Machine Learning for Intelligent Systems

Lecture 12: Stochastic Gradient Descent

Reading: UML 14

Instructors: Nika Haghtalab (this time) and Thorsten Joachims

Regularized Linear Models

Many learning problems can be written as the following optimization on the sample set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

$$\min_{\vec{w},b} \; \boxed{R(\vec{w})} + C\frac{1}{n}\sum_{i=1}^{n} \boxed{L(\vec{w}\cdot\vec{x}_i + b, y_i)}$$

Regularizer

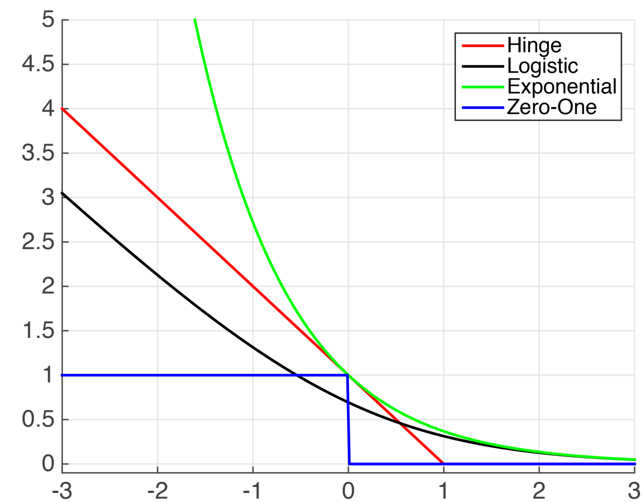
Loss of each instance $(\vec{x}_i, y_i)$

1. Primal SVM: $$\min_{\vec{w},b} \frac{1}{2}\vec{w}\cdot\vec{w} + C\frac{1}{n}\sum_{i=1}^{n} \max(1 - y_i(\vec{w}\cdot\vec{x}_i + b), 0)$$
2. Reg. Logistic Regression: $$\min_{\vec{w},b} \frac{1}{2}\vec{w}\cdot\vec{w} + C\frac{1}{n}\sum_{i=1}^{n} \log(1 + \mathrm{e}^{-y_i(\vec{w}\cdot\vec{x}_i + b)})$$
3. Ridge Regression: $$\min_{\vec{w},b} \frac{1}{2}\vec{w}\cdot\vec{w} + C\frac{1}{n}\sum_{i=1}^{n} (\vec{w}\cdot\vec{x}_i + b - y_i)^2$$

Loss Functions

$$\min_{\vec{w},b} \quad R(\vec{w}) + C\,\frac{1}{n}\sum_{i=1}^{n} L(\vec{w}\cdot\vec{x}_i + b, y_i\,)$$

Loss Function $\mathrm{L}(\bar{y}\,, y_i)$	**Algorithm**
Hinge loss: $\max(1\, - \bar{y}\, y_i, 0)$	SVM
Log loss: $\mathrm{Log}(1 + \exp(-\bar{y}y_i))$	Logistic Regression
Exponential loss: $\exp(-\bar{y}y_i)$	Boosting
0-1 Loss: $\mathbf{1}(\bar{y} \neq y_i)$	Classification loss



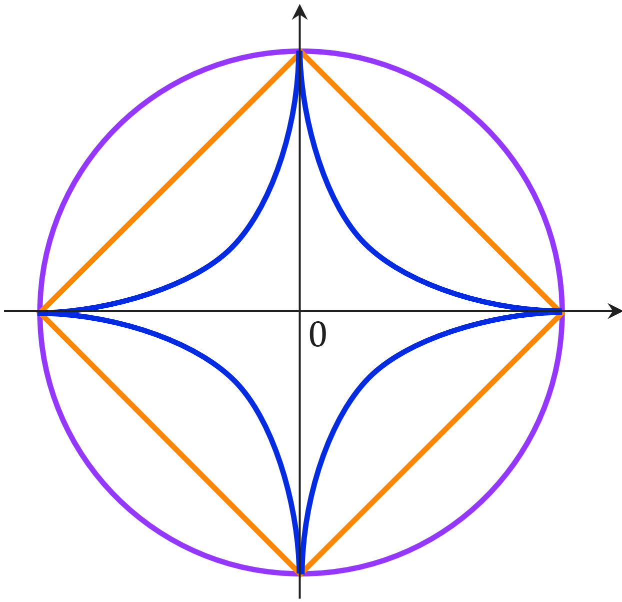
Non-Convex

Convex

Regularizers

$$\min_{\vec{w},b} \quad R(\vec{w}) + C\frac{1}{n}\sum_{i=1}^{n} L(\vec{w}\cdot\vec{x}_i + b, y_i)$$

Regularizer $R(\vec{w})$	Properties
ℓ_2 **regularization:** $\frac{1}{2}\vec{w}\cdot\vec{w}$	Convex
ℓ_1 **regularization:** $\Vert \vec{w} \Vert_1$	Convex, sparse
ℓ_p for $0 \leq p < 1$	Non-convex, very sparse

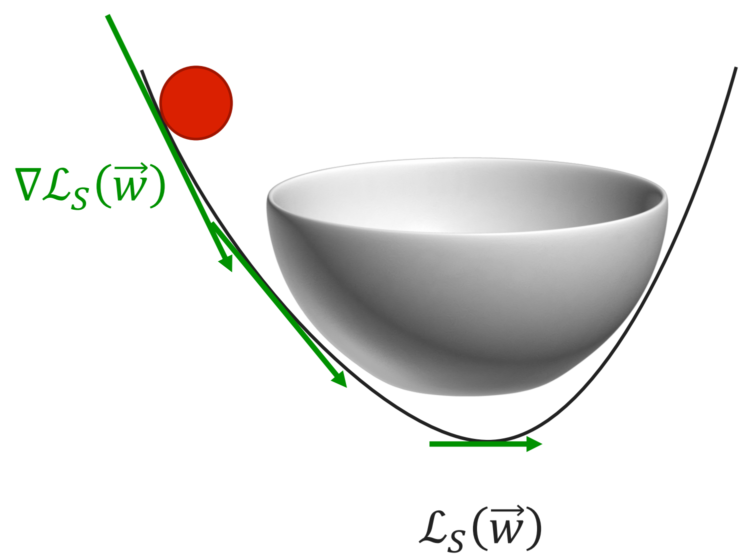


Optimizing Regularized Lin. Models

Many learning problems can be written as the following optimization on the sample set $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$.

$$\min_{\vec{w}} \quad \underbrace{R(\vec{w}) + C\frac{1}{n}\sum_{i=1}^{n} L(\vec{w}\cdot\vec{x}_i, y_i)}_{\mathcal{L}_S(\vec{w})}$$

Formal guarantees for when $\mathcal{L}_S(\vec{w})$ is convex in $\vec{w}$. But these methods are widely used for non-convex optimization as well.

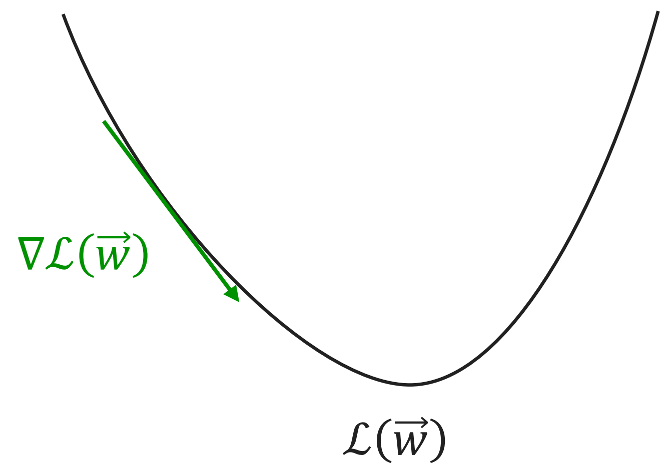


Gradient Descent (GD)

For finding the minimum of a convex function: $\min_{\vec{w}} \mathcal{L}(\vec{w})$

Gradient

$$\nabla \mathcal{L}(\vec{w}) = \left(\frac{\partial \mathcal{L}(\vec{w})}{\partial w_1}, \dots, \frac{\partial \mathcal{L}(\vec{w})}{\partial w_d} \right)$$



Gradient Descent

Input: A function $\mathcal{L}$, number of time steps T, step size η_t

Initialize $\vec{w}^{(0)} = (0, \dots, 0)$

For $t = 1, 2, 3, \dots, T$

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla \mathcal{L}(\vec{w}^{(t)})$$

Output $\vec{w}^{(T)}$

Learning with Gradient Descent

Consider the following optimization on $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$.

$$\min_{\vec{w}} \ \mathcal{L}_S(\vec{w}) = \min_{\vec{w}} \quad R(\vec{w}) + C\frac{1}{n}\sum_{i=1}^{n} L(\vec{w}\cdot\vec{x}_i, y_i)$$

Gradient Descent for Learning

Input: Step sizes η_t, time T, and samples $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$

Initialize $\vec{w}^{(0)} = (0, \dots, 0)$

For $t = 1, 2, 3, \dots, \mathrm{T}$

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t \frac{C}{n}\sum_{i=1}^{n} \nabla L\left(\vec{w}^{(t)}\cdot\vec{x}_i, y_i\right)$$

Output $\vec{w}^{(t)}$

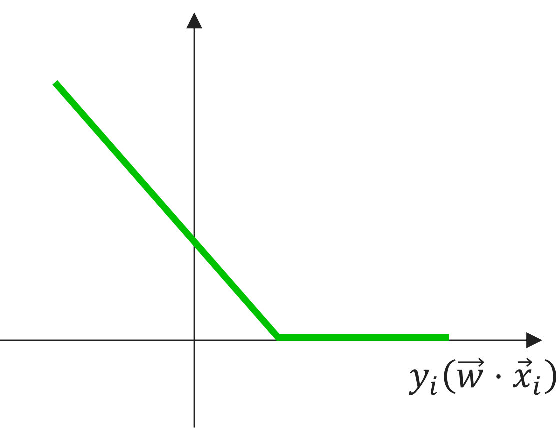
Example: GD for SVM

Example: Consider the SVM primal form as written in a regularized linear model (homogenous).

Gradient of

$$\min_{\vec{w}} \underbrace{\frac{1}{2}\vec{w}\cdot\vec{w}}_{} + \underbrace{C\frac{1}{n}\sum_{i=1}^{n} \overbrace{\max(1 - y_i(\vec{w}\cdot\vec{x}_i), 0)}}_{}$$

$$\nabla R(\vec{w}) = \vec{w} \qquad \nabla L_S(\vec{w}) = \frac{C}{n}\sum_{i=1}^{n} ???$$



$$\max(1 - y_i(\vec{w}\cdot\vec{x}_i), 0)$$

0 if $y_i(\vec{w}\cdot\vec{x}_i) > 1$, $\qquad$ $1 - y_i(\vec{w}\cdot\vec{x}_i)$ if $y_i(\vec{w}\cdot\vec{x}_i) \le 1$

Gradient = 0 $\qquad$ Gradient = $-y_i x_i$

GD update: $\vec{w}^{(t+1)} \leftarrow (1-\eta_t)\vec{w}^{(t)} + \frac{\eta_t C}{n}\sum_{i=1}^{n} y_i x_i \,\mathbf{1}\left(y_i\left(\vec{w}^{(t)}\cdot\vec{x}_i\right) \le 1\right)$

Gradient Descent for Large Scale ML

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t \frac{C}{n} \sum_{i=1}^{n} \nabla L\left(\vec{w}^{(t)} \cdot \vec{x}_i, y_i\right)$$

Challenges?

- Large data set $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$ for very large n.
- High dimensional data sets $x_i \in \mathbb{R}^d$ for very large d.

Using fewer samples for the update

Each time step use fewer samples for update

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t \frac{C}{n} \sum_{i=1}^{n} \nabla L\left(\vec{w}^{(t)} \cdot \vec{x}_i, y_i\right)$$

Take a random $\left(\vec{x}_{(t)}, y_{(t)}\right) \sim S$.

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t C\, \nabla L\left(\vec{w}^{(t)} \cdot \vec{x}_{(t)}, y_{(t)}\right)$$

Stochastic Gradient Descent (SGD)

Gradient Descent for Learning

Input: Function $\mathcal{L}$, step sizes η_t, time T, samples $S = \{(x_1, y_1), \dots, (x_n, y_n)\}$

Initialize $\vec{w}^{(0)} = (0, \dots, 0)$

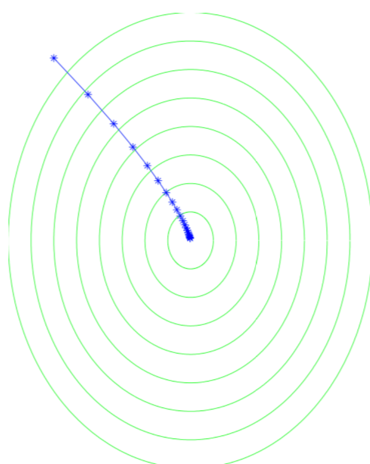
For $t = 1, 2, 3, \dots, T$

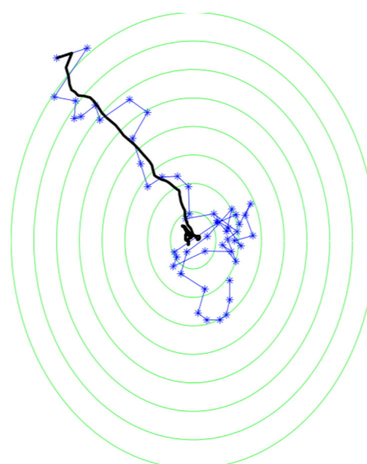
Take a random sample of $(\vec{x}, y) \sim S$

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t \, C \;\; \nabla L\left(\vec{w}^{(t)} \cdot \vec{x}, y\right)$$

Output $\vec{w}^{(T)}$. %% Or the average of $\vec{w}^{(1)}, \dots, \vec{w}^{(T)}$

Gradient Descent





Stochastic Gradient Descent:

Each iteration

Average up to now

Added from the whiteboard

Example: SGD for SVM

Example: Consider the SVM primal form as written in a regularized linear model (homogenous).

$$\min_{\vec{w}} \frac{1}{2}\vec{w}\cdot\vec{w} + C\frac{1}{n}\sum_{i=1}^{n}\max(1 - y_i(\vec{w}\cdot\vec{x}_i), 0)$$

SGD update: Take $(\vec{x}_i, y_i) \sim S$

$$\vec{w}^{(t+1)} \leftarrow (1-\eta_t)\vec{w}^{(t)} + \eta_t C y_i \vec{x}_i \mathbf{1}(y_i(\vec{w}^{(t)}\cdot\vec{x}_i) \leq 1)$$

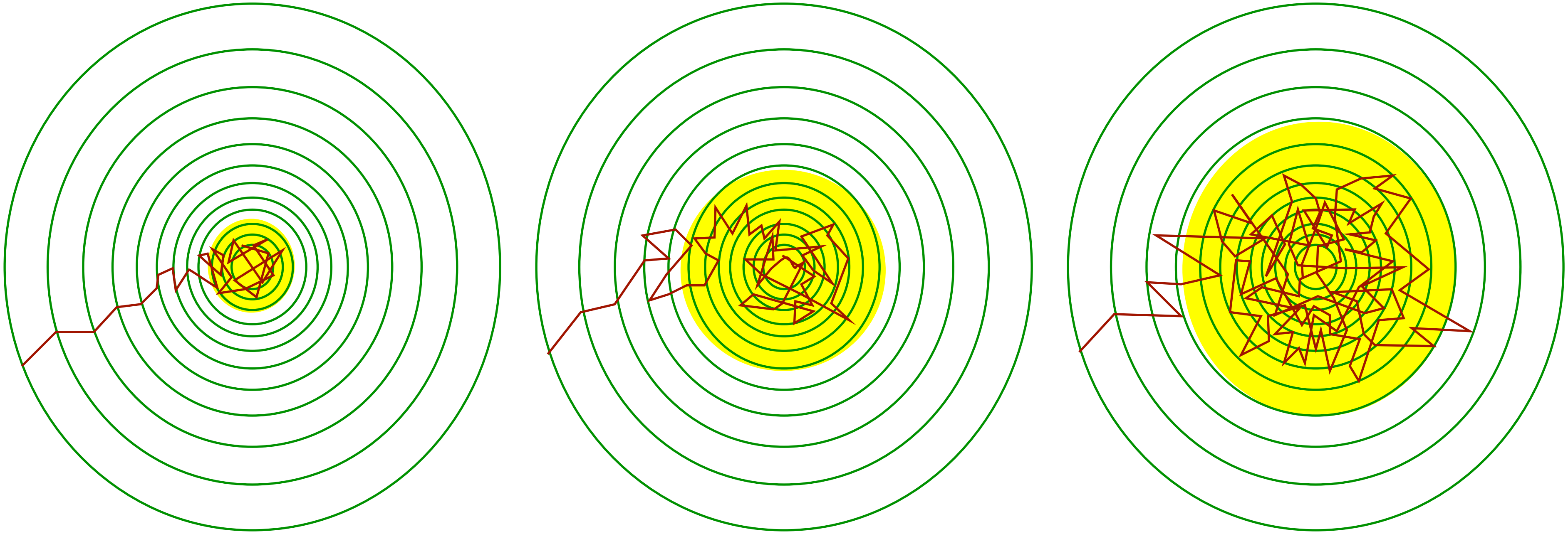
Equivalently:

Take $(\vec{x}_i, y_i) \sim S$

If $y_i(\vec{w}^{(t)}\cdot\vec{x}_i) \leq 1$ then $\vec{w}^{(t+1)} \leftarrow (1-\eta_t)\vec{w}^{(t)} + \eta_t C y_i \vec{x}_i$

Else $\vec{w}^{(t+1)} \leftarrow (1-\eta_t)\vec{w}^{(t)}$.

Effects of Step Size



Small step size   Medium step size   Larger step size

Smaller step size: More similar to Gradient Descent, less stochastic improvement, less uncertainty

Bigger step size: More stochastic improvement, more uncertainty.

What makes SGD work

Gradient Descent:

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t C \underbrace{\frac{1}{n}\sum_{i=1}^{n} \nabla L\left(\vec{w}^{(t)} \cdot \vec{x}_i, y_i\right)}_{\mathbb{E}_{(\vec{x},y)}\left[\nabla L\left(\vec{w}^{(t)} \cdot \vec{x}, y\right)\right]}$$

Stochastic Gradient Descent:

- Uses an “unbiased” estimator for the total gradient.
- Step size helps control the variance.

Noisy Estimates

Reduce Computation

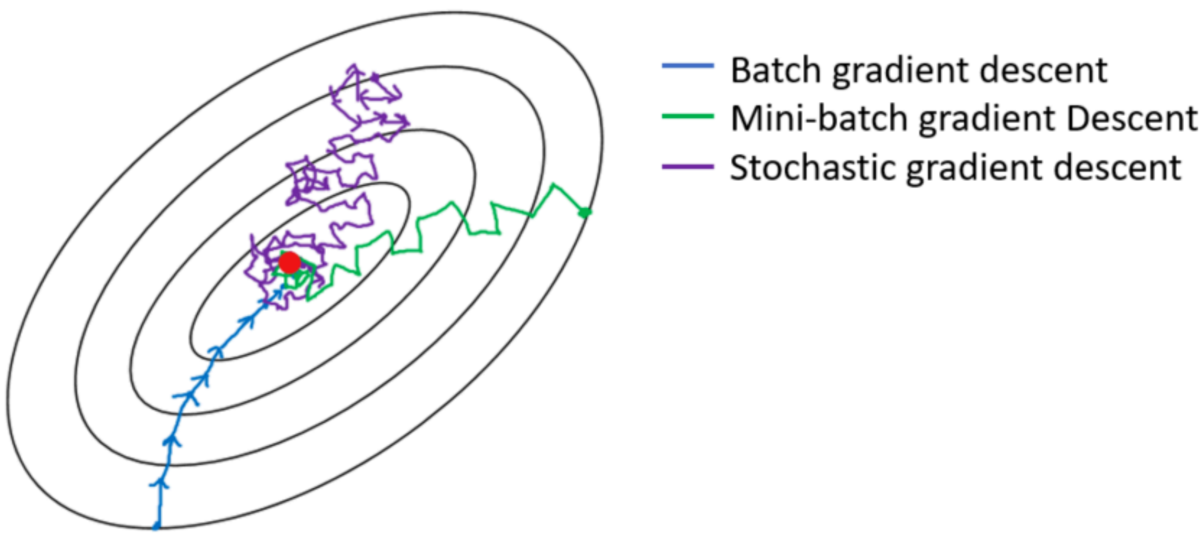
Improve Generalization

Mini-Batch

Go between deterministic Gradient Descent and Stochastic Gradient Descent, take between n and 1 instances.

At each time: Take a random subset S_t of instance

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} - \eta_t \nabla R\left(\vec{w}^{(t)}\right) - \eta_t C \frac{1}{|S_t|} \sum_{(x,y)\in S_t} \nabla L\left(\vec{w}^{(t)} \cdot \vec{x}, y\right)$$



Practical Challenges

Randomly choosing an instance or mini-batch

- In practice: Shuffle and choose without replacement
- Theory: i.i.d, with replacement

Beyond the linear $\vec{w} \cdot \vec{x}$?

- Convex Versus Non-Convex

Setting the step size and mini-batch size?

- Parallel computation.
- Generalization?
- Convex versus Non-convex