

# Gradient Descent and Loss Functions

Mengye Ren

NYU

September 10, 2024

# Homework 1

- Homework 1 will be released on course website today (Sept 10). You have until Oct 3 noon (12pm) to finish.
- Submit PDF to Gradescope.
- Course website: <https://nyu-cs2565.github.io/2024-fall/>

## Review: ERM

# Our Machine Learning Setup

## Prediction Function

A **prediction function** gets input  $x$  and produces an output  $\hat{y} = f(x)$ .

# Our Machine Learning Setup

## Prediction Function

A **prediction function** gets input  $x$  and produces an output  $\hat{y} = f(x)$ .

## Loss Function

A **loss function**  $\ell(\hat{y}, y)$  evaluates an action in the context of the outcome  $y$ .

# Risk and the Bayes Prediction Function

## Definition

The **risk** of a prediction function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  is

$$R(f) = \mathbb{E}\ell(f(x), y).$$

In words, it's the **expected loss** of  $f$  on a new example  $(x, y)$  drawn randomly from  $P_{\mathcal{X} \times \mathcal{Y}}$ .

# Risk and the Bayes Prediction Function

## Definition

The **risk** of a prediction function  $f : \mathcal{X} \rightarrow \mathcal{Y}$  is

$$R(f) = \mathbb{E} \ell(f(x), y).$$

In words, it's the **expected loss** of  $f$  on a new example  $(x, y)$  drawn randomly from  $P_{\mathcal{X} \times \mathcal{Y}}$ .

## Definition

A **Bayes prediction function**  $f^*$  is a function that achieves the *minimal risk* among all possible functions:

$$f^* \in \arg \min_f R(f),$$

- The risk of a Bayes prediction function is called the **Bayes risk**.

# The Empirical Risk

Let  $\mathcal{D}_n = ((x_1, y_1), \dots, (x_n, y_n))$  be drawn i.i.d. from  $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$ .

## Definition

The **empirical risk** of  $f$  with respect to  $\mathcal{D}_n$  is

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i).$$

- The **unconstrained** empirical risk minimizer can overfit.
  - i.e. if we minimize  $\hat{R}_n(f)$  over **all functions**, we overfit.



# Constrained Empirical Risk Minimization

## Definition

A **hypothesis space**  $\mathcal{F}$  is a set of functions mapping  $\mathcal{X} \rightarrow \mathcal{Y}$ .

- This is the collection of prediction functions we are choosing from.

# Constrained Empirical Risk Minimization

## Definition

A **hypothesis space**  $\mathcal{F}$  is a set of functions mapping  $\mathcal{X} \rightarrow \mathcal{Y}$ .

- This is the collection of prediction functions we are choosing from.
- An **empirical risk minimizer** (ERM) in  $\mathcal{F}$  is

$$\hat{f}_n \in \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \ell(f(x_i), y_i).$$

- From now on “ERM” always means “constrained ERM”.
- So we should always specify the hypothesis space when we’re doing ERM.

# Example: Linear Least Squares Regression

## Setup

- Loss:  $\ell(\hat{y}, y) = (y - \hat{y})^2$

# Example: Linear Least Squares Regression

## Setup

- Loss:  $\ell(\hat{y}, y) = (y - \hat{y})^2$
- Hypothesis space:  $\mathcal{F} = \{f : \mathbb{R}^d \rightarrow \mathbb{R} \mid f(x) = w^T x, w \in \mathbb{R}^d\}$
- Given a data set  $\mathcal{D}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ ,
  - Our goal is to find the ERM  $\hat{f} \in \mathcal{F}$ .

## Example: Linear Least Squares Regression

### Objective Function: Empirical Risk

We want to find the function in  $\mathcal{F}$ , parametrized by  $w \in \mathbb{R}^d$ , that minimizes the empirical risk:

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n (w^T x_i - y_i)^2$$

## Example: Linear Least Squares Regression

### Objective Function: Empirical Risk

We want to find the function in  $\mathcal{F}$ , parametrized by  $w \in \mathbb{R}^d$ , that minimizes the empirical risk:

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n (w^T x_i - y_i)^2$$

- How do we solve this optimization problem?

$$\min_{w \in \mathbb{R}^d} \hat{R}_n(w)$$

- (For OLS there's a closed form solution, but in general there isn't.)

# Gradient Descent

# Unconstrained Optimization

## Setting

We assume that the objective function  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is *differentiable*.

We want to find

$$x^* = \arg \min_{x \in \mathbb{R}^d} f(x)$$



# The Gradient

- Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable at  $x_0 \in \mathbb{R}^d$ .
- The **gradient** of  $f$  at the point  $x_0$ , denoted  $\nabla_x f(x_0)$ , is the direction in which  $f(x)$  **increases fastest**, if we start from  $x_0$ .
- The **gradient** of  $f$  is the partial derivatives of all dimensions:  
 $\nabla f(x) = [\partial f / \partial x_1(x), \dots, \partial f / \partial x_d(x)]$ .

# The Gradient

- Let  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be differentiable at  $x_0 \in \mathbb{R}^d$ .
- The **gradient** of  $f$  at the point  $x_0$ , denoted  $\nabla_x f(x_0)$ , is the direction in which  $f(x)$  **increases fastest**, if we start from  $x_0$ .
- The **gradient** of  $f$  is the partial derivatives of all dimensions:  
 $\nabla f(x) = [\partial f / \partial x_1(x), \dots, \partial f / \partial x_d(x)]$ .

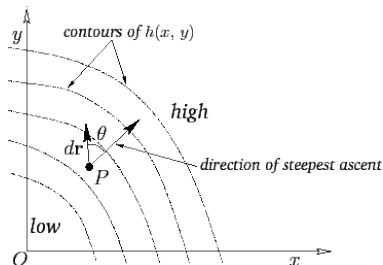


Figure A.111 from Newtonian Dynamics, by Richard Fitzpatrick.

# Gradient Descent

- To reach a local minimum as fast as possible, we want to go in the opposite direction from the gradient.

# Gradient Descent

- To reach a local minimum as fast as possible, we want to go in the opposite direction from the gradient.

## Gradient Descent

- Initialize  $x \leftarrow 0$ .
- Repeat:
  - $x \leftarrow x - \eta \nabla f(x)$
- until the stopping criterion is satisfied.

# Gradient Descent

- To reach a local minimum as fast as possible, we want to go in the opposite direction from the gradient.

## Gradient Descent

- Initialize  $x \leftarrow 0$ .
  - Repeat:
    - $x \leftarrow x - \eta \nabla f(x)$
  - until the stopping criterion is satisfied.
- 
- The “step size”  $\eta$  is not the amount by which we update  $x$ !

# Gradient Descent

- To reach a local minimum as fast as possible, we want to go in the opposite direction from the gradient.

## Gradient Descent

- Initialize  $x \leftarrow 0$ .
  - Repeat:
    - $x \leftarrow x - \eta \nabla f(x)$
  - until the stopping criterion is satisfied.
- 
- The “step size”  $\eta$  is not the amount by which we update  $x$ !
  - “Step size” is also referred to as “learning rate” in neural networks literature.

# Gradient Descent Path

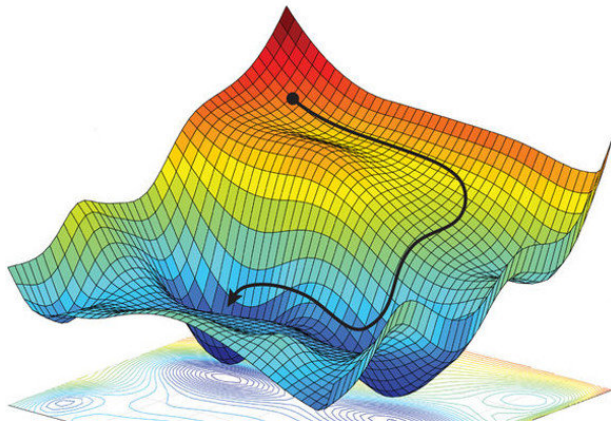


Image credit: Amini et al. Spatial Uncertainty Sampling for End-to-End Control. 2018.

## Gradient Descent: Step Size

A fixed step size will work, eventually, as long as it's small enough



## Gradient Descent: Step Size

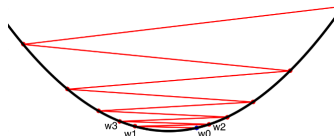
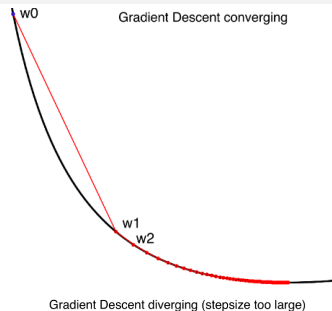
A fixed step size will work, eventually, as long as it's small enough

- If  $\eta$  is too large, the optimization process might diverge

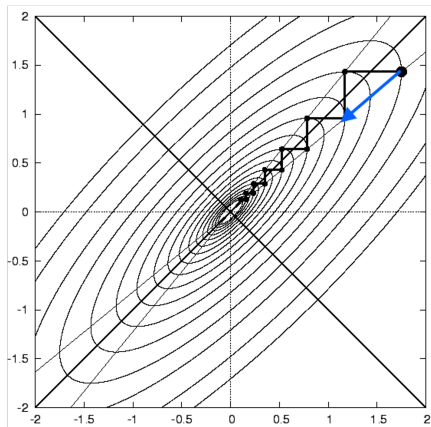
# Gradient Descent: Step Size

A fixed step size will work, eventually, as long as it's small enough

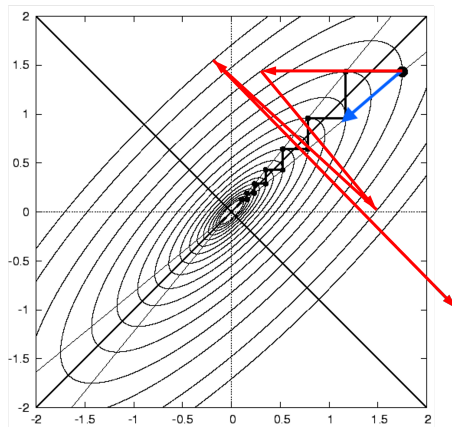
- If  $\eta$  is too large, the optimization process might diverge
- In practice, it often makes sense to try several fixed step sizes
- Intuition on when to take big steps and when to take small steps?



## 2D Divergence example



Small Step Size



Large Step Size

# Notes on Convergence

- Gradient descent with an appropriate step size converges to stationary point (derivative = 0) for differentiable functions.

# Notes on Convergence

- Gradient descent with an appropriate step size converges to stationary point (derivative = 0) for differentiable functions.
- Stationary points can be (local) minima, (local) maxima, saddle points, etc.

# Notes on Convergence

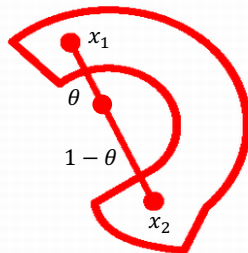
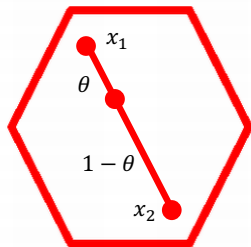
- Gradient descent with an appropriate step size converges to stationary point (derivative = 0) for differentiable functions.
- Stationary points can be (local) minima, (local) maxima, saddle points, etc.
- Gradient descent can converge to global minimum for **convex functions**.

# Convex Sets

## Definition

A set  $C$  is **convex** if for any  $x_1, x_2 \in C$  and any  $\theta$  with  $0 \leq \theta \leq 1$  we have

$$\theta x_1 + (1 - \theta)x_2 \in C.$$

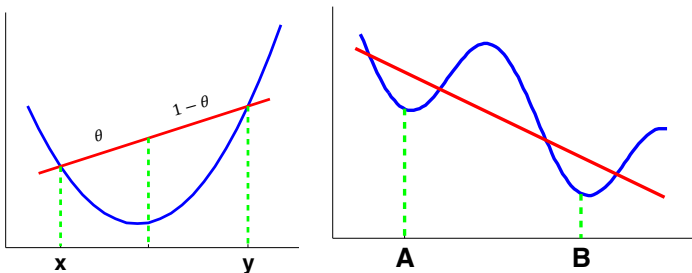


# Convex Functions

## Definition

A function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is **convex** if **dom**  $f$  is a convex set and if for all  $x, y \in \mathbf{dom} f$ , and  $0 \leq \theta \leq 1$ , we have

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y).$$



KPM Fig. 7.5



# Convergence Theorem for Fixed Step Size

## Theorem

Suppose  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  is convex and differentiable, and  $\nabla f$  is **Lipschitz continuous** with constant  $L > 0$  (***L-smooth***), i.e.

$$\|\nabla f(x) - \nabla f(x')\| \leq L\|x - x'\|$$

for any  $x, x' \in \mathbb{R}^d$ . Then gradient descent with fixed step size  $\eta \leq 1/L$  **converges**. In particular,

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|^2}{2\eta k}.$$

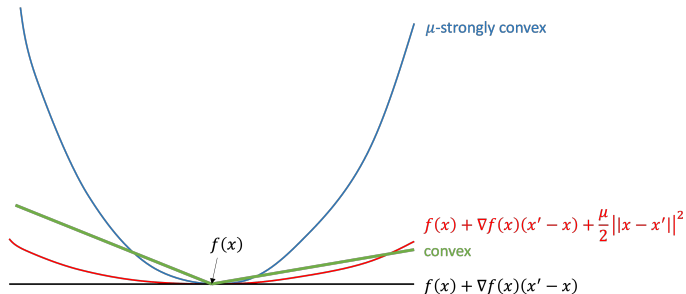
This says that gradient descent is guaranteed to converge and that it converges with rate  $O(1/k)$ .

# Strongly Convex Functions

## Definition

A function  $f$  is  $\mu$ -strongly convex if

$$f(x') \geq f(x) + \nabla f(x) \cdot (x' - x) + \frac{\mu}{2} \|x - x'\|^2$$



# Convergence Theorem for Strongly Convex Functions

## Theorem

*If  $f$  is  $L$ -smooth and  $\mu$ -strongly convex, and step size  $0 < \eta \leq \frac{1}{L}$ , then gradient descent converges with the following inequality:*

$$\|x^{(k)} - x^*\|^2 \leq (1 - \eta\mu)^k \|x^{(0)} - x^*\|^2$$

This means we can get linear convergence, but it depends on  $\mu$ . If the estimate of  $\mu$  is bad then the rate is not great.

# Gradient Descent: When to Stop?

- Wait until  $\|\nabla f(x)\|_2 \leq \varepsilon$ , for some  $\varepsilon$  of your choosing.
  - (Recall  $\nabla f(x) = 0$  at a local minimum.)

# Gradient Descent: When to Stop?

- Wait until  $\|\nabla f(x)\|_2 \leq \varepsilon$ , for some  $\varepsilon$  of your choosing.
  - (Recall  $\nabla f(x) = 0$  at a local minimum.)
- **Early stopping:**
  - evaluate loss on validation data (unseen held out data) after each iteration;
  - stop when the loss does not improve (or gets worse).

## Gradient Descent for Empirical Risk - Scaling Issues

## Quick recap: Gradient Descent for ERM

- We have a hypothesis space of functions  $\mathcal{F} = \{f_w : \mathcal{X} \rightarrow \mathcal{Y} \mid w \in \mathbb{R}^d\}$ 
  - Parameterized by  $w \in \mathbb{R}^d$ .

## Quick recap: Gradient Descent for ERM

- We have a hypothesis space of functions  $\mathcal{F} = \{f_w : \mathcal{X} \rightarrow \mathcal{Y} \mid w \in \mathbb{R}^d\}$ 
  - Parameterized by  $w \in \mathbb{R}^d$ .
- Finding an empirical risk minimizer entails finding a  $w$  that minimizes

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \ell(f_w(x_i), y_i)$$



## Quick recap: Gradient Descent for ERM

- We have a hypothesis space of functions  $\mathcal{F} = \{f_w : \mathcal{X} \rightarrow \mathcal{Y} \mid w \in \mathbb{R}^d\}$ 
  - Parameterized by  $w \in \mathbb{R}^d$ .
- Finding an empirical risk minimizer entails finding a  $w$  that minimizes

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \ell(f_w(x_i), y_i)$$

- Suppose  $\ell(f_w(x_i), y_i)$  is differentiable as a function of  $w$ .
- Then we can do gradient descent on  $\hat{R}_n(w)$

# Gradient Descent: Scalability

- At every iteration, we compute the gradient at the current  $w$ :

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

- How does this scale with  $n$ ?

# Gradient Descent: Scalability

- At every iteration, we compute the gradient at the current  $w$ :

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

- How does this scale with  $n$ ?
- We have to iterate over all  $n$  training points to take a single step. [ $O(n)$ ]

# Gradient Descent: Scalability

- At every iteration, we compute the gradient at the current  $w$ :

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

- How does this scale with  $n$ ?
- We have to iterate over all  $n$  training points to take a single step. [ $O(n)$ ]
- Can we make progress without looking at all the data before updating  $w$ ?

# Stochastic Gradient Descent

# “Noisy” Gradient Descent

- Instead of using the gradient, we use a noisy estimate of the gradient.
- Turns out this can work just fine!

# “Noisy” Gradient Descent

- Instead of using the gradient, we use a noisy estimate of the gradient.
- Turns out this can work just fine!
- **Intuition:**
  - Gradient descent is an iterative procedure anyway.
  - At every step, we have a chance to recover from previous missteps.

# Minibatch Gradient

- The **full gradient** is

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

- It's an average over the **full batch** of data  $\mathcal{D}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ .



# Minibatch Gradient

- The **full gradient** is

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

- It's an average over the **full batch** of data  $\mathcal{D}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ .
- Let's take a random subsample of size  $N$  (called a **minibatch**):

$$(x_{m_1}, y_{m_1}), \dots, (x_{m_N}, y_{m_N})$$

# Minibatch Gradient

- The **full gradient** is

$$\nabla \hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(f_w(x_i), y_i)$$

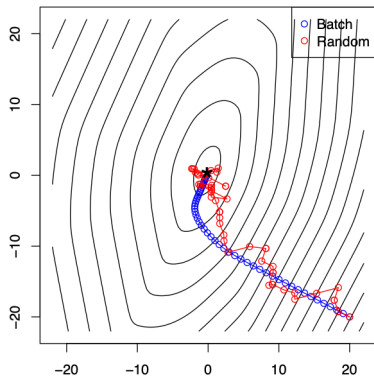
- It's an average over the **full batch** of data  $\mathcal{D}_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ .
- Let's take a random subsample of size  $N$  (called a **minibatch**):

$$(x_{m_1}, y_{m_1}), \dots, (x_{m_N}, y_{m_N})$$

- The **minibatch gradient** is

$$\nabla \hat{R}_N(w) = \frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_{m_i}), y_{m_i})$$

# Batch vs Stochastic Methods



Rule of thumb for stochastic methods:

- Stochastic methods work well far from the optimum
- But struggle close the the optimum

(Slide adapted from Ryan Tibshirani)

# Minibatch Gradient Properties

- The minibatch gradient is an **unbiased estimator** for the [full] batch gradient. What does that mean?

# Minibatch Gradient Properties

- The minibatch gradient is an **unbiased estimator** for the [full] batch gradient. What does that mean?

$$\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$

# Minibatch Gradient Properties

- The minibatch gradient is an **unbiased estimator** for the [full] batch gradient. What does that mean?

$$\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$

- The bigger the minibatch, the better the estimate.

$$\text{Var} \left[ \nabla \hat{R}_N(w) \right] = \text{Var} \left[ \frac{1}{N} \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N^2} \text{Var} \left[ \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N} \text{Var} \left[ \nabla \hat{R}_i(w) \right]$$

# Minibatch Gradient Properties

- The minibatch gradient is an **unbiased estimator** for the [full] batch gradient. What does that mean?

$$\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$

- The bigger the minibatch, the better the estimate.

$$\text{Var} \left[ \nabla \hat{R}_N(w) \right] = \text{Var} \left[ \frac{1}{N} \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N^2} \text{Var} \left[ \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N} \text{Var} \left[ \nabla \hat{R}_i(w) \right]$$

- Tradeoffs of minibatch size:
  - Bigger  $N \implies$  Better estimate of gradient, but slower (more data to process)

# Minibatch Gradient Properties

- The minibatch gradient is an **unbiased estimator** for the [full] batch gradient. What does that mean?

$$\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$

- The bigger the minibatch, the better the estimate.

$$\text{Var} \left[ \nabla \hat{R}_N(w) \right] = \text{Var} \left[ \frac{1}{N} \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N^2} \text{Var} \left[ \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N} \text{Var} \left[ \nabla \hat{R}_i(w) \right]$$

- Tradeoffs of minibatch size:
  - Bigger  $N \implies$  Better estimate of gradient, but slower (more data to process)
  - Smaller  $N \implies$  Worse estimate of gradient, but can be quite fast



# Minibatch Gradient Properties

- The minibatch gradient is an **unbiased estimator** for the [full] batch gradient. What does that mean?

$$\mathbb{E} \left[ \nabla \hat{R}_N(w) \right] = \nabla \hat{R}_n(w)$$

- The bigger the minibatch, the better the estimate.

$$\text{Var} \left[ \nabla \hat{R}_N(w) \right] = \text{Var} \left[ \frac{1}{N} \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N^2} \text{Var} \left[ \sum_i \nabla \hat{R}_i(w) \right] = \frac{1}{N} \text{Var} \left[ \nabla \hat{R}_i(w) \right]$$

- Tradeoffs of minibatch size:
  - Bigger  $N \implies$  Better estimate of gradient, but slower (more data to process)
  - Smaller  $N \implies$  Worse estimate of gradient, but can be quite fast
- Because of vectorization, the computation cost of minibatches is sublinear

# Convergence of SGD

- For convergence guarantee, use **diminishing step sizes**, e.g.  $\eta_k = 1/k$
- Theoretically, GD is much faster than SGD in terms of convergence rate and number of steps:
  - much faster to add a digit of accuracy (more details later)

# Convergence of SGD

- For convergence guarantee, use **diminishing step sizes**, e.g.  $\eta_k = 1/k$
- Theoretically, GD is much faster than SGD in terms of convergence rate and number of steps:
  - much faster to add a digit of accuracy (more details later)
  - costlier to compute a single step

# Convergence of SGD

- For convergence guarantee, use **diminishing step sizes**, e.g.  $\eta_k = 1/k$
- Theoretically, GD is much faster than SGD in terms of convergence rate and number of steps:
  - much faster to add a digit of accuracy (more details later)
  - costlier to compute a single step
  - but most of that advantage comes into play once we're already pretty close to the minimum

# Convergence of SGD

- For convergence guarantee, use **diminishing step sizes**, e.g.  $\eta_k = 1/k$
- Theoretically, GD is much faster than SGD in terms of convergence rate and number of steps:
  - much faster to add a digit of accuracy (more details later)
  - costlier to compute a single step
  - but most of that advantage comes into play once we're already pretty close to the minimum
  - in many ML problems we don't care about optimizing to high accuracy (why?)

# Step Sizes in Minibatch Gradient Descent

## Minibatch Gradient Descent (minibatch size $N$ )

- initialize  $w = 0$
- repeat
  - randomly choose  $N$  points  $\{(x_i, y_i)\}_{i=1}^N \subset \mathcal{D}_n$
  - $w \leftarrow w - \eta \left[ \frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_i), y_i) \right]$
- For SGD, fixed step size can work well in practice.

# Step Sizes in Minibatch Gradient Descent

## Minibatch Gradient Descent (minibatch size $N$ )

- initialize  $w = 0$
- repeat
  - randomly choose  $N$  points  $\{(x_i, y_i)\}_{i=1}^N \subset \mathcal{D}_n$
  - $w \leftarrow w - \eta \left[ \frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_i), y_i) \right]$
- For SGD, fixed step size can work well in practice.
- Typical approach: Fixed step size reduced by constant factor whenever validation performance stops improving (staircase decay).

# Step Sizes in Minibatch Gradient Descent

## Minibatch Gradient Descent (minibatch size $N$ )

- initialize  $w = 0$
- repeat
  - randomly choose  $N$  points  $\{(x_i, y_i)\}_{i=1}^N \subset \mathcal{D}_n$
  - $w \leftarrow w - \eta \left[ \frac{1}{N} \sum_{i=1}^N \nabla_w \ell(f_w(x_i), y_i) \right]$
- For SGD, fixed step size can work well in practice.
- Typical approach: Fixed step size reduced by constant factor whenever validation performance stops improving (staircase decay).
- Other schedules: inverse time decay ( $1/t$ ) etc.



## Convergence of SGD Theorem (Optional)

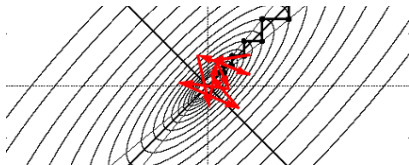
More on why we need a diminishing step size.

### Theorem

If  $f$  is  $L$ -smooth and convex, and SGD has bounded variance  $\text{Var}(\nabla f(x^{(k)})) \leq \sigma^2$  for all  $k$ , then SGD with step size  $\eta \leq \frac{1}{L}$  satisfies:

$$\min_k \mathbb{E}[\|f(x^{(k)})\|^2] \leq \frac{f(x^{(0)}) - f(x^*)}{\sum_k \eta_k} + \frac{L\sigma^2}{2} \frac{\sum_k \eta_k^2}{\sum_k \eta_k}$$

The extra term of variance will dominate if the step size does not decrease. <sup>1</sup>



<sup>1</sup><https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L11.pdf>

# Convergence of SGD Theorem (Optional)

## Theorem

If  $f$  is  $L$ -smooth and convex, and SGD has bounded variance  $\text{Var}(\nabla f(x^{(k)})) \leq \sigma^2$  for all  $k$ , then SGD with step size  $\eta \leq \frac{1}{L}$  satisfies:

$$\min_k \mathbb{E}[\|f(x^{(k)})\|^2] \leq \frac{f(x^{(0)}) - f(x^*)}{\sum_k \eta_k} + \frac{L\sigma^2}{2} \frac{\sum_k \eta_k^2}{\sum_k \eta_k}$$

- If  $\eta_k = \eta$ , then  $\sum_k \eta_k = k\eta$ ,  $\sum_k \eta_k^2 = k\eta^2$ , error =  $O(1/k) + O(\eta)$ .

# Convergence of SGD Theorem (Optional)

## Theorem

If  $f$  is  $L$ -smooth and convex, and SGD has bounded variance  $\text{Var}(\nabla f(x^{(k)})) \leq \sigma^2$  for all  $k$ , then SGD with step size  $\eta \leq \frac{1}{L}$  satisfies:

$$\min_k \mathbb{E}[\|f(x^{(k)})\|^2] \leq \frac{f(x^{(0)}) - f(x^*)}{\sum_k \eta_k} + \frac{L\sigma^2}{2} \frac{\sum_k \eta_k^2}{\sum_k \eta_k}$$

- If  $\eta_k = \eta$ , then  $\sum_k \eta_k = k\eta$ ,  $\sum_k \eta_k^2 = k\eta^2$ , error =  $O(1/k) + O(\eta)$ .
- If  $\eta_k = \eta/k$ , then  $\sum_k \eta_k = O(\log(k))$ ,  $\sum_k \eta_k^2 = O(1)$ , error =  $O(1/\log(k))$ .

# Convergence of SGD Theorem (Optional)

## Theorem

If  $f$  is  $L$ -smooth and convex, and SGD has bounded variance  $\text{Var}(\nabla f(x^{(k)})) \leq \sigma^2$  for all  $k$ , then SGD with step size  $\eta \leq \frac{1}{L}$  satisfies:

$$\min_k \mathbb{E}[\|f(x^{(k)})\|^2] \leq \frac{f(x^{(0)}) - f(x^*)}{\sum_k \eta_k} + \frac{L\sigma^2}{2} \frac{\sum_k \eta_k^2}{\sum_k \eta_k}$$

- If  $\eta_k = \eta$ , then  $\sum_k \eta_k = k\eta$ ,  $\sum_k \eta_k^2 = k\eta^2$ , error =  $O(1/k) + O(\eta)$ .
- If  $\eta_k = \eta/k$ , then  $\sum_k \eta_k = O(\log(k))$ ,  $\sum_k \eta_k^2 = O(1)$ , error =  $O(1/\log(k))$ .
- If  $\eta_k = \eta/\sqrt{k}$ , then  $\sum_k \eta_k = O(\sqrt{k})$ ,  $\sum_k \eta_k^2 = O(\log(k))$ , error =  $O(\log(k)/\sqrt{k}) = \tilde{O}(1/\sqrt{k})$ .

# Summary

- **Gradient descent** or “full-batch” gradient descent
  - Use full data set of size  $n$  to determine step direction

# Summary

- **Gradient descent** or “full-batch” gradient descent
  - Use full data set of size  $n$  to determine step direction
- **Minibatch gradient descent**
  - Use a **random** subset of size  $N$  to determine step direction

# Summary

- **Gradient descent** or “full-batch” gradient descent
  - Use full data set of size  $n$  to determine step direction
- **Minibatch gradient descent**
  - Use a **random** subset of size  $N$  to determine step direction
- **Stochastic gradient descent**
  - Minibatch with  $N = 1$ .
  - Use a single randomly chosen point to determine step direction.

# Summary

- **Gradient descent** or “full-batch” gradient descent
  - Use full data set of size  $n$  to determine step direction
- **Minibatch gradient descent**
  - Use a **random** subset of size  $N$  to determine step direction
- **Stochastic gradient descent**
  - Minibatch with  $N = 1$ .
  - Use a single randomly chosen point to determine step direction.

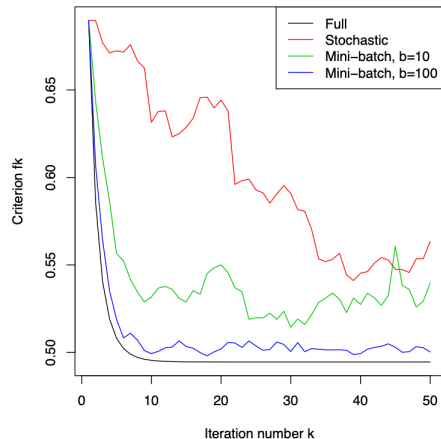
These days terminology isn't used so consistently, so when referring to SGD, always clarify the [mini]batch size.

SGD is much more efficient in time and memory cost and has been quite successful in large-scale ML.



## Example: Logistic regression with $\ell_2$ regularization

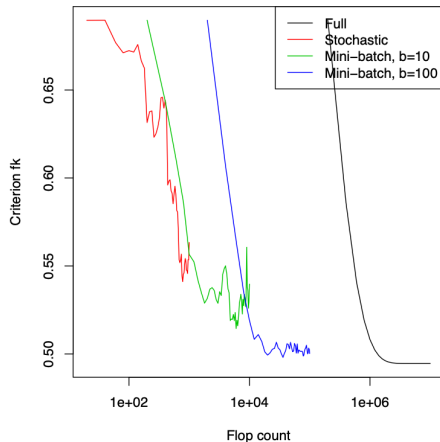
Batch methods converge faster :



(Example from Ryan Tibshirani)

## Example: Logistic regression with $\ell_2$ regularization

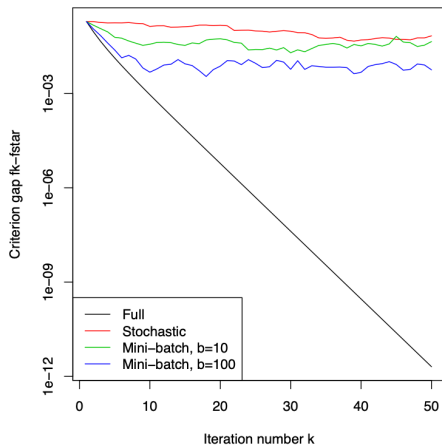
Stochastic methods are computationally more efficient:



(Example from Ryan Tibshirani)

## Example: Logistic regression with $\ell_2$ regularization

Batch methods are much faster close to the optimum:



(Example from Ryan Tibshirani)

## Loss Functions: Regression

# Regression Problems

- Examples:
  - Predicting the stock price given history prices

# Regression Problems

- Examples:
  - Predicting the stock price given history prices
  - Predicting medical cost of given age, sex, region, BMI etc.

# Regression Problems

- Examples:
  - Predicting the stock price given history prices
  - Predicting medical cost of given age, sex, region, BMI etc.
  - Predicting the age of a person based on their photos

# Regression Problems

- Examples:
  - Predicting the stock price given history prices
  - Predicting medical cost of given age, sex, region, BMI etc.
  - Predicting the age of a person based on their photos
- Notation:
  - $\hat{y}$  is the predicted value (the action)
  - $y$  is the actual observed value (the outcome)



# Loss Functions for Regression

- A loss function in general:

$$(\hat{y}, y) \mapsto \ell(\hat{y}, y) \in \mathbb{R}$$

# Loss Functions for Regression

- A loss function in general:

$$(\hat{y}, y) \mapsto \ell(\hat{y}, y) \in \mathbb{R}$$

- Regression losses usually only depend on the **residual**  $r = y - \hat{y}$ .
  - what you have to add to your prediction to get the correct answer.

# Loss Functions for Regression

- A loss function in general:

$$(\hat{y}, y) \mapsto \ell(\hat{y}, y) \in \mathbb{R}$$

- Regression losses usually only depend on the **residual**  $r = y - \hat{y}$ .
  - what you have to add to your prediction to get the correct answer.
- A loss  $\ell(\hat{y}, y)$  is called **distance-based** if:
  - 1 It only depends on the residual:

$$\ell(\hat{y}, y) = \psi(y - \hat{y}) \quad \text{for some } \psi: \mathbb{R} \rightarrow \mathbb{R}$$

- 2 It is zero when the residual is 0:

$$\psi(0) = 0$$

# Distance-Based Losses are Translation Invariant

- Distance-based losses are translation-invariant. That is,

$$\ell(\hat{y} + b, y + b) = \ell(\hat{y}, y) \quad \forall b \in \mathbb{R}.$$

# Distance-Based Losses are Translation Invariant

- Distance-based losses are translation-invariant. That is,

$$\ell(\hat{y} + b, y + b) = \ell(\hat{y}, y) \quad \forall b \in \mathbb{R}.$$

- When might you not want to use a translation-invariant loss?

# Distance-Based Losses are Translation Invariant

- Distance-based losses are translation-invariant. That is,

$$\ell(\hat{y} + b, y + b) = \ell(\hat{y}, y) \quad \forall b \in \mathbb{R}.$$

- When might you not want to use a translation-invariant loss?
- Sometimes the relative error  $\frac{\hat{y} - y}{y}$  is a more natural loss (but not translation-invariant)

# Distance-Based Losses are Translation Invariant

- Distance-based losses are translation-invariant. That is,

$$\ell(\hat{y} + b, y + b) = \ell(\hat{y}, y) \quad \forall b \in \mathbb{R}.$$

- When might you not want to use a translation-invariant loss?
- Sometimes the relative error  $\frac{\hat{y} - y}{y}$  is a more natural loss (but not translation-invariant)
- Often you can transform response  $y$  so it's translation-invariant (e.g. log transform)

## Some Losses for Regression

- **Residual:**  $r = y - \hat{y}$
- **Square or  $\ell_2$  Loss:**  $\ell(r) = r^2$



## Some Losses for Regression

- **Residual:**  $r = y - \hat{y}$
- **Square** or  $\ell_2$  Loss:  $\ell(r) = r^2$
- **Absolute** or **Laplace** or  $\ell_1$  Loss:  $\ell(r) = |r|$

## Some Losses for Regression

- **Residual:**  $r = y - \hat{y}$
- **Square** or  $\ell_2$  Loss:  $\ell(r) = r^2$
- **Absolute** or **Laplace** or  $\ell_1$  Loss:  $\ell(r) = |r|$

$y$	$\hat{y}$	$ r  =  y - \hat{y} $	$r^2 = (y - \hat{y})^2$
1	0	1	1
5	0	5	25
10	0	10	100
50	0	50	2500

- An outlier is a data point that differs significantly from other observations.

## Some Losses for Regression

- **Residual:**  $r = y - \hat{y}$
- **Square** or  $\ell_2$  Loss:  $\ell(r) = r^2$
- **Absolute** or **Laplace** or  $\ell_1$  Loss:  $\ell(r) = |r|$

$y$	$\hat{y}$	$ r  =  y - \hat{y} $	$r^2 = (y - \hat{y})^2$
1	0	1	1
5	0	5	25
10	0	10	100
50	0	50	2500

- An outlier is a data point that differs significantly from other observations.
- Outliers typically have large residuals.

## Some Losses for Regression

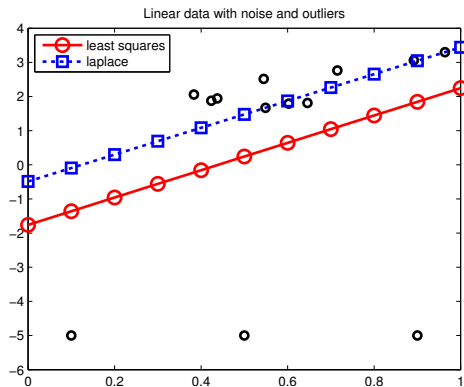
- **Residual:**  $r = y - \hat{y}$
- **Square** or  $\ell_2$  Loss:  $\ell(r) = r^2$
- **Absolute** or **Laplace** or  $\ell_1$  Loss:  $\ell(r) = |r|$

$y$	$\hat{y}$	$ r  =  y - \hat{y} $	$r^2 = (y - \hat{y})^2$
1	0	1	1
5	0	5	25
10	0	10	100
50	0	50	2500

- An outlier is a data point that differs significantly from other observations.
- Outliers typically have large residuals.
- Square loss much more affected by outliers than absolute loss.

# Loss Function Robustness

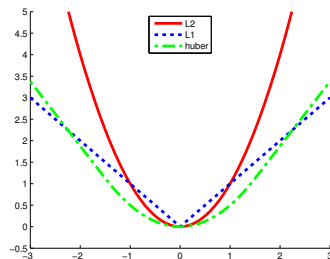
- **Robustness** refers to how affected a learning algorithm is by outliers.



KPM Figure 7.6

# Some Losses for Regression

- **Square** or  $\ell_2$  Loss:  $\ell(r) = r^2$  (*not robust*)
- **Absolute** or **Laplace** Loss:  $\ell(r) = |r|$  (*not differentiable*)
  - gives **median regression**
- **Huber** Loss: Quadratic for  $|r| \leq \delta$  and linear for  $|r| > \delta$  (*robust and differentiable*)
  - Equal values and slopes at  $r = \delta$



KPM Figure 7.6

# Classification Loss Functions

# The Classification Problem

- Examples:
  - Predict whether the image contains a cat
  - Predict whether the email is spam



# The Classification Problem

- Examples:
  - Predict whether the image contains a cat
  - Predict whether the email is spam
- Classification spaces:
  - Input space  $\mathcal{X}$
  - Outcome space  $\mathcal{Y} = \{-1, 1\}$

# The Classification Problem

- Examples:
  - Predict whether the image contains a cat
  - Predict whether the email is spam
- Classification spaces:
  - Input space  $\mathcal{X}$
  - Outcome space  $\mathcal{Y} = \{-1, 1\}$
- Inference:

$$f(x) > 0 \implies \text{Predict } 1$$

$$f(x) < 0 \implies \text{Predict } -1$$

# The Classification Problem

- Examples:
  - Predict whether the image contains a cat
  - Predict whether the email is spam
- Classification spaces:
  - Input space  $\mathcal{X}$
  - Outcome space  $\mathcal{Y} = \{-1, 1\}$
- Inference:

$$f(x) > 0 \implies \text{Predict } 1$$

$$f(x) < 0 \implies \text{Predict } -1$$

How can we optimize the model output?

# The Score Function

- Output space  $\mathcal{Y} = \{-1, 1\}$
- Real-valued prediction function  $f : \mathcal{X} \rightarrow \mathbb{R}$

## Definition

The value  $f(x)$  is called the **score** for the input  $x$ .

# The Score Function

- Output space  $\mathcal{Y} = \{-1, 1\}$
- Real-valued prediction function  $f : \mathcal{X} \rightarrow \mathbb{R}$

## Definition

The value  $f(x)$  is called the **score** for the input  $x$ .

- In this context,  $f$  may be called a **score function**.
- The magnitude of the score can be interpreted as our **confidence of our prediction**.

# The Margin

## Definition

The **margin** (or **functional margin**) for a predicted score  $\hat{y}$  and the true class  $y \in \{-1, 1\}$  is  $y\hat{y}$ .

# The Margin

## Definition

The **margin** (or **functional margin**) for a predicted score  $\hat{y}$  and the true class  $y \in \{-1, 1\}$  is  $y\hat{y}$ .

- The margin is often written as  $yf(x)$ , where  $f(x)$  is our score function.

# The Margin

## Definition

The **margin** (or **functional margin**) for a predicted score  $\hat{y}$  and the true class  $y \in \{-1, 1\}$  is  $y\hat{y}$ .

- The margin is often written as  $yf(x)$ , where  $f(x)$  is our score function.
- The margin is a measure of how **correct** we are:
  - If  $y$  and  $\hat{y}$  are the same sign, prediction is **correct** and margin is **positive**.
  - If  $y$  and  $\hat{y}$  have different sign, prediction is **incorrect** and margin is **negative**.



# The Margin

## Definition

The **margin** (or **functional margin**) for a predicted score  $\hat{y}$  and the true class  $y \in \{-1, 1\}$  is  $y\hat{y}$ .

- The margin is often written as  $yf(x)$ , where  $f(x)$  is our score function.
- The margin is a measure of how **correct** we are:
  - If  $y$  and  $\hat{y}$  are the same sign, prediction is **correct** and margin is **positive**.
  - If  $y$  and  $\hat{y}$  have different sign, prediction is **incorrect** and margin is **negative**.
- We want to **maximize the margin**.

# The Margin

## Definition

The **margin** (or **functional margin**) for a predicted score  $\hat{y}$  and the true class  $y \in \{-1, 1\}$  is  $y\hat{y}$ .

- The margin is often written as  $yf(x)$ , where  $f(x)$  is our score function.
- The margin is a measure of how **correct** we are:
  - If  $y$  and  $\hat{y}$  are the same sign, prediction is **correct** and margin is **positive**.
  - If  $y$  and  $\hat{y}$  have different sign, prediction is **incorrect** and margin is **negative**.
- We want to **maximize the margin**.
- Most classification losses depend only on the margin (they are **margin-based losses**).

## Classification Losses: 0–1 Loss

- If  $\tilde{f}$  is the inference function (1 if  $f(x) > 0$  and  $-1$  otherwise), then
- The **0-1 loss** for  $f : \mathcal{X} \rightarrow \{-1, 1\}$ :

$$\ell(f(x), y) = \mathbb{1}[\tilde{f}(x) \neq y]$$

## Classification Losses: 0–1 Loss

- If  $\tilde{f}$  is the inference function (1 if  $f(x) > 0$  and  $-1$  otherwise), then
- The **0-1 loss** for  $f : \mathcal{X} \rightarrow \{-1, 1\}$ :

$$\ell(f(x), y) = \mathbb{1}[\tilde{f}(x) \neq y]$$

- Empirical risk for 0–1 loss:

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[y_i f(x_i) \leq 0]$$

## Classification Losses: 0–1 Loss

- If  $\tilde{f}$  is the inference function (1 if  $f(x) > 0$  and  $-1$  otherwise), then
- The **0-1 loss** for  $f : \mathcal{X} \rightarrow \{-1, 1\}$ :

$$\ell(f(x), y) = \mathbb{1}[\tilde{f}(x) \neq y]$$

- Empirical risk for 0–1 loss:

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[y_i f(x_i) \leq 0]$$

Minimizing empirical 0–1 risk not computationally feasible.

## Classification Losses: 0–1 Loss

- If  $\tilde{f}$  is the inference function (1 if  $f(x) > 0$  and  $-1$  otherwise), then
- The **0-1 loss** for  $f : \mathcal{X} \rightarrow \{-1, 1\}$ :

$$\ell(f(x), y) = \mathbb{1}[\tilde{f}(x) \neq y]$$

- Empirical risk for 0–1 loss:

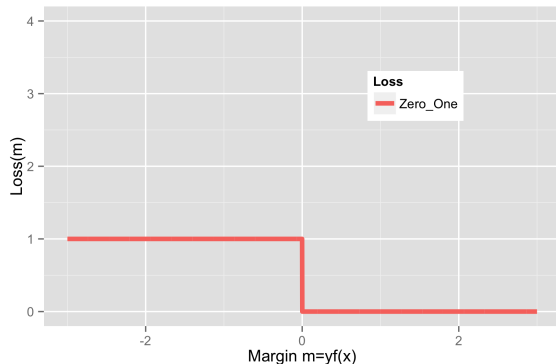
$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}[y_i f(x_i) \leq 0]$$

Minimizing empirical 0–1 risk not computationally feasible.

$\hat{R}_n(f)$  is non-convex, not differentiable, and even discontinuous.

# Classification Losses

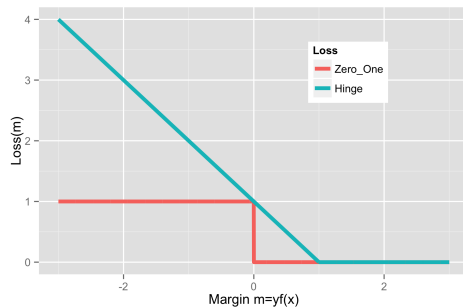
Zero-One loss:  $\ell_{0-1} = \mathbb{1}[m \leq 0]$



- x-axis is **margin**:  $m > 0 \iff$  correct classification

# Hinge Loss

SVM/Hinge loss:  $\ell_{\text{Hinge}} = \max(1 - m, 0)$



Hinge is a **convex, upper bound** on 0–1 loss. Not differentiable at  $m = 1$ .

We will cover SVM and Hinge loss in more details in future lectures.



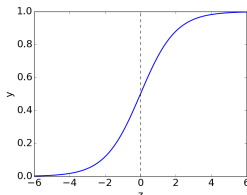
# Logistic Regression

- Also known as linear classification. Logistic regression is not actually “regression.”
- Two equivalent types of logistic regression losses, depending on the labels.

# Logistic Regression

- Also known as linear classification. Logistic regression is not actually “regression.”
- Two equivalent types of logistic regression losses, depending on the labels.
- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$ , where  $\sigma$  is the sigmoid function, and  $z = f(x) = w^\top x$ .

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



# Logistic Regression

- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$ , where  $\sigma$  is the sigmoid function.

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

# Logistic Regression

- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$ , where  $\sigma$  is the sigmoid function.

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- The loss is binary cross entropy:

$$\ell_{\text{Logistic}} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

# Logistic Regression

- If the label is 0 or 1:
- $\hat{y} = \sigma(z)$ , where  $\sigma$  is the sigmoid function.

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

- The loss is binary cross entropy:

$$\ell_{\text{Logistic}} = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

- Remember the negative sign!

# Logistic Regression

- If the label is -1 or 1:
- Note:  $1 - \sigma(z) = \sigma(-z)$

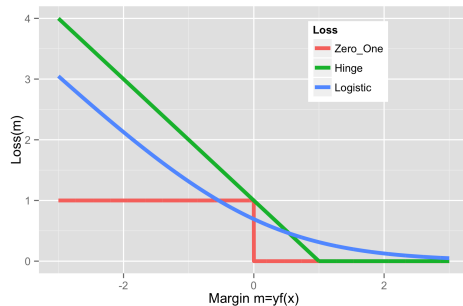
# Logistic Regression

- If the label is -1 or 1:
- Note:  $1 - \sigma(z) = \sigma(-z)$
- Now we can derive an equivalent loss form:

$$\begin{aligned}\ell_{\text{Logistic}} &= \begin{cases} -\log(\sigma(z)) & \text{if } y = 1 \\ -\log(\sigma(-z)) & \text{if } y = -1 \end{cases} \\ &= -\log(\sigma(yz)) \\ &= -\log\left(\frac{1}{1 + e^{-yz}}\right) \\ &= \log(1 + e^{-m}).\end{aligned}$$

# Logistic Loss

Logistic/Log loss:  $\ell_{\text{Logistic}} = \log(1 + e^{-m})$



Logistic loss is differentiable. Logistic loss always rewards a larger margin (the loss is never 0).



# What About Square Loss for Classification?

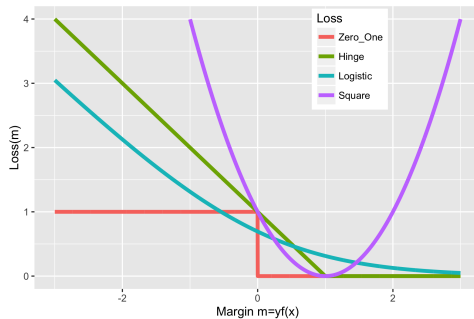
- Loss  $\ell(f(x), y) = (f(x) - y)^2$ .

# What About Square Loss for Classification?

- Loss  $\ell(f(x), y) = (f(x) - y)^2$ .
- Turns out, can write this in terms of margin  $m = f(x)y$ :
- Using fact that  $y^2 = 1$ , since  $y \in \{-1, 1\}$ .

$$\begin{aligned}\ell(f(x), y) &= (f(x) - y)^2 \\ &= f^2(x) - 2f(x)y + y^2 \\ &= f^2(x)y^2 - 2f(x)y + 1 \\ &= (1 - f(x)y)^2 \\ &= (1 - m)^2\end{aligned}$$

# What About Square Loss for Classification?



Heavily penalizes outliers (e.g. mislabeled examples).

# Summary

- Gradient descent: step size/learning rate, batch size, convergence

# Summary

- Gradient descent: step size/learning rate, batch size, convergence
- Loss functions for regression and classification problems.

# Summary

- Gradient descent: step size/learning rate, batch size, convergence
- Loss functions for regression and classification problems.
- Regression: Squared (L2) loss, Absolute (L1) loss, Huber loss.

# Summary

- Gradient descent: step size/learning rate, batch size, convergence
- Loss functions for regression and classification problems.
- Regression: Squared (L2) loss, Absolute (L1) loss, Huber loss.
- Classification: Hinge loss, Logistic loss.

# Summary

- Gradient descent: step size/learning rate, batch size, convergence
- Loss functions for regression and classification problems.
- Regression: Squared (L2) loss, Absolute (L1) loss, Huber loss.
- Classification: Hinge loss, Logistic loss.
- Residual, margin



# Summary

- Gradient descent: step size/learning rate, batch size, convergence
- Loss functions for regression and classification problems.
- Regression: Squared (L2) loss, Absolute (L1) loss, Huber loss.
- Classification: Hinge loss, Logistic loss.
- Residual, margin
- Logistic regression