CSE 250A. Principles of AI

Probabilistic Reasoning and Decision-Making

Lecture 9 – Linear regression and least-squares

Lawrence Saul
Department of Computer Science and Engineering
University of California, San Diego

Fall 2021

Outline

Learning in BNs with discrete nodes

- **ML estimation for complete data:**

$$P_{\mathrm{ML}}(X_i\!=\!x|\mathrm{pa}_i\!=\!\pi) \;=\; \frac{\mathrm{count}(X_i\!=\!x, \mathrm{pa}_i\!=\!\pi)}{\sum_{x'} \mathrm{count}(X_i\!=\!x', \mathrm{pa}_i\!=\!\pi)}$$

- **For nodes with parents:**

$$P_{\mathrm{ML}}(X_i\!=\!x|\mathrm{pa}_i\!=\!\pi) \;=\; \frac{\mathrm{count}(X_i\!=\!x, \mathrm{pa}_i\!=\!\pi)}{\mathrm{count}(\mathrm{pa}_i\!=\!\pi)}$$

- **For root nodes:**

$$P_{\mathrm{ML}}(X_i\!=\!x|) \;=\; \frac{\mathrm{count}(X_i\!=\!x)}{T}$$

Markov models for statistical language processing

- ***n*-gram models of word sequences:**

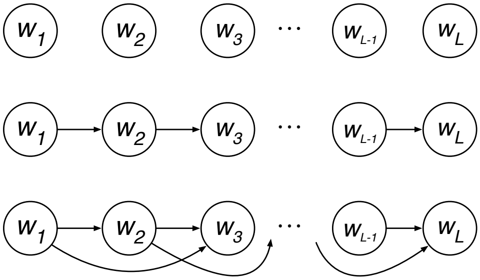
$$P(w_1, w_2, \ldots, w_L) = \prod_{\ell} P(w_\ell | \underbrace{w_{\ell-(n-1)}, \ldots, w_{\ell-1}}_{\text{previous words}}))$$

- **As belief networks:**

$n = 1$ **unigram**

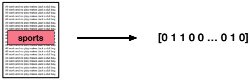
$n = 2$ **bigram**

$n = 3$ **trigram**



Naive Bayes model for document classification

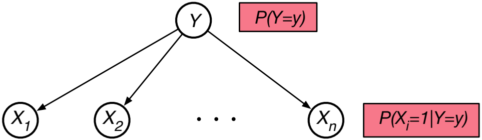
- **Random variables**



$Y \in \{1, 2, \ldots, m\}$ topic of document

$X_i \in \{0, 1\}$ i^{th} word appears?

- **Belief network**



- **Naive Bayes assumption**

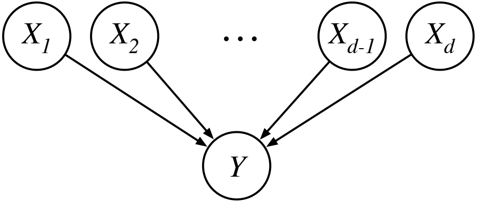
$$P(X_1, \ldots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y)$$

Outline

1. **Review**
2. $\boxed{\text{Linear regression}}$
3. **Numerical optimization**

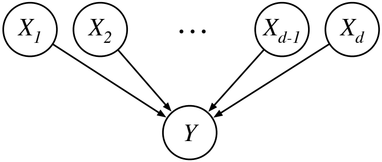
Learning with parametric models

If the parent nodes are real-valued, then it is no longer possible to enumerate a conditional probability table.



How to predict Y from real-valued parents $\vec{X} \in \mathbb{R}^d$?

Gaussian conditional distribution

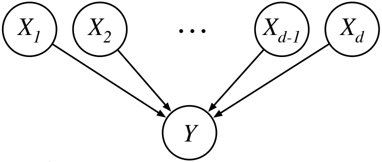


Suppose $Y \in \mathbb{R}$ is a real-valued random variable.
Then we can use a **Gaussian conditional distribution**:

$$P(y|\vec{x}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y-\vec{w}\cdot\vec{x})^2}{2\sigma^2}\right\}$$

How to learn the variance σ^2 and weights $\vec{w} = (w_1, w_2, \ldots, w_d)$?
This is the problem of **linear regression**.

Learning from complete data



- **Training examples**

$$\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_T, y_T)\}$$

T complete pairs of inputs and outputs

- **Conditional likelihood for IID data**

$$P(y_1, y_2, \ldots, y_T | \vec{x}_1, \ldots, \vec{x}_T) = \prod_{t=1}^{T} P(y_t | \vec{x}_t)$$

Note: if the parents are always observed, we needn't model $P(\vec{x})$.

Computing the log conditional likelihood

$$\begin{aligned}
\mathcal{L}(\vec{w}, \sigma^2) &= \log P(y_1, y_2, \ldots, y_T | \vec{x}_1, \ldots, \vec{x}_T) \\
&= \log \prod_{t=1}^{T} P(y_t | \vec{x}_t) \qquad \boxed{\textbf{IID}} \\
&= \sum_{t=1}^{T} \log P(y_t | \vec{x}_t) \\
&= \sum_{t=1}^{T} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(y_t - \vec{w} \cdot \vec{x}_t)^2}{2\sigma^2} \right\} \right] \\
&= -\frac{1}{2} \sum_{t=1}^{T} \left[\log(2\pi\sigma^2) + \frac{(y_t - \vec{w} \cdot \vec{x}_t)^2}{\sigma^2} \right]
\end{aligned}$$

Interpreting the log conditional likelihood

$$\boxed{\mathcal{L}(\vec{w}, \sigma^2) \;=\; -\frac{1}{2}\sum_{t=1}^{T}\left[\log(2\pi\sigma^2) \;+\; \frac{(y_t - \vec{w}\cdot\vec{x}_t)^2}{\sigma^2}\right]}$$

Consider the weights $\vec{w}$ that maximize $\mathcal{L}(\vec{w}, \sigma^2)$.
The same weights also minimize the mean squared error:

$$\mathcal{E}(\vec{w}) \;=\; \frac{1}{T}\sum_{t}(y_t - \vec{w}\cdot\vec{x}_t)^2$$

Maximum likelihood linear regression is simply the least-squares problem for a linear fit.

Maximizing the log conditional likelihood

$$\mathcal{L}(\vec{w}, \sigma^2) = -\frac{1}{2}\sum_{t=1}^{T}\left[\log(2\pi\sigma^2) + \frac{(y_t - \vec{w}\cdot\vec{x}_t)^2}{\sigma^2}\right]$$

- **Computing the partial derivatives:**

$$\frac{\partial\mathcal{L}}{\partial w_\alpha} = \sum_t -\frac{1}{2\sigma^2}\cdot 2\cdot(y - \vec{w}\cdot\vec{x}_t)(-x_{\alpha t}) \quad \boxed{\textbf{chain rule}}$$

$$= \frac{1}{\sigma^2}\sum_t (y - \vec{w}\cdot\vec{x}_t)x_{\alpha t}$$

- **Solving for where they vanish:**

$$\sum_t y_t x_{\alpha t} = \sum_t (\vec{w}\cdot\vec{x}_t)x_{\alpha t} \quad \left\{\begin{array}{l} d \text{ equations } (\alpha = 1, 2, \ldots d) \\ d \text{ unknowns } (w_1, w_2, \ldots, w_d) \end{array}\right.$$

Maximizing the log conditional likelihood (con't)

- **System of linear equations:**

$$\begin{aligned}
\sum_{t=1}^{T} y_t x_{\alpha t} &= \sum_{t=1}^{T} (\vec{w} \cdot \vec{x}_t) x_{\alpha t} \\
&= \sum_{t=1}^{T} \left(\sum_{\beta=1}^{d} w_\beta x_{\beta t} \right) x_{\alpha t} && \boxed{\textbf{dot product}} \\
&= \sum_{\beta=1}^{d} \left(\sum_{t=1}^{T} x_{\alpha t} x_{\beta t} \right) w_\beta && \boxed{\textbf{reorder sums}}
\end{aligned}$$

- **Matrix-vector notation:**

$$A_{\alpha\beta} = \textstyle\sum_t x_{\alpha t} x_{\beta t} \quad \boxed{d \times d \text{ matrix}} \quad \mathbf{A} = \textstyle\sum_t \vec{x}_t \vec{x}_t^\top$$

$$b_\alpha = \textstyle\sum_t y_t x_{\alpha t} \quad \boxed{d \times 1 \text{ vector}} \quad \vec{b} = \textstyle\sum_t y_t \vec{x}_t$$

Maximizing the log conditional likelihood (con't)

- **Solving the system of linear equations:**

$$\left.\begin{aligned}\mathbf{A} &= \textstyle\sum_t \vec{x}_t \vec{x}_t^\top \\ \vec{b} &= \textstyle\sum_t y_t \vec{x}_t\end{aligned}\right\} \quad \vec{b} = \mathbf{A}\vec{w} \quad \Longrightarrow \quad \boxed{\vec{w}_{\mathrm{ML}} = \mathbf{A}^{-1}\vec{b}}$$

  Note that $\mathbf{A}$ and $\vec{b}$ can be computed in one pass through the training data.

- **Ill-conditioned problems:**

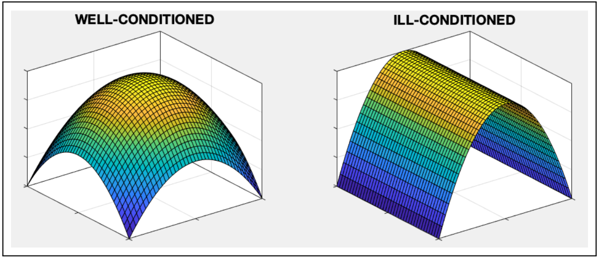
  This solution assumes that the matrix $\mathbf{A}$ is invertible.
  This is true as long as the inputs $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T\}$ span $\mathbb{R}^d$.

- **Failure modes:**

  The input dimensionality d exceeds the number of examples T.
  The inputs lie in (or very near) a proper subspace of $\mathbb{R}^d$.

More on ill-conditioned problems

Log conditional likelihood



When the system of linear equations is ill-conditioned, there remains a (unique) **minimum norm** solution:

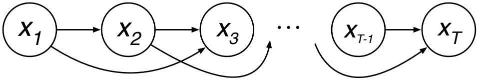
$$\textbf{Minimize} \quad \|\vec{w}\| \quad \textbf{such that} \quad \frac{\partial \mathcal{L}}{\partial \vec{w}} = \mathbf{0}.$$

Application

- **Time series prediction**

  Let $\{x_1, x_2, \ldots, x_T\}$ be a time series with $x_t \in \mathbb{R}$.
  How well can we predict the future from the past?

- **Linear predictive model**



$$P(x_t | x_1, \ldots, x_{t-1}) \;=\; \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} \left(x_t - \sum_{\alpha=1}^{d} w_\alpha x_{t-\alpha} \right)^2 \right\}$$

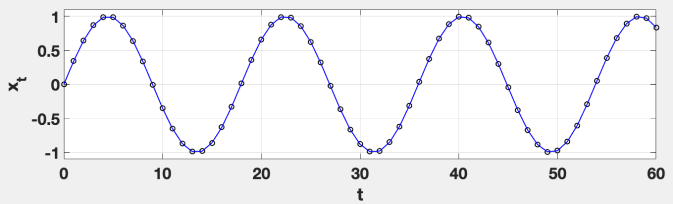
- **Discrete vs continuous**

discrete	HW 4.3	*n*-gram models of word sequences
continuous	HW 4.4	linear prediction of stock prices

Test your understanding

Q: If x_t is a linear combination of (say) x_{t-1} and x_{t-2}, is x_t a linear function of the time t?

A: **No!** As a counterexample, consider $x_t = \sin(\omega t)$.



A little trigonometry shows that this can be **perfectly** predicted by a linear model:

$$x_t \;=\; (2\cos\omega)\,x_{t-1} \;-\; x_{t-2}$$

Outline

1. **Review**
2. **Linear regression**
3. **Numerical optimization**

Optimization

How to maximize (or minimize) a multivariable function $f(\vec{\theta})$ over $\vec{\theta} = (\theta_1, \theta_2, \ldots, \theta_d) \in \mathbb{R}^d$?

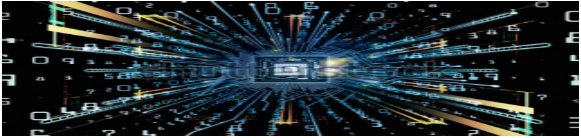
1. **Analytically**

   Compute the gradient and solve for where it vanishes:

$$\nabla f \;=\; \left(\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2}, \ldots, \frac{\partial f}{\partial \theta_d}\right) \;=\; (0, 0, \ldots, 0)$$

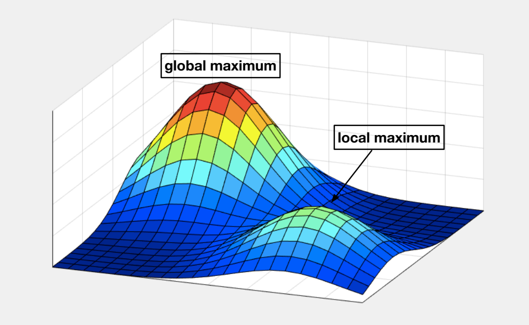
   This is easy for quadratic functions but *hard in general*.

2. **Numerically** (**or algorithmically**)



Hillclimbing methods

Hillclimbing methods are based on **iterative local search** for a local or global maximum of $f(\vec{\theta})$.



Given some estimate $\vec{\theta}_n$ at the n^{th} iteration, can you derive an improved estimate $\vec{\theta}_{n+1}$ with $f(\vec{\theta}_{n+1}) > f(\vec{\theta}_n)$?

Hillclimbing methods

What we'll cover today:

1. Gradient ascent (or descent)
2. Newton's method

What we'll cover later:

3. Auxiliary function methods

   These updates are specialized for monotonic convergence.
   They exploit inequalities (such as $\mathrm{KL}(q, p) \geq 0$).

Gradient ascent (or descent)

- **Local search direction**

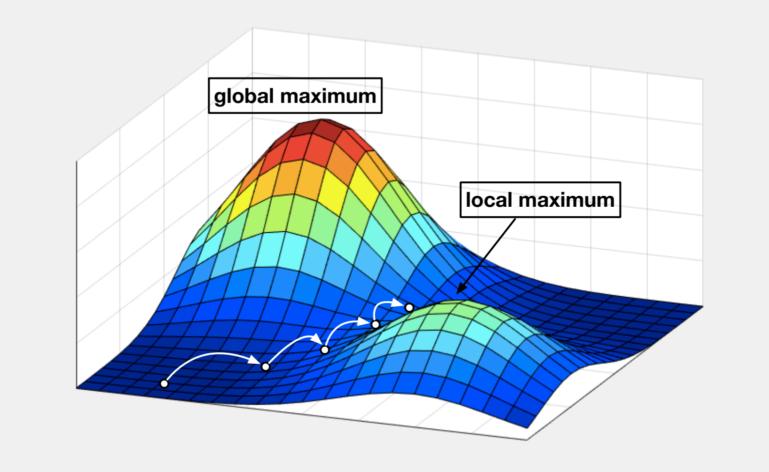
  Recall that the gradient $\nabla f = \partial f / \partial \vec{\theta}$ points in the direction of fastest increase in $f(\vec{\theta})$.

- **Iterative update**

$$\vec{\theta} \leftarrow \vec{\theta} \pm \eta_t \left(\frac{\partial f}{\partial \vec{\theta}} \right) \qquad \left(\begin{array}{l} + \text{ to ascend} \\ - \text{ to descend} \end{array} \right)$$

  The parameter $\eta_t > 0$ is called the **step size** or **learning rate**. Often it must be tuned for convergence.

Visualization



And so on, until reaching a local maximum.

Pros and cons of gradient ascent

Pros

- simple and universal hillclimbing procedure for any once-differentiable function
- asymptotic convergence to some local optimum (but only for a sufficiently small learning rate)

Cons

- sometimes tricky in practice to tune the learning rate
- no guarantee of monotonic convergence
- no guarantee of global optimality

Newton's method versus gradient ascent

In a nutshell:

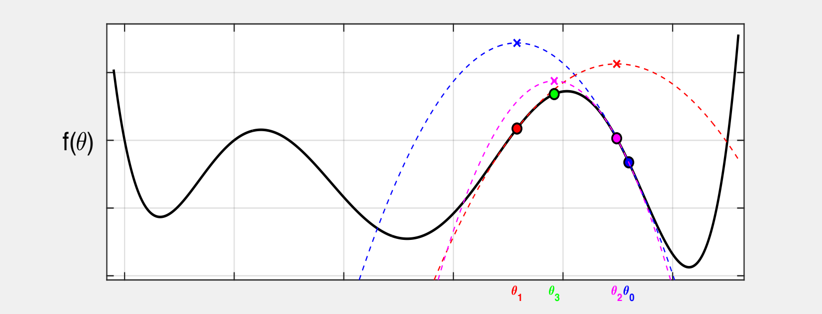
- Gradient ascent is based on the **linear** approximation

$$f(\vec{\theta}) \;\approx\; f(\vec{\theta}_0) \;+\; \frac{\partial f}{\partial \vec{\theta}} \cdot (\vec{\theta} - \vec{\theta}_0)$$

- Newton's method is based on the **quadratic** approximation

$$\begin{aligned} f(\vec{\theta}) \;&\approx\; f(\vec{\theta}_0) \;+\; \frac{\partial f}{\partial \vec{\theta}} \cdot (\vec{\theta} - \vec{\theta}_0) \\ &\quad + \frac{1}{2}(\vec{\theta} - \vec{\theta}_0)^\top \left(\frac{\partial^2 f}{\partial \vec{\theta}\, \partial \vec{\theta}^\top}\right) (\vec{\theta} - \vec{\theta}_0) \end{aligned}$$

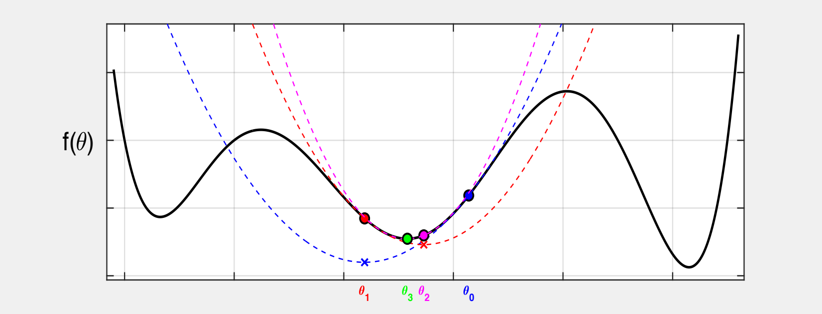
Newton's method in one dimension — maximizing $f(\theta)$



Repeat until convergence:

- Fit a parabola with matching 1st and 2nd derivatives.
- Move to the **maximum** of the parabola.

Newton's method in one dimension — minimizing $f(\theta)$



Repeat until convergence:

- Fit a parabola with matching 1st and 2nd derivatives.
- Move to the **minimum** of the parabola.

Update rule in one dimension

- **Quadratic fit from Taylor series:**

$$f(\theta) \;\approx\; f(\theta_0) \;+\; f'(\theta_0)(\theta - \theta_0) \;+\; \tfrac{1}{2} f''(\theta_0)(\theta - \theta_0)^2$$

- **Optimizing the right hand side:**

$$\begin{aligned} 0 \;&=\; \frac{d}{d\theta}\left[f(\theta_0) \;+\; f'(\theta_0)(\theta - \theta_0) \;+\; \tfrac{1}{2} f''(\theta_0)(\theta - \theta_0)^2 \right] \\ &=\; f'(\theta_0) \;+\; f''(\theta_0) \cdot (\theta - \theta_0) \end{aligned}$$

- **Update rule:**

$$\boxed{\theta_1 \;=\; \theta_0 \;-\; \frac{f'(\theta_0)}{f''(\theta_0)}} \qquad \textbf{or} \qquad \boxed{\theta \;\leftarrow\; \theta \;-\; \frac{f'(\theta)}{f''(\theta)}}$$

Update rule in d dimensions

SCALAR	MULTIVARIABLE	
$f(\theta)$	$f(\theta_1, \theta_2, \ldots, \theta_d)$	
$f'(\theta) = \frac{df}{d\theta}$	$\nabla f = \left(\frac{\partial f}{\partial \theta_1}, \frac{\partial f}{\partial \theta_2}, \ldots, \frac{\partial f}{\partial \theta_d}\right)$	**gradient**
$f''(\theta) = \frac{d^2 f}{d\theta^2}$	$H_{\alpha\beta} = \frac{\partial^2 f}{\partial \theta_\alpha \partial \theta_\beta}$	**Hessian**
$\theta \leftarrow \theta - \frac{f'(\theta)}{f''(\theta)}$	$\vec{\theta} \leftarrow \vec{\theta} - \mathbf{H}^{-1} \nabla f$	**update**

In the general update:

- The gradient ∇f is a d-dimensional vector.
- The Hessian $\mathbf{H}$ is a symmetric $d \times d$ matrix.
- $\mathbf{H}^{-1}$ denotes the **matrix inverse**.
- $\mathbf{H}^{-1}\nabla f$ denotes matrix-vector multiplication.

Pros and cons of Newton's method

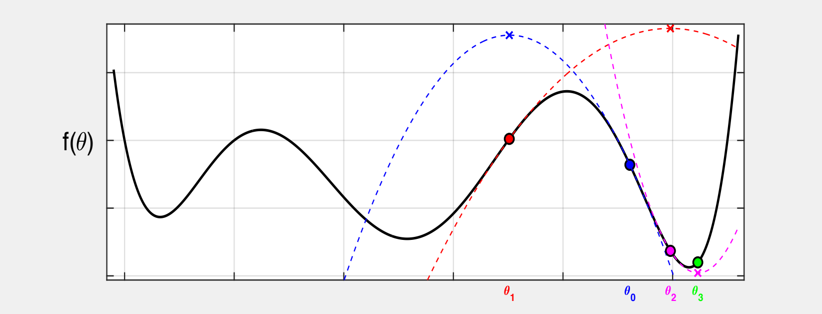
Pros

- simple and universal hillclimbing procedure for any twice-differentiable function
- rapid convergence (when it converges)
- no learning rate to tune

Cons

- expensive to compute Hessian matrix $O(d^2)$
- expensive to perform update $O(d^2)$
- no guarantee of global optimality
- unpredictable and/or unstable when initial estimate is poor

Newton's method can be unstable



The method often goes awry when the initial estimate is poor.

Next lecture

- **Logistic regression**
- **Learning from incomplete data**
- **Auxiliary functions**

Lots of good stuff to come ...