COMPUTER SCIENCE
INDIANA UNIVERSITY
School of Informatics and Computing
Bloomington

Stochastic optimization

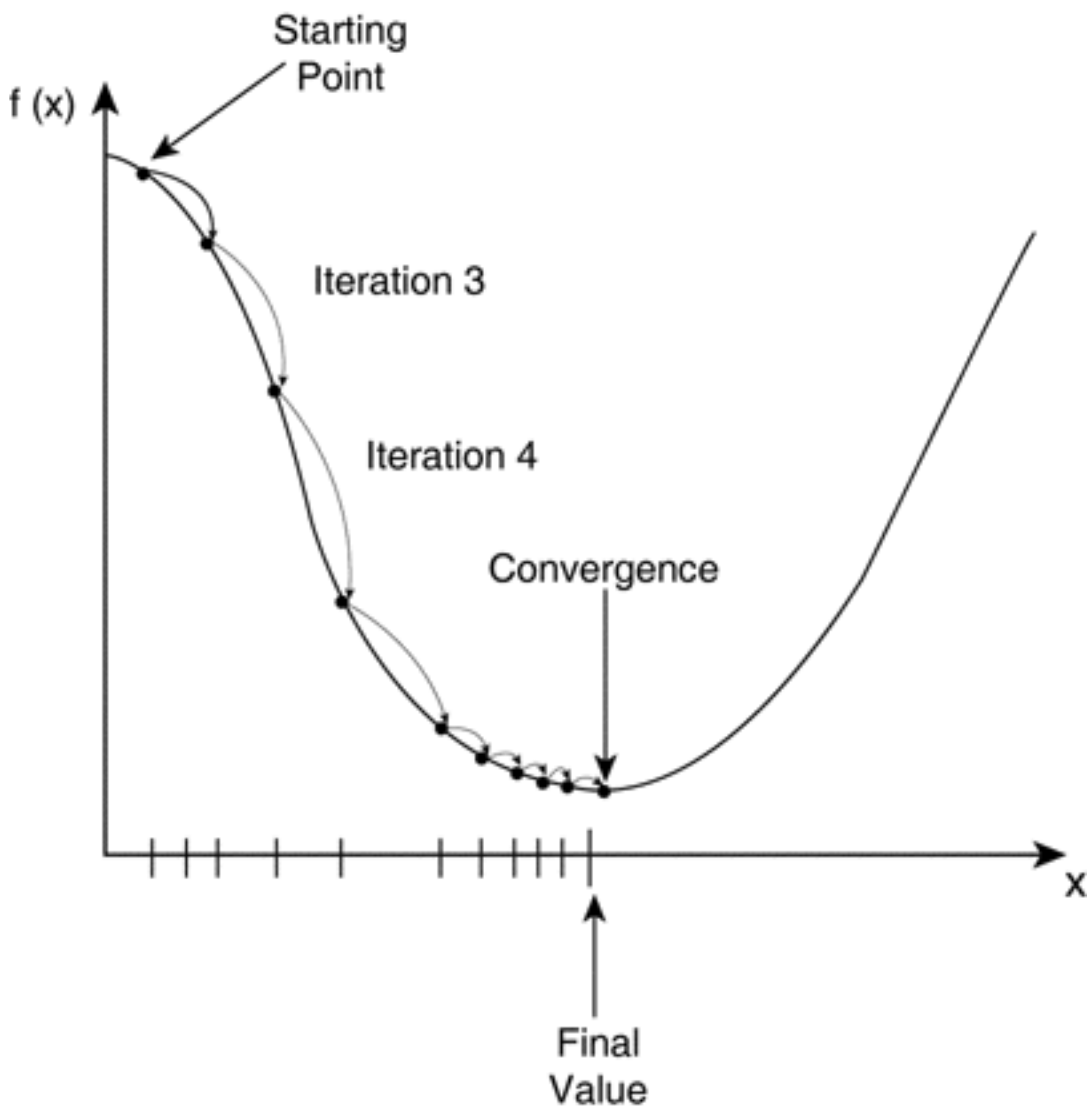
Reminders/Comments

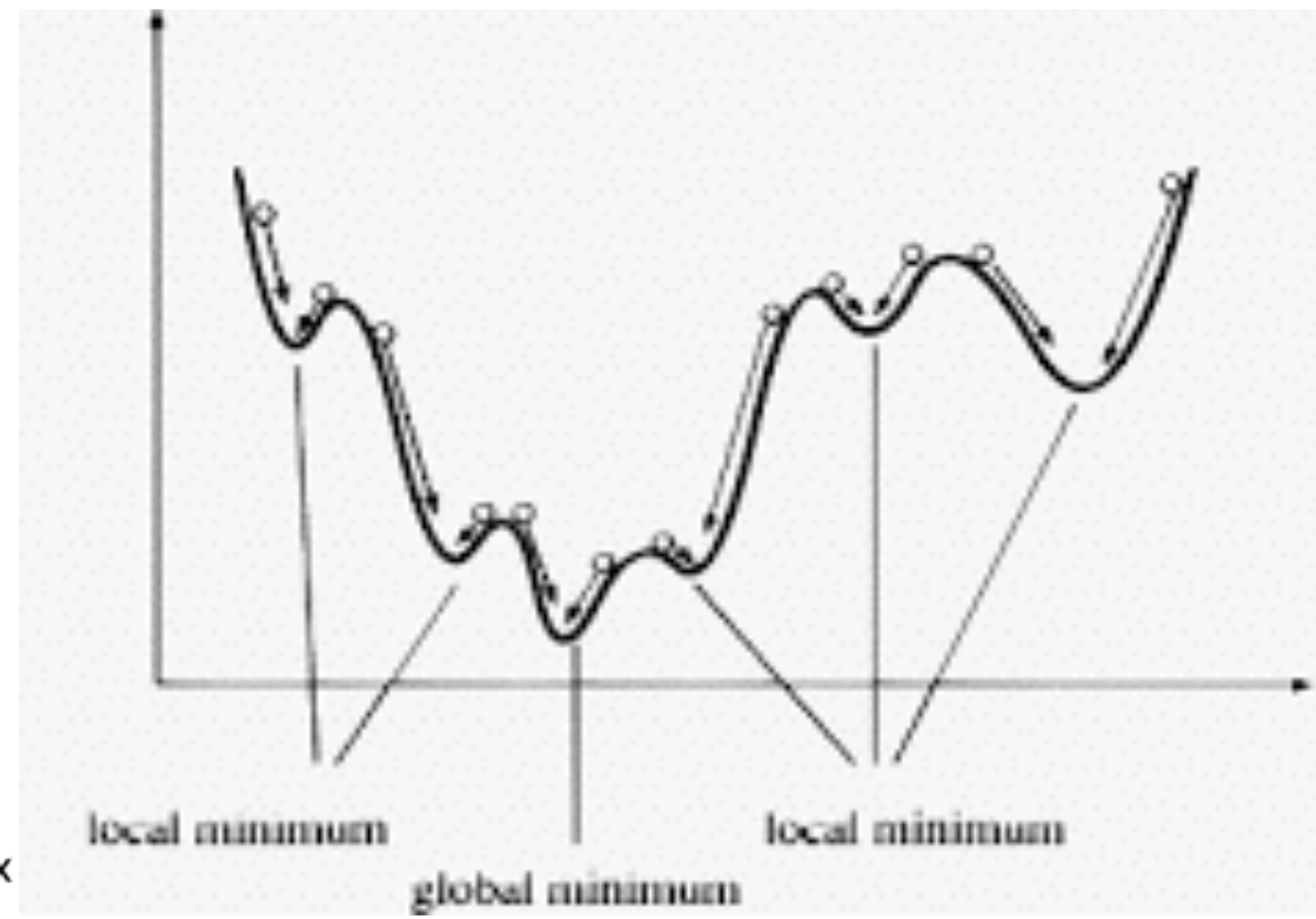
- Notes will be updated on Wednesday; this includes
  - chapters on classification and representations
  - appendix chapters on optimization
- Current appendix linked to on the notes page, with second-order optimization described
- Feel free to also give comments about any topics that you find particularly confusing/difficult in the anonymous survey

Thought question

- Can we have an initial Parameter estimation without ‘experience’? And later incrementally improve the estimation/model with experience?
  - Yes!
  - For a point-estimate (which is what we’ve been doing), can start with an initial guess for parameters you believe to be true and then use (stochastic) gradient descent to incorporate (new) samples
  - When we talk about Bayesian estimation, there we start with a distribution over parameters and update that distribution with samples

Gradient descent intuition





Convex function

Non-convex function

Gradient descent

Algorithm 1: Batch Gradient Descent$(E, \mathbf{X}, \mathbf{y})$

// A non-optimized, basic implementation of batch gradient descent
$\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$
err $\leftarrow \infty$
tolerance $\leftarrow 10e^{-4}$
$\alpha \leftarrow 0.1$
while $|E(\mathbf{w}) - \text{err}| >$ tolerance **do**
    // The step-size α should be chosen by line-search
    $\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla E(\mathbf{w}) = \mathbf{w} - \alpha \mathbf{X}^\top (\mathbf{X}\mathbf{w} - \mathbf{y})$
end while
return $\mathbf{w}$

Recall: for error function $E(\mathbf{w})$ goal is to solve $\nabla E(\mathbf{w}) = \mathbf{0}$

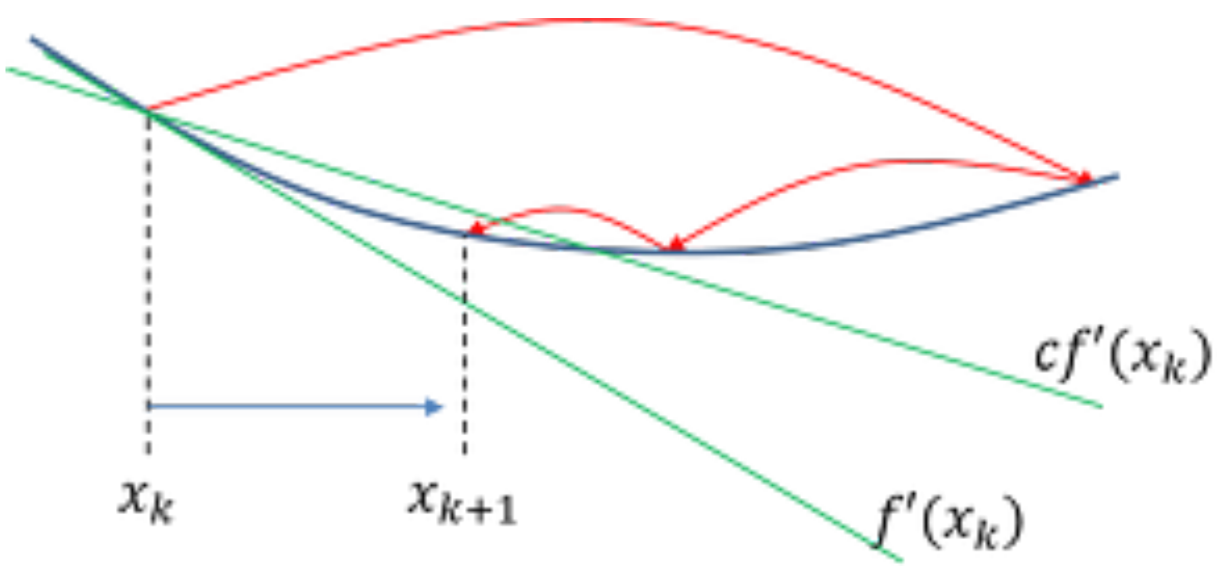
Line search

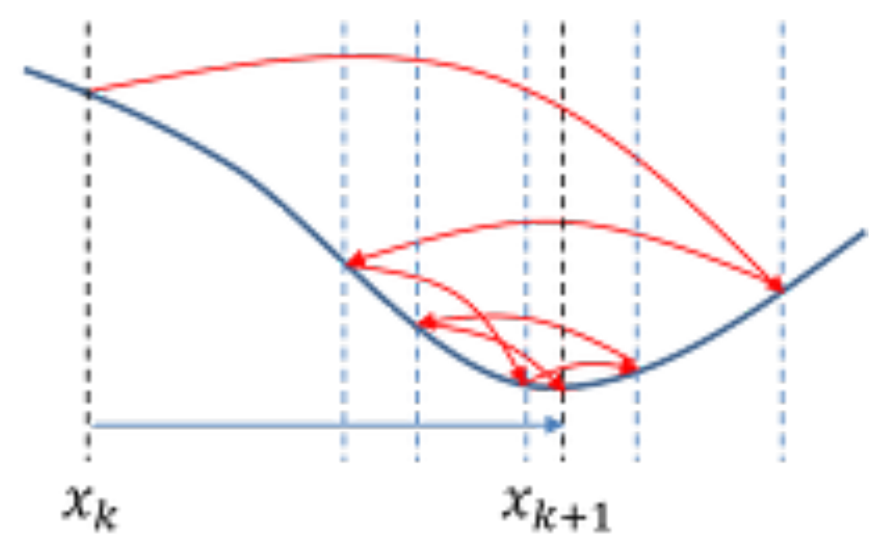
Want step-size such that

$$\alpha = \arg\min_{\alpha} E(\mathbf{w} - \alpha \nabla E(\mathbf{w}))$$

Backtracking line search:

1. Start with relatively large α (say $\alpha = 1$)
2. Check if $E(\mathbf{w} - \alpha \nabla E(\mathbf{w}) < E(\mathbf{w})$
3. If yes, use that α
4. Otherwise, decrease α (e.g., $\alpha = \alpha/2$), and check again





Second-order optimization

- If have Hessian, can use second-order techniques
  - Mostly removes the need for a step-size parameter, and/or makes the choice of this parameter much less sensitive
- For certain situations, computing the Hessian is too expensive (in space and computation) and so first-order methods are used OR quasi-second order (LBFGS)
  - e.g., huge number of features
  - e.g., stochastic gradient descent

Second-order: Newton-Raphson for single-variate setting

A function $f(x)$ in the neighborhood of point x_0, can be approximated using the Taylor series as

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n,$$

where $f^{(n)}(x_0)$ is the n-th derivative of function $f(x)$ evaluated at point x_0.

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2}(x - x_0)^2 f''(x_0).$$

$$f'(x) \approx f'(x_0) + (x - x_0)f''(x_0) = 0.$$

Solving this equation for x gives us

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}.$$

Iterating gives us:

$$x^{(i+1)} = x^{(i)} - \frac{f'(x^{(i)})}{f''(x^{(i)})}.$$

Second-order: Newton-Raphson for multi-variate setting

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T \cdot (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \cdot H_{f(\mathbf{x}_0)} \cdot (\mathbf{x} - \mathbf{x}_0),$$

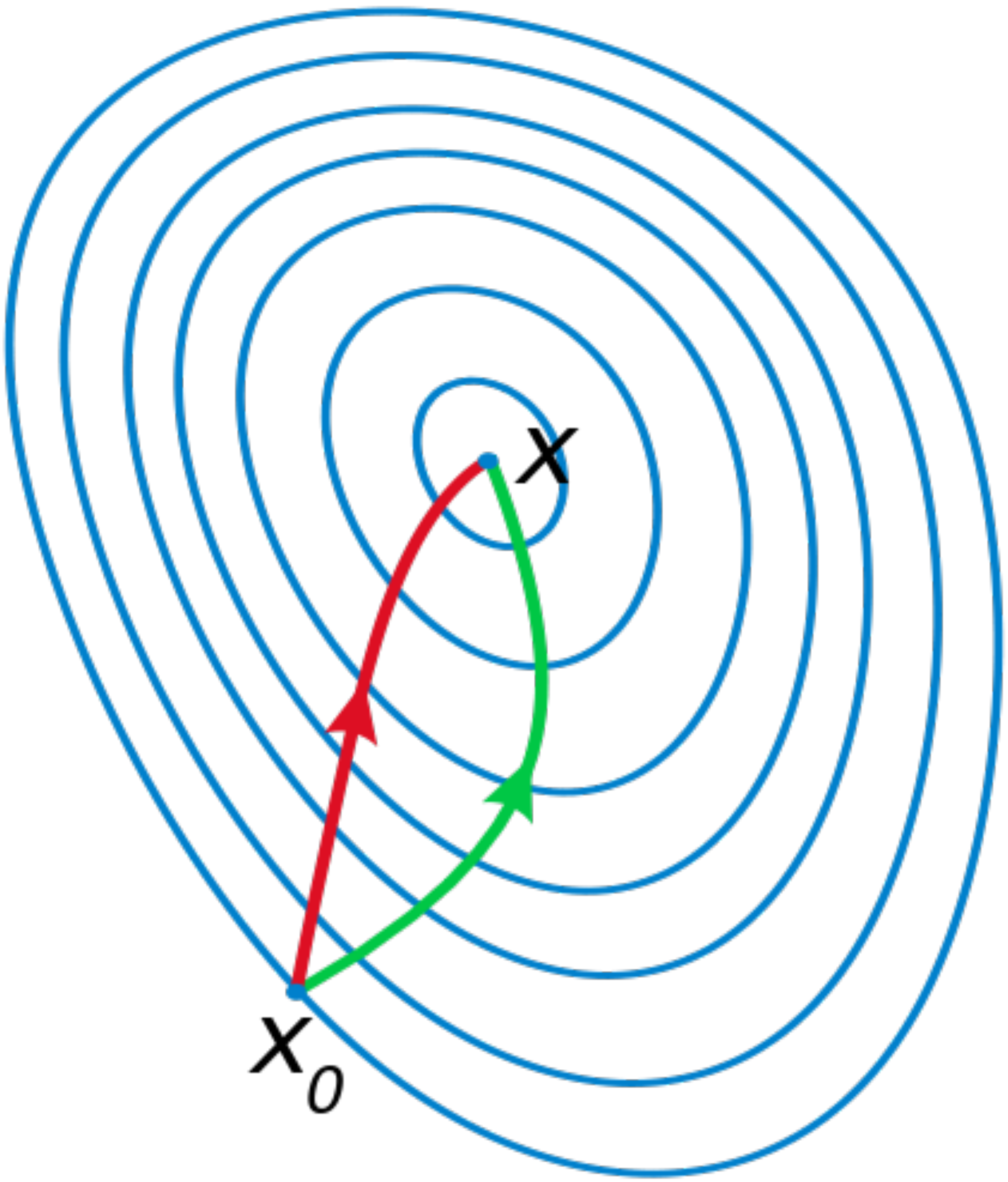
$$\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_k} \right)$$

$$H_{f(\mathbf{x})} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_k} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \\ \vdots & & \ddots & \\ \frac{\partial^2 f}{\partial x_k \partial x_1} & & & \frac{\partial^2 f}{\partial x_k^2} \end{bmatrix}$$

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \left(H_{f(\mathbf{x}^{(i)})} \right)^{-1} \cdot \nabla f(\mathbf{x}^{(i)}),$$

Intuition for first and second order

- Locally approximate function at current point
- For first order, locally approximate as linear and step in the direction of the minimum of that linear function
- For second order, locally approximate as quadratic and step in the direction of the minimum of that quadratic function
  - a quadratic approximation is more accurate
- What happens if the true function is quadratic?



$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \left(H_{f(\mathbf{x}^{(i)})}\right)^{-1} \cdot \nabla f(\mathbf{x}^{(i)}),$$

Stochastic gradient descent

Algorithm 2: Stochastic Gradient Descent($E, \mathbf{X}, \mathbf{y}$)

1: $\mathbf{w} \leftarrow$ random vector in $\mathbb{R}^d$
2: **for** $t = 1, \ldots, n$ **do**
3: // For some settings, we need the step-size α_t to decrease with time
4: $\mathbf{w} \leftarrow \mathbf{w} - \alpha_t \nabla E_t(\mathbf{w}) = \mathbf{w} - \alpha_t(\mathbf{x}_t^\top \mathbf{w} - y_t)\mathbf{x}_t$
5: **end for**
6: **return** $\mathbf{w}$

For batch error: $\hat{E}(\mathbf{w}) = \sum_{t=1}^{n} E_t(\mathbf{w})$

e.g., $E_t(\mathbf{w}) = (\mathbf{x}_t^\top \mathbf{w} - y_t)^2$

$\hat{E}(\mathbf{w}) = \sum_{t=1}^{n} E_t(\mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$

$\nabla \hat{E}(\mathbf{w}) = \sum_{t=1}^{n} \nabla E_t(\mathbf{w})$

$E(\mathbf{w}) = \int_{\mathcal{X}} \int_{\mathcal{Y}} f(\mathbf{x}, y)(\mathbf{x}^\top \mathbf{w} - y)^2 dy d\mathbf{x}$

- Stochastic gradient descent (stochastic approximation) minimizes with an unbiased sample of the gradient $\mathbb{E}[\nabla E_t(\mathbf{w})] = \nabla E(\mathbf{w})$

Stochastic gradient descent

$$
\begin{aligned}
\mathbb{E}\left[\frac{1}{n}\nabla\hat{E}(\mathbf{w})\right] &= \frac{1}{n}\mathbb{E}\left[\sum_{i=1}^{n}\nabla E_i(\mathbf{w})\right] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\nabla E_i(\mathbf{w})] \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[\nabla E(\mathbf{w})] \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla E(\mathbf{w}) \\
&= \nabla E(\mathbf{w})
\end{aligned}
$$

Stochastic gradient descent

- Can also approximate gradient with more than one sample (e.g., mini-batch), as long as $\mathbb{E}[\nabla E_t(\mathbf{w})] = \nabla E(\mathbf{w})$

- Proof of convergence and conditions on step-size: Robbins-Monro ("A Stochastic Approximation Method", Robbins and Monro, 1951)

- For more, consider taking my course next year called "Stochastic optimization for machine learning"
  - will also include algorithms for temporal setting: time series and reinforcement learning

Whiteboard

- **Exercise**: derive an algorithm to compute the solution to l1-regularized linear regression (i.e., MAP estimation with a Gaussian likelihood p(y | x, w) and Laplace prior)
  - First write down the Laplacian
  - Then write down the MAP optimization
  - Then determine how to solve this optimization
- Generalized linear models