# ENM 3600: Data-driven Modeling and Probabilistic Scientific Computing

## *Lecture #8: Variational inference*

Penn
University *of* Pennsylvania

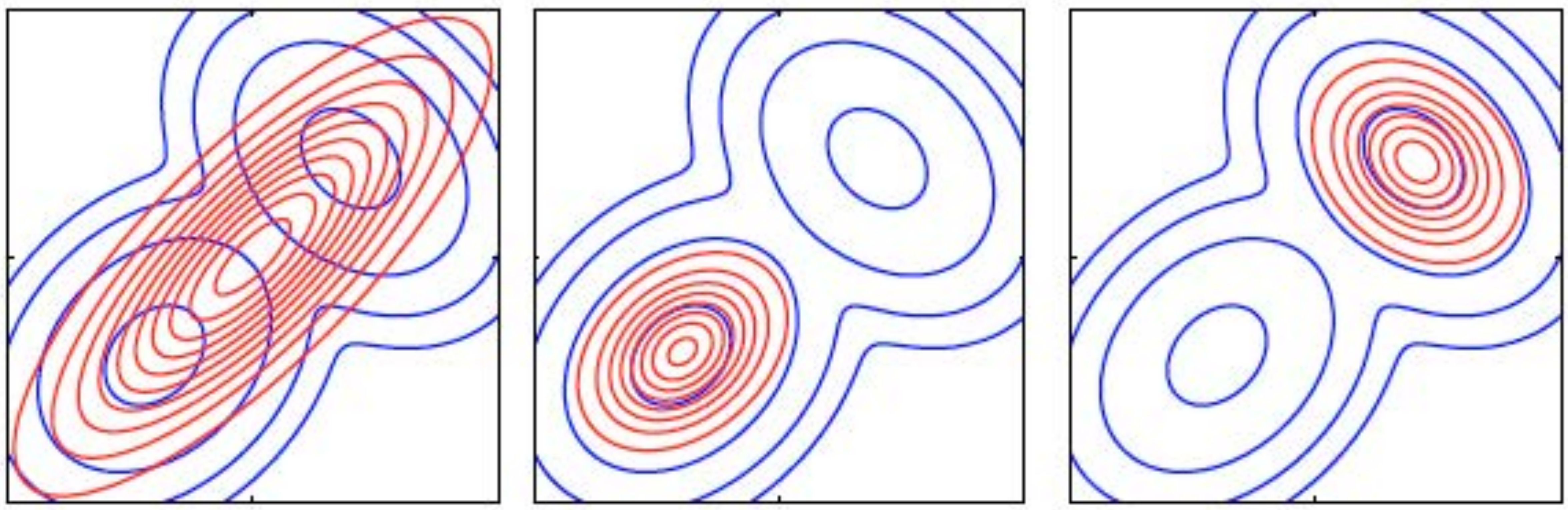
# Variational inference

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta|)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|\theta|)p(\theta)}{\int p(\mathcal{D}|\theta|)p(\theta)d\theta}$$

- Variational inference provides a computational framework for approximate Bayesian inference.
- The idea is that we'll approximate the posterior distribution with a family of distributions that is easy to work with.
- It will provide us with a set of tools for transforming the sampling problem (integration) to an optimization problem, that can be scaled to large models (i.e. with many parameters) and large data-sets.
- It also tends to favor approximations that underestimate the variance, and it usually will result in approximate distributions that get the means right but underestimate the variance.

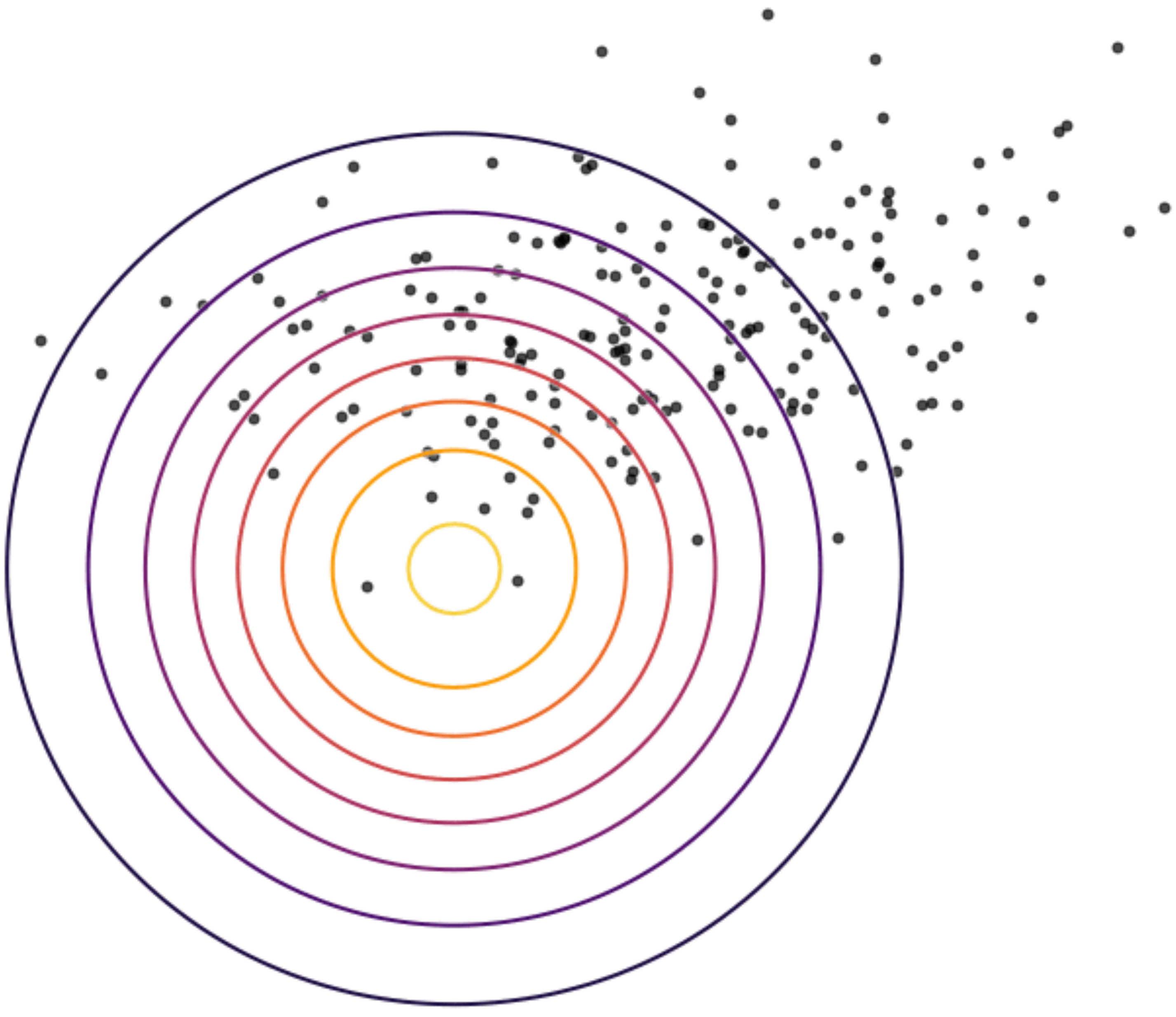
# Variational inference

$\mathbb{KL}[q_\phi(x)||p(x)]$ $\mathbb{KL}[p(x)||q_\phi(x)]$ $\mathbb{KL}[p(x)||q_\phi(x)]$



KL Reverse KL Reverse KL

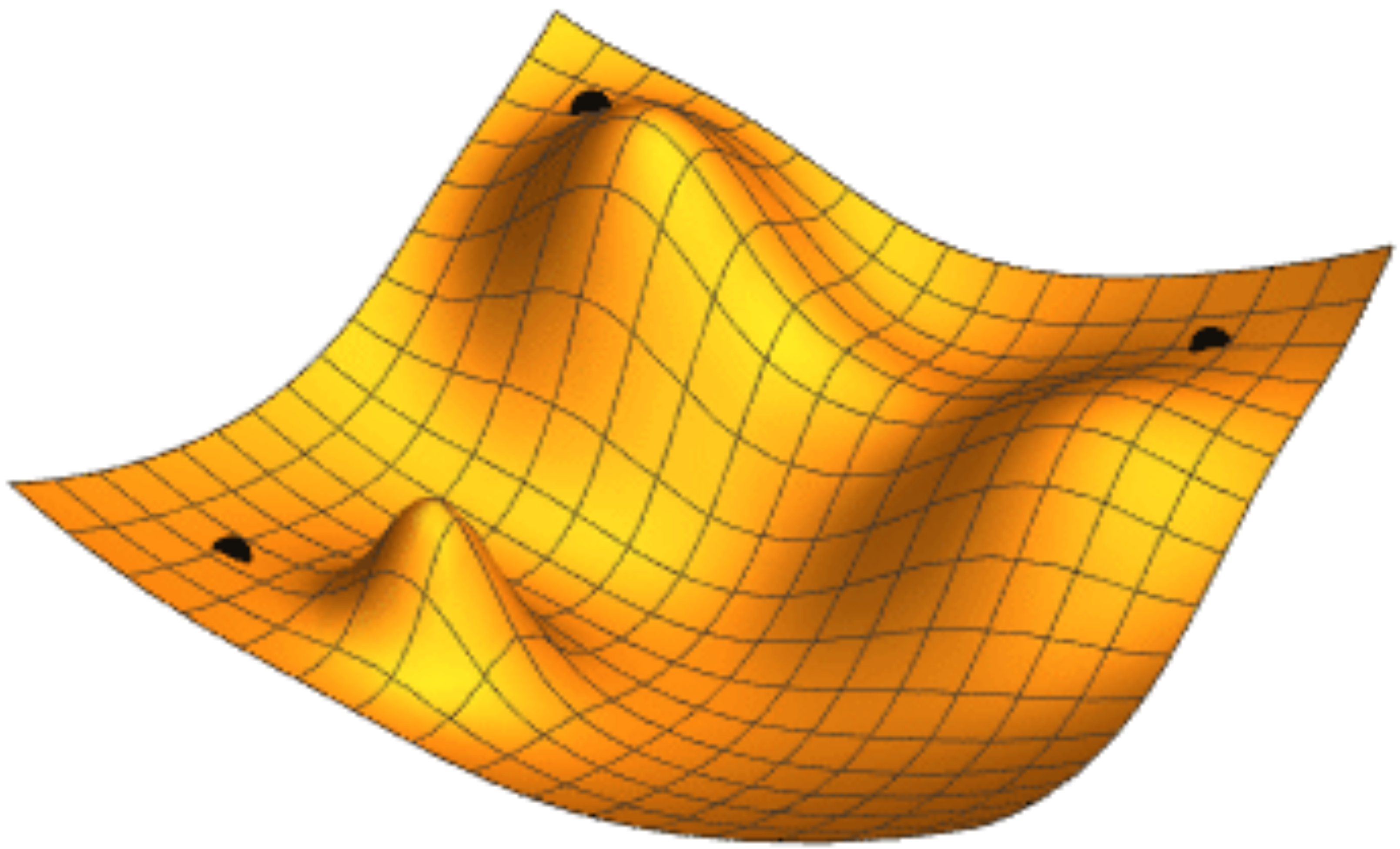
# Variational inference



# Gradient descent

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} J(\boldsymbol{\theta})$$

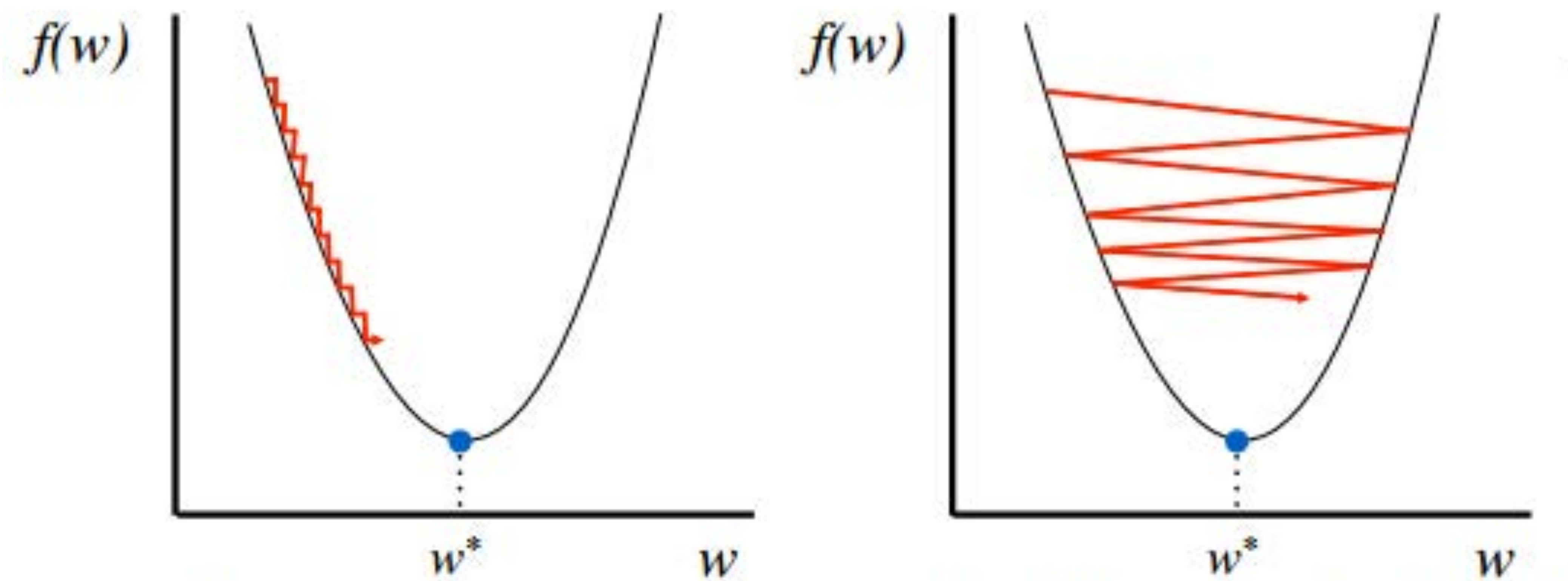
$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



# Gradient descent

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Effect of the learning rate

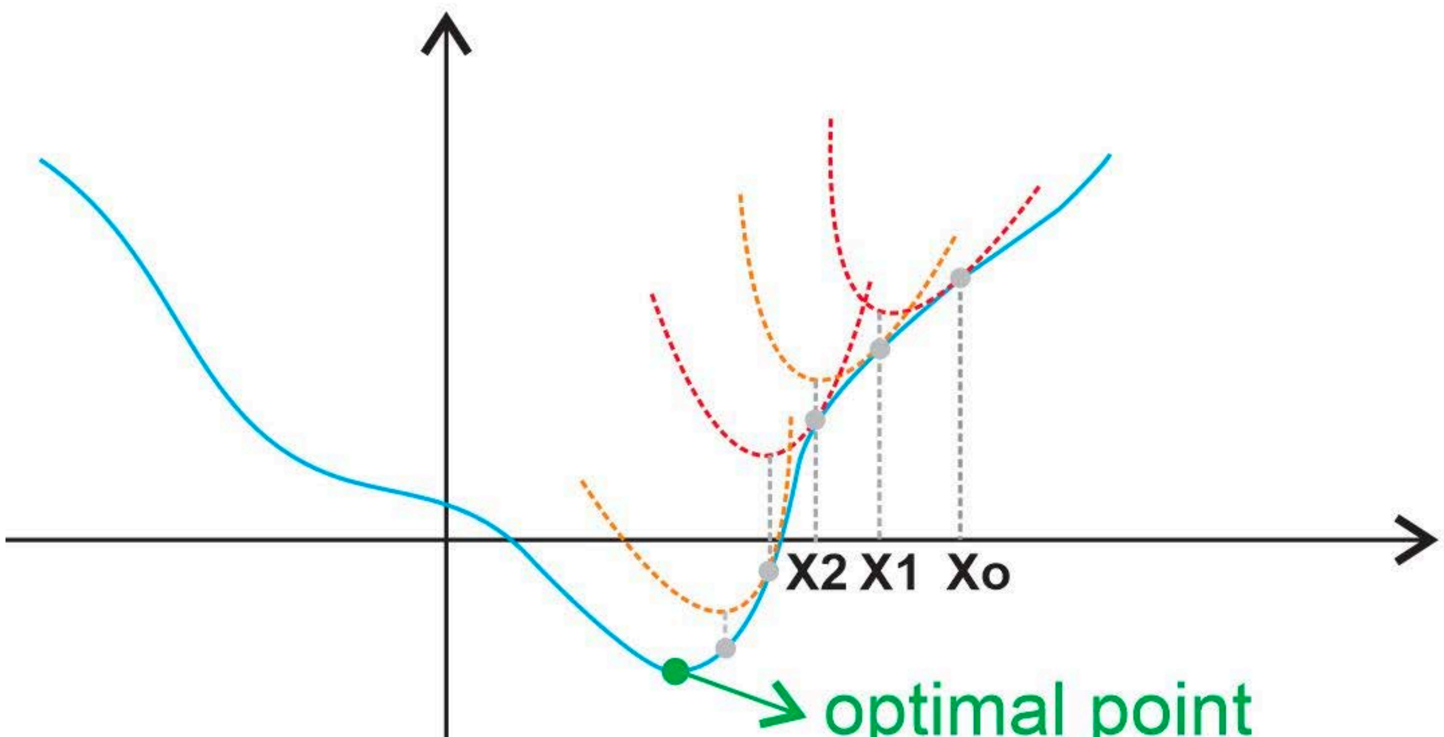



Too small: converge very slowly

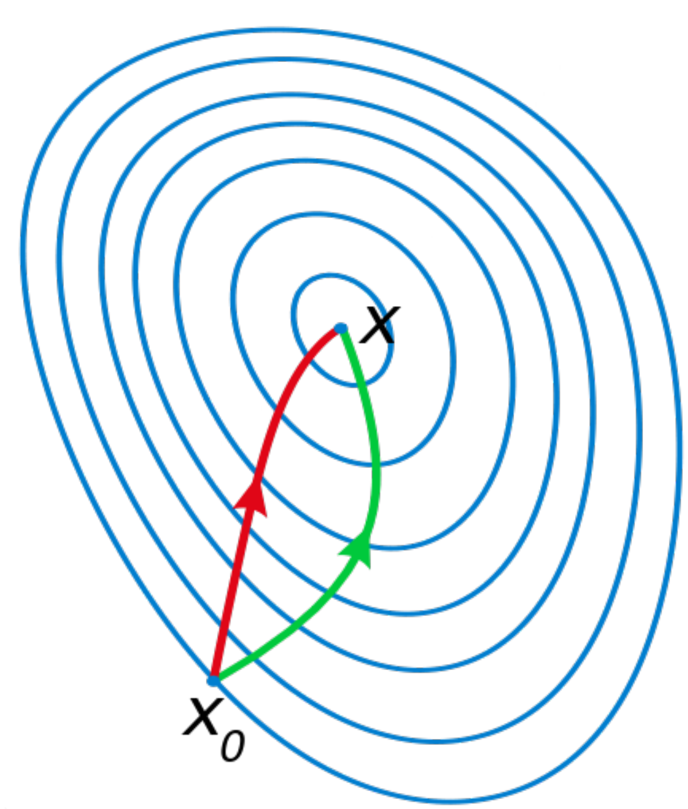
Too big: overshoot and even diverge

# Hessian

$$\nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_1^2} & \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_1 \partial \theta_2} & \cdots & \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_1 \partial \theta_n} \\ \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_2^2} & \cdots & \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_2 \partial \theta_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_d \partial \theta_1} & \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_d \partial \theta_2} & \cdots & \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_d^2} \end{bmatrix}$$




Local quadratic approximation
of the loss == Newton's method




Gradient descent vs Newton

# BFGS

WIKIPEDIA
The Free Encyclopedia

Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

## Broyden–Fletcher–Goldfarb–Shanno algorithm

From Wikipedia, the free encyclopedia

**This article has multiple issues.** Please help **improve it** or discuss these issues on the **talk page**. [hide]

*(Learn how and when to remove these template messages)*

- This article **may be too technical for most readers to understand**. Please help improve it to make it understandable to non-experts, without removing the technical details. *(September 2010)*
- This article **needs additional citations for verification**. *(March 2016)*

In numerical optimization, the **Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm** is an iterative method for solving unconstrained nonlinear optimization problems.[1]

The BFGS method belongs to quasi-Newton methods, a class of hill-climbing optimization techniques that seek a stationary point of a (preferably twice continuously differentiable) function. For such problems, a necessary condition for optimality is that the gradient be zero. Newton's method and the BFGS methods are not guaranteed to converge unless the function has a quadratic Taylor expansion near an optimum. However, BFGS has proven to have good performance even for non-smooth optimizations.[2]

In quasi-Newton methods, the Hessian matrix of second derivatives doesn't need to be evaluated directly. Instead, the Hessian matrix is approximated using updates specified by gradient evaluations (or approximate gradient evaluations). Quasi-Newton methods are generalizations of the secant method to find the root of the first derivative for multidimensional problems. In multi-dimensional problems, the secant equation does not specify a unique solution, and quasi-Newton methods differ in how they constrain the solution. The BFGS method is one of the most popular members of this class.[3] Also in common use is L-BFGS, which is a limited-memory version of BFGS that is particularly suited to problems with very large numbers of variables (e.g., >1000). The BFGS-B[4] variant handles simple box constraints.

The algorithm is named after Charles George Broyden, Roger Fletcher, Donald Goldfarb and David Shanno.

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

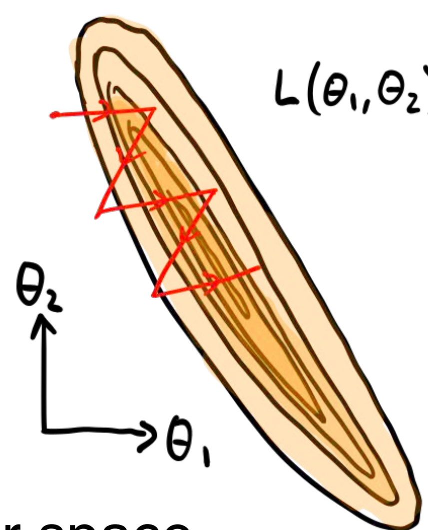
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools

What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item

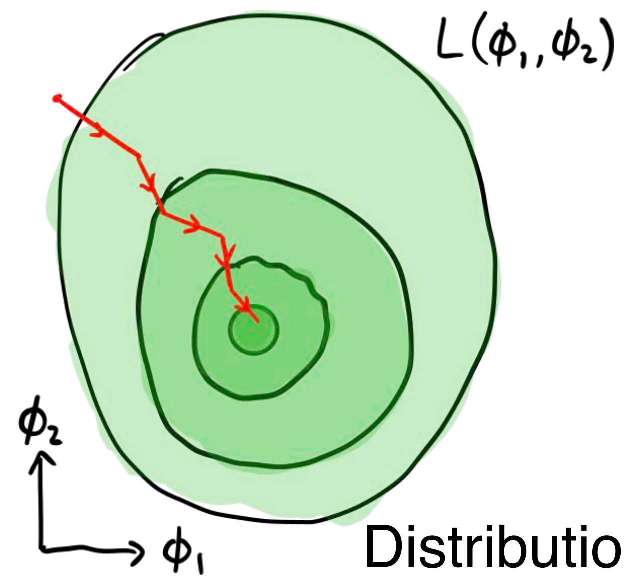
# Gradient descent vs natural gradient descent

Motivation: If our objective is to minimize the loss function (maximizing the likelihood), then it is natural that we taking step in the space of all possible likelihoods, realizable by the parameters θ. As the likelihood function itself is a probability distribution, we call this space distribution space. Thus it makes sense to take the steepest descent direction in this distribution space instead of parameter space.




Parameter space

$\Rightarrow$




Distribution space

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\theta_{n+1} = \theta_n - \eta F^{-1} \nabla \mathscr{L}(\theta)$$

Cramer-Rao bound: The inverse of the Fisher information is a lower bound on the variance of any unbiased estimator of $\theta$

Fisher Information Matrix == negative expected Hessian of log likelihood

$$\begin{aligned}
\mathrm{H}_{\mathrm{KL}[p(x|\theta)\,\|\,p(x|\theta')]} &= -\int p(x|\theta)\ \nabla^2_{\theta'} \log p(x|\theta')\big|_{\theta'=\theta}\, \mathrm{d}x \\
&= -\int p(x|\theta)\, \mathrm{H}_{\log p(x|\theta)}\, \mathrm{d}x \\
&= -\underset{p(x|\theta)}{\mathbb{E}}\left[\mathrm{H}_{\log p(x|\theta)}\right] \\
&= \mathbf{F}\,.
\end{aligned}$$

# Probabilistic programming

http://mc-stan.org/

https://github.com/pymc-devs/pymc3

http://edwardlib.org/

https://github.com/uber/pyro