Machine Learning (CSE 446): Variations on the Theme of Gradient Descent

Noah Smith

University of Washington
nasmith@cs.washington.edu

October 27, 2017

Learning as Loss Minimization

$$\mathbf{z}^* = \operatorname*{argmin}_{\mathbf{z}} \frac{1}{N} \sum_{n=1}^{N} \underbrace{L(\mathbf{x}_n, y_n, \mathbf{z})}_{L_n(\mathbf{z})} + R(\mathbf{z})$$

For our hyperplane/neuron-inspired classifier, $\mathbf{z} = (\mathbf{w}, b)$.

Subderivatives and Subgradients

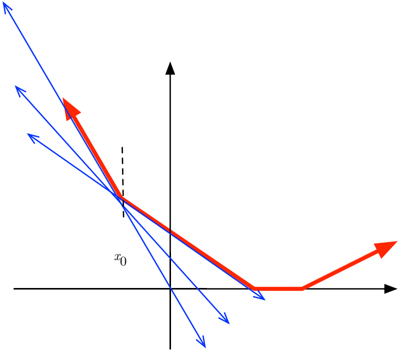
A **subderivative** of F at x_0 is any c such that, for all x:

$$F(x) - F(x_0) \geq c(x - x_0)$$

This is a generalization of derivatives (for differentiable functions, there is only one subderivative at x_0, and it's the derivative).

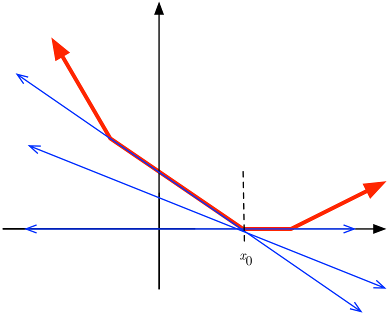
Vector of subderivatives in all dimensions: **subgradient**.

Subderivative



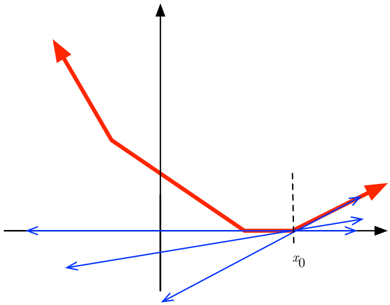
The set of subderivatives for the function at a point x_0 consists of the slopes of all tangent lines fully below the function.

Subderivative



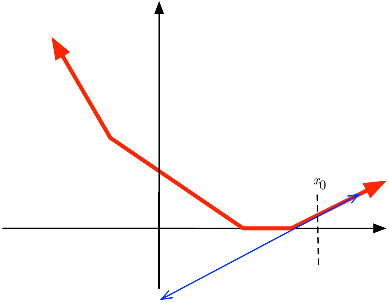
The set of subderivatives for the function at a point x_0 consists of the slopes of all tangent lines fully below the function.

Subderivative



The set of subderivatives for the function at a point x_0 consists of the slopes of all tangent lines fully below the function.

Subderivative



The set of subderivatives for the function at a point x_0 consists of the slopes of all tangent lines fully below the function.

Variation 1

Data: function $F : \mathbb{R}^d \rightarrow \mathbb{R}$, number of iterations K, step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$
Result: $\mathbf{z} \in \mathbb{R}^d$
initialize: $\mathbf{z}^{(0)} = \mathbf{0}$;
for $k \in \{1, \ldots, K\}$ **do**
  # choose a subgradient; doesn't matter which one;
  $\mathbf{g}^{(k)} = \nabla_{\mathbf{z}} F(\mathbf{z}^{(k-1)})$;
  $\mathbf{z}^{(k)} = \mathbf{z}^{(k-1)} - \eta^{(k)} \cdot \mathbf{g}^{(k)}$;
end
return $\mathbf{z}^{(K)}$;

Algorithm 1: SubgradientDescent

Variation 2

Data: loss functions $L_1, \ldots, L_N$, regularization function R, number of iterations K, step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$

Result: parameters $\mathbf{z} \in \mathbb{R}^d$

initialize: $\mathbf{z}^{(0)} = \mathbf{0}$;

for $k \in \{1, \ldots, K\}$ **do**

$\quad i \sim \text{Uniform}(\{1, \ldots, N\})$;

$\quad \mathbf{g}^{(k)} = \nabla_{\mathbf{z}} L_i(\mathbf{z}^{(k-1)}) + \nabla_{\mathbf{z}} R(\mathbf{z}^{(k-1)})$;

$\quad \mathbf{z}^{(k)} = \mathbf{z}^{(k-1)} - \eta^{(k)} \cdot \mathbf{g}^{(k)}$;

end

return $\mathbf{z}^{(K)}$;

Algorithm 2: $\textsc{Stochastic(Sub)GradientDescent}$ for minimizing $\frac{1}{N}\sum_{n=1}^{N} L_n(\mathbf{z}) + R(\mathbf{z})$.

Observation

If you let L be the perceptron loss and don't regularize, and run stochastic subgradient descent with all $\eta = 1$, you have recovered the perceptron algorithm.

Variation 3

Data: loss functions $L_1, \ldots, L_N$, regularization function R, number of iterations K, step sizes $\langle \eta^{(1)}, \ldots, \eta^{(K)} \rangle$, minibatch size B

Result: parameters $\mathbf{z} \in \mathbb{R}^d$

initialize: $\mathbf{z}^{(0)} = \mathbf{0}$;

for $k \in \{1, \ldots, K\}$ **do**

$\quad I \sim \mathrm{Uniform}(\{1, \ldots, N\}^B)$;

$\quad \mathbf{g}^{(k)} = \frac{1}{B} \sum_{i \in I} \nabla_{\mathbf{z}} L_i(\mathbf{z}^{(k-1)}) + \nabla_{\mathbf{z}} R(\mathbf{z}^{(k-1)})$;

$\quad \mathbf{z}^{(k)} = \mathbf{z}^{(k-1)} - \eta^{(k)} \cdot \mathbf{g}^{(k)}$;

end

return $\mathbf{z}^{(K)}$;

Algorithm 3: MinibatchStochastic(Sub)GradientDescent for minimizing $\frac{1}{N} \sum_{n=1}^{N} L_n(\mathbf{z}) + R(\mathbf{z})$.

General-Purpose Optimization Algorithms

$\{\text{batch, minibatch, stochastic}\} \times \text{(sub)gradient descent}$

General-Purpose Optimization Algorithms

$\{$batch, minibatch, stochastic$\} \times$ (sub)gradient descent

Ninja: treat minibatch size $B \in \{1, \ldots, N\}$ as a hyperparameter!

Regularization

(Review)

Choose your loss function L. To fit the training data:

$$\min_{\mathbf{w},b} \frac{1}{N}\sum_{n=1}^{N} L\left(y_n \cdot (\mathbf{w} \cdot \mathbf{x}_n + b)\right) + R(\mathbf{w}, b)$$

Regularization: add a penalty to the objective function to encourage generalization.

Most common: $R(\mathbf{w}, b) = \lambda\|\mathbf{w}\|_2^2$.

- Note that this term is convex and differentiable.

This is called **(squared)** L_2 **regularization** or **ridge regularization**.

Some Regularization Functions

$$\text{ridge or (squared) } L_2 \qquad \lambda\|\mathbf{w}\|_2^2 = \lambda \sum_d \mathbf{w}[d]^2$$

$$\text{“}L_0\text{”} \qquad \lambda\|\mathbf{w}\|_0 = \lambda \sum_d [\![\mathbf{w}[d] \neq 0]\!]$$

$$\text{lasso or } L_1 \qquad \lambda\|\mathbf{w}\|_1 = \lambda \sum_d |\mathbf{w}[d]|$$

Inductive bias for ridge: small change in $\mathbf{x}[d]$ should have a small effect on prediction. Penalizing $\|\mathbf{w}\|_2$ is the same as penalizing $\|\mathbf{w}\|_2^2$, but to get the same effect you'll need a different λ.

Some Regularization Functions

$$\text{ridge or (squared) } L_2 \qquad \lambda\|\mathbf{w}\|_2^2 = \lambda \sum_d \mathbf{w}[d]^2$$

$$\text{“}L_0\text{”} \qquad \lambda\|\mathbf{w}\|_0 = \lambda \sum_d [\![\mathbf{w}[d] \neq 0]\!]$$

$$\text{lasso or } L_1 \qquad \lambda\|\mathbf{w}\|_1 = \lambda \sum_d |\mathbf{w}[d]|$$

Inductive bias for L_0: use fewer features.

Some Regularization Functions

$$
\begin{aligned}
\text{ridge or (squared) } L_2 \quad & \lambda\|\mathbf{w}\|_2^2 = \lambda \sum_d \mathbf{w}[d]^2 \\
\text{“}L_0\text{”} \quad & \lambda\|\mathbf{w}\|_0 = \lambda \sum_d [\![\mathbf{w}[d] \neq 0]\!] \\
\text{lasso or } L_1 \quad & \lambda\|\mathbf{w}\|_1 = \lambda \sum_d |\mathbf{w}[d]|
\end{aligned}
$$

Inductive bias for L_0 and lasso: use fewer features.

A Constrained View of the Regularized Loss Minimization Problem

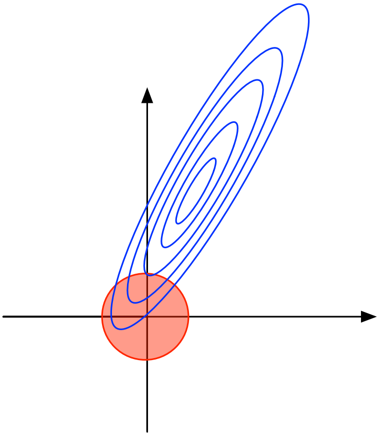
Tikhonov regularization:

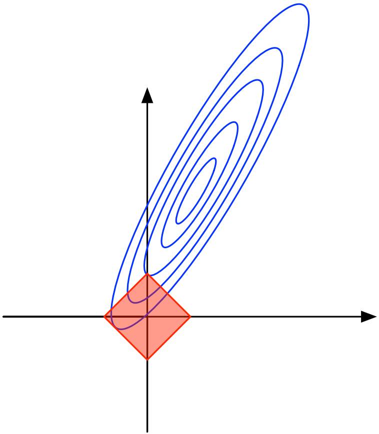
$$\mathbf{z}^* = \operatorname*{argmin}_{\mathbf{z}} \frac{1}{N} \sum_{n=1}^{N} L_n(\mathbf{z}) + \lambda \|\mathbf{z}\|_p$$

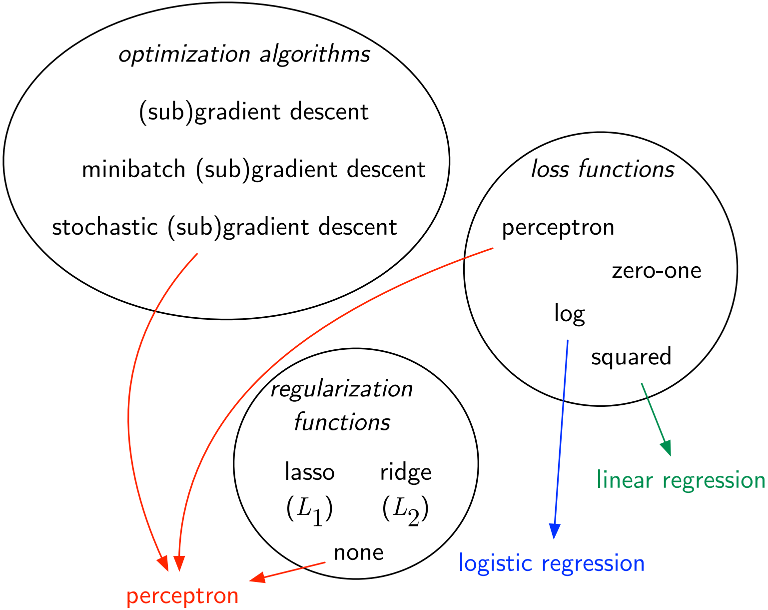
Ivanov regularization:

$$\mathbf{z}^* = \operatorname*{argmin}_{\mathbf{z}} \frac{1}{N} \sum_{n=1}^{N} L_n(\mathbf{z})$$

$$\text{s.t.} \quad \|\mathbf{z}\|_p \leq \tau$$






optimization algorithms
(sub)gradient descent
minibatch (sub)gradient descent
stochastic (sub)gradient descent
loss functions
perceptron
zero-one
log
squared
regularization functions
lasso (L_1)
ridge (L_2)
none
perceptron
logistic regression
linear regression