Math Review

Lecture 1

Machine Learning Decal

Hosted by Machine Learning at Berkeley

Agenda

Let's build a rocket

Linear Algebra

Probability and Information Theory

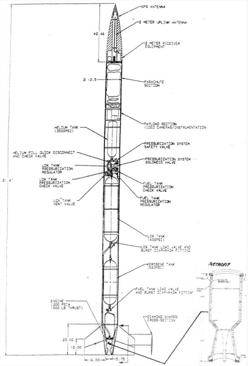
Numerical Computation

Python and Numpy Review Demo

Questions

Let's build a rocket

Designing an ML algorithm is like building a rocket



- Fuel: **Data**
- The design of rocket propulsion system: **Linear Algebra**
- The knowledge of physics and chemistry needed to ensure that the combustion of fuel provides enough thrust: **Statistics (Probability and Information Theory)**
- The principles of engineering to close the gap between the ideal and the reality: **Numerical Computation**

Linear Algebra

Scalars

A scalar is a single number

Integers $\mathbb{Z}$, real numbers $\mathbb{R}$, rational numbers $\mathbb{Q}$, etc.

Example notation: Italic font x, y, m, n, a

Vectors

A vector is a 1-D array of numbers:

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} \tag{0.1}$$

The entries can be integers $\mathbb{Z}$, real numbers $\mathbb{R}$, rational numbers $\mathbb{Q}$, binary etc.

Example notation for type and size:

$$\mathbb{Z}^N \tag{0.2}$$

Matrices

A matrix is a 2-D array of numbers:

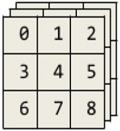
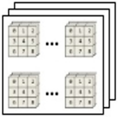
$$\mathbf{M} = \begin{bmatrix} M_{1,1} & M_{1,2} \\ M_{2,1} & M_{2,2} \end{bmatrix} \tag{0.3}$$

Example notation for type and shape:

$$\mathbf{M} \in \mathbb{Q}^{mxn} \tag{0.4}$$

Tensors

A tensor is an n-D array of numbers and can have:

Dimensions	Example	Terminology
1	$\begin{bmatrix} 0 & 1 & 2 \end{bmatrix}$	Vector
2	$\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \\ 6 & 7 & 8 \end{bmatrix}$	Matrix
3		3D Array (3rd order Tensor)
N		ND Array

Matrix Transpose

$$\mathbf{A}^\top{}_{i,j} = \mathbf{A}_{j,i}$$

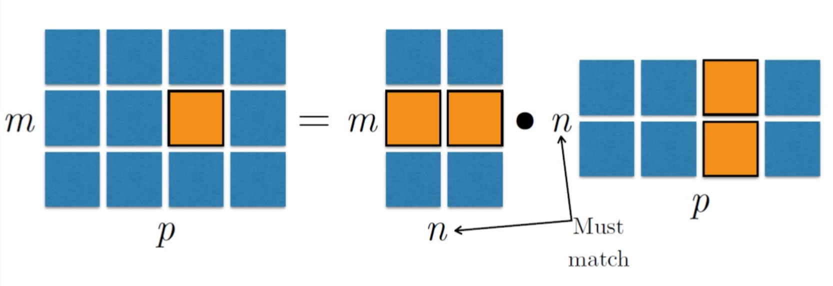
The transpose of a matrix is like a reflection along the main diagonal.

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \\ A_{3,1} & A_{3,2} \end{bmatrix} \implies \mathbf{A}^\top = \begin{bmatrix} A_{1,1} & A_{2,1} & A_{3,1} \\ A_{1,2} & A_{2,2} & A_{3,2} \end{bmatrix} \tag{0.5}$$

$$(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$$

$$\mathbf{C} = \mathbf{AB}$$

$$C_{i,j} = \sum_k A_{i,k} B_{k,j}$$



Element-wise multiplication: **Hadamard product** $A \odot B$

Identity Matrix

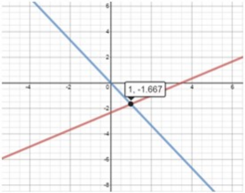
$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

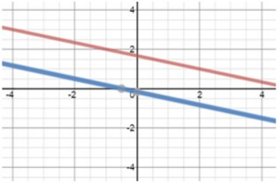
$$\forall x \in \mathbb{R}^n, \mathbf{I}_n x = x$$

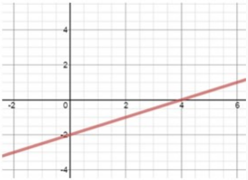
Systems of Equations

Three Possibilities – 1 solution, no solution, infinite number of solutions

$2x - 3y = 7$	$x + 3y = 5$	$.25x - .5y = 1$
$5x + 3y = 0$	$-2x - 6y = 1$	$-x + 2y = -4$







Solving Systems of Equations using Matrix Inversion

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}
\quad \Rightarrow \quad
\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}
=
\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}
$$

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}_n$$

Numerically unstable, but useful for abstract analysis.

$$\mathbf{A}x = b$$

$$\mathbf{A}^{-1}\mathbf{A}x = \mathbf{A}^{-1}b$$

$$\mathbf{I}_n x = \mathbf{A}^{-1}b$$

Invertibility

Matrix can't be inverted if:

- More rows than columns
- More columns than rows
- Redundant rows/columns (linearly dependent/low rank)

Norms

1) Functions that measure the "size" of a vector

2) Similar to the distance between zero and the point represented by the vector

- $f(x) = 0 \implies x = 0$
- $f(x + y) \leqslant f(x) + f(y)$ (the triangle equality)
- $\forall a \in \mathbb{R}, \quad f(ax) = |a| f(x)$

Norms

- L^p norm

$$||x||_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$$

- L1 norm, $p = 1$:

$$||x||_1 = \sum_i |x_i|$$

- L2 norm, $p = 2$: (most popular)

$$||x||_2 = (\sum_i |x_i|^2)^{\frac{1}{2}}$$

- Max norm, infinite p:

$$||x||_\infty = \max_i |x_i|$$

Special Matrices and Vectors

- Unit vector:

$$||x||_2 = 1$$

- Symmetric Matrix:

$$\mathbf{A} = \mathbf{A}^T$$

- Orthogonal Matrix:

$$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$$

$$\mathbf{A}^{-1} = \mathbf{A}^T$$

Eigendecomposition

- Eigenvector and eigenvalue:

$$\mathbf{A}v = \lambda v$$

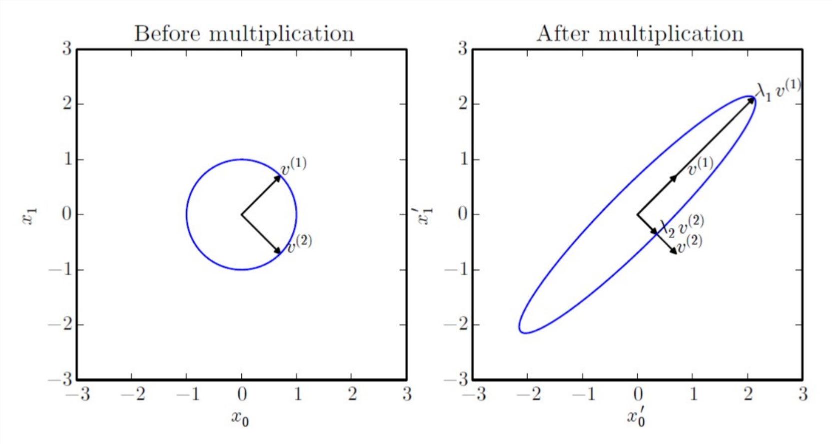
- Eigendecomposition of a diagonalizable matrix:

$$\mathbf{A} = \mathbf{V} diag(\lambda) \mathbf{V}^{-1}$$

- Every real symmetric matrix has a real, orthogonal eigendecomposition:

$$\mathbf{A} = \mathbf{QDQ}^T$$

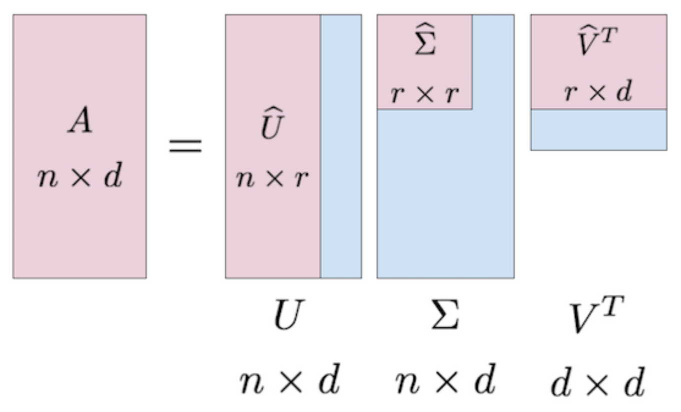
Effect of Eigenvalues



Singular Value Decomposition

Similar to eigendecomposition but more general: matrix need not be square.

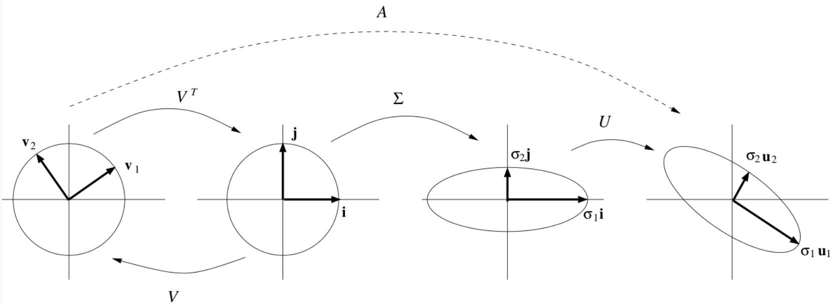
$$\mathbf{A} = \mathbf{UDV}^T$$



Singular Value Decomposition

Similar to eigendecomposition but more general: matrix need not be square.

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$



Moore-Penrose Pseudoinverse

$$x = \mathbf{A}^{+} y$$

If the equation has:

- Exactly one solution: this is the same as the inverse.
- No solution: this gives us the solution with the smallest error

$$||\mathbf{A}x - y||_2$$

- Many solutions: this gives us the solution with the smallest norm of x
- Use SVD to compute pseudoinverse

$$\mathbf{A}^{+} = \mathbf{U}\mathbf{D}^{+}\mathbf{V}^{T}$$

$\mathbf{D}^{+}$: Take reciprocal of non-zero entries

Trace

Trace is the sum of the diagonal entries of a matrix.

$$Tr(\mathbf{A}) = \sum_i \mathbf{A}_{i,i}$$

Useful identities:

$$Tr(\mathbf{A}) = Tr(\mathbf{A}^T)$$

$$Tr(\mathbf{ABC}) = Tr(\mathbf{CAB}) = Tr(\mathbf{BCA})$$

Probability and Information Theory

Probability Mass Function

The domain of P must be the set of all possible states of x.

$$\forall x \in X, 0 \leqslant P(x) \leqslant 1$$

$$\sum_{x \in X} P(x) = 1$$

Example: uniform distribution

$$P(X = x) = \frac{1}{k}$$

Probability Density Function

The domain of p must be the set of all possible states of x.

$$\forall x \in X, p(x) \geqslant 0$$

Note: do not require $p(x) \leqslant 1$

$$\int p(x)dx = 1$$

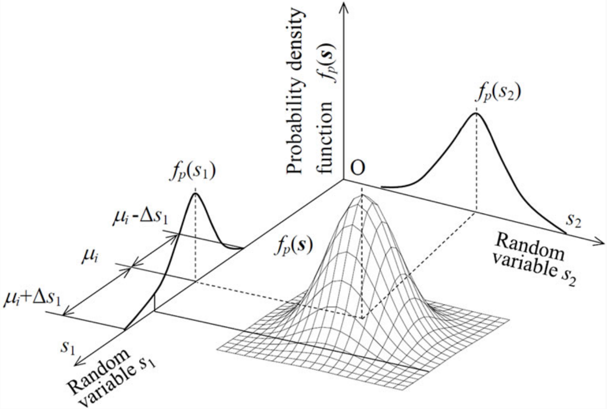
Example: uniform distribution

$$u(x; a, b) = \frac{1}{b - a}$$

Computing Marginal Probability with the Sum Rule

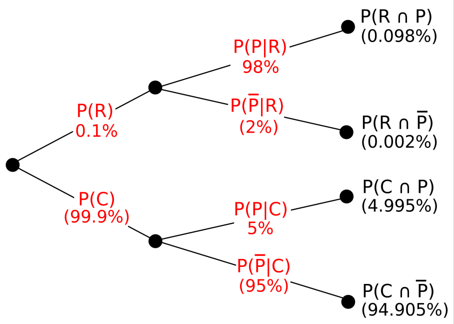
$$\forall x \in X, P(X = x) = \sum_{y} P(X = x, Y = y)$$

$$p(x) = \int p(x, y) dy$$



Conditional Probability

$$P(Y = y|X = x) = \frac{P(Y = y, X = x)}{P(X = x)}$$



Bayes Rule:

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)}$$

Chain Rule of Probability

$$P(x_1, ..., x_n) = P(x_1) \prod_{i=2}^{n} P(x_i | x_1, ..., x_{i-1})$$

Markov property:

$$P(x_i | x_{i-1}, ..., x_1) = P(x_i | x_{i-1}) \; for \; i > 1$$

$$P(x_n, ..., x_1) = P(x_1) \prod_{i=2}^{n} P(x_i | x_{i-1}) = P(x_1) P(x_2 | x_1) ... P(x_n | x_{n-1})$$

Independence

$$\forall x \in X, y \in Y$$

$$p(X = x, Y = y) = p(X = x)p(Y = y)$$

Conditional Independence:

$$\forall x \in X, y \in Y, z \in Z$$

$$p(X = x, Y = y|Z = z) = p(X = x|Z = z)p(Y = y|Z = z)$$

Expectation

$$\mathbb{E}_{x \sim P}[f(x)] = \sum_x P(x)f(x)$$

$$\mathbb{E}_{x \sim p}[f(x)] = \int p(x)f(x)dx$$

Linearity of Expectations:

$$\mathbb{E}_x[\alpha f(x) + \beta g(x)] = \alpha \mathbb{E}_x[f(x)] + \beta \mathbb{E}_x[g(x)]$$

Variance and Covariance

$$Var(f(x)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$$

$$Cov(f(x), g(y)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])(g(y) - \mathbb{E}[g(y)])]$$

Covariance matrix:

$$Cov(\mathbf{x})_{i,j} = Cov(x_i, x_j)$$

Bernoulli Distribution

$$P(X = 1) = \phi$$

$$P(X = 0) = 1 - \phi$$

$$P(X = x) = \phi^x (1 - \phi)^{1-x}$$

$$\mathbb{E}_x[X] = \phi$$

$$Var_x(X) = \phi(1 - \phi)$$

Gaussian Distribution

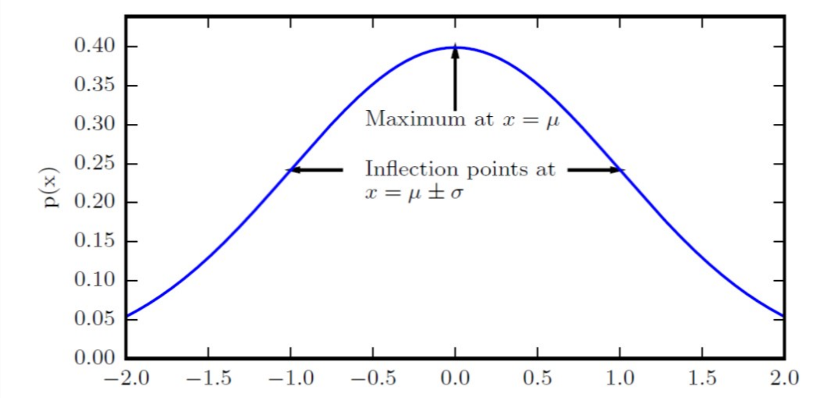
Parametrized by variance:

$$\mathcal{N}(x;\mu,\sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}} exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$$

Parametrized by precision:

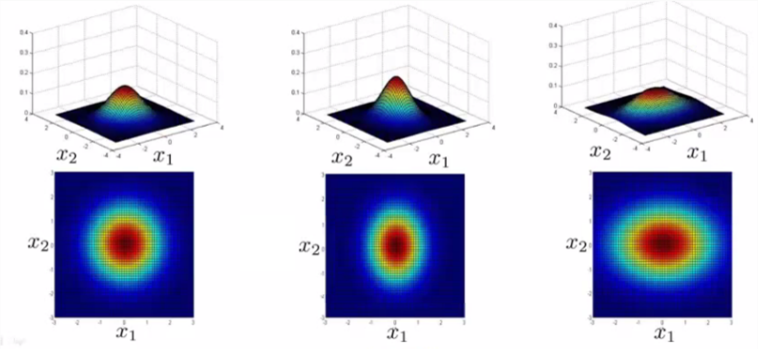
$$\mathcal{N}(x;\mu,\beta^{-1}) = \sqrt{\frac{\beta}{2\pi}} exp(-\frac{1}{2}\beta(x-\mu)^2)$$

Gaussian Distribution



$$\mathcal{N}(x;\mu,\Sigma) = \sqrt{\frac{1}{(2\pi)^n det(\Sigma)}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

$$\mathbf{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

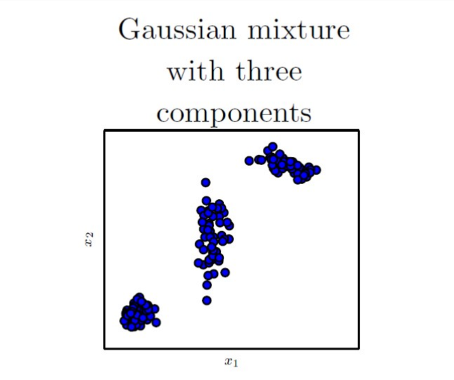


Empirical Distribution

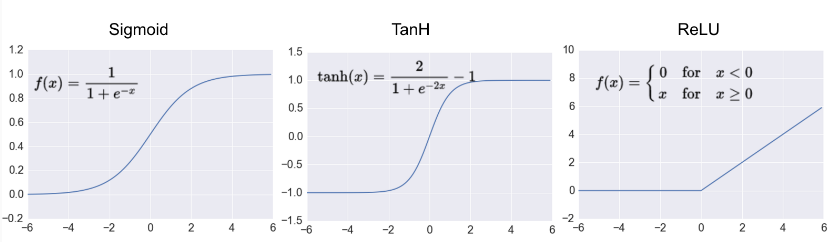
$$\hat{p} = \frac{1}{m} \sum_{i=1}^{m} \delta(x - x^{(i)})$$

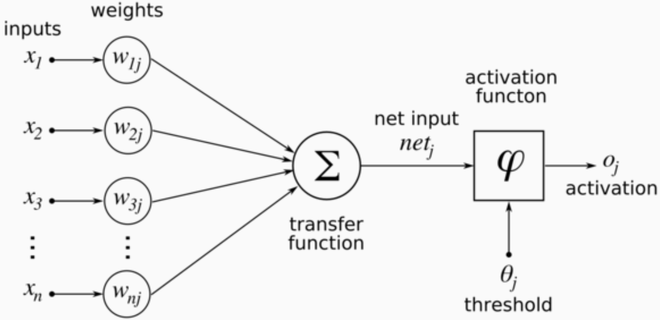
Mixture Distribution

$$P(X) = \sum_i P(c = i)P(x|c = i)$$

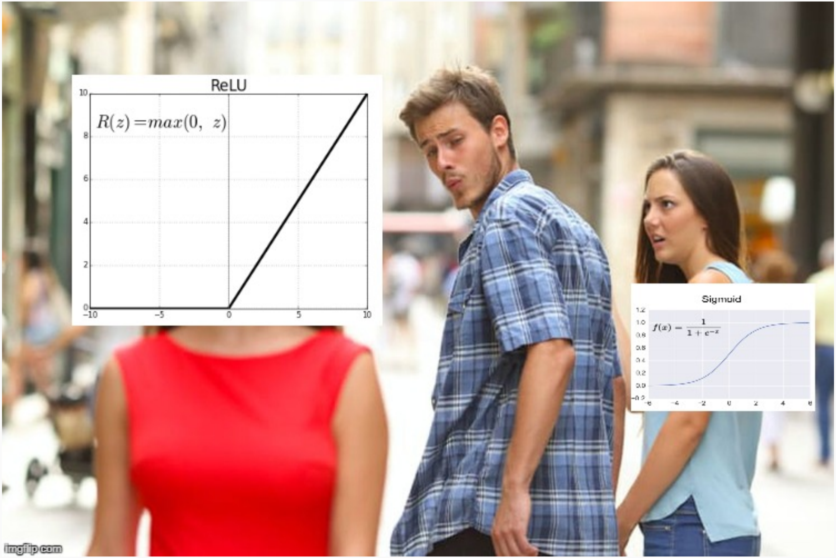


Activation Functions





Vanishing Gradient Problem



Information Theory

Information:

$$I(x) = -logP(x)$$

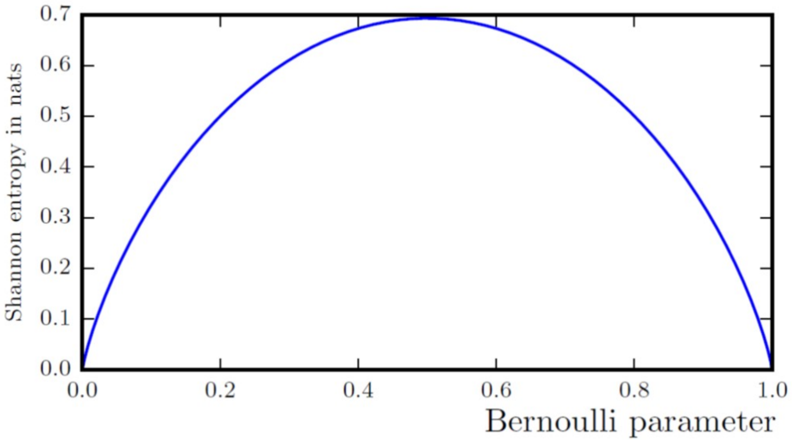
Shannon's Entropy:

$$H(X) = \mathbb{E}_{X\sim P}[I(x)] = -\mathbb{E}_{X\sim P}[logP(x)]$$

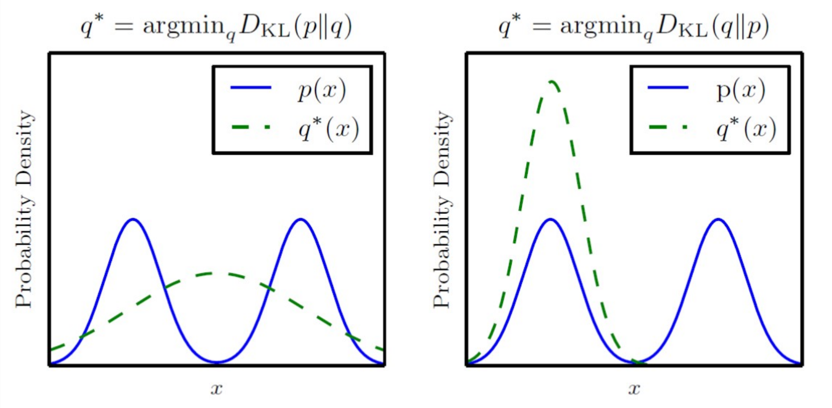
KL divergence:

$$D_{KL}(P||Q) = \mathbb{E}_{X\sim P}[log\frac{P(x)}{Q(x)}] = \mathbb{E}_{X\sim P}[logP(x) - logQ(x)]$$

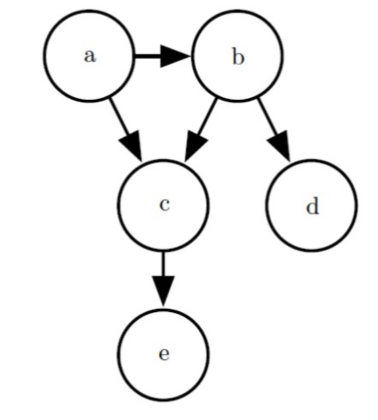
Entropy of a Bernoulli Variable



KL Divergence is Asymmetric



Directed Model



$$p(a, b, c, d, e) = p(a)p(b|a)p(c|a, b)p(d|b)p(e|c)$$

Undirected Model

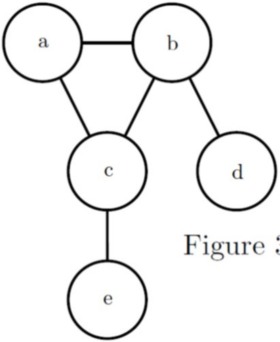


Figure 3.8

$$p(a, b, c, d, e) = \frac{1}{Z}\phi^{(1)}(a, b, c)\phi^{(2)}(b, d)\phi^{(3)}(c, e)$$

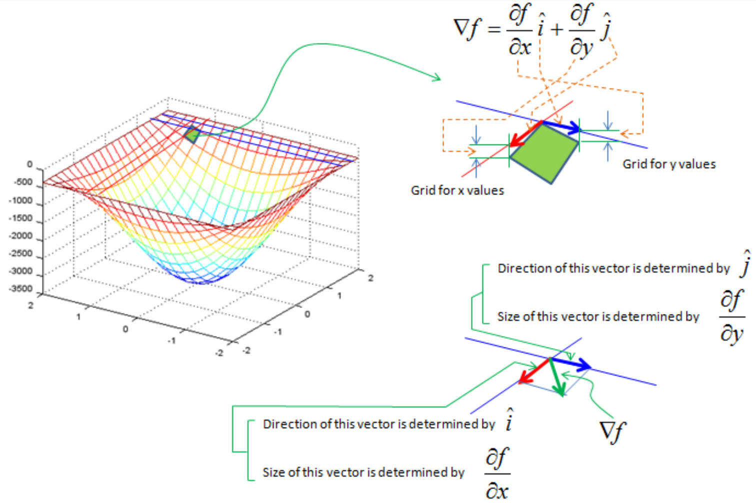
Numerical Computation

Numerical concerns for implementation

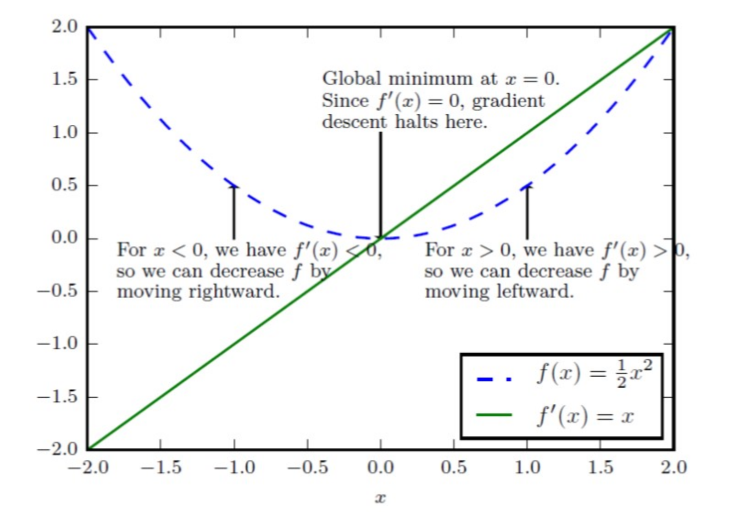
Algorithms are often specified in terms of real numbers but $\mathbb{R}$ cannot be implemented in a finite computer.
To implement deep learning algorithms with a finite number of bits, we need **Iterative Optimization**.

- Gradient descent
- Curvature

Gradient



$\nabla f = \frac{\partial f}{\partial x}\hat{i} + \frac{\partial f}{\partial y}\hat{j}$
Grid for y values
Grid for x values
Direction of this vector is determined by $\hat{j}$
Size of this vector is determined by $\frac{\partial f}{\partial y}$
Direction of this vector is determined by $\hat{i}$
Size of this vector is determined by $\frac{\partial f}{\partial x}$
∇f
0
-500
-1000
-1500
-2000
-2500
-3000
-3500
2
1
0
-1
-2
2
1
0
-1
-2



Global minimum at $x = 0$.
Since $f'(x) = 0$, gradient
descent halts here.
For $x < 0$, we have $f'(x) < 0$,
so we can decrease f by
moving rightward.
For $x > 0$, we have $f'(x) > 0$,
so we can decrease f by
moving leftward.
$f(x) = \frac{1}{2}x^2$
$f'(x) = x$
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

Gradient Descent Algorithm

repeat until convergence $\{$

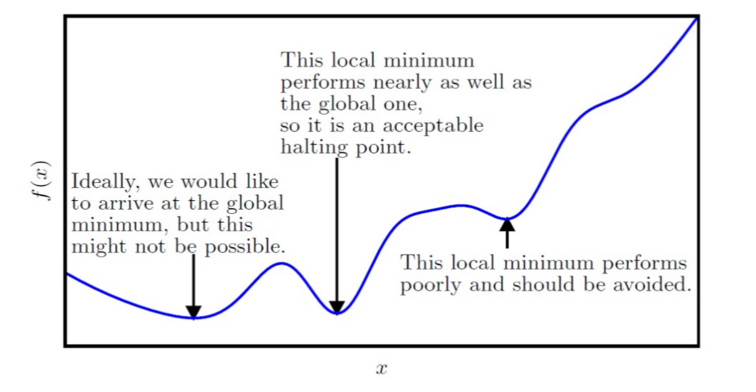
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

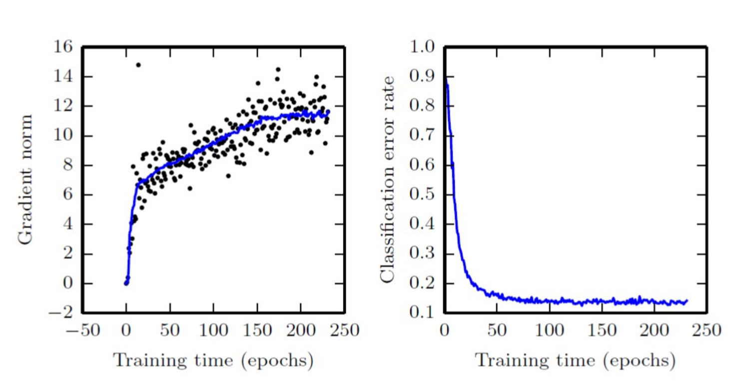
$\}$

update θ_0 and θ_1 simultaneously

Approximate Optimization



Usually don't even reach a local minimum



Optimization: Pure math vs Deep Learning

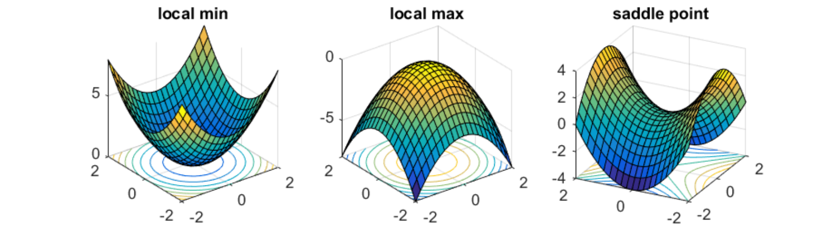
Pure Math (Calculus: setting derivative to zero/ Lagrange multipliers)

- Find literally the smallest value of f(x)
- Or maybe: find some critical point of f(x) where the value is locally smallest by solving equations

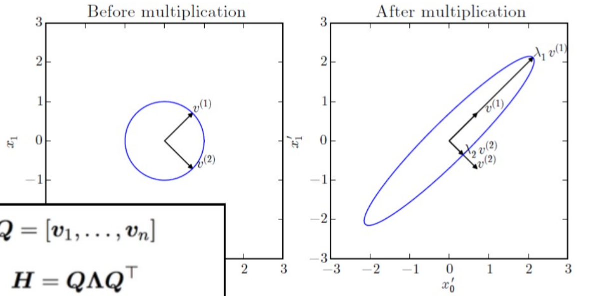
Deep Learning

- Just decrease the value of f(x) a lot iteratively until a point of convergence is approached

Critical Points



Directional Second Derivatives



$$\boldsymbol{Q} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$$

$$\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$$

Second derivative in direction d:

$$\boldsymbol{d}^\top \boldsymbol{H} \boldsymbol{d} = \sum_i \lambda_i \cos^2 \angle(\boldsymbol{v}_i, \boldsymbol{d})$$

(Goodfellow 2017)

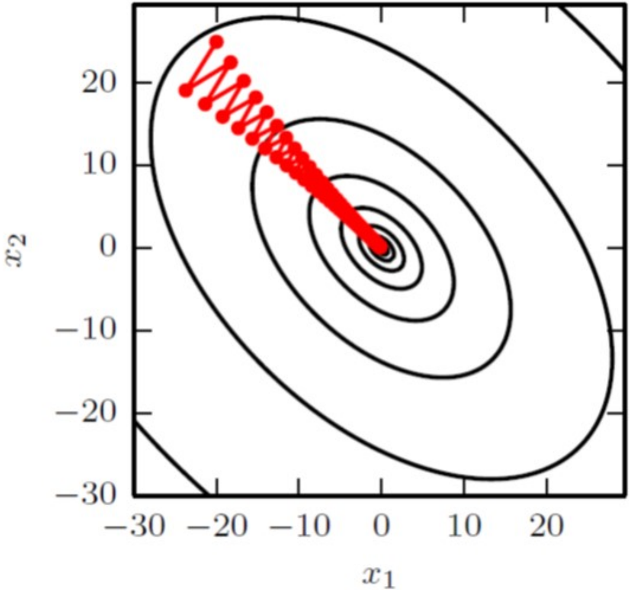
Predicting optimal step size using Taylor series

$f(x^{(0)} - \epsilon g) \approx f(x^{(0)}) - \epsilon g^T g + \frac{1}{2}\epsilon^2 g^T \mathbf{H} g$

$\epsilon^* = \frac{g^T g}{g^T \mathbf{H} g}$

Numerator: Big gradients speed you up
Denominator: Big eigenvalues slow you down if you align with their eigenvectors

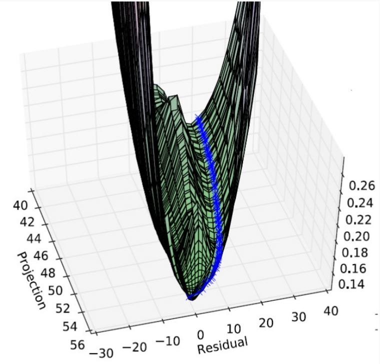
Gradient Descent and Poor Conditioning



Neural net visualization

At the end of learning:

- gradient is still large
- curvature is huge



Python and Numpy Review Demo

Questions