Lecture 9: Gradient Descent

Introduction to Learning and Analysis of Big Data
Efficient learning algorithms

- How to implement learning algorithms for linear predictors efficiently?
- Is there a general method that works for many learning algorithms?
- Which types of learning algorithms can be implemented efficiently?
- How to make the implementation even more efficient in practice?
  - Run faster;
  - Use less RAM;
  - Work even if training examples cannot be stored.
Regularized Loss Minimization

- Many learning algorithms can be written as a minimization problem.

- Learning linear predictors:
  - ERM for some $\mathcal{B} \subseteq \mathbb{R}^d$: Minimize $\min_{w \in \mathcal{B}} \text{err}(h_w, S)$.
  - Soft SVM: Minimize $\min_{w \in \mathbb{R}^d} \lambda \|w\|^2 + \ell^h(w, S)$.
  - Hard SVM: Minimize $\min_{w \in \mathbb{R}^d : \forall i \leq m, y_i \langle w, x_i \rangle \geq 1} \|w\|^2$.

- In general:
  $$\min_{w \in \mathcal{B}} R(w) + \ell(w, S).$$

- $\ell$ is a loss function, $\ell(w, S) = \frac{1}{m} \sum_{i=1}^{m} \ell(w, (x_i, y_i))$.
- $R$ is a regularization term: controls overfitting.

- ERM: $\mathcal{B} = \mathcal{H}$, $\ell(w, S) := \text{err}(h_w, S)$.
- Soft SVM: $\mathcal{B} = \mathbb{R}^d$, $R(h_w) := \lambda \|w\|^2$.
- Hard SVM: $\mathcal{B} = \{w \in \mathbb{R}^d : \forall i \leq m, y_i \langle w, x_i \rangle \geq 1\}$, $R(h_w) = \|w\|^2$. 
Learning as Optimization

- Learning algorithms as a minimization problem:

  \[
  \text{Minimize}_{w \in \mathcal{B}} \quad R(w) + \ell(w, S).
  \]

- **Optimization algorithm**: Find the solution of a minimization problem.

- Efficient optimization $\implies$ efficient learning.

- When can we have efficient optimization?
  - Not always: If \( \ell(w, S) = \text{err}(h_w, S) \) and \( \mathcal{B} = \mathbb{R}^d \), NP-hard.

- A convenient case: The objective function has a single minimum.
Convex Sets

- Which objective functions have this nice property?

**Definition: Convex Sets**

A set $C \subseteq \mathbb{R}^n$ is convex if for any two vectors $u, v \in C$, the straight line between them is also in $C$:

$$\forall \alpha \in [0, 1], \quad \alpha u + (1 - \alpha)v \in C.$$
Convex Functions

Definition: Convex Functions

Let $C$ be a convex set. Let $f : C \to \mathbb{R}$ be a function. $f$ is convex if for every $u, v \in C$, $\alpha \in [0, 1]$,

$$f(\alpha u + (1 - \alpha)v) \leq \alpha f(u) + (1 - \alpha)f(v).$$

- Geometric meaning:
  - A straight line between points on the graph of $f$ is above (or on) the graph of $f$.
  - The area above the graph of $f$ is a convex set.
Convex functions

Theorem

If $C \subseteq \mathbb{R}^n$ is convex and $f : C \rightarrow \mathbb{R}$ is convex, then every local minimum of $f$ is also a global minimum.

- Learning with a minimization problem on linear predictors:

  \[
  \text{Minimize}_{w \in \mathcal{B}} f(w)
  \]

- If $\mathcal{B}$ and $f$ are convex, this is a \textit{convex minimization problem}.
- The solution of a convex minimization problem can be found efficiently using the \textbf{Gradient Descent} algorithm.
  - We will see it soon.

- How can we identify convex functions?
Identifying Convex functions

- Convexity: \( \forall \alpha \in [0, 1], u, v \in C, f(\alpha u + (1 - \alpha)v) \leq \alpha f(u) + (1 - \alpha)f(v) \).
- If \( C \subseteq \mathbb{R} \), and \( f \) is twice differentiable, then \( f \) is convex iff \( f'' \geq 0 \).
- (In higher dimensions the second derivative is a matrix and it should have non-negative eigenvalues)
- Linear functions are convex.
- **Conic combinations** of convex functions are convex:
  \[
g(u) = \sum_{i=1}^{k} a_i f_i(u) \text{ for } a_i \geq 0
\]
- Maximum of two convex functions is convex.
  - \( g(u) := \max\{f_1(u), f_2(u)\} \)
  \[
g(\alpha u + (1 - \alpha)v) = \max\{f_1(\alpha u + (1 - \alpha)v), f_2(\alpha u + (1 - \alpha)v)\}
\leq \max\{\alpha f_1(u) + (1 - \alpha)f_1(v), \alpha f_2(u) + (1 - \alpha)f_2(v)\}
\leq \alpha \max\{f_1(u), f_2(u)\} + (1 - \alpha) \max\{f_1(v), f_2(v)\}
= \alpha g(u) + (1 - \alpha)g(v).
- Composition of convex and linear function is convex.
- Let’s check some learning objectives.
Is ERM on linear predictors convex?

- ERM on linear predictors:

\[
\text{Minimize}_{w \in \mathbb{R}^d} \quad \text{err}(h_w, S) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}[y_i \langle w, x_i \rangle < 0].
\]

- \( f(w) = \text{err}(h_w, S) \), is it convex?
- Take \( d = 1, \ m = 1, \ y_1 = 1, \ x_1 = 1 \).
- Is \( f(w) = \mathbb{I}[w \leq 0] \) convex?

- No 😞. (makes sense: this problem is NP-hard!)
Is Hard SVM Convex?

- Hard SVM:
  \[
  \text{Minimize } \quad \min_{w \in \mathbb{R}^d : \forall i \leq m, y_i \langle w, x_i \rangle \geq 1} \quad \|w\|^2
  \]

- Is the set \( B = \{ w \in \mathbb{R}^d : \forall i \leq m, y_i \langle w, x_i \rangle \geq 1 \} \) convex?
  - Convex set: For all \( \alpha \in [0, 1], u, v \in B \), show \( \alpha u + (1 - \alpha)v \in B \).
  - \( u, v \in B \), then for \( i \leq m \), \( y_i \langle v, x_i \rangle \geq 1 \) and \( y_i \langle u, x_i \rangle \geq 1 \).

\[
y_i \langle \alpha u + (1 - \alpha)v, x_i \rangle = \alpha y_i \langle u, x_i \rangle + (1 - \alpha)y_i \langle v, x_i \rangle \geq 1.
\]

- Is \( f(w) = \|w\|^2 \) convex?
  - \( f(w) = \sum_{i=1}^{d} w(i)^2 \).
  - Define \( f_i : \mathbb{R}^d \to \mathbb{R} \) by \( f_i(w) = w(i)^2 \).
  - \( f(w) = \sum_{i=1}^{m} f_i(w) \), a conic combination of functions. Show \( f_i \) are convex.
  - Define \( g : \mathbb{R} \to \mathbb{R} \) by \( g(z) = z^2 \), \( p : \mathbb{R}^d \to \mathbb{R} \) by \( p(w) = w(i) \).
  - \( f_i = g \circ p \).
  - \( g''(z) = 2 \geq 0 \) so \( g \) is convex. \( p \) is linear (also convex).
  - \( f_i \) is the composition of linear and convex, so \( f_i \) is convex.
  - Conclusion: \( f \) is convex.

- Conclusion: Hard-SVM is a convex minimization problem. 😊
Is Soft SVM Convex?

- Soft-SVM minimization problem:
  \[
  \text{Minimize}_{w \in \mathbb{R}^d} \quad \lambda \|w\|^2 + \ell^h(w, S).
  \]

- Is \(f(w) = \lambda \|w\|^2 + \ell^h(w, S)\) convex?

- \(f\) is a conic combination of \(g_1(w) := \|w\|^2\) and \(g_2(w) := \ell^h(w, S)\).

- \(g_1\) is convex.

- \(g_2(w) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\}\). Is it convex?

- \(g_2\) is a conic combination of functions: \(z_i(w) := \max\{0, 1 - y_i \langle w, x_i \rangle\}\).

- \(z_i\) is the maximum of two functions:
  - \(z_{i,1}(w) = 0\): linear in \(w\), hence convex
  - \(z_{i,2}(w) = 1 - y_i \langle w, x_i \rangle = 1 - \langle w, y_i x_i \rangle\): linear in \(w\), hence convex.

- \(z_i\) is convex \(\implies g_2\) is convex \(\implies f\) is convex 😊.
Efficient convex minimization: Gradient Descent

Minimize \( w \in B f(w) \)

- \( f \) is convex, hence any local minimum it has is also global.

- Idea:
  - Start with some \( w \)
  - Do in iterations:
    - Move \( w \) slightly in a direction that decreases \( f(w) \).
    - Stop and return some average of the \( w \)’s.

- What is a *direction that decreases* \( f(w) \)?

- One-dimensional case (\( w \in \mathbb{R} \)): Move in direction of \(-f'(w)\).

- Multi-dimensional case: Move in direction of the negative gradient.
Gradient of function

- Assume $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable.
- Partial derivative of $f$ according to $w(i)$: $\frac{\partial f(w)}{\partial w(i)}$.
- Example: $f(w) = \|w\|^2$. Then $\frac{\partial f(w)}{\partial w(i)} = 2w(i)$.
- The gradient of $f$ at $w$ is $\nabla f(w) := (\frac{\partial f(w)}{\partial w(1)}, \ldots, \frac{\partial f(w)}{\partial w(d)})$.
- Example: $f(w) = \|w\|^2$. Then $\nabla f(w) = (2w(1), \ldots, 2w(d))$.
- If $f$ is one-dimensional, $\nabla f(w) = f'(w)$.
- $\nabla f(w)$ is the direction that would **increase** $f(w)$ the most.
- To get closer to the minimum of $f$, move in the direction $-\nabla f(w)$.
The gradient descent algorithm

- $\nabla f(w)$ is the direction that would increase $f(w)$ the most.
- To get closer to the minimum of $f$, move in the direction $-\nabla f(w)$.
- We show here Gradient Descent for $B = \mathbb{R}^d$.

---

**Gradient Descent**

**input** Number of iterations $T$, step size $\eta > 0$

**output** $w \in \mathbb{R}^d$ that (approximately) solves $\text{Minimize}_{w \in \mathbb{R}^d} f(w)$

1: $w^{(1)} \leftarrow (0, \ldots, 0)$.
2: for $t = 1 : T$ do
3: \hspace{1em} $w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla f(w^{(t)})$.
4: end for
5: Return $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$.

- Can also return $w^{(T)}$ or average of a few $w$'s. Depends on properties of $f$.
- For $B \subsetneq \mathbb{R}^d$, an extra projection step is needed (see book).
The gradient descent algorithm

- Update step: \( w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla f(w^{(t)}) \).
- Step size \( \eta \) is important:
  - Too small: will take a long time to get close to the minimum
  - Too large: might oscillate and miss the minimum.

**Theorem**

Let \( w^* \in \arg\min_{w \in \mathbb{R}^d} f(w) \). If \( f \) is a convex function, and \( \eta := \frac{C}{\sqrt{T}} \) \((C > 0 \text{ depends on } f \)), then

\[
f(\bar{w}) - f(w^*) \leq O\left(\frac{1}{\sqrt{T}}\right).
\]
Subgradients

- Gradient descent uses $\nabla f(w) := \left( \frac{\partial f(w)}{\partial w(1)}, \ldots, \frac{\partial f(w)}{\partial w(d)} \right)$.

- What if $f$ is not differentiable?

- Example: Hinge-loss.

$$\text{Minimize}_{w \in \mathbb{R}^d} \quad \lambda \|w\|^2 + \ell^h(w, S).$$

$$\ell^h(w, S) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\}.$$ $f(w) = \ell^h(w, S)$ is not differentiable at $w$ if $\exists i \leq m, y_i \langle w, x_i \rangle = 1$.

- $w \in \mathbb{R}, m = 1, y_1 = 1, x_1 = 1$:
Subgradients

- What to do when $\nabla f(w)$ does not exist? ($f$ is not differentiable at $w$)
- $v \in \mathbb{R}^n$ is a **sub-gradient** of $f : C \to \mathbb{R}$ at $w$ if
  \[
  \forall u \in C, f(u) \geq f(w) + \langle v, u - w \rangle.
  \]
- A sub-gradient of $f$ at $w$ is a direction that is always below $f(w)$.
- Formally: it is a vector $z$ such that for all $u$, $f(w) + \langle u - w, z \rangle \leq f(u)$.

Set of sub-gradients at $w$: $\partial f(w)$.

- If $f : C \to \mathbb{R}$ is convex, $\partial f(w) \neq \emptyset$ for all $w \in C$. (even if $f$ is not differentiable!)
- When $\nabla f(w)$ exists, $\partial f(w) = \{ \nabla f(w) \}$. (no other sub-gradients)
- Any $v \in \partial f(w)$ can be used in Gradient Descent instead of $\nabla f(w)$.
- **Theorem**: $w$ is a minimizer of $f$ $\iff$ $(0, \ldots, 0) \in \partial f(w)$. 

\[ f(w) \]
\[ w \]
\[ \nabla f(w) \]
Subgradients for soft-SVM

- For the case of hinge-loss, we can use the following rule:

Subgradients for maximum functions

Let \( f(w) := \max \{g_1(w), g_2(w)\} \), where \( g_1, g_2 \) are convex differentiable. If \( f(w) = g_i(w) \), then \( \nabla g_i(w) \in \partial f(w) \).

- In gradient descent algorithm, choose some \( v_w \in \partial f(w) \) instead of \( \nabla f(w) \).
- What should \( v_w \) be for soft-SVM?

\[
 f(w) = \lambda \|w\|^2 + \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i \langle w, x_i \rangle\}
\]

Define: \( g(w) := \lambda \|w\|^2, z_i(w) := 1 - y_i \langle w, x_i \rangle, g_i(w) := \max\{0, z_i(w)\} \).

- So

\[
 f(w) = g(w) + \frac{1}{m} \sum_{i=1}^{m} g_i(w).
\]

\( \nabla g(w) = 2 \lambda w. \)
\( \nabla z_i(w) = -y_i x_i. \)

Choose \( v_i(w) \in \partial g_i(w) \) according to the maximum rule:

\[
 v_i(w) := \begin{cases} 
 0 & y_i \langle w, x_i \rangle \geq 1 \\
 -y_i x_i & \text{otherwise}
\end{cases}
\]

Set \( v_w := 2 \lambda w + \frac{1}{m} \sum_{i=1}^{m} v_i(w) \). Use instead of \( \nabla f(w) \) in GD algorithm.
Stochastic Gradient Descent

- Recall the general learning minimization problem for linear predictors:

\[
\text{Minimize}_{w \in B} \ R(w) + \ell(w, S).
\]

- In Gradient Descent, \( \nabla f(w^{(t)}) = \nabla R(w^{(t)}) + \nabla \ell(w^{(t)}, S) \) is calculated in every round.

- Calculating \( \nabla \ell(w, S) \) in the GD algorithm can be computationally heavy:
  - Each iteration costs \( O(m) \) calculations
  - Each iteration requires retrieving \( O(m) \) examples into the RAM.

- Solution: **Stochastic Gradient Descent** (SGD).

  - Idea: In each iteration estimate \( \nabla \ell(w, S) \).

\[
\ell(w, S) = \frac{1}{m} \sum_{i=1}^{m} \ell(w, (x_i, y_i)) \quad \Rightarrow \quad \nabla \ell(w, S) = \frac{1}{m} \sum_{i=1}^{m} \nabla \ell(w, (x_i, y_i))
\]

  \[
  \Rightarrow \quad \nabla \ell(w, S) = \mathbb{E}_{(X, Y) \sim S}[\nabla \ell(w, (X, Y))].
  \]

- \((X, Y)\) is a random pair selected uniformly from \(S\).

- SGD: Select a random \(i \in \{1, \ldots, m\}\), and use \( \nabla \ell(w, (x_i, y_i)) \) instead of \( \nabla \ell(w, S) \).
Stochastic Gradient Descent

- SGD: Select a random $i \in \{1, \ldots, m\}$, and use $\nabla \ell(w, (x_i, y_i))$ instead of $\nabla \ell(w, S)$.

---

**Stochastic Gradient Descent**

<table>
<thead>
<tr>
<th>input</th>
<th>Number of iterations $T$, step size $\eta &gt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>$w \in \mathbb{R}^d$ that (approximately) solves $\text{Minimize}_{w \in \mathbb{R}^d} R(w) + \ell(w, S)$</td>
</tr>
</tbody>
</table>

1: $w^{(1)} \leftarrow (0, \ldots, 0)$.
2: for $t = 1 : T$ do
3: Draw a random $i$ uniformly from $\{1, \ldots, m\}$
4: $w^{(t+1)} \leftarrow w^{(t)} - \eta (\nabla R(w^{(t)}) + \nabla \ell(w^{(t)}, (x_i, y_i)))$.
5: end for
6: Return $\bar{w} = \frac{1}{T} \sum_{t=1}^{T} w^{(t)}$.

- As in Gradient Descent, can use sub-gradients instead of gradients.
Stochastic Gradient Descent

Theorem

Let \( w^* \in \text{argmin}_{w \in \mathbb{R}^d} f(w) \), where \( f(w) = R(w) + \ell(w, S) \). If \( f \) is a convex function, and the step size is set to \( \eta = \frac{C}{\sqrt{T}} \) (\( C > 0 \) depends on \( f \)), then

\[
\mathbb{E}[f(\bar{w})] - f(w^*) \leq O\left(\frac{1}{\sqrt{T}}\right).
\]

- The same guarantee as for GD, except “in expectation”
- Hidden constants are also the same
- \( \Rightarrow \) same \#iterations, each iteration is \( O(1) \) instead of \( O(m) \).
- However, SGD suffers from variance unlike GD.
- SGD is used in practice for optimization with Big Data.
- What if \( f \) is not convex?
  - SGD can converge to a local minimum.
SGD update step for soft-SVM

- The objective: Minimize $w \in \mathbb{R}^d R(w) + \ell(w, S)$
- The update step:
  \[ w^{(t+1)} \leftarrow w^{(t)} - \eta (\nabla R(w^{(t)}) + \nabla \ell(w^{(t)}, (x_i, y_i))). \]
- Soft-SVM objective:
  \[
  \begin{align*}
  &\text{Minimize}_{w \in \mathbb{R}^d} \lambda \|w\|^2 + \ell^h(w, S) \\
  &R(w) := \lambda \|w\|^2 \\
  &\ell(w, (x_i, y_i)) = \max\{0, 1 - y_i \langle w, x_i \rangle\}. 
  \end{align*}
  \]
- We showed: $\nabla R(w) = 2\lambda w$.
- And can choose $v_i(w) \in \partial \ell^h(w, (x_i, y_i))$, where
  \[ v_i(w) := \begin{cases} 0 & y_i \langle w, x_i \rangle \geq 1 \\ -y_i x_i & \text{otherwise} \end{cases} \]
- Soft SVM SGD update step: for a random $i \in \{1, \ldots, m\}$,
  \[ w^{(t+1)} \leftarrow w^{(t)} - \eta (2\lambda w^{(t)} + v_i(w^{(t)})). \]
A surprising similarity

- Soft SVM SGD update step: for a random $i \in \{1, \ldots, m\}$,
  \[ w^{(t+1)} \leftarrow w^{(t)} - \eta(2\lambda w^{(t)} + v_i(w^{(t)})). \]

Definition of $v_i(w)$:
\[
v_i(w) := \begin{cases} 
0 & y_i \langle w, x_i \rangle \geq 1 \\
-y_i x_i & \text{otherwise}
\end{cases}
\]

- What is the update step if we set $\lambda = 0$ and $\eta = 1$?
  - If $y_i \langle w, x_i \rangle \leq 0$,
    \[ w^{(t+1)} \leftarrow w^{(t)} + y_i x_i. \]
  - Otherwise, no update.

- This is exactly the **Perceptron** update rule!
SGD step size

- The optimal step size $\eta$ depends on the objective function $f$.
- For some $f$, it is better to use a variable step size $\eta_t$.
- Update step: $w^{(t+1)} \leftarrow w^{(t)} - \eta_t (\nabla R(w^{(t)}) + \nabla \ell(w^{(t)}, (x_i, y_i)))$.
- For soft-SVM, a step size of $\eta_t = \frac{1}{\lambda t}$ gives better guarantees.
An infinite training sample?

- A server on the Internet accepts requests from users
- Rate of requests: 100,000 per second.
- Learning problem: Identify faulty requests.
- What is the size of our training sample?
  - We can keep updating $w(t)$ forever
  - When need to predict, use the current $\tilde{w}$.
- Cannot store all training examples.
- Think of this as an infinite training sample $S = ((x_1, y_1), (x_2, y_2), \ldots)$
- Can still apply SGD!
Stochastic Gradient Descent on a stream of examples

1: $w^{(1)} \leftarrow (0, \ldots, 0)$.  
2: while true do  
3: Get next sample $(x_t, y_t)$  
4: $w^{(t+1)} \leftarrow w^{(t)} - \eta_t (\nabla R(w^{(t)}) + \nabla \ell(w^{(t)}, (x_t, y_t)))$.  
5: end while

- Input is constantly fed to the algorithm: can be infinite 😊
- Output: whenever we want, we can use current $ar{w}_t = \frac{1}{t} \sum_{i=1}^{t} w^{(i)}$.
- No need to store examples 😊.
Stochastic Gradient Descent on a stream of examples

- Compare to batch (non-stream) SGD:
  - In batch SGD, \((x_i, y_i)\) is a random example from \(S\).
  - In stream SGD, \((x_t, y_t)\) is a random example from \(D\).

**Theorem**

Let \(w^* \in \arg\min_{w \in \mathbb{R}^d} f(w)\), where \(f(w) = R(w) + \ell(w, D)\). If \(f\) is a convex function, and the step size is set to \(\eta_t = O\left(\frac{C}{t}\right)\), then at time \(t\),

\[
\mathbb{E}[f(\bar{w})] - f(w^*) \leq O\left(\frac{1}{\sqrt{t}}\right).
\]

- There are better guarantees for some cases, such as soft-SVM.
- Sample complexity of soft-SVM SGD \(\approx\) sample complexity of standard soft-SVM.
Gradient Descent: Summary

- Many learning algorithms can be written as minimization objectives.
- Some of these objectives are convex, including soft-SVM.
- Gradient Descent iteratively moves the linear predictor $\mathbf{w}$ in the direction of the negative gradient.
- For convex functions, there is always a sub-gradient that can be used, even if the function is not differentiable.
- For convex objectives, Gradient Descent gets close to the optimal solution.
- Stochastic Gradient Descent uses a single example in each iteration to estimate the gradient.
- SGD has similar guarantees to GD and requires much less memory.
- SGD can also be used to learn from an infinite training stream.