Neural Computation

Week 3 - Gradient Descent

Yunwen Lei

School of Computer Science, University of Birmingham

Outline

Gradient Descent

Visualization of the Gradient

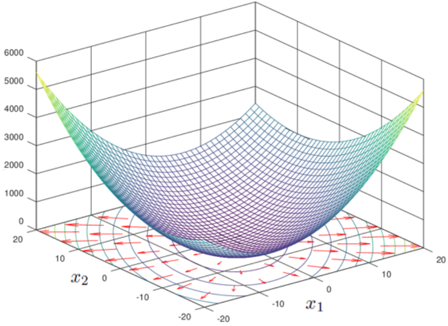
- The gradient vector at a point $\mathbf{x}$ points in the direction of greatest increase of the function f: each element of the gradient shows how fast $f(\mathbf{x})$ is changing w.r.t. each of the coordinate axes

Example: Consider
$f(x_1, x_2) = 6x_1^2 + 4x_2^2 - 4x_1x_2$

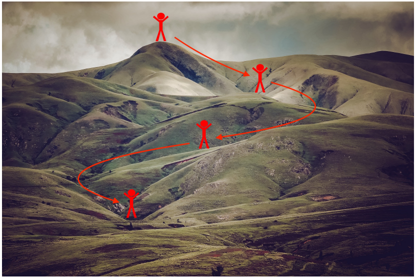
The gradient vector is

$$\nabla f(x_1, x_2) = \begin{pmatrix} 12x_1 - 4x_2 \\ 8x_2 - 4x_1 \end{pmatrix}$$

Right: the graph of the function and its gradient vector field



Optimization



Minimizing the cost is like finding the lowest point in a hilly landscape

Template Optimization Algorithm

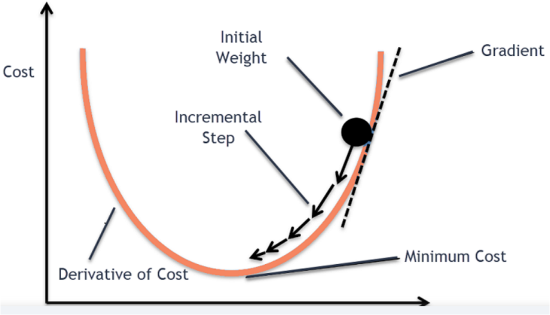
- Start with a point $\mathbf{w}$ (initial guess)
- Find a direction $\mathbf{d}$ to move on, and determine how far (η) to move along $\mathbf{d}$
- Gets updated $\mathbf{w} \leftarrow \mathbf{w} + \eta\mathbf{d}$

Gradient Descent

- Gradient descent is one of the simplest, but very general algorithm for minimizing an objective function $C(\mathbf{w})$ (first proposed by Cauchy in 1847):
- It is an iterative algorithm, starting from $\mathbf{w}^{(0)}$ and producing a new $\mathbf{w}^{(t+1)}$ at each iteration as

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla C(\mathbf{w}^{(t)}), \quad t = 0, 1, \ldots .$$

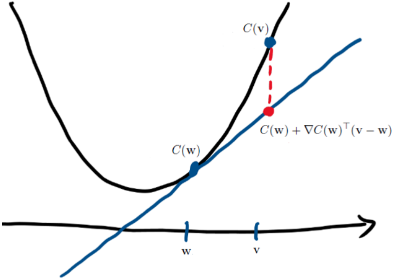
- $\eta_t > 0$ is the learning rate or step size



Why Gradient Descent Works?

Recall we can use gradient to get a tangent as approximation of C

$$C(\mathbf{v}) \approx \underbrace{C(\mathbf{w}) + \nabla C(\mathbf{w})^\top (\mathbf{v} - \mathbf{w})}_{\text{linear function of } \mathbf{v}} \qquad (1)$$



- By Eq. (1), if $\mathbf{d}$ is small then

$$C(\mathbf{w}^{(t)} + \mathbf{d}) \approx C(\mathbf{w}^{(t)}) + \nabla C(\mathbf{w}^{(t)})^\top \mathbf{d}$$

- Hence, we want to minimize the approximation while staying close to $\mathbf{w}^{(t)}$

$$\mathbf{d}^* = \arg\min_{\mathbf{w}} \underbrace{\frac{1}{2\eta_t}\|\mathbf{d}\|^2}_{\text{penalty}} + \underbrace{\left(C(\mathbf{w}^{(t)}) + \nabla C(\mathbf{w}^{(t)})^\top \mathbf{d}\right)}_{\text{linear approximation}}$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \mathbf{d}^*.$$

As an exercise, show that the above update is exactly GD!

Illustration of Gradient Descent

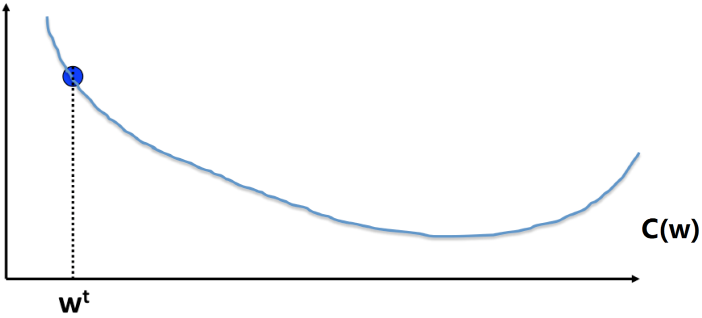
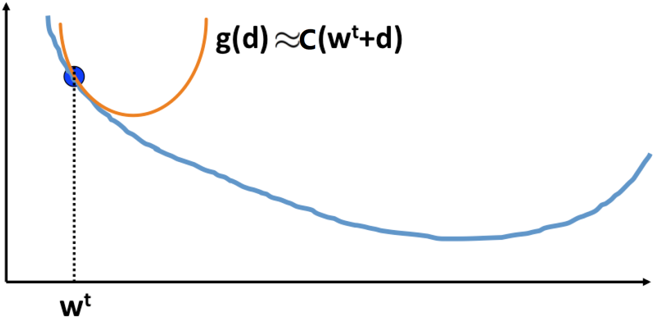


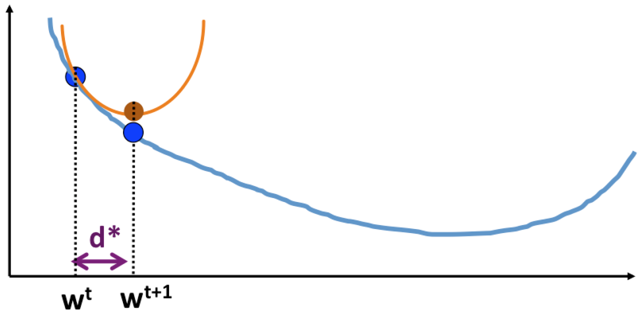
Illustration of Gradient Descent



Form a quadratic approximation

$$C(\mathbf{w}^{(t)} + \mathbf{d}) \approx g(\mathbf{d}) := \underbrace{C(\mathbf{w}^{(t)}) + \nabla C(\mathbf{w}^{(t)})^\top \mathbf{d}}_{\text{linear approximation}} + \underbrace{\frac{1}{2\eta_t}\|\mathbf{d}\|_2^2}_{\text{penalty}}.$$

Illustration of Gradient Descent



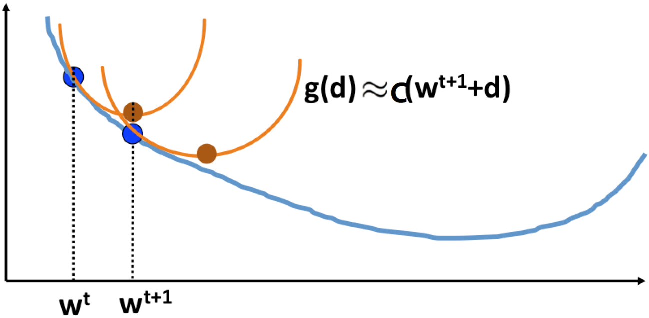
Minimize $g(d)$ gets

$$d^* = -\eta_t \nabla C(\mathbf{w}^{(t)})$$

and update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + d^* = \mathbf{w}^{(t)} - \eta_t \nabla C(\mathbf{w}^{(t)})$$

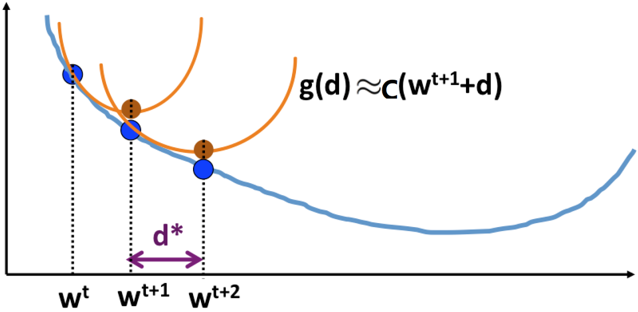
Illustration of Gradient Descent



Form another quadratic approximation

$$C(\mathbf{w}^{(t+1)} + \mathbf{d}) \approx g(\mathbf{d}) := \underbrace{C(\mathbf{w}^{(t+1)}) + \nabla C(\mathbf{w}^{(t+1)})^\top \mathbf{d}}_{\text{linear approximation}} + \underbrace{\frac{1}{2\eta_{t+1}} \|\mathbf{d}\|_2^2}_{\text{penalty}}.$$

Illustration of Gradient Descent



Minimize $g(d)$ gets

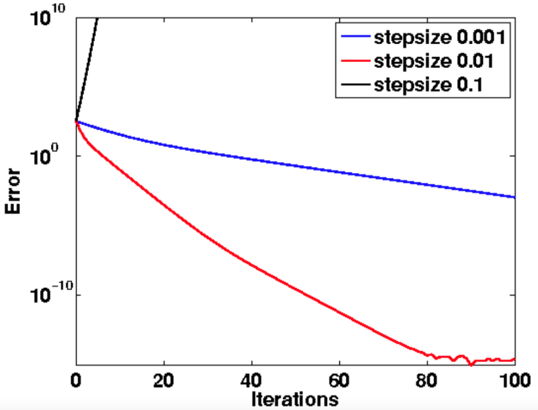
$$d^* = -\eta_{t+1} \nabla C(\mathbf{w}^{(t+1)})$$

and update

$$\mathbf{w}^{(t+2)} = \mathbf{w}^{(t+1)} + d^* = \mathbf{w}^{(t+1)} - \eta_{t+1} \nabla C(\mathbf{w}^{(t+1)})$$

Choosing a Step Size

- Choosing a good step-size is important
- It step size is too large, algorithm may never converge
- If step size is too small, convergence may be very slow
- May want a time-varying step size



Gradient Descent for Least Square Regression

- For least square regression, recall

$$C(\mathbf{w}) = \frac{1}{2n}\Big(\underbrace{\mathbf{w}^\top X^\top X\mathbf{w}}_{\text{quadratic}} - 2\underbrace{\mathbf{w}^\top X^\top \mathbf{y}}_{\text{linear}} + \underbrace{\mathbf{y}^\top \mathbf{y}}_{\text{constant}}\Big)$$

- The gradient is computed in the last week by

$$\nabla C(\mathbf{w}) = \frac{1}{n}\Big(X^\top X\mathbf{w} - X^\top \mathbf{y}\Big).$$

- GD updates $\mathbf{w}^{(t)}$ by

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta\nabla C(\mathbf{w}^{(t)}) = \mathbf{w}^{(t)} - \frac{\eta}{n}\Big(X^\top X\mathbf{w}^{(t)} - X^\top \mathbf{y}\Big).$$

Simple to Implement!

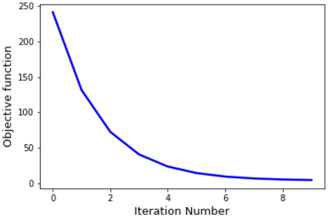
Back to Toy Example (Commute Time)

Recall $n = 5, d = 3$ and

$$\mathbf{y} = \begin{pmatrix} 25 \\ 33 \\ 15 \\ 45 \\ 22 \end{pmatrix}, \quad X = \begin{pmatrix} 1 & 2.7 & 1 \\ 1 & 4.1 & 1 \\ 1 & 1.0 & 0 \\ 1 & 5.2 & 1 \\ 1 & 2.8 & 0 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

If we run GD with $\mathbf{w}^{(1)} = (0, 0, 0)^\top$ and $\eta = 0.02$, then we get (python implementation)

```
the grad of 1-th iteration is [ -28.    -102.68   -20.6 ]
the weight of 1-th iteration is [0.56   2.0536 0.412 ]
the grad of 2-th iteration is [-20.703424  -75.2866144 -15.08816   ]
the weight of 2-th iteration is [0.97406848 3.55933229 0.7137632 ]
the grad of 3-th iteration is [-15.35018357 -55.1911618  -11.0449035 ]
the weight of 3-th iteration is [1.28107215 4.66315552 0.93466127]
the grad of 4-th iteration is [-11.42255963 -40.44941129  -8.07898669]
the weight of 4-th iteration is [1.50952334 5.47214375 1.096241   ]
the grad of 5-th iteration is [ -8.5407578  -29.6350914   -5.90339639]
the weight of 5-th iteration is [1.6803385  6.06484558 1.21430893]
the grad of 6-th iteration is [ -6.42616412 -21.70190135  -4.30758215]
the weight of 6-th iteration is [1.80886178 6.4988836  1.30046057]
the grad of 7-th iteration is [ -4.87438968 -15.88228366  -3.13708593]
the weight of 7-th iteration is [1.90634958 6.81652928 1.36320229]
the grad of 8-th iteration is [ -3.73549653 -11.61316462  -2.27859861]
the weight of 8-th iteration is [1.98105951 7.04879257 1.40877427]
the grad of 9-th iteration is [-2.89949141 -8.48147805 -1.64899757]
the weight of 9-th iteration is [2.03904933 7.21842213 1.44175422]
the grad of 10-th iteration is [-2.2856842  -6.18420209 -1.18730475]
the weight of 10-th iteration is [2.08476302 7.34210617 1.46550031]
```



Stochastic Gradient Descent

Stochastic Gradient Descent

- GD is easy to implement since gradient computation is required
- GD is computationally expensive as it require to go through all the examples

Sum Structure

$$C(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} C_i(\mathbf{w}), \quad C_i(\mathbf{w}) \text{ often corresponds to a loss with } i\text{-th example}$$

Can we boost the optimization by using the sum structure of C?

Yes and replace the computationally expensive term $\nabla C(\mathbf{w}^{(t)})$ by a stochastic gradient computed on a random example

- At the t-th iteration, we randomly choose an index i_t uniformly from $\{1, 2, \ldots, n\}$
- We compute a stochastic gradient $\nabla C_{i_t}(\mathbf{w}^{(t)})$
- We update the model as follows

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla C_{i_t}(\mathbf{w}^{(t)}).$$

Stochastic Gradient Descent

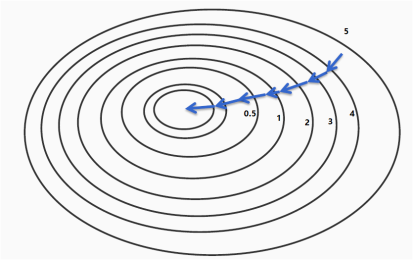
Stochastic Gradient Descent (Robbins & Monro 1951)

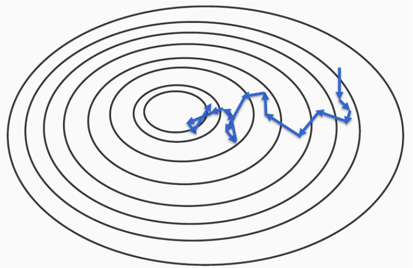
- Initialize the weights $\mathbf{w}^{(0)}$
- For $t = 0, 1, \ldots, T$
  - Draw i_t from $\{1, \ldots, n\}$ with equal probability
  - Compute stochastic gradient $\nabla C_{i_t}(\mathbf{w}^{(t)})$ and update

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \nabla C_{i_t}(\mathbf{w}^{(t)})$$

- ∇C_{i_t} is not a true gradient and therefore SGD is not as smooth as GD
- On-average of stochastic gradient is $\frac{1}{n}\sum_{i=1}^{n} \nabla C_i(\mathbf{w}^{(t)}) = \nabla C(\mathbf{w}^{(t)})$. In the long run, SGD would solve the optimization problem

Comparison Between GD and SGD





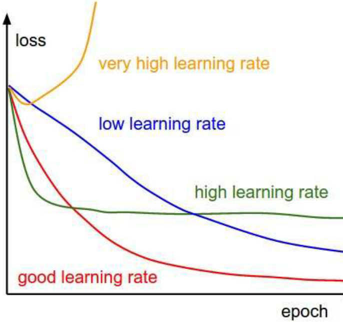
- GD requires more computation per iteration but make a good progress per iteration. It needs few iteration to get a good solution.

- SGD requires less computation per iteration but makes less update per iteration. Therefore, it needs more iteration to get a good solution.

- GD and SGD cannot always dominate the other
  - If we want a high accuracy and n is small, then GD is better
  - If we want a moderate accuracy and n is large, then SGD is better.

Effect of Learning Rates

- If we choose a low learning rate, then SGD would converge very slowly
- If we choose a large learning rate, then SGD would not go further as we run more and more iterations
- If we choose a huge learning rate, then SGD would become unstable
- A typical choice is $\eta_t = c/\sqrt{t}$, where c is a parameter needed to tune.



Minibatch SGD

Minibatch SGD

- Instead of using only one example to compute a stochastic gradient, we can use several examples to compute a stochastic gradient.
- we first randomly select a batch of indices: $B_t \subseteq \{1, 2, \ldots, n\}$ and update the model by (b is the batch size)

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta_t}{b} \sum_{i \in B_t} \nabla C_i(\mathbf{w}^{(t)}).$$

- For example, if $b = 2$ and $B_t = \{2, 6\}$ then

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta_t}{2} \Big(\nabla C_2(\mathbf{w}^{(t)}) + \nabla C_6(\mathbf{w}^{(t)}) \Big).$$

- If $b = 1$, it is clear that minibatch SGD is SGD.

Minibatch SGD

Minibatch SGD

Let $\{\eta_t\}$ be a sequence of step sizes

- Initialize the weights $\mathbf{w}^{(0)}$
- For $t = 0, 1, \ldots, T$
  - Randomly selecting a batch $B_t \subseteq \{1, 2, \ldots, n\}$ of size b
  - Compute stochastic gradient $\nabla C_{i_t}(\mathbf{w}^{(t)})$ and update

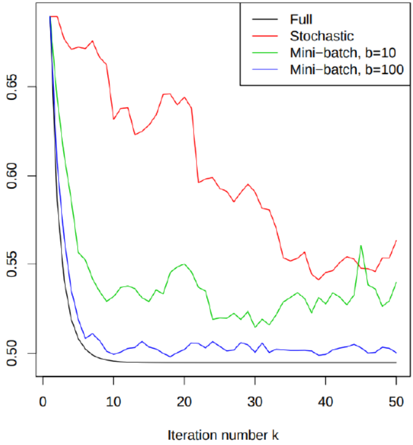
$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \frac{\eta_t}{b} \sum_{i \in B_t} \nabla C_i(\mathbf{w}^{(t)}).$$

There are two ways to sample the minibatch B_t

- sampling with replacement: we put it back to the bag after selecting the index (if $b = 2$, it is possible $B_t = \{2, 2\}$)
- sampling without replacement: we do not put it back to the bag after selecting the index (if $b = 2$, it is not possible $B_t = \{2, 2\}$)

Remarks on Minibatch SGD

- Minibatch SGD requires more computation than SGD per iteration to build stochastic gradient.
- Stochastic gradient is more accurate. Therefore it converges faster w.r.t. the iteration number
- We need to balance the accuracy and computation by choosing an appropriate b.



Typical choice: $b = 32, 64, 128$.

Linear Models for Classification

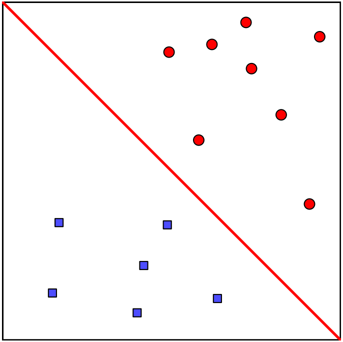
Linear Models for Classification

Suppose we have

$$S = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}, y_i \in \{-1, +1\}.$$

We want to build a linear model to separate positive examples from negative examples

- positive examples are on one-hand of the linear function
- negative examples are on the other hand.



- A linear model $\mathbf{x} \mapsto \mathbf{w}^\top \mathbf{x}$ outputs a real number, we predict as

$$\hat{y} = \mathrm{sgn}(\mathbf{w}^\top \mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}^\top \mathbf{x} > 0 \\ -1, & \text{otherwise.} \end{cases}$$

0-1 Loss

- The performance of a model $\mathbf{w}$ on $(\mathbf{x}, y)$ can be measured by the 0-1 loss

$$L(\hat{y}, y) = \mathbb{I}[\hat{y} \neq y] = \begin{cases} 1, & \text{if } \hat{y} \neq y \\ 0, & \text{otherwise.} \end{cases}$$

- Then the behavior of a model on S can be measured by

$$C(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[\mathrm{sgn}(\mathbf{w}^\top \mathbf{x}^{(i)}) \neq y^{(i)}].$$

- The minimization of $C(\mathbf{w})$ is challenging since it is not continuous!

We need to find a surrogate of $C(\mathbf{w})$ which is easy to minimize!

Margin-based Loss

$$\hat{y} = \begin{cases} 1, & \text{if } \mathbf{w}^\top \mathbf{x} > 0 \\ -1, & \text{otherwise.} \end{cases} \Longrightarrow \begin{cases} y = \hat{y} = 1, & \text{if } y = 1, y(\mathbf{w}^\top \mathbf{x}) > 0 \\ y = \hat{y} = -1, & \text{if } y = -1, y(\mathbf{w}^\top \mathbf{x}) \geq 0 \\ y = 1, \hat{y} = -1, & \text{if } y = 1, y(\mathbf{w}^\top \mathbf{x}) \leq 0 \\ y = -1, \hat{y} = 1, & \text{if } y = -1, y(\mathbf{w}^\top \mathbf{x}) < 0 \end{cases}$$

$\hat{y} \neq y$ amounts to saying $y\mathbf{w}^\top \mathbf{x} \leq 0$ (ignoring the case of being 0)

Margin

The margin of a model $\mathbf{w}$ on an example $(\mathbf{x}, y)$ is defined as $y\mathbf{w}^\top x$.

- A model with a positive margin means a correct prediction
- A model with a negative margin means a incorrect prediction

This further motivates a model with large margin: a large margin means the model is robust in making a correct prediction.

Margin-based Loss

Margin-based loss

We consider the loss function for the form

$$L(\hat{y}, y) = g(y\hat{y}),$$

where g is decreasing.

- We want to minimize the loss $L(\hat{y}, y) = g(\hat{y}y)$. Since g is decreasing, minimizing L means maximizing the margin.
- Maximizing the margin means getting a model with good performance.
- If g is differentiable, then we can get an easy optimization problem.

Surrogate Loss

- We mainly consider

$$g(t) = \frac{1}{2}\big(\max\{0, 1-t\}\big)^2 = \begin{cases} 0, & \text{if } t \geq 1 \\ \frac{1}{2}(1-t)^2, & \text{otherwise.} \end{cases}.$$

- The loss function becomes

$$L(\hat{y}, y) = \frac{1}{2}\big(\max\{0, 1-\hat{y}y\}\big)^2 = \frac{1}{2}\big(\max\{0, 1-y\mathbf{w}^\top\mathbf{x}\}\big)^2.$$

- The behavior on S is quantified by

$$C(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\big(\max\{0, 1-y^{(i)}\mathbf{w}^\top\mathbf{x}^{(i)}\}\big)^2.$$

Minimization: First Attempt

- By first-order necessary condition, the optimal $\mathbf{w}^*$ satisfies $\nabla C(\mathbf{w}^*) = 0$
- The gradient can be computed by

$$\begin{cases} 0, \\ \frac{1}{2}(1 - y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)})^2, \end{cases} \implies \nabla C_i(\mathbf{w}) = \begin{cases} 0, & \text{if } y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)} \geq 1 \\ y^{(i)}(y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)} - 1)\mathbf{x}^{(i)}, & \text{otherwise.} \end{cases}$$

- We further get

$$\nabla C_i(\mathbf{w}) = \begin{cases} 0, & \text{if } y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)} \geq 1 \\ (\mathbf{w}^\top \mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}, & \text{otherwise.} \end{cases} \tag{2}$$

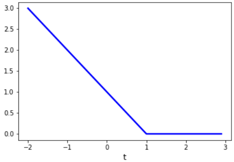
- The first-order optimality condition implies

$$\frac{1}{n}\sum_{i=1}^{n} \nabla C_i(\mathbf{w}^*) = 0.$$

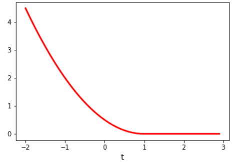
Difficult to solve since this is a highly nonlinear problem!

Other Choices of Surrogate Loss

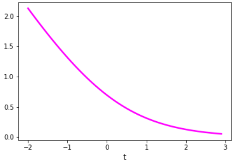
There are some other choices, including $g(t) = \max\{0, 1 - t\}$ and $g(t) = \log(1 + \exp(-t))$



$g(t) = \max\{0, 1 - t\}$



$g(t) = \frac{1}{2}\left(\max\{0, 1 - t\}\right)^2$



$g(t) = \log(1 + \exp(-t))$

Other Choices of Surrogate Loss

- If we consider $g(t) = \max\{0, 1 - t\}$, then this leads to the following optimization problem

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \max\{0, 1 - y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)}\},$$

  which corresponds to the support vector machine.

- If one considers $g(t) = \log(1 + \exp(-t))$, then this leads to

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \log\left(1 + \exp(-y^{(i)} \mathbf{w}^\top \mathbf{x}^{(i)})\right),$$

  which corresponds to another popular method called logistic regression.

We recover different popular methods by considering different margin-based loss functions!

SGD for Linear Classification

- Suppose we consider the following loss function for linear classification

$$C_i(\mathbf{w}) = \begin{cases} 0, & \text{if } y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)} \geq 1 \\ \frac{1}{2}(1 - y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)})^2, & \text{otherwise.} \end{cases}$$

- If we choose the index i_t, then the update should be

$$\mathbf{w}^{(t+1)} = \begin{cases} \mathbf{w}^{(t)}, & \text{if } y^{(i_t)}(\mathbf{w}^{(t)})^\top \mathbf{x}^{(i_t)} \geq 1 \\ \mathbf{w}^{(t)} - \eta_t y^{(i_t)}(y^{(i_t)}\mathbf{w}^\top \mathbf{x}^{(i_t)} - 1)\mathbf{x}^{(i_t)}, & \text{otherwise.} \end{cases}$$

Algorithm 1: SGD for Linear Classification

for $t = 0, 1, \ldots$ **to** T **do**
  $i_t \leftarrow$ random index from $\{1, 2, \ldots, n\}$
  if $y^{(i_t)}(\mathbf{w}^{(t)})^\top x^{(i_t)} \geq 1$ **then**

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)}$$

  else

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta_t(1 - y^{(i_t)}\mathbf{w}^\top \mathbf{x}^{(i_t)})y^{(i_t)}\mathbf{x}^{(i_t)}$$

$$\mathbf{w}^{(t+1)} = \begin{cases} \mathbf{w}^{(t)}, & \text{if } y^{(i_t)}(\mathbf{w}^{(t)})^\top \mathbf{x}^{(i_t)} \geq 1 \\ \mathbf{w}^{(t)} + \eta_t(1 - y^{(i_t)}\mathbf{w}^\top \mathbf{x}^{(i_t)})y^{(i_t)}\mathbf{x}^{(i_t)}, & \text{otherwise.} \end{cases}$$

- No update if $y^{(i_t)}(\mathbf{w}^{(t)})^\top x^{(i_t)} \geq 1$ (doing well with this example and no need to update).
- An update if the margin is smaller than 1 (we are not doing quite well on the example). Since $1 > y^{(i_t)}\mathbf{w}^\top \mathbf{x}^{(i_t)}$ in this case, we know

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \alpha y^{(i_t)}\mathbf{x}^{(i_t)}$$

for $\alpha = \eta_t(1 - y^{(i_t)}\mathbf{w}^\top \mathbf{x}^{(i_t)}) > 0$. That is

$$\begin{aligned} y^{(i_t)}(\mathbf{w}^{(t+1)})^\top \mathbf{x}^{(i_t)} &= y^{(i_t)}(\mathbf{w}^{(t)})^\top \mathbf{x}^{(i_t)} + \alpha y^{(i_t)} y^{(i_t)} \|\mathbf{x}^{(i_t)}\|^2 \\ &> y^{(i_t)}(\mathbf{w}^{(t)})^\top \mathbf{x}^{(i_t)}. \end{aligned}$$

The margin of $\mathbf{w}^{(t+1)}$ at $(\mathbf{x}^{(i_t)}, y^{i_t})$ is larger than the margin of $\mathbf{w}^{(t)}$!

Questions

Question

If we consider linear classification with the hinge loss

$$C_i(\mathbf{w}) = \max\{0, 1 - y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)}\},$$

what is the formula for SGD update?

$$\mathbf{w}^{(t+1)} = \begin{cases} \mathbf{w}^{(t)}, & \text{if } y^{(i_t)}(\mathbf{w}^{(t)})^\top \mathbf{x}^{(i_t)} \geq 1 \\ \mathbf{w}^{(t)} + \eta_t y^{(i_t)} \mathbf{x}^{(i_t)}, & \text{otherwise.} \end{cases}$$

Question

If we consider a regularization problem with

$$C_i(\mathbf{w}) = \frac{1}{2}\big(\max\{0, 1 - y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)}\}\big)^2 + \lambda\|\mathbf{w}\|_2^2/2,$$

what is the formula for SGD update?

$$\mathbf{w}^{(t+1)} = \begin{cases} \mathbf{w}^{(t)} - \lambda \mathbf{w}^{(t)}, & \text{if } y^{(i_t)}(\mathbf{w}^{(t)})^\top \mathbf{x}^{(i_t)} \geq 1 \\ \mathbf{w}^{(t)} - \lambda \mathbf{w}^{(t)} + \eta_t(1 - y^{(i_t)}\mathbf{w}^\top \mathbf{x}^{(i_t)}) y^{(i_t)} \mathbf{x}^{(i_t)}, & \text{otherwise.} \end{cases}$$

Next Lecture

Perceptron and Neural Network