escape sub-optimal local minima and “wander/explore” the loss landscape more.
    - For giant datasets SGD is also more computationally feasible
  - **Cons:**
    - Noisy updates mean you could also wander off the optimal path if the loss landscape is simpler

Gradient descent is theoretically nice but computationally expensive…

SGD is noisy (maybe too noisy) but computationally efficient…

Meet in the middle with MiniBatch GD!

Gradient Descent Variants (for this class)

Gradient Descent (GD)

- Take a step after looking at **every** single data point in the training set

Stochastic Gradient Descent (SGD)

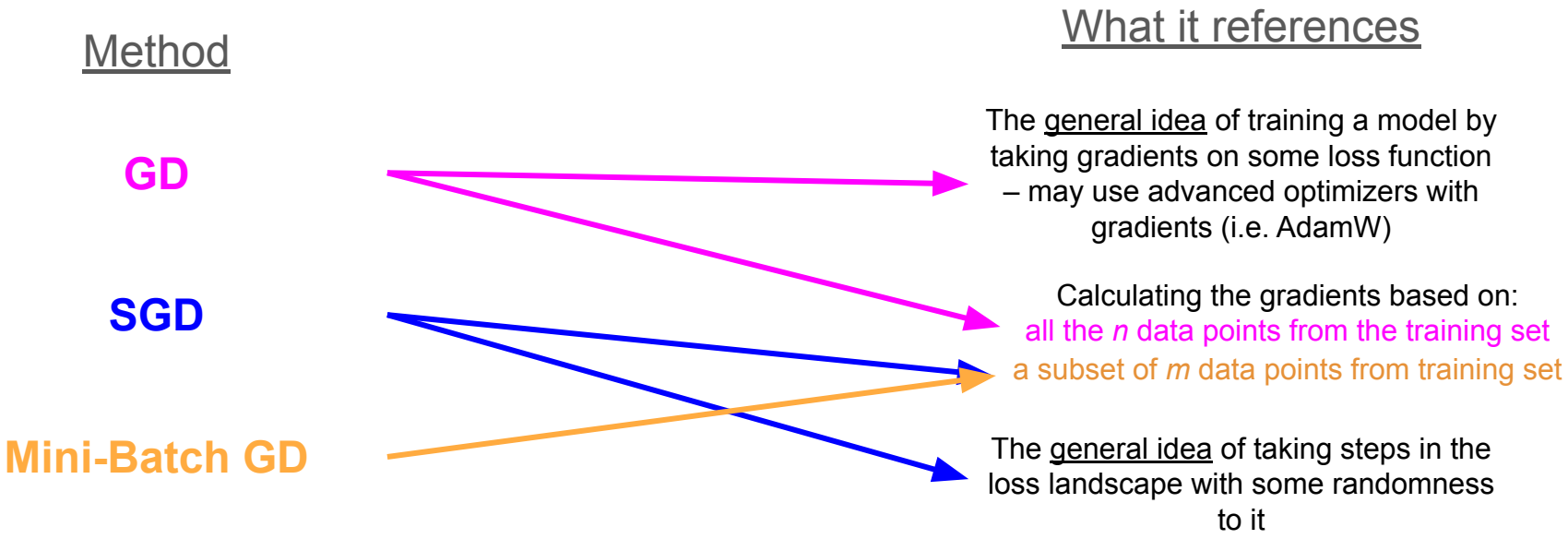
- Take a step after looking at **1** randomly sampled point from the training set

Mini-Batch Gradient Descent (Mini-Batch GD)

- Take a step after looking at ***k*** sampled data points from the training set

These definitions are specific and important for this class, but…

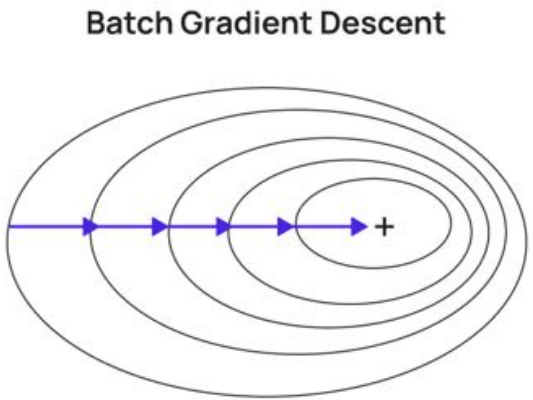
Gradient Descent Variants (as discussed in industry)

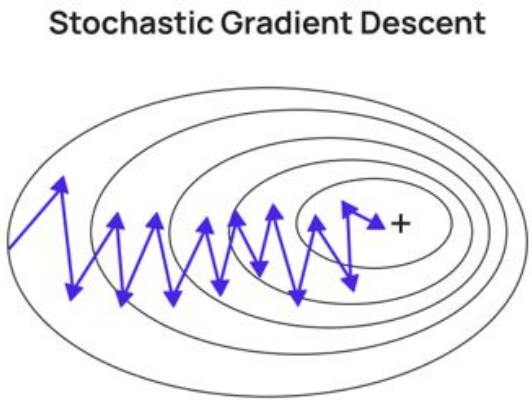


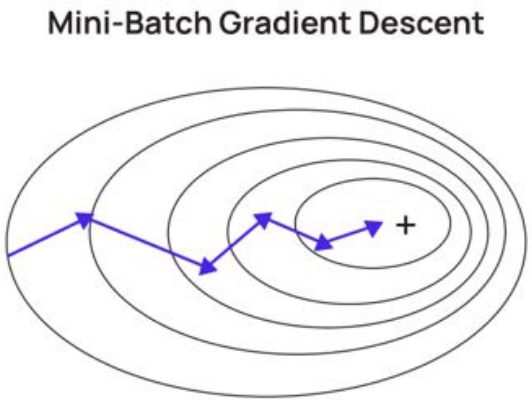
This is how people talk about them IRL. Note that SGD & Minibatch are used interchangeably.

Note: From here on out in this class, the definitions of these terms are **these ones →**

Gradient Descent (GD)
Take a step after looking at **every** single datapoint in the training set
Stochastic Gradient Descent (SGD)
Take a step after looking at **1** randomly sampled point from the training set
Mini-Batch Gradient Descent (Mini-Batch GD)
Take a step after looking at ***n*** sampled datapoints from the training set







Another way of looking at things!

4c. MiniBatch Gradient Descent

(c) Are there other extensions or variants of the basic stochastic gradient descent algorithm?

Many improvements, which are listed below, on the basic SGD algorithm have been developed and used.

- Implicit updates (ISGD)
- Momentum
- Averaged stochastic gradient descent
- Adaptive gradient algorithm (AdaGrad)
- Root Mean Square Propagation (RMSProp)
- Adaptive Moment Estimation (Adam)

Basically, these methods consider to fine-tune the step size parameter, take previous update magnitude into account, or introduce the second moments of the gradients when updating. For example, Momentum remembers the previous update magnitude so that $w^{(t)}$ tends to keep traveling in the same direction, preventing oscillations:

$$w^{(t+1)} = w^{(t)} - \eta \nabla \ell_{I_t}(w^{(t)}) + \alpha(w^{(t)} - w^{(t-1)}).$$

Adam, as another example, considers to tune to step size with the second moments of the gradients:

$$w^{(t+1)} = w^{(t)} - \eta G\Big(\nabla \ell^{(t)}, \nabla \ell^{(t-1)}, ..., (\nabla \ell^{(t)})^2, (\nabla \ell^{(t-1)})^2, ...\Big),$$

where $\nabla \ell^{(t)} = \nabla \ell_{I_t}(w^{(t)})$ and G is a function that involves element-wise square of all previous gradients. The paper below provides more details on Adam: https://arxiv.org/pdf/1412.6980.pdf.

Note: you'll be using Adam/AdamW quite often on HWs (and in the real world)