On Logistic Regression: Gradients of the Log Loss, Multi-Class Classification, and Other Optimization Techniques

Karl Stratos

June 20, 2018

Recall: Logistic Regression

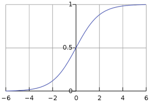
- **Task.** Given input $\boldsymbol{x} \in \mathbb{R}^d$, predict either 1 or 0 (on or off).

Recall: Logistic Regression

- **Task.** Given input $\boldsymbol{x} \in \mathbb{R}^d$, predict either 1 or 0 (on or off).

- **Model.** The probability of on is parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ as a dot product squashed under the **sigmoid/logistic** function $\sigma : \mathbb{R} \to [0,1]$.

$$p\left(1|\boldsymbol{x}, \boldsymbol{w}\right) := \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) := \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})}$$

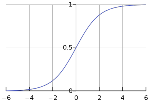


Recall: Logistic Regression

- **Task.** Given input $\boldsymbol{x} \in \mathbb{R}^d$, predict either 1 or 0 (on or off).

- **Model.** The probability of on is parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ as a dot product squashed under the **sigmoid/logistic** function $\sigma : \mathbb{R} \to [0, 1]$.

$$p(1|\boldsymbol{x}, \boldsymbol{w}) := \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) := \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})}$$



The probability of off is

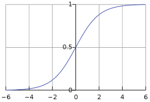
$$p(0|\boldsymbol{x}, \boldsymbol{w}) = 1 - \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) = \sigma(-\boldsymbol{w} \cdot \boldsymbol{x})$$

Recall: Logistic Regression

- **Task.** Given input $\boldsymbol{x} \in \mathbb{R}^d$, predict either 1 or 0 (on or off).

- **Model.** The probability of on is parameterized by $\boldsymbol{w} \in \mathbb{R}^d$ as a dot product squashed under the **sigmoid/logistic** function $\sigma : \mathbb{R} \to [0, 1]$.

$$p\left(1|\boldsymbol{x}, \boldsymbol{w}\right) := \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) := \frac{1}{1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})}$$



The probability of off is

$$p\left(0|\boldsymbol{x}, \boldsymbol{w}\right) = 1 - \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) = \sigma(-\boldsymbol{w} \cdot \boldsymbol{x})$$

- Today's focus:
  1. **Optimizing the log loss by gradient descent**
  2. **Multi-class classification** to handle more than two classes
  3. **More on optimization**: Newton, stochastic gradient descent

Overview

Gradient Descent on the Log Loss

Multi-Class Classification

More on Optimization

- Newton's Method
- Stochastic Gradient Descent (SGD)

Negative Log Probability Under Logistic Function

- Using $y \in \{0, 1\}$ to denote the two classes,

$$
\begin{aligned}
-\log p\,(y|\boldsymbol{x}, \boldsymbol{w}) &= -y \log \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) - (1 - y) \log \sigma(-\boldsymbol{w} \cdot \boldsymbol{x}) \\
&= \begin{cases} \log(1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})) & \text{if } y = 1 \\ \log(1 + \exp(\boldsymbol{w} \cdot \boldsymbol{x})) & \text{if } y = 0 \end{cases}
\end{aligned}
$$

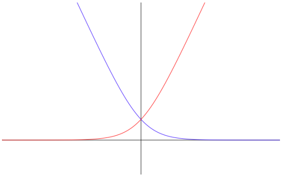
  is convex in $\boldsymbol{w}$.

Negative Log Probability Under Logistic Function

- Using $y \in \{0, 1\}$ to denote the two classes,

$$-\log p\,(y|\boldsymbol{x}, \boldsymbol{w}) = -y \log \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) - (1 - y) \log \sigma(-\boldsymbol{w} \cdot \boldsymbol{x})$$
$$= \begin{cases} \log(1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})) & \text{if } y = 1 \\ \log(1 + \exp(\boldsymbol{w} \cdot \boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

is convex in $\boldsymbol{w}$.

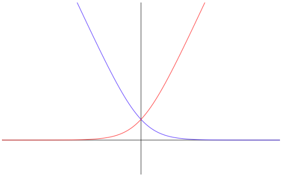


Negative Log Probability Under Logistic Function

- Using $y \in \{0, 1\}$ to denote the two classes,

$$-\log p\left(y|\boldsymbol{x}, \boldsymbol{w}\right) = -y \log \sigma(\boldsymbol{w} \cdot \boldsymbol{x}) - (1-y) \log \sigma(-\boldsymbol{w} \cdot \boldsymbol{x})$$
$$= \begin{cases} \log(1 + \exp(-\boldsymbol{w} \cdot \boldsymbol{x})) & \text{if } y = 1 \\ \log(1 + \exp(\boldsymbol{w} \cdot \boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

is convex in $\boldsymbol{w}$.



- Gradient given by

$$\nabla_{\boldsymbol{w}} \left(-\log p\left(y|\boldsymbol{x}, \boldsymbol{w}\right)\right) = -(y - \sigma(\boldsymbol{w} \cdot \boldsymbol{x}))\boldsymbol{x}$$

Logistic Regression Objective

- Given iid samples $S = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$, find $\boldsymbol{w}$ that minimizes the empirical negative log likelihood of S ("log loss"):

$$J_S^{\mathrm{LOG}}(\boldsymbol{w}) := -\frac{1}{n} \sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{w}\right)$$

- Gradient?

Logistic Regression Objective

- Given iid samples $S = \left\{ (\boldsymbol{x}^{(i)}, y^{(i)}) \right\}_{i=1}^{n}$, find $\boldsymbol{w}$ that minimizes the empirical negative log likelihood of S ("log loss"):

$$J_S^{\mathrm{LOG}}(\boldsymbol{w}) := -\frac{1}{n} \sum_{i=1}^{n} \log p \left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{w} \right)$$

- Gradient?

$$\nabla J_S^{\mathrm{LOG}}(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \sigma \left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \right) \right) \boldsymbol{x}^{(i)}$$

Logistic Regression Objective

- Given iid samples $S = \left\{(\boldsymbol{x}^{(i)}, y^{(i)})\right\}_{i=1}^{n}$, find $\boldsymbol{w}$ that minimizes the empirical negative log likelihood of S ("log loss"):

$$J_S^{\mathrm{LOG}}(\boldsymbol{w}) := -\frac{1}{n}\sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{w}\right)$$

- Gradient?

$$\nabla J_S^{\mathrm{LOG}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}$$

- Unlike in linear regression, there is no closed-form solution for

$$\boldsymbol{w}_S^{\mathrm{LOG}} := \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathbb{R}^d} J_S^{\mathrm{LOG}}(\boldsymbol{w})$$

Logistic Regression Objective

- Given iid samples $S = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$, find $\boldsymbol{w}$ that minimizes the empirical negative log likelihood of S ("log loss"):

$$J_S^{\mathrm{LOG}}(\boldsymbol{w}) := -\frac{1}{n}\sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{w}\right)$$

- Gradient?

$$\nabla J_S^{\mathrm{LOG}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}$$

- Unlike in linear regression, there is no closed-form solution for

$$\boldsymbol{w}_S^{\mathrm{LOG}} := \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathbb{R}^d} J_S^{\mathrm{LOG}}(\boldsymbol{w})$$

- But $J_S^{\mathrm{LOG}}(\boldsymbol{w})$ is **convex** and **differentiable**! So we can do gradient descent and approach an optimal solution.

Gradient Descent for Logistic Regression

Input: training objective

$$J_S^{\mathrm{LOG}}(\boldsymbol{w}) := -\frac{1}{n} \sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{w}\right)$$

number of iterations T

Output: parameter $\hat{\boldsymbol{w}} \in \mathbb{R}^n$ such that $J_S^{\mathrm{LOG}}(\hat{\boldsymbol{w}}) \approx J_S^{\mathrm{LOG}}(\boldsymbol{w}_S^{\mathrm{LOG}})$

1. Initialize θ^0 (e.g., randomly).
2. For $t = 0 \ldots T - 1$,

$$\theta^{t+1} = \theta^t + \frac{\eta^t}{n} \sum_{i=1}^{n} \left(y^{(i)} - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right) \boldsymbol{x}^{(i)}$$

3. Return θ^T.

Overview

Gradient Descent on the Log Loss

Multi-Class Classification

More on Optimization

- Newton's Method
- Stochastic Gradient Descent (SGD)

From Binary to m Classes

- A logistic regressor has a single $\boldsymbol{w} \in \mathbb{R}^d$ to define the probability of "on" (against "off"):

$$p\left(1 | \boldsymbol{x}, \boldsymbol{w}\right) = \frac{\exp(\boldsymbol{w} \cdot \boldsymbol{x})}{1 + \exp(\boldsymbol{w} \cdot \boldsymbol{x})}$$

From Binary to m Classes

- A logistic regressor has a single $\boldsymbol{w} \in \mathbb{R}^d$ to define the probability of "on" (against "off"):

$$p\left(1|\boldsymbol{x}, \boldsymbol{w}\right) = \frac{\exp(\boldsymbol{w} \cdot \boldsymbol{x})}{1 + \exp(\boldsymbol{w} \cdot \boldsymbol{x})}$$

- A **log-linear model** has $\boldsymbol{w}^y \in \mathbb{R}^d$ for each possible class $y \in \{1 \ldots m\}$ to define the probability of the class equaling y:

$$p\left(y \middle| \boldsymbol{x}, \theta\right) = \frac{\exp(\boldsymbol{w}^y \cdot \boldsymbol{x})}{\sum_{y'=1}^{m} \exp(\boldsymbol{w}^{y'} \cdot \boldsymbol{x})}$$

where $\theta = \left\{\boldsymbol{w}^t\right\}_{t=1}^{m}$ denotes the set of parameters

From Binary to m Classes

- A logistic regressor has a single $\boldsymbol{w} \in \mathbb{R}^d$ to define the probability of "on" (against "off"):

$$p\left(1 | \boldsymbol{x}, \boldsymbol{w}\right) = \frac{\exp(\boldsymbol{w} \cdot \boldsymbol{x})}{1 + \exp(\boldsymbol{w} \cdot \boldsymbol{x})}$$

- A **log-linear model** has $\boldsymbol{w}^y \in \mathbb{R}^d$ for each possible class $y \in \{1 \ldots m\}$ to define the probability of the class equaling y:

$$p\left(y \middle| \boldsymbol{x}, \theta\right) = \frac{\exp(\boldsymbol{w}^y \cdot \boldsymbol{x})}{\sum_{y'=1}^m \exp(\boldsymbol{w}^{y'} \cdot \boldsymbol{x})}$$

  where $\theta = \left\{\boldsymbol{w}^t\right\}_{t=1}^m$ denotes the set of parameters

- Is this a valid probability distribution?

Softmax Formulation

- We can transform any vector $z \in \mathbb{R}^m$ into a probability distribution over m elements by the **softmax function** $\text{softmax} : \mathbb{R}^m \to \Delta^{m-1}$,

$$\text{softmax}_i(\boldsymbol{z}) := \frac{\exp(\boldsymbol{z}_i)}{\sum_{j=1}^{m} \exp(\boldsymbol{z}_j)} \qquad \forall i = 1 \ldots m$$

Softmax Formulation

- We can transform any vector $z \in \mathbb{R}^m$ into a probability distribution over m elements by the **softmax function** $\text{softmax} : \mathbb{R}^m \to \Delta^{m-1}$,

$$\text{softmax}_i(\boldsymbol{z}) := \frac{\exp(\boldsymbol{z}_i)}{\sum_{j=1}^m \exp(\boldsymbol{z}_j)} \qquad \forall i = 1 \dots m$$

- A log-linear model is a linear transformation by a matrix $W \in \mathbb{R}^{m \times d}$ (with rows $\boldsymbol{w}^1 \dots \boldsymbol{w}^m$) followed by softmax:

$$p\left(y | \boldsymbol{x}, W\right) = \text{softmax}_y(W\boldsymbol{x})$$

Negative Log Probability Under Log-Linear Model

- For any $y \in \{1 \dots m\}$,

$$-\log p\left(y \middle| \boldsymbol{x}, \theta\right) = \underbrace{\log \left(\sum_{y'=1}^{m} \exp\left(\boldsymbol{w}^{y'} \cdot \boldsymbol{x} \right) \right)}_{\text{constant wrt. } y} - \underbrace{\boldsymbol{w}^{y} \cdot \boldsymbol{x}}_{\text{linear}}$$

is convex in each $\boldsymbol{w}^{l} \in \theta$.

Negative Log Probability Under Log-Linear Model

- For any $y \in \{1 \ldots m\}$,

$$-\log p\left(y \middle| \boldsymbol{x}, \theta\right) = \underbrace{\log \left(\sum_{y'=1}^{m} \exp\left(\boldsymbol{w}^{y'} \cdot \boldsymbol{x}\right) \right)}_{\text{constant wrt. } y} - \underbrace{\boldsymbol{w}^{y} \cdot \boldsymbol{x}}_{\text{linear}}$$

  is convex in each $\boldsymbol{w}^{l} \in \theta$.

- For any $l \in \{1 \ldots m\}$, the gradient wrt. $\boldsymbol{w}^{l}$ is given by

$$\nabla_{\boldsymbol{w}^{l}} \left(-\log p\left(y \middle| \boldsymbol{x}, \theta\right)\right) = \begin{cases} -(1 - p\left(l \middle| \boldsymbol{x}, \theta\right))\boldsymbol{x} & \text{if } l = y \\ p\left(l \middle| \boldsymbol{x}, \theta\right) \boldsymbol{x} & \text{if } l \neq y \end{cases}$$

Log-Linear Model Objective

- Given iid samples $S = \left\{(\boldsymbol{x}^{(i)}, y^{(i)})\right\}_{i=1}^{n}$, find $\boldsymbol{w}$ that minimizes the empirical negative log likelihood of S

$$J_S^{\mathrm{LLM}}(\boldsymbol{w}) := -\frac{1}{n}\sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \theta\right)$$

  This is the so-called **cross-entropy loss**.

- Again convex and differentiable, can be optimized by gradient descent to reach an optimal solution.

Overview

One Way to Motivate Gradient Descent

- At current parameter value $\theta \in \mathbb{R}^d$, choose update $\Delta \in \mathbb{R}^d$ by minimizing the first-order Taylor approximation around θ with squared l_2 regularization:

$$J(\theta + \Delta) \approx \underbrace{J(\theta) + \nabla J(\theta)^\top \Delta + \frac{1}{2} ||\Delta||_2^2}_{J_1(\Delta)}$$

$$\Delta^{\mathrm{GD}} = \underset{\Delta \in \mathbb{R}^d}{\arg\min}\, J_1(\Delta) = -\nabla J(\theta)$$

One Way to Motivate Gradient Descent

- At current parameter value $\theta \in \mathbb{R}^d$, choose update $\Delta \in \mathbb{R}^d$ by minimizing the first-order Taylor approximation around θ with squared l_2 regularization:

$$J(\theta + \Delta) \approx \underbrace{J(\theta) + \nabla J(\theta)^\top \Delta + \frac{1}{2} ||\Delta||_2^2}_{J_1(\Delta)}$$

$$\Delta^{\mathrm{GD}} = \operatorname*{arg\,min}_{\Delta \in \mathbb{R}^d} J_1(\Delta) = -\nabla J(\theta)$$

- Equivalent to minimizing a second-order Taylor approximation but with uniform curvature

$$J(\theta) + \nabla J(\theta)\Delta + \frac{1}{2}\Delta^\top I_{d\times d}\Delta$$

Newton's Method

- Actually minimize the second-order Taylor approximation:

$$J(\theta + \Delta) \approx \underbrace{J(\theta) + \nabla J(\theta)^\top \Delta + \frac{1}{2}\Delta^\top \nabla^2 J(\theta)\Delta}_{J_2(\Delta)}$$

$$\Delta^{\mathrm{NEWTON}} = \arg\min_{\Delta \in \mathbb{R}^d} J_2(\Delta) = -\nabla^2 J(\theta)^{-1} \nabla J(\theta)$$

Newton's Method

- Actually minimize the second-order Taylor approximation:

$$J(\theta + \Delta) \approx \underbrace{J(\theta) + \nabla J(\theta)^\top \Delta + \frac{1}{2}\Delta^\top \nabla^2 J(\theta)\Delta}_{J_2(\Delta)}$$

$$\Delta^{\text{NEWTON}} = \arg\min_{\Delta \in \mathbb{R}^d} J_2(\Delta) = -\nabla^2 J(\theta)^{-1} \nabla J(\theta)$$

- Equivalent to gradient descent after a change of coordinates by $\nabla^2 J(\theta)^{1/2}$

Overview

Gradient Descent on the Log Loss

Multi-Class Classification

More on Optimization

- Newton's Method
- Stochastic Gradient Descent (SGD)

Loss: a Function of All Samples

- Empirical loss $J_S(\theta)$ is a function of the **entire data** S.*

$$\underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)^2}_{J_S^{\text{LS}}(\boldsymbol{w})} \qquad \underbrace{-\frac{1}{n}\sum_{i=1}^{n}\log p\left(y^{(i)}\middle|\boldsymbol{x}^{(i)},\boldsymbol{w}\right)}_{J_S^{\text{LOG}}(\boldsymbol{w})}$$

*We're normalizing by a constant for convenience without loss of generality.

Loss: a Function of All Samples

- Empirical loss $J_S(\theta)$ is a function of the **entire data** S.*

$$\underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)^2}_{J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{-\frac{1}{n}\sum_{i=1}^{n}\log p\left(y^{(i)}\middle|\boldsymbol{x}^{(i)},\boldsymbol{w}\right)}_{J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

- So the gradient is also a function of the entire data.

$$\underbrace{-\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\boldsymbol{x}^{(i)}}_{\nabla J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}}_{\nabla J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

*We're normalizing by a constant for convenience without loss of generality.

Loss: a Function of All Samples

- Empirical loss $J_S(\theta)$ is a function of the **entire data** S.*

$$\underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)^2}_{J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{-\frac{1}{n}\sum_{i=1}^{n}\log p\left(y^{(i)}\middle|\boldsymbol{x}^{(i)},\boldsymbol{w}\right)}_{J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

- So the gradient is also a function of the entire data.

$$\underbrace{-\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\boldsymbol{x}^{(i)}}_{\nabla J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}}_{\nabla J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

- Thus one update in gradient descent requires <u>summing over all n samples</u>.

*We're normalizing by a constant for convenience without loss of generality.

Gradient Estimation Based on a Single Sample

- What if we use a **single uniformly random** sample $i \in \{1 \ldots n\}$ to estimate the gradient?

$$\underbrace{-\left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(i)} J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\left(y^{(i)} - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(i)} J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

Gradient Estimation Based on a Single Sample

- What if we use a **single uniformly random** sample $i \in \{1 \ldots n\}$ to estimate the gradient?

$$\underbrace{-\left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(i)} J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\left(y^{(i)} - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(i)} J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

- In expectation, the gradient is consistent:

$$\mathbf{E}_i \left[\widehat{\nabla}^{(i)} J_S(\boldsymbol{w})\right] = \frac{1}{n} \sum_{i=1}^{n} \widehat{\nabla}^{(i)} J_S(\boldsymbol{w}) = \nabla J_S(\boldsymbol{w})$$

Gradient Estimation Based on a Single Sample

- What if we use a **single uniformly random** sample $i \in \{1 \ldots n\}$ to estimate the gradient?

$$\underbrace{-\left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(i)} J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\left(y^{(i)} - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(i)} J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

- In expectation, the gradient is consistent:

$$\mathbf{E}_i\left[\widehat{\nabla}^{(i)} J_S(\boldsymbol{w})\right] = \frac{1}{n} \sum_{i=1}^{n} \widehat{\nabla}^{(i)} J_S(\boldsymbol{w}) = \nabla J_S(\boldsymbol{w})$$

- This is **stochastic gradient descent (SGD)**: estimate the gradient with a single random sample. This is justified as long as the gradient is *consistent* in expectation.

SGD with Mini-Batches

- Instead of estimating the gradient based on a single random example i, use a random "mini-batch" $B \subseteq \{1 \ldots n\}$.

$$\underbrace{-\frac{1}{|B|} \sum_{i \in B} \left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(B)} J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\frac{1}{|B|} \sum_{i \in B} \left(y^{(i)} - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \right) \right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(B)} J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

SGD with Mini-Batches

- Instead of estimating the gradient based on a single random example i, use a random "mini-batch" $B \subseteq \{1 \dots n\}$.

$$\underbrace{-\frac{1}{|B|}\sum_{i \in B}\left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(B)} J_S^{\text{LS}}(\boldsymbol{w})} \qquad \underbrace{\frac{1}{|B|}\sum_{i \in B}\left(y^{(i)} - \sigma\left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(B)} J_S^{\text{LOG}}(\boldsymbol{w})}$$

- Still consistent: $\mathbf{E}_B\left[\widehat{\nabla}^{(B)} J_S(\boldsymbol{w})\right] = \nabla J_S(\boldsymbol{w})$

SGD with Mini-Batches

- Instead of estimating the gradient based on a single random example i, use a random "mini-batch" $B \subseteq \{1 \dots n\}$.

$$\underbrace{-\frac{1}{|B|} \sum_{i \in B} \left(y^{(i)} - \boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(B)} J_S^{\mathrm{LS}}(\boldsymbol{w})} \qquad \underbrace{\frac{1}{|B|} \sum_{i \in B} \left(y^{(i)} - \sigma \left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} \right) \right) \boldsymbol{x}^{(i)}}_{\widehat{\nabla}^{(B)} J_S^{\mathrm{LOG}}(\boldsymbol{w})}$$

- Still consistent: $\mathbf{E}_B \left[\widehat{\nabla}^{(B)} J_S(\boldsymbol{w}) \right] = \nabla J_S(\boldsymbol{w})$

- Mini-batches allow for a more stable gradient estimation.
  - SGD is a special case with $|B| = 1$.

Stochastic Gradient Descent

Input: training objective $J(\theta) \in \mathbb{R}$ of form

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} J_i(\theta)$$

, number of iterations T
Output: parameter $\hat{\theta} \in \mathbb{R}^n$ such that $J(\hat{\theta})$ is small

1. Initialize $\hat{\theta}$ (e.g., randomly).
2. For $t = 0 \ldots T - 1$,
   2.1 For $i \in \{1 \ldots n\}$ in **random order**,

$$\hat{\theta} \leftarrow \hat{\theta} - \eta^{t,i} \nabla J_i(\hat{\theta})$$

3. Return $\hat{\theta}$.

Stochastic Gradient Descent for Linear Regression

Input: training objective

$$J_S^{\mathrm{LS}}(\boldsymbol{w}) := \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{w}\cdot\boldsymbol{x}^{(i)}\right)^2$$

number of iterations T

Output: parameter $\hat{\boldsymbol{w}} \in \mathbb{R}^n$ such that $J_S^{\mathrm{LS}}(\hat{\boldsymbol{w}}) \approx J_S^{\mathrm{LS}}(\boldsymbol{w}_S^{\mathrm{LS}})$

1. Initialize $\hat{\boldsymbol{w}}$ (e.g., randomly).
2. For $t = 0 \ldots T-1$,
   2.1 For $i \in \{1 \ldots n\}$ in **random order**,

$$\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} - \eta^{t,i}\left(y^{(i)} - \hat{\boldsymbol{w}}^\top \cdot \boldsymbol{x}^{(i)}\right)\boldsymbol{x}^{(i)}$$

3. Return $\hat{\boldsymbol{w}}$.

Stochastic Gradient Descent for Logistic Regression

Input: training objective

$$J_S^{\mathrm{LOG}}(\boldsymbol{w}) := -\frac{1}{n}\sum_{i=1}^{n} \log p\left(y^{(i)} \middle| \boldsymbol{x}^{(i)}, \boldsymbol{w}\right)$$

number of iterations T

Output: parameter $\hat{\boldsymbol{w}} \in \mathbb{R}^n$ such that $J_S^{\mathrm{LOG}}(\hat{\boldsymbol{w}}) \approx J_S^{\mathrm{LOG}}(\boldsymbol{w}_S^{\mathrm{LOG}})$

1. Initialize $\hat{\boldsymbol{w}}$ (e.g., randomly).
2. For $t = 0 \ldots T-1$,
   2.1 For $i \in \{1 \ldots n\}$ in **random order**,

$$\hat{\boldsymbol{w}} \leftarrow \hat{\boldsymbol{w}} + \eta^{t,i}\left(y^{(i)} - \sigma\left(\hat{\boldsymbol{w}} \cdot \boldsymbol{x}^{(i)}\right)\right)\boldsymbol{x}^{(i)}$$

3. Return $\hat{\boldsymbol{w}}$.

Summary

- **Logistic regression**: binary classifier that can be trained by optimizing the log loss
- **Log-linear model**: multi-class classifier that can be trained by optimizing the cross-entropy loss
- **Newton's method**: local search using the curvature of the loss function
- **SGD**: gradient descent with stochastic gradient estimation
  - Cornerstone of modern large-scale machine learning