

# Inference from Statistical Models and from Data

EE599 Deep Learning

Keith M. Chugg  
Spring 2020



**USC** University of  
Southern California

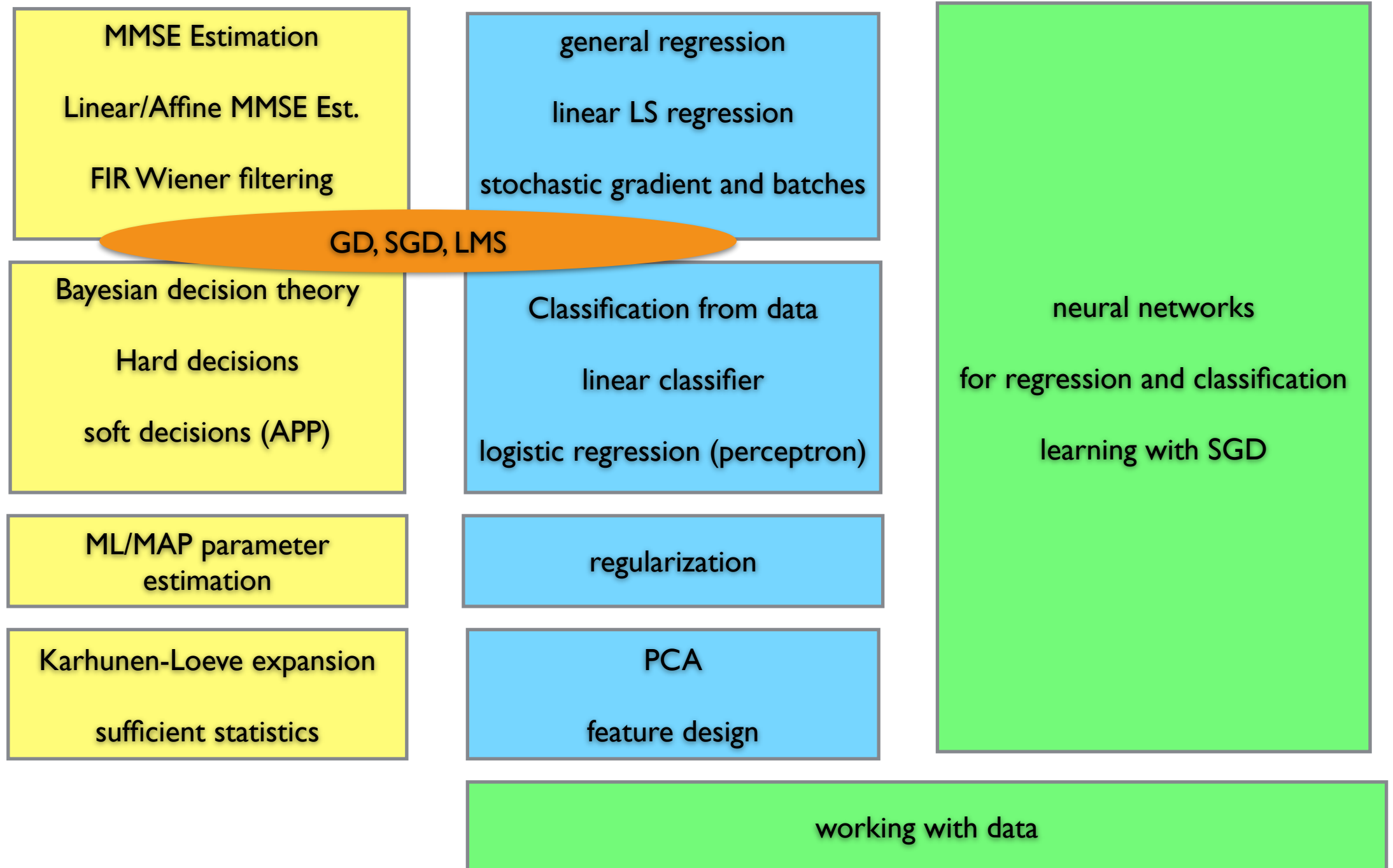
# Course Topics (from Syllabus)

- Course Introduction
- **Estimation and Detection with Statistical Descriptions**
- Regression (data fitting)
- Optimization with Steepest Descent
- Multi-Layer Perceptrons (Feedforward Neural Networks)
- Variations on SGD
- Working with Data
- Convolutional Neural Networks
- Recurrent Neural Networks
- Additional Topics (Reinforcement Learning, GANs, etc)

# Detection, Estimation, Regression

statistical models

data driven



# Key Ideas for Random Vectors

- $N \times 1$  random vectors — generalization of  $2 \times 1$
- Complete statistical descriptions vs Second moment descriptions
  - Direction preference (KL expansion)
- Gaussian processes and linear processing
- Linearity of the expectation operator

$$\mathbb{E} \{L(\mathbf{x}(u))\} = L(\mathbb{E} \{\mathbf{x}(u)\})$$

expectation commutes with any linear operation



# Random Vectors

random vector

$$\mathbf{x}(u) = \begin{bmatrix} x_0(u) \\ x_1(u) \\ \vdots \\ x_{N-1}(u) \end{bmatrix} \quad (N \times 1)$$

Complete  
statistical  
description

$$p_{\mathbf{x}(u)}(\mathbf{x}) = p_{x_0(u), x_1(u), \dots, x_{N-1}(u)}(x_0, x_1, \dots, x_{N-1}) \quad (\text{pdf or cdf or pmf})$$

$$\mathbf{m}_{\mathbf{x}} = \mathbb{E} \{ \mathbf{x}(u) \}$$

mean vector

$$\mathbf{R}_{\mathbf{x}} = \mathbb{E} \{ \mathbf{x}(u) \mathbf{x}^t(u) \}$$

correlation matrix

$$[\mathbf{R}_{\mathbf{x}}]_{i,j} = \mathbb{E} \{ x_i(u) x_j(u) \}$$

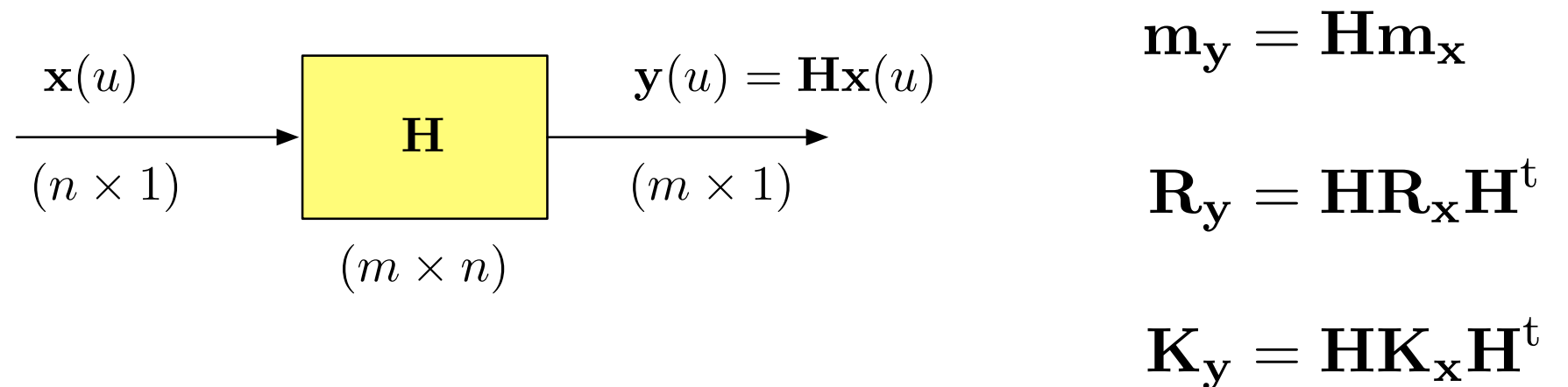
Second  
Moment  
Description

$$\begin{aligned} \mathbf{K}_{\mathbf{x}} &= \mathbb{E} \{ (\mathbf{x}(u) - \mathbf{m}_{\mathbf{x}})(\mathbf{x}(u) - \mathbf{m}_{\mathbf{x}})^t \} \\ &= \mathbf{R}_{\mathbf{x}} - \mathbf{m}_{\mathbf{x}} \mathbf{m}_{\mathbf{x}}^t \end{aligned}$$

covariance matrix

$$[\mathbf{K}_{\mathbf{x}}]_{i,j} = \text{cov} [x_i(u), x_j(u)]$$

# Random Vectors



## Special case

$$y(u) = \mathbf{b}^t \mathbf{x}(u) \quad (1 \times 1)$$

$$m_y = \mathbf{b}^t \mathbf{m}_x$$

$$\mathbb{E} \{y^2(u)\} = \mathbf{b}^t \mathbf{R}_x \mathbf{b}$$

$$\sigma_y^2 = \mathbf{b}^t \mathbf{K}_x \mathbf{b}$$

## example math

$$\begin{aligned} \mathbf{R}_y &= \mathbb{E} \{ \mathbf{y}(u) \mathbf{y}^t(u) \} \\ &= \mathbb{E} \{ (\mathbf{H}\mathbf{x}(u)) (\mathbf{H}\mathbf{x}(u))^t \} \\ &= \mathbb{E} \{ \mathbf{H}\mathbf{x}(u) \mathbf{x}^t(u) \mathbf{H}^t \} \\ &= \mathbf{H} \mathbb{E} \{ \mathbf{x}(u) \mathbf{x}^t(u) \} \mathbf{H}^t \\ &= \mathbf{H} \mathbf{R}_x \mathbf{H}^t \end{aligned}$$

Note that covariance/correlation matrices are symmetric, non-negative definite

# KL-Expansion

Can always find orthonormal set of e-vectors of **K**

These are an alternate coordinate systems (rotations, reflections)

in this eigen-coordinate system, the components are uncorrelated

*(principle components)*

The eigen-values are the variance (energy) in each of these principle directions

*(can be used to reduce dimensions by throwing out components with low energy)*

# KL-Expansion

$$\mathbf{K}_{\mathbf{x}} \mathbf{e}_k = \lambda_k \mathbf{e}_k \quad k = 0, 1, \dots, N-1 \quad (\text{Eigen equation})$$

$$\mathbf{e}_k^{\text{t}} \mathbf{e}_l = \delta[k-l] \quad \lambda_k \geq 0 \quad (\text{orthonormal e-vectors})$$

$$\mathbf{x}(u) = \sum_{k=0}^{N-1} X_k(u) \mathbf{e}_k \quad (\text{change of coordinates})$$

$$X_k(u) = \mathbf{e}_k^{\text{t}} \mathbf{x}(u)$$

$$\mathbb{E} \{X_k(u) X_l(u)\} = \mathbf{e}_k^{\text{t}} \mathbf{K}_{\mathbf{x}} \mathbf{e}_l = \lambda_k \delta[k-l] \quad (\text{uncorrelated components})$$

$$\mathbf{K}_{\mathbf{x}} = \sum_{k=0}^{N-1} \lambda_k \mathbf{e}_k \mathbf{e}_k^{\text{t}} = \mathbf{E} \mathbf{\Lambda} \mathbf{E}^{\text{t}} \quad (\text{Mercer's Theorem})$$

$$\mathbb{E} \{ \|\mathbf{x}(u)\|^2 \} = \text{tr}(\mathbf{K}_{\mathbf{x}}) = \sum_{k=0}^{N-1} \lambda_k \quad (\text{Total Energy})$$

Always exists because  $\mathbf{K}$  is nnd-symmetric

# KL-Expansion Examples

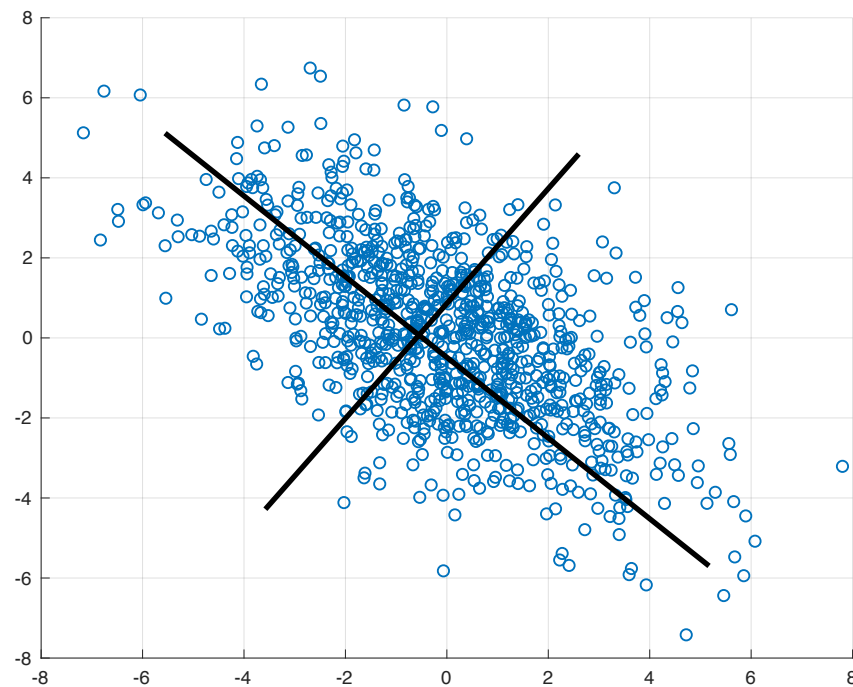
$$\mathbf{x}(u) = \mathbf{H}\mathbf{w}(u)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 2 \\ 1 & -2 \end{bmatrix}$$

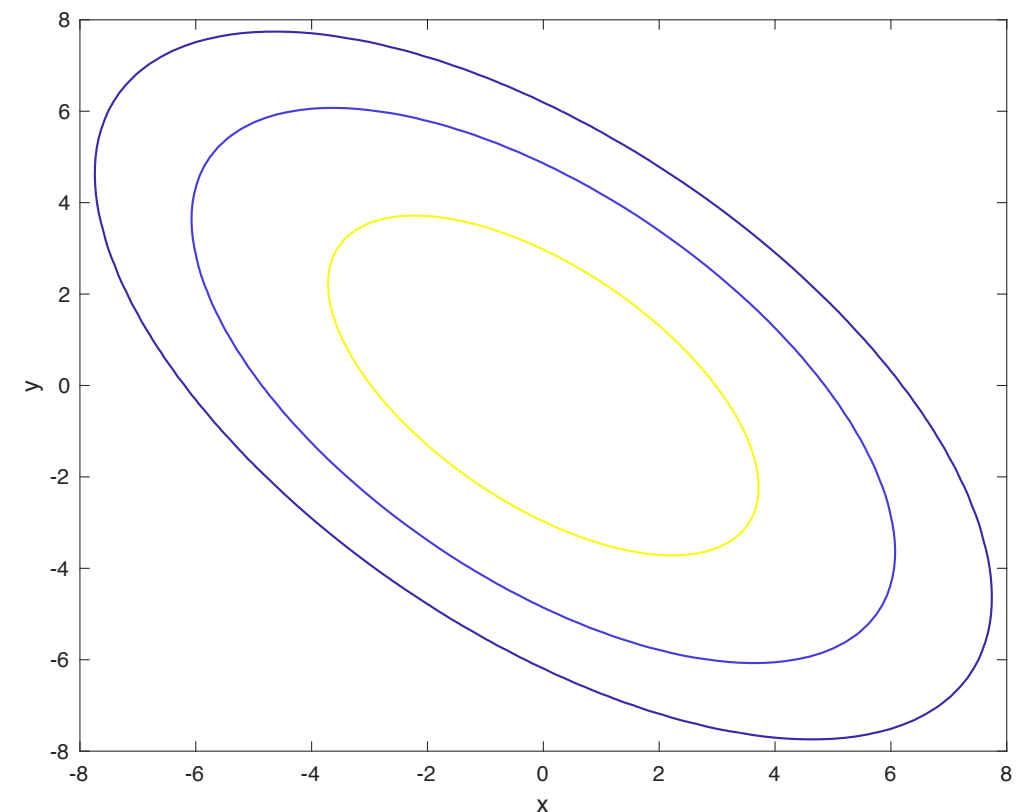
$$\mathbf{K} = \mathbf{H}\mathbf{K}_w\mathbf{H}^t = \mathbf{H}\mathbf{H}^t = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$$

$$\mathbf{E} = \frac{1}{\sqrt{2}} \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$



generated with  $\mathbf{w}(u)$  Gaussian



Gaussian pdf contours

# KL-Expansion Examples

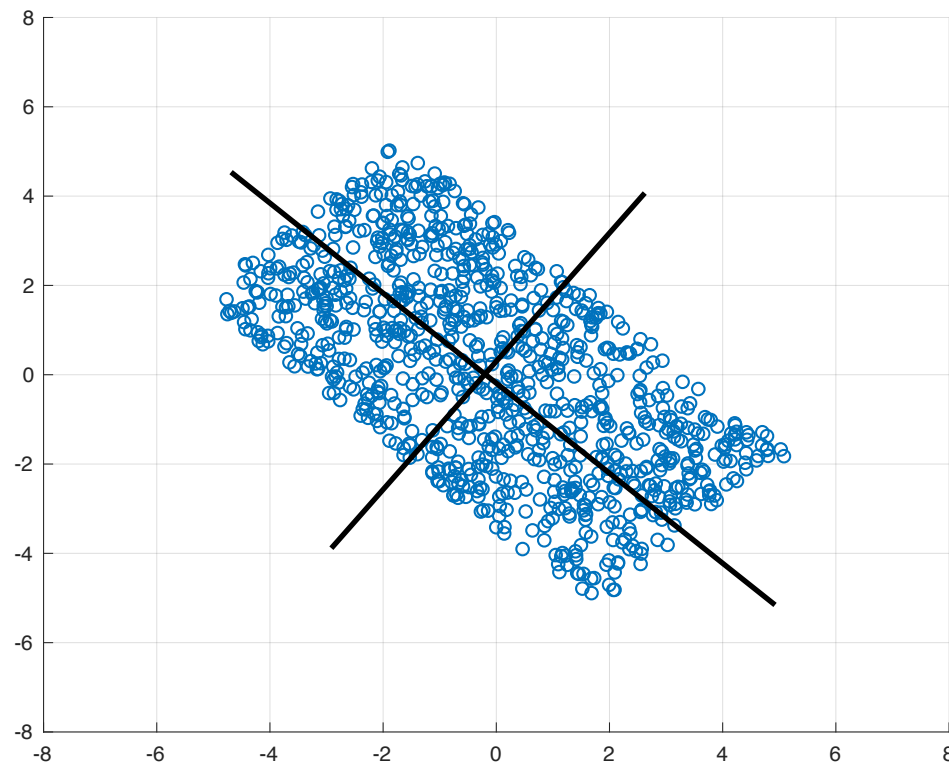
$$\mathbf{x}(u) = \mathbf{H}\mathbf{w}(u)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 2 \\ 1 & -2 \end{bmatrix}$$

$$\mathbf{K} = \mathbf{H}\mathbf{K}_w\mathbf{H}^t = \mathbf{H}\mathbf{H}^t = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$$

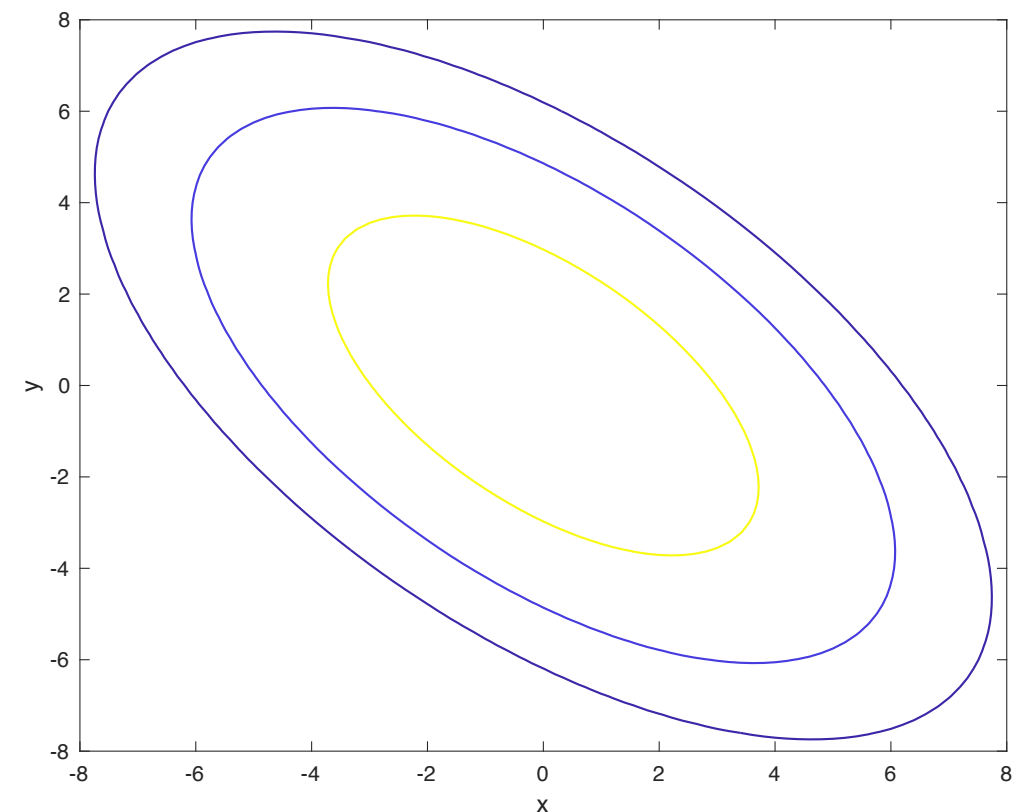
$$\mathbf{E} = \frac{1}{\sqrt{2}} \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$



generated with  $\mathbf{w}(u)$  uniform

$$\text{PR} \left\{ (\mathbf{x}(u) - \mathbf{m}_x)^t \mathbf{K}_x^{-1} (\mathbf{x}(u) - \mathbf{m}_x) \geq \epsilon^2 \right\} \leq \frac{n}{\epsilon^2}.$$



Chebyshev bound regions

# KL-Expansion Examples

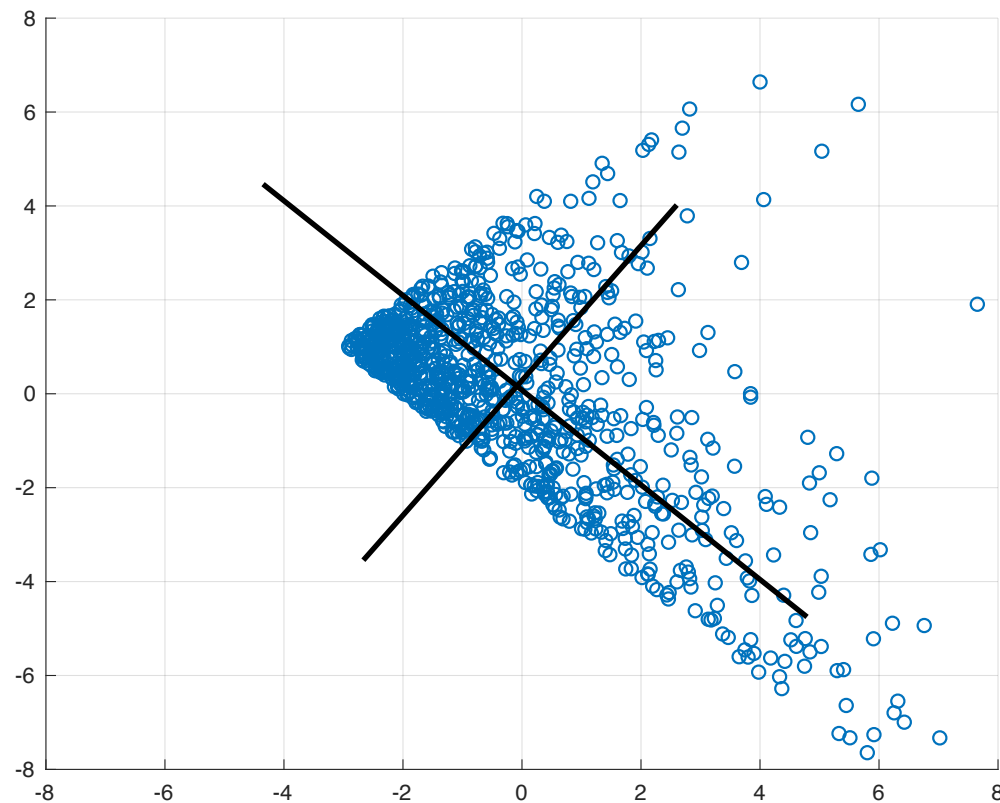
$$\mathbf{x}(u) = \mathbf{H}\mathbf{w}(u)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 2 \\ 1 & -2 \end{bmatrix}$$

$$\mathbf{K} = \mathbf{H}\mathbf{K}_w\mathbf{H}^t = \mathbf{H}\mathbf{H}^t = \begin{bmatrix} 5 & -3 \\ -3 & 5 \end{bmatrix}$$

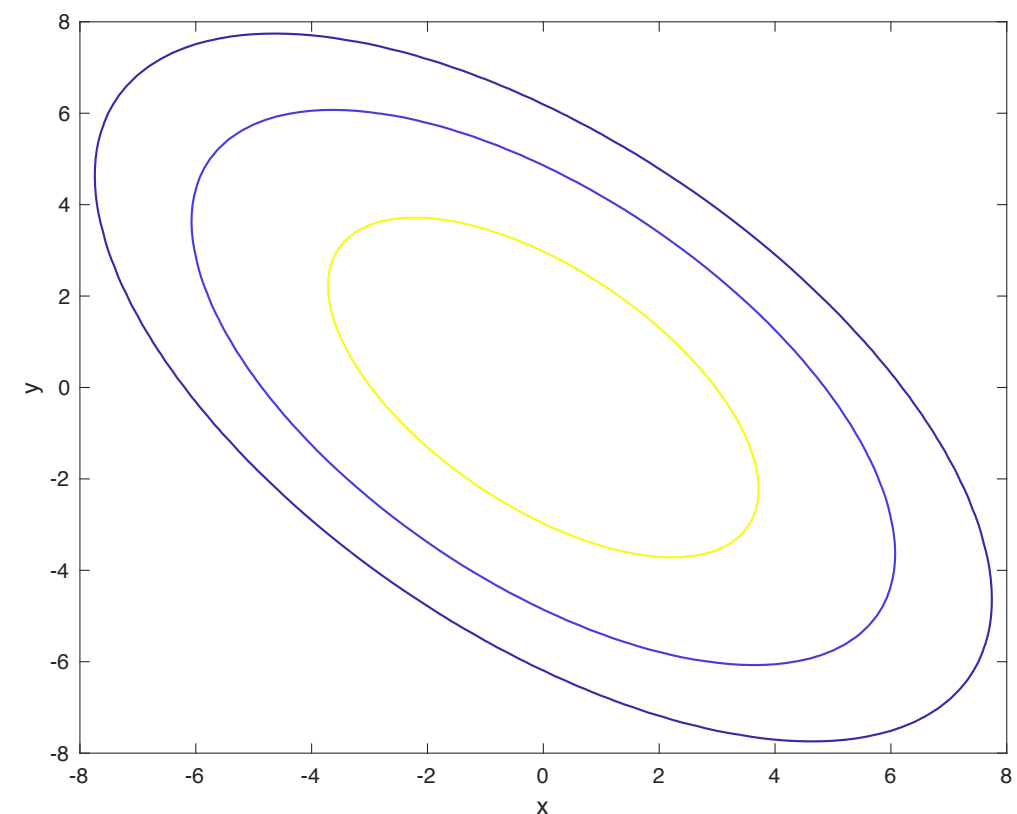
$$\mathbf{E} = \frac{1}{\sqrt{2}} \begin{bmatrix} +1 & +1 \\ +1 & -1 \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$$



generated with  $\mathbf{w}(u)$  exponential

$$\text{Pr} \{ (\mathbf{x}(u) - \mathbf{m}_x)^t \mathbf{K}_x^{-1} (\mathbf{x}(u) - \mathbf{m}_x) \geq \epsilon^2 \} \leq \frac{n}{\epsilon^2}.$$



Chebyshev bound regions

# Gaussian Random Vectors

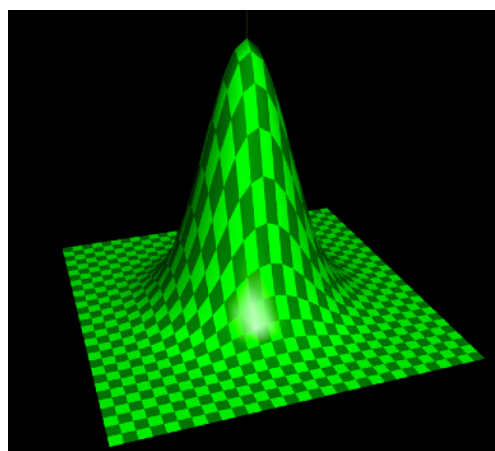
$$p_{\mathbf{x}(u)}(\mathbf{x}) = \mathcal{N}_N(\mathbf{x}; \mathbf{m}_{\mathbf{x}}; \mathbf{K}_{\mathbf{x}})$$

$$= \frac{1}{(2\pi)^{N/2} \sqrt{|\mathbf{K}_{\mathbf{x}}|}} \exp \left( -\frac{1}{2} (\mathbf{x} - \mathbf{m}_{\mathbf{x}})^{\mathsf{t}} \mathbf{K}_{\mathbf{x}}^{-1} (\mathbf{x} - \mathbf{m}_{\mathbf{x}}) \right)$$

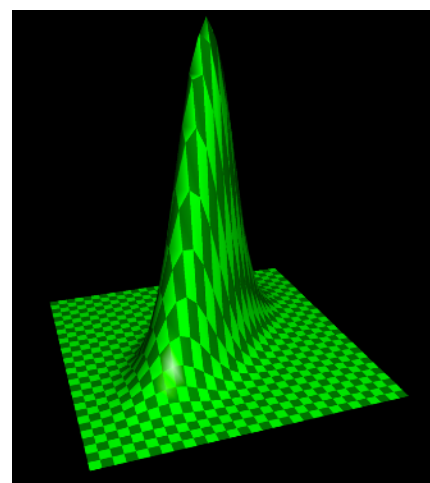
$$p_{\mathbf{x}(u)}(\mathbf{x}) = \mathcal{N}_N(\mathbf{x}; \mathbf{0}; \mathbf{I})$$

$$= \frac{1}{(2\pi)^{N/2}} \exp \left( -\frac{1}{2} \|\mathbf{x} - \mathbf{m}_{\mathbf{x}}\|^2 \right)$$

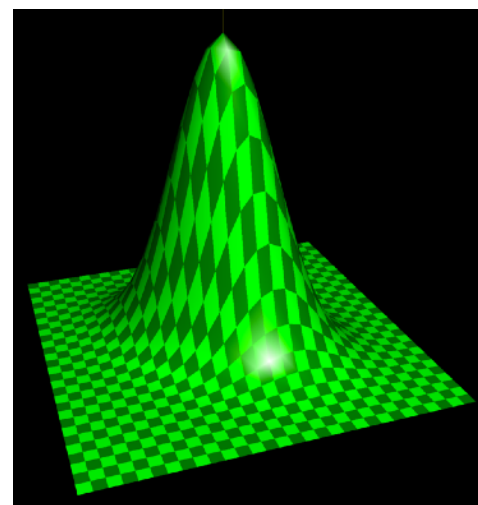
$$= \prod_{n=0}^{N-1} \mathcal{N}_1(x_n; 0; 1)$$



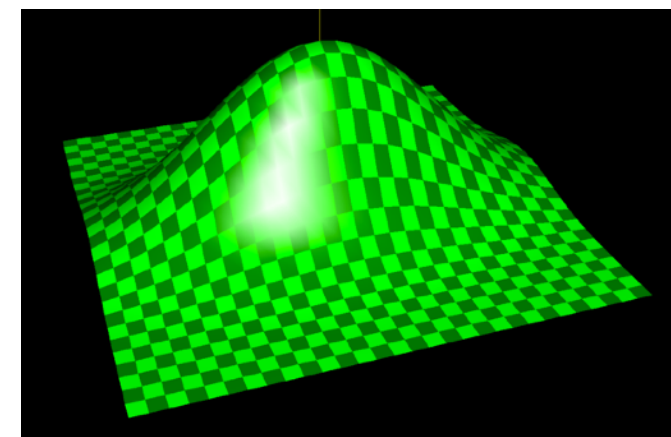
$$\mathbf{K} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$



$$\mathbf{K} = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$



$$\mathbf{K} = \begin{bmatrix} 1 & -0.4 \\ -0.4 & 1 \end{bmatrix}$$



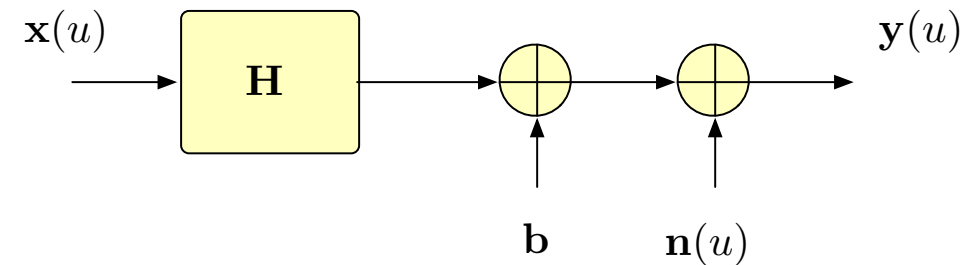
$$\mathbf{K} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

macOS plotter — source posted



# Gaussian Random Vectors



**any linear processing of Gaussians yields Gaussians**

if  $\begin{bmatrix} \mathbf{x}(u) \\ \mathbf{n}(u) \end{bmatrix}$  is Gaussian ( $\mathbf{x}(u)$  and  $\mathbf{w}(u)$  jointly-Gaussian), then:

$\mathbf{y}(u)$  is Gaussian

$\begin{bmatrix} \mathbf{x}(u) \\ \mathbf{y}(u) \end{bmatrix}$  is Gaussian

$\begin{bmatrix} \mathbf{x}(u) \\ \mathbf{n}(u) \\ \mathbf{y}(u) \end{bmatrix}$  is Gaussian

any subset of these random variables is also Gaussian

jointly-Gaussian is  
common in EE...

# Linear/Affine MMSE

estimate  $\mathbf{y}(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

$$\min_{\mathbf{f}(\mathbf{x})} \mathbb{E} \{ \|\mathbf{y}(u) - \mathbf{f}(\mathbf{x}(u))\|^2 \}$$

with no constraint, this is the conditional expectation function

$$\begin{aligned} \mathbf{f}_{\text{opt}}(\mathbf{x}) &= \mathbf{m}_{\mathbf{y}|\mathbf{x}}(\mathbf{x}) \\ &= \mathbb{E} \{ \mathbf{y}(u) | \mathbf{x}(u) = \mathbf{x} \} \\ &= \int \mathbf{y} p_{\mathbf{y}(u)|\mathbf{x}(u)}(\mathbf{y}|\mathbf{x}) d\mathbf{y} \end{aligned}$$

# Minimum Mean-Square Error Estimation (MMSE)

estimate  $\mathbf{y}(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

“cross covariance matrix”

$$\mathbf{K}_{\mathbf{y}\mathbf{x}} = \mathbb{E} \{ (\mathbf{y}(u) - \mathbf{m}_{\mathbf{y}})(\mathbf{x}(u) - \mathbf{m}_{\mathbf{x}})^t \} = [\mathbf{K}_{\mathbf{xy}}]^t$$

## Affine MMSE:

$$\min_{\mathbf{f}(\mathbf{x})=\mathbf{F}\mathbf{x}+\mathbf{b}} \mathbb{E} \{ \|\mathbf{y}(u) - \mathbf{f}(\mathbf{x}(u))\|^2 \}$$

$$\mathbf{F}_{\text{AMMSE}} = \mathbf{K}_{\mathbf{yx}}\mathbf{K}_{\mathbf{x}}^{-1}$$

$$\mathbf{b}_{\text{AMMSE}} = \mathbf{m}_{\mathbf{y}} - \mathbf{F}_{\text{AMMSE}}\mathbf{m}_{\mathbf{x}}$$

$$\min_{\mathbf{F}, \mathbf{b}} \mathbb{E} \{ \|\mathbf{y}(u) - [\mathbf{F}\mathbf{x}(u) + \mathbf{b}]\|^2 \}$$

$$\hat{\mathbf{y}} = \mathbf{K}_{\mathbf{yx}}\mathbf{K}_{\mathbf{x}}^{-1}(\mathbf{x} - \mathbf{m}_{\mathbf{x}}) + \mathbf{m}_{\mathbf{y}} \quad \text{AMMSE} = \text{tr}(\mathbf{K}_{\mathbf{y}} - \mathbf{K}_{\mathbf{yx}}\mathbf{K}_{\mathbf{x}}^{-1}\mathbf{K}_{\mathbf{xy}})$$

## Linear MMSE:

$$\min_{\mathbf{f}(\mathbf{x})=\mathbf{F}\mathbf{x}} \mathbb{E} \{ \|\mathbf{y}(u) - \mathbf{f}(\mathbf{x}(u))\|^2 \}$$

$$\mathbf{F}_{\text{LMMSE}} = \mathbf{R}_{\mathbf{yx}}\mathbf{R}_{\mathbf{x}}^{-1}$$

$$\text{LMMSE} = \text{tr}(\mathbf{R}_{\mathbf{y}} - \mathbf{R}_{\mathbf{yx}}\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{R}_{\mathbf{xy}})$$

$$\min_{\mathbf{F}} \mathbb{E} \{ \|\mathbf{y}(u) - \mathbf{F}\mathbf{x}(u)\|^2 \}$$

$$\hat{\mathbf{y}} = \mathbf{R}_{\mathbf{yx}}\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{x}$$

## notes:

- affine often called linear... same when means are zero
- Conditional Expectation better than Affine, better than Linear

# Proof for LMMSE

$$\min_{\mathbf{F}} \mathbb{E} \{ \|\mathbf{y}(u) - \mathbf{F}\mathbf{x}(u)\|^2 \}$$

Proof:

$$\text{MSE}(\mathbf{F}) = \mathbb{E} \{ \|\mathbf{y}(u) - \mathbf{F}\mathbf{x}(u)\|^2 \}$$

$$= \mathbb{E} \{ \|(\mathbf{y}(u) - \mathbf{F}_{\text{opt}}\mathbf{x}(u)) + (\mathbf{F}_{\text{opt}} - \mathbf{F})\mathbf{x}(u)\|^2 \}$$

$$= \mathbb{E} \{ \|(\mathbf{y}(u) - \mathbf{F}_{\text{opt}}\mathbf{x}(u))\|^2 \} + \text{tr} \left( (\mathbf{F}_{\text{opt}} - \mathbf{F})\mathbf{R}_{\mathbf{x}}(\mathbf{F}_{\text{opt}} - \mathbf{F})^{\text{t}} \right)$$

$$+ 2\text{tr} \left( (\mathbf{R}_{\mathbf{y}\mathbf{x}} - \mathbf{F}_{\text{opt}}\mathbf{R}_{\mathbf{x}})(\mathbf{F}_{\text{opt}} - \mathbf{F})^{\text{t}} \right)$$

use:

$$\mathbf{v}^{\text{t}}\mathbf{w} = \text{tr}(\mathbf{w}\mathbf{v}^{\text{t}})$$

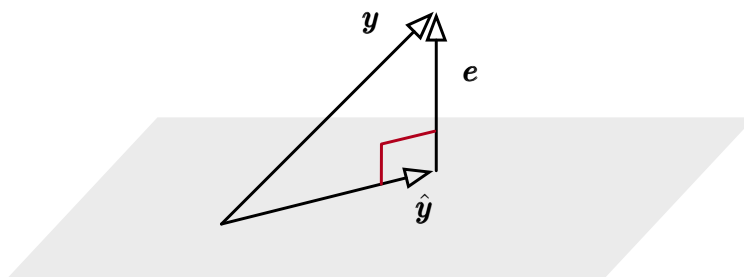
if:

$$\mathbf{F}_{\text{opt}}\mathbf{R}_{\mathbf{x}} = \mathbf{R}_{\mathbf{y}\mathbf{x}}$$

(Wiener-Hopf eq.)  
(aka Orthogonality Principle)

then:

$$\text{MSE}(\mathbf{F}) = \mathbb{E} \{ \|(\mathbf{y}(u) - \mathbf{F}_{\text{opt}}\mathbf{x}(u))\|^2 \} + \underbrace{\text{tr} \left( (\mathbf{F}_{\text{opt}} - \mathbf{F})\mathbf{R}_{\mathbf{x}}(\mathbf{F}_{\text{opt}} - \mathbf{F})^{\text{t}} \right)}_{\geq 0 \ \forall \ \mathbf{F}, \text{ since } \mathbf{R}_{\mathbf{x}} \text{ is nnd}}$$



space of all estimates/approximations

because of orthogonality principle

$$\begin{aligned} \mathbb{E} \{ \|\mathbf{y}(u) - \hat{\mathbf{y}}(u)\|^2 \} &= \mathbb{E} \{ \|\mathbf{y}(u)\|^2 \} - \mathbb{E} \{ \|\hat{\mathbf{y}}(u)\|^2 \} \\ &= \text{tr}(\mathbf{R}_{\mathbf{y}} - \mathbf{R}_{\mathbf{y}\mathbf{x}}\mathbf{R}_{\mathbf{x}}^{-1}\mathbf{R}_{\mathbf{x}\mathbf{y}}) \end{aligned}$$

# MMSE Estimation: special case jointly-Gaussian

estimate  $\mathbf{y}(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

Conditional Gaussian pdf:

$$\begin{aligned} p_{\mathbf{y}(u)|\mathbf{x}(u)}(\mathbf{y}|\mathbf{x}) &= \frac{p_{\mathbf{x}(u),\mathbf{y}(u)}(\mathbf{x},\mathbf{y})}{p_{\mathbf{x}(u)}(\mathbf{x})} \\ &= \frac{\mathcal{N}_{M+N} \left( \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}; \begin{bmatrix} \mathbf{m}_x \\ \mathbf{m}_y \end{bmatrix}; \begin{bmatrix} \mathbf{K}_x & \mathbf{K}_{xy} \\ \mathbf{K}_{yx} & \mathbf{K}_y \end{bmatrix} \right)}{\mathcal{N}_N(\mathbf{x}; \mathbf{m}_x; \mathbf{K}_x)} \\ &= \mathcal{N}_M \left( \mathbf{y}; \underbrace{\mathbf{m}_y + \mathbf{K}_{yx}\mathbf{K}_x^{-1}(\mathbf{x} - \mathbf{m}_x)}_{\text{the AMMSE estimator}}; \underbrace{\mathbf{K}_y - \mathbf{K}_{yx}\mathbf{K}_x^{-1}\mathbf{K}_{xy}}_{\text{the error covariance}} \right) \end{aligned}$$

jointly-Gaussian is  
common in EE...

don't need ML or  
deep learning in  
this case!

For jointly-Gaussian observation and desired,  $E\{\mathbf{Y}|\mathbf{x}\}$  is the Affine MMSE estimator

# LMMSE Special Case: scalars

estimate  $y(u)$  from  $x(u)=x$

Affine MMSE:

$$\hat{y} = \frac{\text{cov}[y(u), x(u)]}{\sigma_x^2} (x - m_x) + m_y$$

$$\text{AMMSE} = \sigma_y^2 (1 - \rho_{xy}^2)$$

$$= \rho_{xy} \frac{\sigma_y}{\sigma_x} (x - m_x) + m_y$$

# LMMSE Special Case: scalar desired, vector observed

estimate  $y(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

Linear MMSE (typ. means are zero):

$$\hat{y} = \mathbf{w}^t \mathbf{x}$$

$$\mathbf{w} = \mathbf{F}^t = \mathbf{R}_{\mathbf{x}}^{-1} \mathbf{r}_{\mathbf{x}y}$$

$$\mathbf{r}_{\mathbf{x}y} = \mathbf{R}_{\mathbf{x}y} = \mathbb{E} \{ \mathbf{x}(u) y(u) \} = [\mathbb{E} \{ y(u) \mathbf{x}^t(u) \}]^t$$

$$\text{LMMSE} = \sigma_y^2 - \mathbf{r}_{\mathbf{x}y}^t \mathbf{R}_{\mathbf{x}}^{-1} \mathbf{r}_{\mathbf{x}y}$$

This is an important special case, so let's develop it...

we will use gradients so that we arrive at stochastic gradient and LMS

# LMMSE Special Case: scalar desired, vector observed

estimate  $y(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

Linear MMSE (typ. means are zero):

$$\begin{aligned} E(\mathbf{w}) &= \mathbb{E} \left\{ [y(u) - \mathbf{w}^t \mathbf{x}]^2 \right\} \\ &= \mathbb{E} \left\{ [y(u)]^2 + \mathbf{w}^t \mathbf{x}(u) \mathbf{x}^t(u) \mathbf{w} - 2y(u) \mathbf{w}^t \mathbf{x}(u) \right\} \end{aligned}$$

Let's differentiate and seek critical point:

$$\nabla_{\mathbf{w}} E = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_{N-1}} \end{bmatrix}$$

this is a linear operation so...

$$\begin{aligned} \nabla_{\mathbf{w}} E &= \nabla_{\mathbf{w}} \mathbb{E} \left\{ [y(u)]^2 + \mathbf{w}^t \mathbf{x}(u) \mathbf{x}^t(u) \mathbf{w} - 2y(u) \mathbf{w}^t \mathbf{x}(u) \right\} \\ &= \mathbb{E} \left\{ \nabla_{\mathbf{w}} [y(u)]^2 + \nabla_{\mathbf{w}} [\mathbf{w}^t \mathbf{x}(u) \mathbf{x}^t(u) \mathbf{w}] - 2y(u) \nabla_{\mathbf{w}} [\mathbf{w}^t \mathbf{x}(u)] \right\} \end{aligned}$$



# Some Vector Derivative Results

These are simple to verify

$$\nabla_{\mathbf{w}} [\mathbf{w}^t \mathbf{A} \mathbf{w}] = (\mathbf{A}^t + \mathbf{A}) \mathbf{w}$$

$$\nabla_{\mathbf{w}} [\mathbf{w}^t \mathbf{b}] = \mathbf{b}$$

$$\nabla_{\mathbf{w}} (y - \mathbf{w}^t \mathbf{x})^2 = 2(\mathbf{x} \mathbf{x}^t \mathbf{w} - \mathbf{x} y)$$

As we build up more of these relations, I will make a table to post on Piazza

# LMMSE Special Case: scalar desired, vector observed

estimate  $y(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

$$E(\mathbf{w}) = \mathbb{E} \left\{ [y(u) - \mathbf{w}^t \mathbf{x}(u)]^2 \right\}$$

Let's differentiate and seeks critical point:

$$\begin{aligned} \nabla_{\mathbf{w}} E &= 2 \left( \mathbb{E} \left\{ \mathbf{w} \mathbf{x}(u) \mathbf{x}^t(u) - y(u) \mathbf{x}(u) \right\} \right) \\ &= 2(\mathbf{w}^t \mathbf{R}_{\mathbf{x}} - \mathbf{r}_{\mathbf{x}y}) \end{aligned}$$

$$\nabla_{\mathbf{w}} E = 0 \quad \Longleftrightarrow \quad \mathbf{R}_{\mathbf{x}} \mathbf{w} = \mathbf{r}_{\mathbf{x}y} \quad (\text{Wiener-Hopf equation})$$

This yields our solution  
(can verify global minimum)

# Steepest Descent and LMS

estimate  $y(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

$$E(\mathbf{w}) = \mathbb{E} \left\{ [y(u) - \mathbf{w}^t \mathbf{x}(u)]^2 \right\}$$

$$\begin{aligned} \nabla_{\mathbf{w}} E &= 2 \left( \mathbb{E} \{ \mathbf{w} \mathbf{x}(u) \mathbf{x}^t(u) - y(u) \mathbf{x}(u) \} \right) \\ &= 2(\mathbf{w}^t \mathbf{R}_{\mathbf{x}} - \mathbf{r}_{\mathbf{x}y}) \end{aligned}$$

Steepest descent using (ensemble average) gradient:

$$\begin{aligned} \hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n - (\eta/2) \nabla_{\mathbf{w}} E \\ &= \hat{\mathbf{w}}_n + \eta(\mathbf{r}_{\mathbf{x}y} - \mathbf{R}_{\mathbf{x}} \hat{\mathbf{w}}_n) \end{aligned}$$

Single Point Stochastic Gradient Descent:

$$\begin{aligned} -\frac{1}{2} \nabla_{\mathbf{w}} E &= \mathbf{r}_{\mathbf{x}y} - \mathbf{R}_{\mathbf{x}} \mathbf{w} \\ &= \mathbb{E} \{ y(u) \mathbf{x}(u) - \mathbf{x}(u) \mathbf{x}^t(u) \mathbf{w} \} \\ &\approx y_n \mathbf{x}_n - \mathbf{x}_n \mathbf{x}_n^t \mathbf{w} \\ &= (y_n - \mathbf{x}_n^t \hat{\mathbf{w}}_n) \mathbf{x}_n \end{aligned}$$

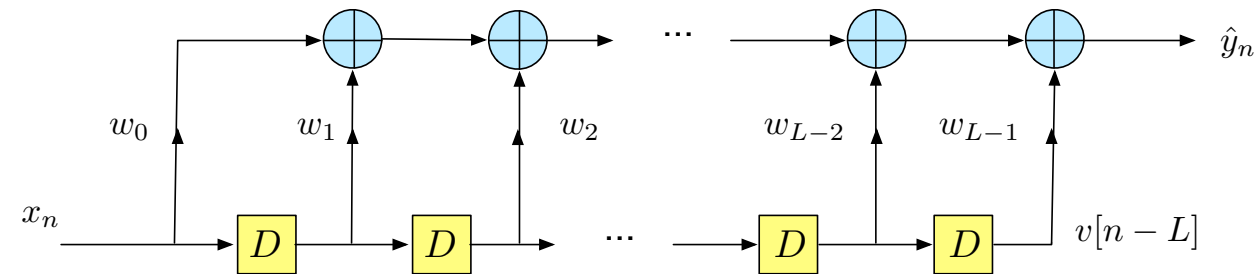
$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta(y_n - \hat{\mathbf{w}}_n^t \mathbf{x}_n) \mathbf{x}_n$$

this is called “on-line learning”

when  $n \sim$  time, this is the Least Mean Square (LMS) adaptive filter

when  $n$  does not represent time, we can easily average the gradient over more data points for a better approximation (batches)

# LMS Algorithm as Adaptive FIR filter



$$\hat{y}_n = \sum_{l=0}^{L-1} w_l x_{n-l} = \mathbf{w}^t \mathbf{v}_n$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{L-1} \end{bmatrix}$$

$$\mathbf{v}_n = \mathbf{x}_{n-(L-1)}^n = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{n-L+1} \end{bmatrix}$$

## Single Point Stochastic Gradient Descent:

$$-\frac{1}{2} \nabla_{\mathbf{w}} E = \mathbf{r}_{\mathbf{v}_n y_n} - \mathbf{R}_{\mathbf{v}_n} \mathbf{w}$$

$$= \mathbb{E} \{ y(u) \mathbf{v}_n(u) - \mathbf{v}_n(u) \mathbf{v}_n^t(u) \mathbf{w} \}$$

$$\approx (y_n - \mathbf{w}_n^t \mathbf{v}_n) \mathbf{v}_n$$

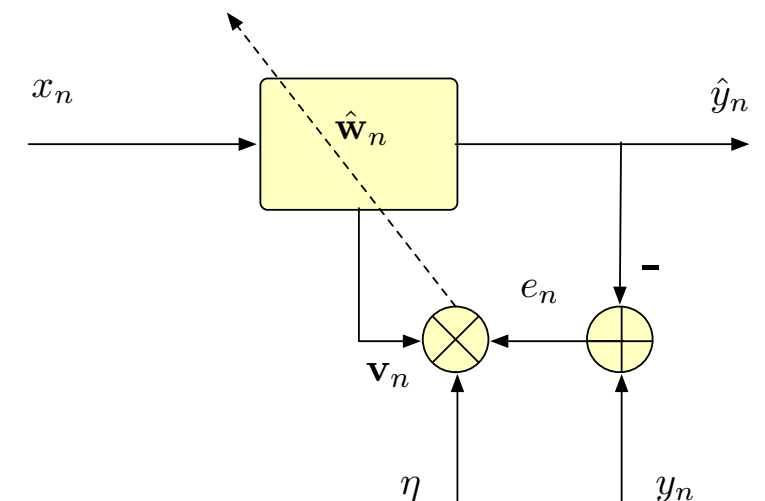
### LMS Algorithm

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta (y_n - \hat{\mathbf{w}}_n^t \mathbf{v}_n) \mathbf{v}_n$$

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta (y_n - \hat{y}_n) \mathbf{v}_n$$

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta e_n \mathbf{v}_n$$

(this is an on-line linear regressor)



# LMS Algorithm as Adaptive FIR filter

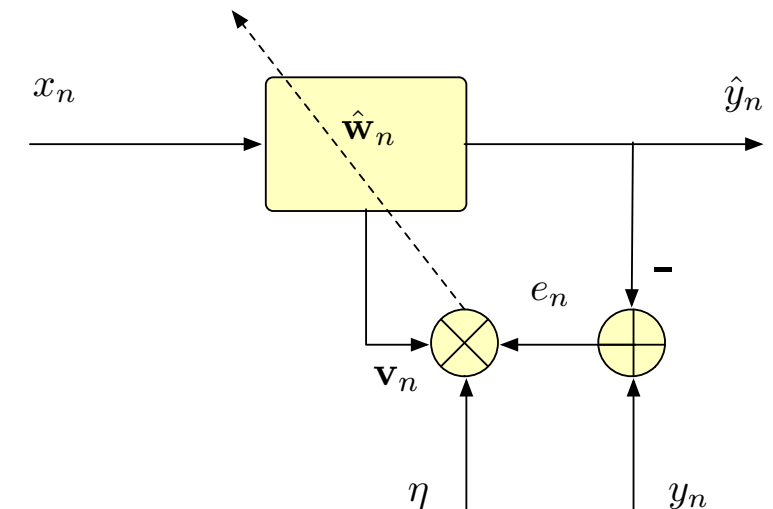
LMS algorithm:

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta(y_n - \hat{\mathbf{w}}_n^t \mathbf{v}_n) \mathbf{v}_n$$

If  $\mathbf{R}_{\mathbf{v}_n}$  and  $\mathbf{r}_{\mathbf{v}_n y} = \mathbb{E} \{ \mathbf{v}_n(u) y(u) \}$  do not change with  $n$ ,

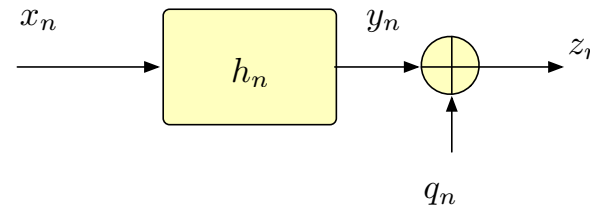
$$\hat{\mathbf{w}}_n \longrightarrow \approx \mathbf{w}_{\text{LMMSE}} = \mathbf{R}_{\mathbf{v}}^{-1} \mathbf{r}_{\mathbf{v}y}$$

If these correlations vary with time, the LMS filter will **adaptively track** them



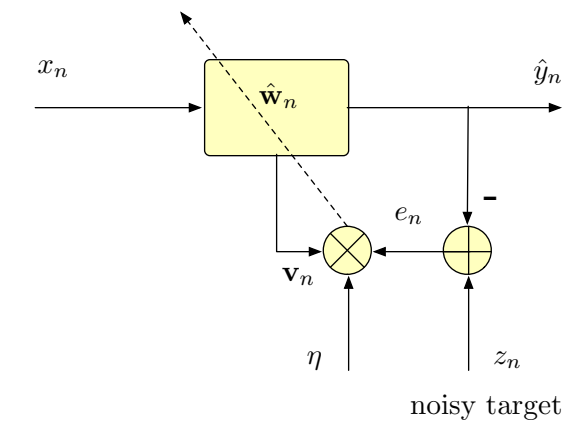
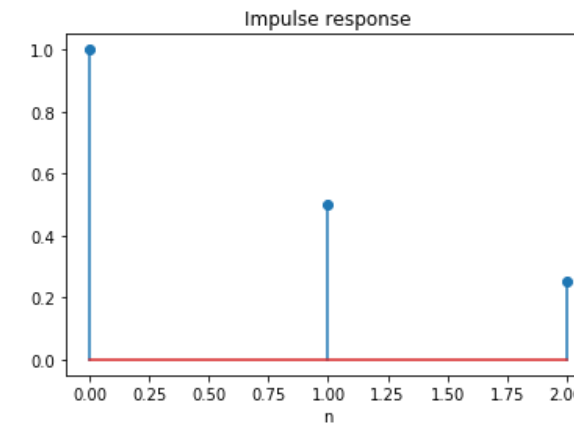
# LMS Experiment Example

data generated using:

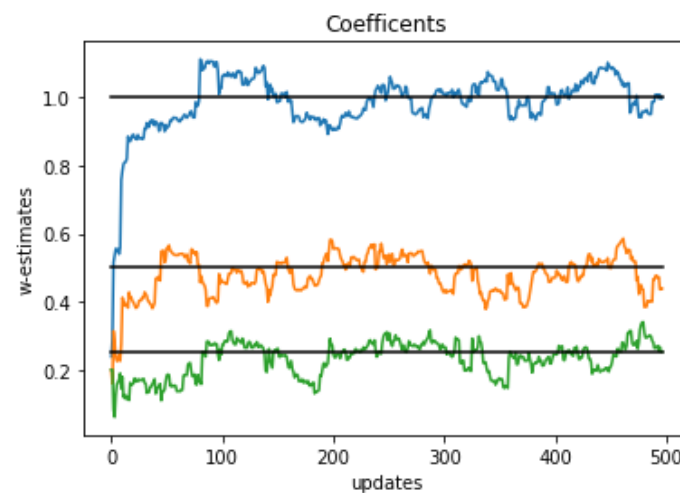


this is the ideal case as the model and data are matched

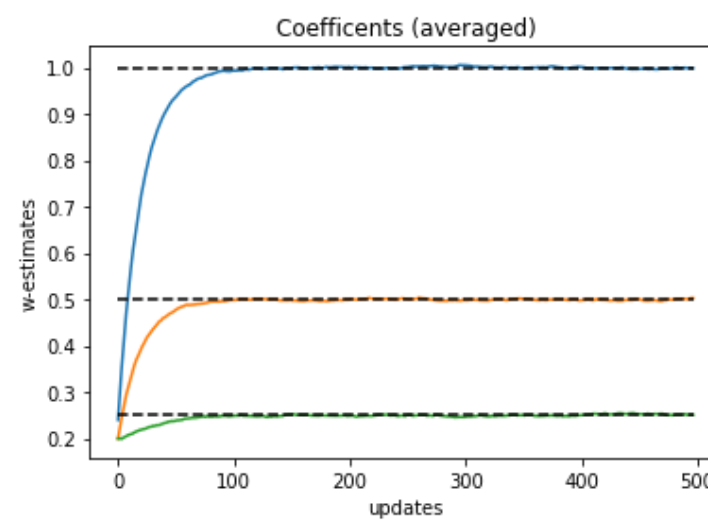
you have HW with mismatches



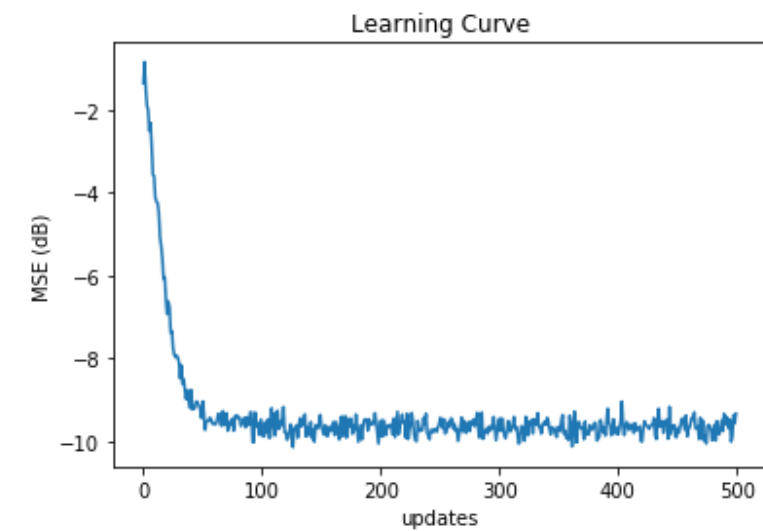
$\eta = 0.05$ ,  $\text{SNR} = 10 \text{ dB}$



single run



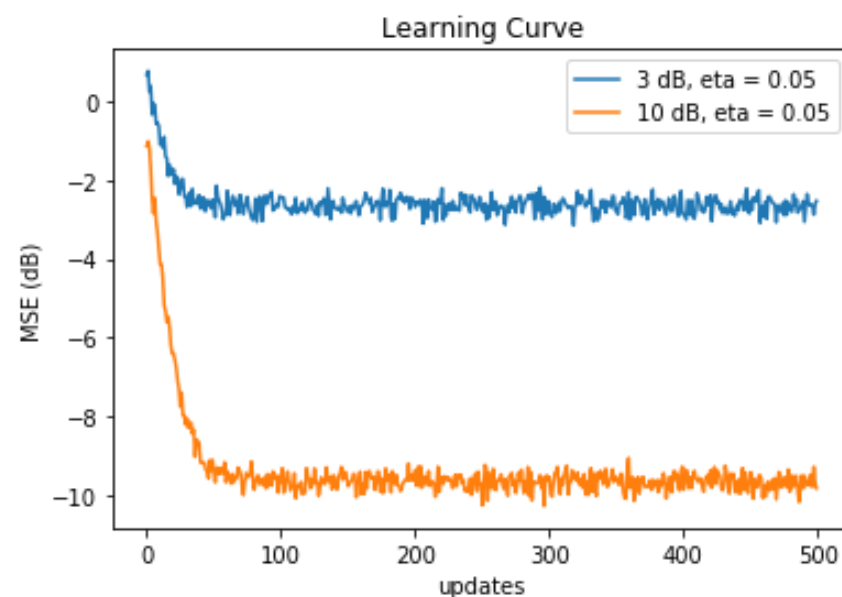
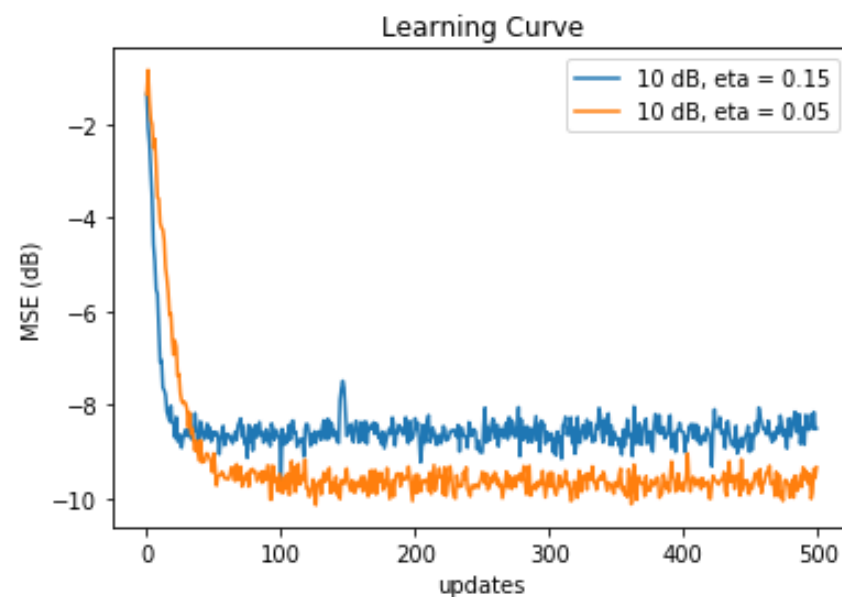
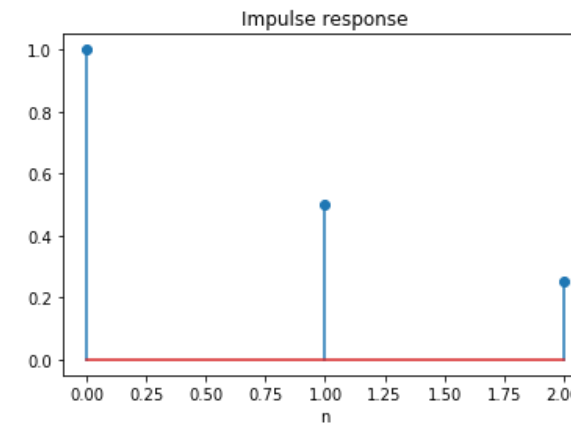
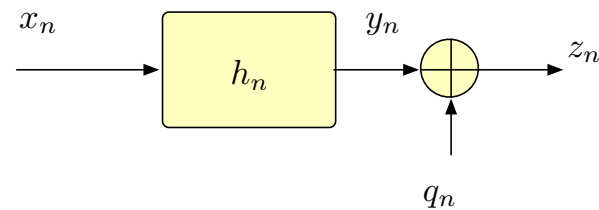
averaged over 500 runs



averaged over 500 runs

# LMS Experiment Example

data generated using:



larger learning rate means faster convergence but more misalignment (gradient noise)

even the optimal Wiener (LMMSE) filter will have higher MMSE when the SNR is lower

# LMS History/Example

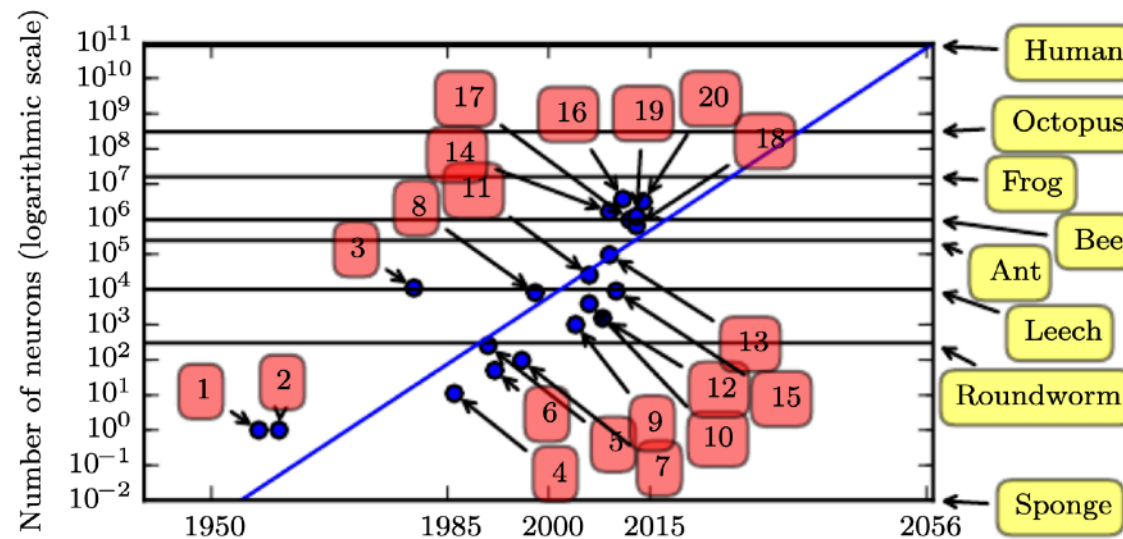


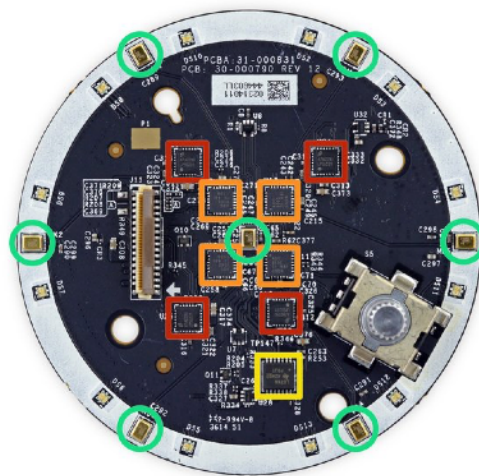
Figure 1.11: Increasing neural network size over time. Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from [Wikipedia \(2015\)](#).

1. Perceptron ([Rosenblatt, 1958, 1962](#))
2. Adaptive linear element ([Widrow and Hoff, 1960](#))
3. Neocognitron ([Fukushima, 1980](#))
4. Early back-propagation network ([Rumelhart et al., 1986b](#))
5. Recurrent neural network for speech recognition ([Robinson and Fallside, 1991](#))
6. Multilayer perceptron for speech recognition ([Bengio et al., 1991](#))
7. Mean field sigmoid belief network ([Saul et al., 1996](#))
8. LeNet-5 ([LeCun et al., 1998b](#))
9. Echo state network ([Jaeger and Haas, 2004](#))
10. Deep belief network ([Hinton et al., 2006](#))
11. GPU-accelerated convolutional network ([Chellapilla et al., 2006](#))
12. Deep Boltzmann machine ([Salakhutdinov and Hinton, 2009a](#))
13. GPU-accelerated deep belief network ([Raina et al., 2009](#))
14. Unsupervised convolutional network ([Jarrett et al., 2009](#))
15. GPU-accelerated multilayer perceptron ([Ciresan et al., 2010](#))
16. OMP-1 network ([Coates and Ng, 2011](#))
17. Distributed autoencoder ([Le et al., 2012](#))
18. Multi-GPU convolutional network ([Krizhevsky et al., 2012](#))
19. COTS HPC unsupervised convolutional network ([Coates et al., 2013](#))
20. GoogLeNet ([Szegedy et al., 2014a](#))

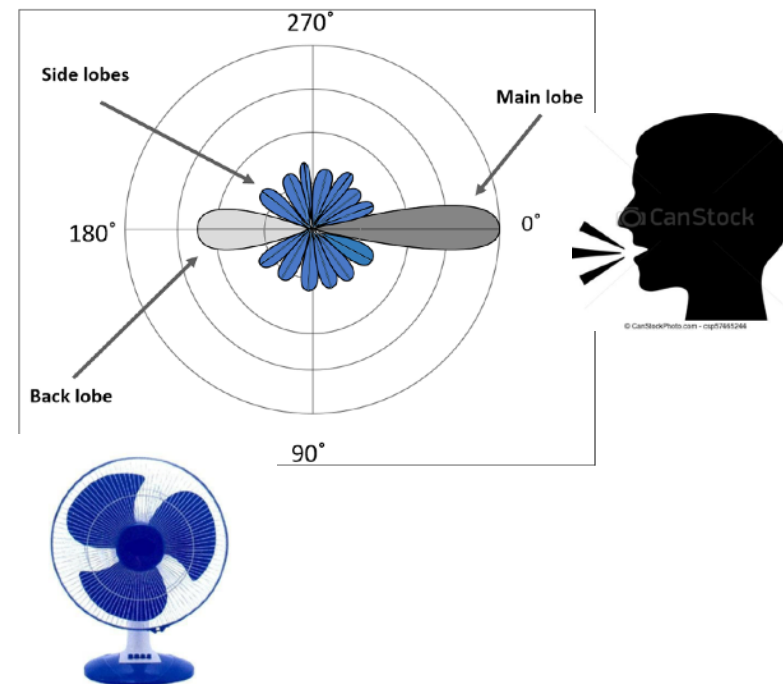


# LMS History/Example

Widrow and Hoff, Adaptive Linear Element (ADALINE)  
(developed LMS for adaptive antenna array processing)



[ifixit.com](https://www.ifixit.com)



point a beam at the desired speaker and learn to cancel  
noise energy in other directions using LMS

# MMSE Summary

1. Estimation using statistical models

2. Best MMSE estimator (unconstrained) is conditional expectation

*/. Requires complete statistical description of observed and desired — i.e.,  $p(\mathbf{y}|\mathbf{x})$*

3. Linear/affine MMSE estimator have closed form equations

*/. Require only the second moment description of observed and desired — i.e., means, correlations*

4. For jointly Gaussian observed and desired, 2 & 3 are the same!

5. The LMS algorithm may be viewed as approximating the gradient of the LMMSE cost function by a single realization.

# Regression Overview

- Regression is data fitting to a specific parameterized function class
- Linear regression
  - Same as LMMSE, but with data averages replacing expectation (ensemble averages)
    - *Linear least-squares*
  - Generalize on-line learning to full-batch and mini-batches
- Regularization (after decision theory)
- Logistic Regression (after decision theory)

# General Regression Problem

Given a data set:  $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=0}^{N-1}$

General regression problem:

$$\min_{\Theta} \langle C(\mathbf{y}, \mathbf{g}(\mathbf{x}; \Theta)) \rangle_{\mathcal{D}} \quad \Theta_{\text{opt}} = \arg \min_{\Theta} \langle C(\mathbf{y}, \mathbf{g}(\mathbf{x}; \Theta)) \rangle_{\mathcal{D}} \quad \hat{\mathbf{y}} = \mathbf{g}(\mathbf{x}; \Theta_{\text{opt}})$$

$\mathbf{x} \sim$  regressor (observed)  
 $\mathbf{y} \sim$  target (desired)

Empirical expectation (average over data):

$$\langle \mathbf{h}(\mathbf{x}, \mathbf{y}) \rangle_{\mathcal{S}} \triangleq \frac{1}{|\mathcal{S}|} \sum_{(\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{S}} \mathbf{h}(\mathbf{x}_n, \mathbf{y}_n)$$

For large averaging sets (i.e., many realizations):

$$\mathbb{E} \{ \mathbf{h}(\mathbf{x}(u), \mathbf{y}(u)) \} = \int \mathbf{h}(\mathbf{x}, \mathbf{y}) p_{\mathbf{x}(u), \mathbf{y}(u)}(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \approx \langle \mathbf{h}(\mathbf{x}, \mathbf{y}) \rangle_{\mathcal{S}}$$

sample mean

Monte Carlo method

# Least-Squares Regression Problem

$$\min_{\Theta} \langle \|\mathbf{y} - \mathbf{g}(\mathbf{x}; \Theta)\|^2 \rangle_{\mathcal{D}} \iff \min_{\Theta} \sum_{n=0}^{N-1} \|\mathbf{y}_n - \mathbf{g}(\mathbf{x}_n; \Theta)\|^2$$

$$\Theta_{\text{opt}} = \arg \min_{\Theta} \langle \|\mathbf{y} - \mathbf{g}(\mathbf{x}; \Theta)\|^2 \rangle_{\mathcal{D}}$$

Squared-error is a common cost function in (electrical) engineering

this corresponds to power or energy in many applications

# Linear and Affine (LS) Regression

Linear regression problem:

$$\min_{\mathbf{W}} \langle \|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2 \rangle_{\mathcal{D}} \iff \min_{\mathbf{W}} \sum_{n=0}^{N-1} \|\mathbf{y}_n - \mathbf{W}\mathbf{x}_n\|^2$$

$$\mathbf{W}_{\text{LLSE}} = \arg \min_{\mathbf{W}} \langle \|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2 \rangle_{\mathcal{D}}$$
$$\hat{\mathbf{y}} = \mathbf{W}_{\text{LLSE}} \mathbf{x}$$

Affine regression problem: (aka: Linear regression)

$$\mathbf{W}_{\text{ALSE}}, \mathbf{b}_{\text{ALSE}} = \arg \min_{\mathbf{W}, \mathbf{b}} \langle \|\mathbf{y} - [\mathbf{W}\mathbf{x} + \mathbf{b}]\|^2 \rangle_{\mathcal{D}}$$

$$\hat{\mathbf{y}} = \mathbf{W}_{\text{ALSE}} \mathbf{x} + \mathbf{b}_{\text{ALSE}}$$

# Linear and Affine (LS) Regression Solution

Note that the data averaging operator has the same linearity property as the expectation operator

$$\mathbb{E} \{L(\mathbf{x}(u))\} = L(\mathbb{E} \{\mathbf{x}(u)\})$$

$$\langle L(\mathbf{x}) \rangle = L(\langle \mathbf{x} \rangle)$$

This means the solutions are the same as the MMSE solutions with the expectation replaces by data averaging

For example, Linear LS regression:

$$\mathbf{W}_{\text{LLSE}} = \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}} \hat{\mathbf{R}}_{\mathbf{x}}^{-1}$$

$$\hat{\mathbf{y}} = \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}} \hat{\mathbf{R}}_{\mathbf{x}}^{-1} \mathbf{x}$$

$$\text{LLSE} = \langle \|\mathbf{y} - \mathbf{W}_{\text{LLSE}} \mathbf{x}\|^2 \rangle$$

$$= \langle \|\mathbf{y}\|^2 - \langle \|\mathbf{W}_{\text{LLSE}} \mathbf{x}\|^2 \rangle \rangle_{\mathcal{D}}$$

$$= \text{tr} \left( \hat{\mathbf{R}}_{\mathbf{y}} - \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}} \hat{\mathbf{R}}_{\mathbf{x}}^{-1} \hat{\mathbf{R}}_{\mathbf{x}\mathbf{y}} \right)$$

$$\hat{\mathbf{R}}_{\mathbf{x}} = \langle \mathbf{x} \mathbf{x}^t \rangle_{\mathcal{D}}$$

$$\hat{\mathbf{R}}_{\mathbf{x}\mathbf{y}} = \langle \mathbf{x} \mathbf{y}^t \rangle_{\mathcal{D}}$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}_n \mathbf{x}_n^t$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}_n \mathbf{y}_n^t$$

# Proof for LLSE Regression

$$\min_{\mathbf{W}} \langle \|\mathbf{y} - \mathbf{W}\mathbf{x}\|^2 \rangle_{\mathcal{D}}$$

Proof:

$$\begin{aligned} \text{LSE}(\mathbf{G}) &= \langle \|\mathbf{y} - \mathbf{G}\mathbf{x}\|^2 \rangle \\ &= \langle \|(\mathbf{y} - \mathbf{G}_{\text{opt}}\mathbf{x}) + (\mathbf{G}_{\text{opt}} - \mathbf{G})\mathbf{x}\|^2 \rangle \\ &= \langle \|(\mathbf{y} - \mathbf{G}_{\text{opt}}\mathbf{x})\|^2 \rangle + \text{tr} \left( (\mathbf{G}_{\text{opt}} - \mathbf{G})\hat{\mathbf{R}}_{\mathbf{x}}(\mathbf{G}_{\text{opt}} - \mathbf{G})^t \right) \\ &\quad + 2\text{tr} \left( (\hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}} - \mathbf{G}_{\text{opt}}\hat{\mathbf{R}}_{\mathbf{x}})(\mathbf{G}_{\text{opt}} - \mathbf{G})^t \right) \end{aligned}$$

use:

$$\mathbf{v}^t \mathbf{w} = \text{tr}(\mathbf{w}\mathbf{v}^t)$$

if:

$$\mathbf{G}_{\text{opt}}\hat{\mathbf{R}}_{\mathbf{x}} = \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}$$

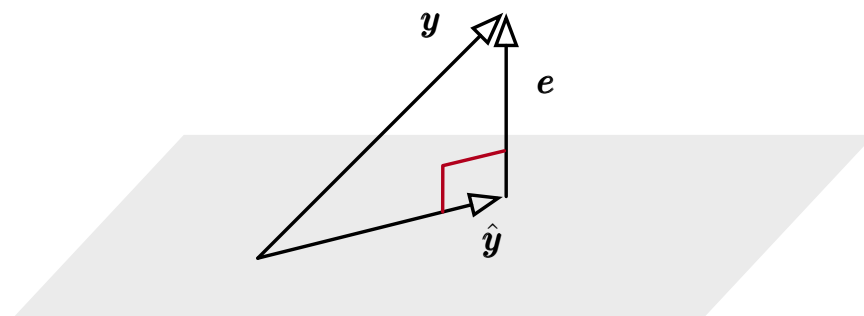
(Normal Equation(s))  
(aka Orthogonality Principle)

then:

$$\text{LSE}(\mathbf{G}) = \langle \|(\mathbf{y} - \mathbf{G}_{\text{opt}}\mathbf{x})\|^2 \rangle + \underbrace{\text{tr} \left( (\mathbf{G}_{\text{opt}} - \mathbf{G})\hat{\mathbf{R}}_{\mathbf{x}}(\mathbf{G}_{\text{opt}} - \mathbf{G})^t \right)}_{\geq 0 \ \forall \ \mathbf{G}, \text{ since } \hat{\mathbf{R}}_{\mathbf{x}} \text{ is nnd}}$$

because of orthogonality principle

$$\begin{aligned} \langle \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \rangle &= \langle \|\mathbf{y}\|^2 \rangle - \langle \|\hat{\mathbf{y}}\|^2 \rangle \\ &= \text{tr} \left( \hat{\mathbf{R}}_{\mathbf{y}} - \hat{\mathbf{R}}_{\mathbf{y}\mathbf{x}}\hat{\mathbf{R}}_{\mathbf{x}}^{-1}\hat{\mathbf{R}}_{\mathbf{x}\mathbf{y}} \right) \end{aligned}$$



space of all estimates/approximations

looks familiar...



# Linear and Affine (LS) Regression Solution

This makes perfect intuitive sense:

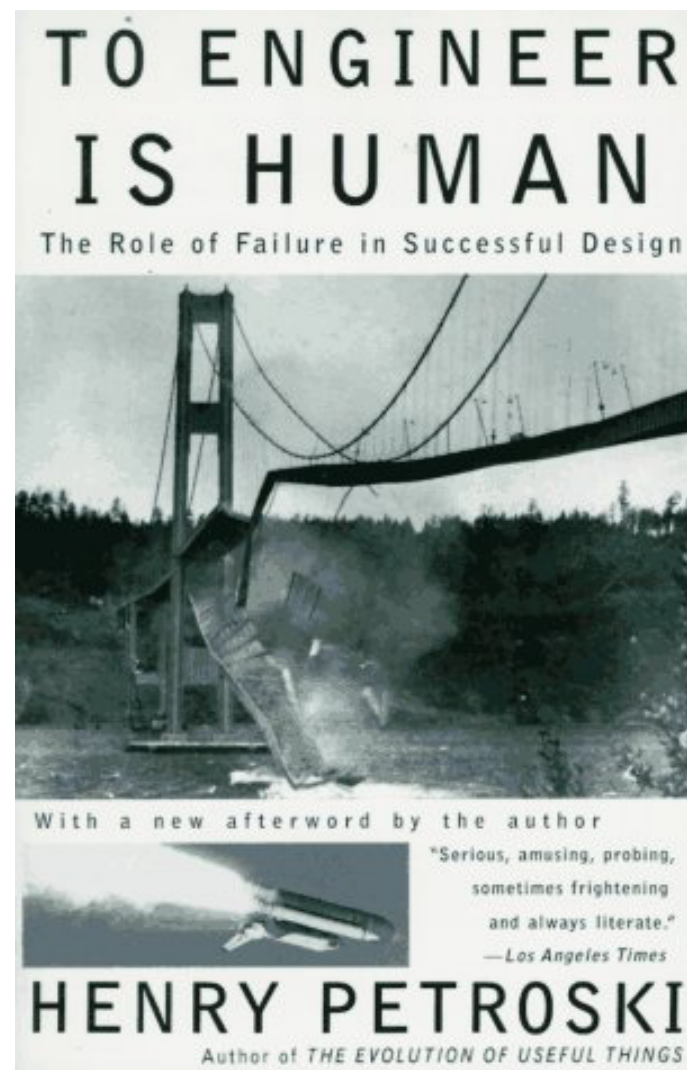
*if you did not know the second moments and wanted to do LMMSE estimation, you would estimate these correlations from data*

*in addition to optimality in the Gaussian case, linear MMSE estimation is extremely popular because it **takes much less data to accurately estimate second moments** than to do so for complete statistical descriptions (or higher moments)*

# To Engineer is Human

there is no purely “model based” approach to any engineering

*cannot really separate modeling/data or frequentist/Bayesian views*

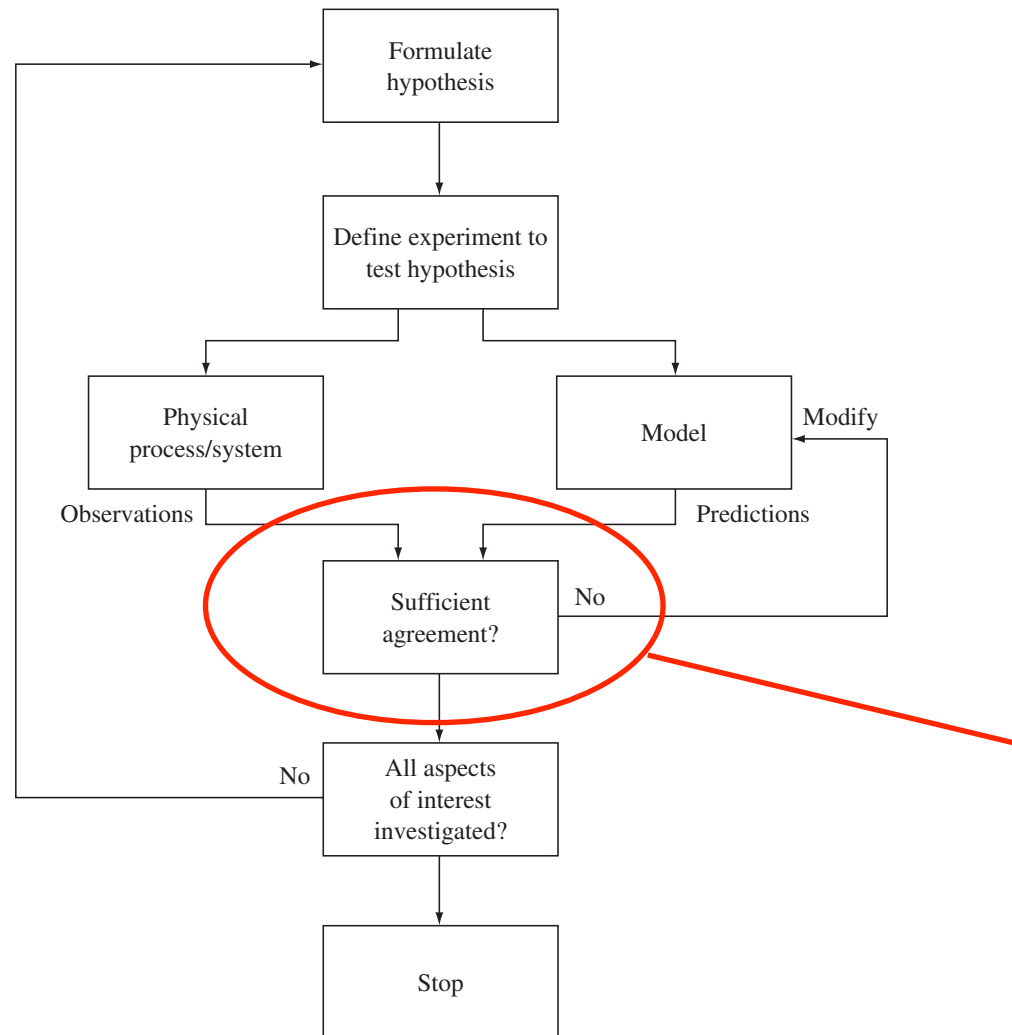


big part of engineering

*trial and error,  
design - test - refine,  
iterative design*

*if you want to be a good  
engineer, you should be  
adventuresome and make lots  
mistakes to **learn** from*

# Data, Experiments, and Models



**FIGURE 1.1**  
The modeling process.

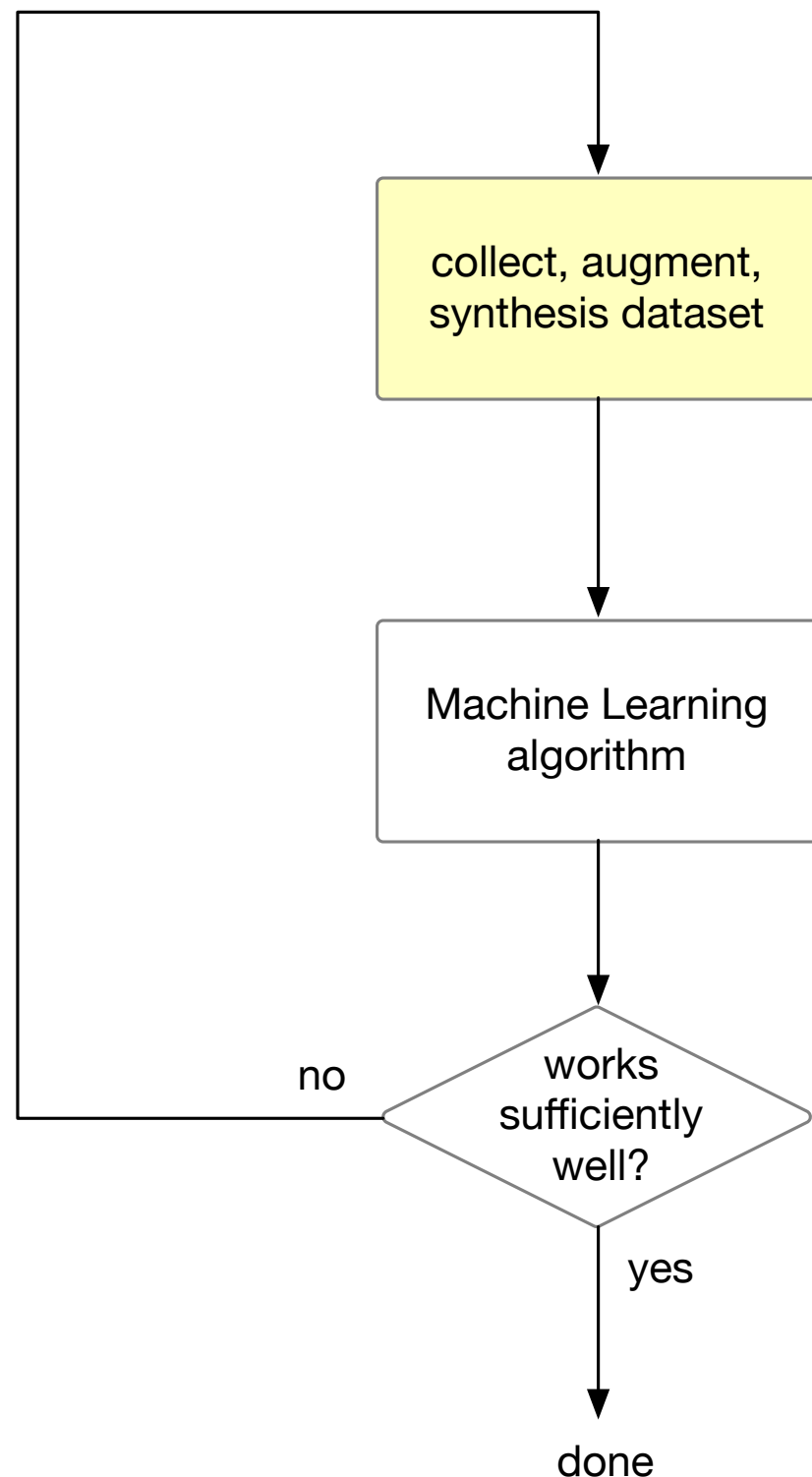
[Leon-Garcia, Probability Statistics, and Random Processes for Engineers]

there is no purely “model based” approach  
to any engineering problem

“All models are wrong, but some are useful”

*George Box (paraphrased)*

# Data Driven Version of this Design Process



data (and ML) evaluated via  
end-to-end performance

much of the attention is here, but  
in practice, more iteration/time  
spent on data engineering

“all data are wrong, but some are useful”

# LLSE Regression: scalar on scalar special case

approximate  $y$  from  $x$

Linear regression problem:

$$\min_w \langle (y - wx)^2 \rangle \iff \min_w \frac{1}{N} \sum_{n=0}^{N-1} (y_n - wx_n)^2$$

Solution (special case):

$$w_{\text{LLSE}} = \frac{\hat{r}_{yx}}{\hat{r}_x}$$

$$\hat{y} = \frac{\hat{r}_{yx}}{\hat{r}_x} x$$

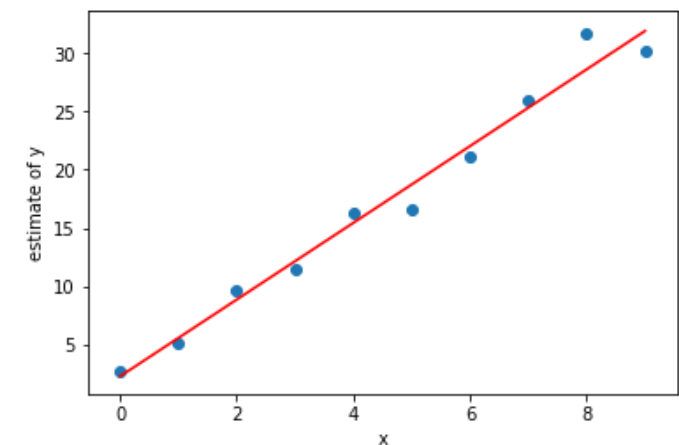
$$\hat{r}_x = \langle x^2 \rangle = \frac{1}{N} \sum_{n=0}^{N-1} x_n^2$$

$$\hat{r}_{yx} = \langle yx \rangle = \frac{1}{N} \sum_{n=0}^{N-1} y_n x_n$$

$$\begin{aligned} \text{LLSE} &= \langle [y - w_{\text{LLSE}}x]^2 \rangle \\ &= \langle y^2 \rangle - \langle [w_{\text{LLSE}}x]^2 \rangle \\ &= \hat{r}_y - \hat{r}_{yx}^2 \hat{r}_x \end{aligned}$$

when sample means are 0:

$$= \hat{\sigma}_y^2 (1 - \hat{\rho}^2)$$



# LLSE Regression: scalar on scalar special case

approximate  $y$  from  $x$

Linear regression problem:

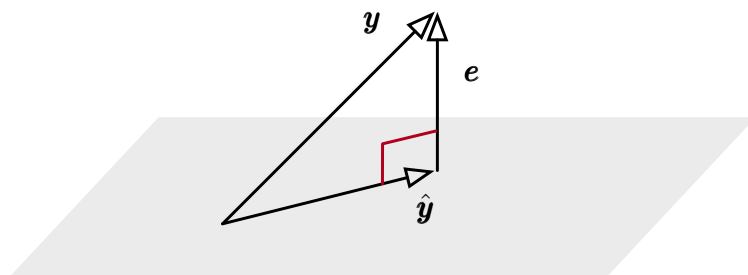
$$\min_w \langle (y - wx)^2 \rangle \iff \min_w \frac{1}{N} \sum_{n=0}^{N-1} (y_n - wx_n)^2 \iff \|\mathbf{y} - w\mathbf{x}\|^2$$

Solution (special case):

$$\mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

$$w_{\text{LLSE}} = \frac{\mathbf{y}^t \mathbf{x}}{\mathbf{x}^t \mathbf{x}} \quad (N)\text{LLSE} = \|\mathbf{y}\|^2 - \left( \frac{\mathbf{y}^t \mathbf{x}}{\mathbf{x}^t \mathbf{x}} \right)^2 \|\mathbf{x}\|^2$$

$$\hat{\mathbf{y}} = \frac{\mathbf{y}^t \mathbf{x}}{\mathbf{x}^t \mathbf{x}} \mathbf{x} \quad = \|\mathbf{y}\|^2 - \frac{(\mathbf{y}^t \mathbf{x})^2}{\|\mathbf{x}\|^2}$$



space of all estimates/approximations

this “stacked” approach yields the same as the  $\langle . \rangle$  approach on the previous slides!!

$\hat{y}$ -hat stacked in a vector

# LLSE Regression: scalar on vector special case

approximate  $y$  from  $\mathbf{x}$

Linear regression problem:

$$\min_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle \quad \Longleftrightarrow \quad \min_{\mathbf{w}} \frac{1}{N} \sum_{n=0}^{N-1} (y_n - \mathbf{w}^t \mathbf{x})^2$$

$$\hat{y} = \mathbf{w}^t \mathbf{x}$$

$$\mathbf{w} = \hat{\mathbf{R}}_{\mathbf{x}}^{-1} \hat{\mathbf{r}}_{\mathbf{x}y}$$

$$\hat{\mathbf{r}}_{\mathbf{x}y} = \hat{\mathbf{R}}_{\mathbf{x}y} = \langle \mathbf{x}y \rangle$$

$$\hat{\mathbf{R}}_{\mathbf{x}} \mathbf{w} = \hat{\mathbf{r}}_{\mathbf{x}y}$$

Normal Equations

$$\text{LLSE} = \hat{r}_y^2 - \hat{\mathbf{r}}_{\mathbf{x}y}^t \mathbf{R}_{\mathbf{x}}^{-1} \hat{\mathbf{r}}_{\mathbf{x}y}$$

$$\hat{\mathbf{R}}_{\mathbf{x}} \mathbf{w} = \hat{\mathbf{r}}_{\mathbf{x}y}$$

again, just change  $E\{.\}$  to  $\langle . \rangle$  from LMMSE result

what about the “stacked” approach for this case??

# LLSE Regression: scalar on vector special case

approximate  $y$  from  $\mathbf{x}$

Linear regression problem:

$$\min_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle \iff \min_{\mathbf{w}} \frac{1}{N} \sum_{n=0}^{N-1} (y_n - \mathbf{w}^t \mathbf{x})^2 \iff \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

Solution (special case):

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0^t \\ \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_{N-1}^t \end{bmatrix} \quad \mathbf{X}^t = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{N-1} \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

$$\mathbf{w} = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}$$

$$(N) \text{ LLSE} = \text{tr} (\|\mathbf{y}\|^2 - \|\mathbf{P}_x \mathbf{y}\|^2)$$

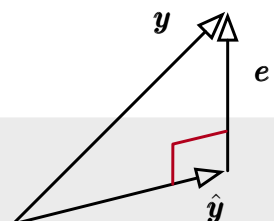
$$\hat{\mathbf{y}} = \mathbf{X} (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}$$

$$= \text{tr} (\|(\mathbf{I} - \mathbf{P}_x) \mathbf{y}\|^2)$$

$$\mathbf{X}^t \mathbf{X} = \mathbf{X}^t \mathbf{y}$$

normal equations

$$= \mathbf{P}_x \mathbf{y}$$



this is the same as  $\langle . \rangle$  case, with all  $y$ -hat stacked in a vector

$$\hat{\mathbf{R}}_x^{-1} \hat{\mathbf{r}}_{xy} = \left( \frac{1}{N} \mathbf{X}^t \mathbf{X} \right)^{-1} \left[ \frac{1}{N} \mathbf{X}^t \mathbf{y} \right]$$



# LLSE Regression: scalar on vector special case

approximate  $y$  from  $\mathbf{x}$

Linear regression problem:

$$\min_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle \iff \min_{\mathbf{w}} \frac{1}{N} \sum_{n=0}^{N-1} (y_n - \mathbf{w}^t \mathbf{x})^2 \iff \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

Solution (special case):

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_0^t \\ \mathbf{x}_1^t \\ \vdots \\ \mathbf{x}_{N-1}^t \end{bmatrix} \quad \mathbf{X}^t = \begin{bmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_{N-1} \end{bmatrix}$$

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

$$\mathbf{w} = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}$$

$$(N) \text{ LLSE} = \text{tr} (\|\mathbf{y}\|^2 - \|\mathbf{P}_x \mathbf{y}\|^2)$$

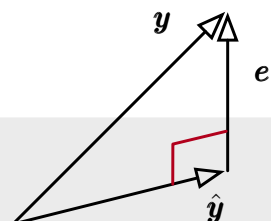
$$\hat{\mathbf{y}} = \mathbf{X} (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}$$

$$= \text{tr} (\|(\mathbf{I} - \mathbf{P}_x) \mathbf{y}\|^2)$$

$$\mathbf{X}^t \mathbf{X} = \mathbf{X}^t \mathbf{y}$$

normal equations

$$= \mathbf{P}_x \mathbf{y}$$



this is the same as  $\langle . \rangle$  case, with all  $y$ -hat stacked in a vector

$$\hat{\mathbf{R}}_x^{-1} \hat{\mathbf{r}}_{xy} = \left( \frac{1}{N} \mathbf{X}^t \mathbf{X} \right)^{-1} \left[ \frac{1}{N} \mathbf{X}^t \mathbf{y} \right]$$

## “stacked” vs $\langle . \rangle$ approach

The stacked approach is used by all of the books I see...

maybe because they start from frequentist (data first) perspective

for EE students with MMSE background,  $\langle . \rangle$  makes it obvious  
what is going on in Linear LSE regression

also makes it simple to see the case when regression vector  $\mathbf{y}$  on vector  $\mathbf{x}$   
*(this would require 3D tensors in stacked approach)*

will show the general case (Projection Theorem), but first, what  
about using Gradient Descent??

# Trick for Doing Affine with Linear Math

$$\min_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle \iff \min_{\mathbf{w}} \frac{1}{N} \sum_{n=0}^{N-1} (y_n - \mathbf{w}^t \mathbf{x})^2$$

$$\hat{y} = \left[ \mathbf{x}^t \mid 1 \right] \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

$$= \mathbf{x}^t \mathbf{w} + b$$

$$\hat{\mathbf{y}} = \left[ \mathbf{X} \mid \mathbf{1} \right] \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$

$$= \mathbf{X} \mathbf{w} + b \mathbf{1}$$

$$\left[ \mathbf{X} \mid \mathbf{1} \right] = \left[ \begin{array}{c|c} \mathbf{x}_0^t & 1 \\ \mathbf{x}_1^t & 1 \\ \vdots & \\ \mathbf{x}_{N-1}^t & 1 \end{array} \right]$$

allows for compact notation while including the b (bias) term

# Recall: Steepest Descent and LMS

estimate  $y(u)$  from  $\mathbf{x}(u)=\mathbf{x}$

$$E(\mathbf{w}) = \mathbb{E} \left\{ [y(u) - \mathbf{w}^t \mathbf{x}(u)]^2 \right\}$$

$$\begin{aligned} \nabla_{\mathbf{w}} E &= 2 \left( \mathbb{E} \left\{ \mathbf{w} \mathbf{x}(u) \mathbf{x}^t(u) - y(u) \mathbf{x}(u) \right\} \right) \\ &= 2(\mathbf{w}^t \mathbf{R}_{\mathbf{x}} - \mathbf{r}_{\mathbf{x}y}) \end{aligned}$$

Steepest descent using (ensemble average) gradient:

$$\begin{aligned} \hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n - (\eta/2) \nabla_{\mathbf{w}} E \\ &= \hat{\mathbf{w}}_n + \eta(\mathbf{r}_{\mathbf{x}y} - \mathbf{R}_{\mathbf{x}} \hat{\mathbf{w}}_n) \end{aligned}$$

Single Point Stochastic Gradient Descent:

$$\begin{aligned} -\frac{1}{2} \nabla_{\mathbf{w}} E &= \mathbf{r}_{\mathbf{x}y} - \mathbf{R}_{\mathbf{x}} \mathbf{w} \\ &= \mathbb{E} \left\{ y(u) \mathbf{x}(u) - \mathbf{x}(u) \mathbf{x}^t(u) \mathbf{w} \right\} \\ &\approx y_n \mathbf{x}_n - \mathbf{x}_n \mathbf{x}_n^t \mathbf{w} \\ &= (y_n - \mathbf{x}_n^t \hat{\mathbf{w}}_n) \mathbf{x}_n \end{aligned}$$

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta(y_n - \hat{\mathbf{w}}_n^t \mathbf{x}_n) \mathbf{x}_n$$

this is called “on-line learning”

when  $n \sim$  time, this is the Least Mean Square (LMS) adaptive filter

when  $n \sim!$  time, we can easily average the gradient over more data points for a better approximation (batches)

# LMS/Stochastic Gradient from LLSE Regression POV

approximate  $y$  from  $\mathbf{x}$

$$\min_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle$$

$$\begin{aligned} \nabla_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle &= 2 \langle \mathbf{x} \mathbf{x}^t \mathbf{w} - \mathbf{x} y \rangle \\ &= 2(\hat{\mathbf{R}}_{\mathbf{x}} \mathbf{w} - \hat{\mathbf{r}}_{\mathbf{x}y}) \end{aligned}$$

Steepest descent using (data average) gradient:

$$\begin{aligned} \hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n - (\eta/2) \nabla_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle \\ &= \hat{\mathbf{w}}_n + \eta(\hat{\mathbf{r}}_{\mathbf{x}y} - \hat{\mathbf{R}}_{\mathbf{x}} \hat{\mathbf{w}}_n) \end{aligned}$$

Stochastic Gradient Descent with mini-batches:

SGD with mini-batch updating  
(average the gradient over subset of data)

$$\hat{\mathbf{w}}_{n+1} = \hat{\mathbf{w}}_n + \eta \frac{1}{|\mathcal{B}|} \left[ \sum_{(x_n, y_n) \in \mathcal{B}_n} (y_n - \mathbf{x}_n^t \hat{\mathbf{w}}_n) \mathbf{x}_n \right]$$

mini-batch size  $l$  is on-line learning (LMS is an example)

$$\begin{aligned} -\frac{1}{2} \nabla_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle &= \hat{\mathbf{r}}_{\mathbf{x}y} - \hat{\mathbf{R}}_{\mathbf{x}} \mathbf{w} \\ &= \langle \mathbf{x} y - \mathbf{x} \mathbf{x}^t \mathbf{w} \rangle_{\mathcal{D}} \\ &\approx \langle \mathbf{x} y - \mathbf{x} \mathbf{x}^t \mathbf{w} \rangle_{\mathcal{B}_n} \\ &= \frac{1}{|\mathcal{B}|} \sum_{(x_n, y_n) \in \mathcal{B}_n} (y_n - \mathbf{x}_n^t \hat{\mathbf{w}}_n) \mathbf{x}_n \end{aligned}$$

# A Word on the “Exact Gradient”

## Model Based View

approximate data  
average gradient

$$\nabla_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle_{\mathcal{D}}$$

approximate data  
average gradient  
(more noisy)

$$\nabla_{\mathbf{w}} \langle (y - \mathbf{w}^t \mathbf{x})^2 \rangle_{\mathcal{B}}, \quad \mathcal{B} \subset \mathcal{D}$$

Exact gradient

$$\nabla_{\mathbf{w}} \mathbb{E} \{ (y(u) - \mathbf{w}^t \mathbf{x}(u))^2 \}$$

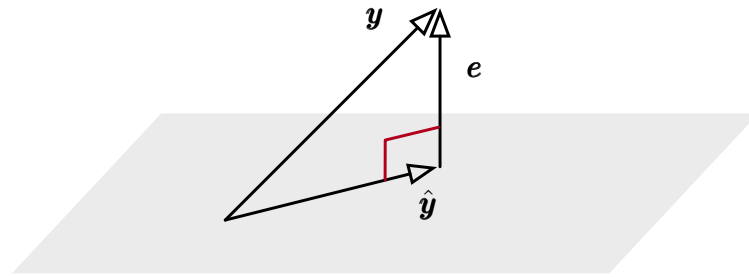
## Data Driven View

exact (full-batch)  
gradient

mini-batch gradient  
approximates full-batch  
(noisy)

hypothetical model for  
data (unreliable?)

# Hilbert Space Projection Theorem (advanced topic)



space of all estimates/approximations

This is why we see the same results so often —  
many spaces are Hilbert spaces

in machine learning, we also consider costs that are not squared error and are not  
in a Hilbert space (do not come from inner product)

in most cases, we use SGD regardless

**Note:** in all of the cases of gradient descent, the gradient is zero when the  
orthogonality principle holds (e.g., Wiener-Hopf, Normal Equations)

# Hilbert Space Projection Theorem (advanced topic)

**Theorem (HSPT):** Let  $\mathcal{H}$  be a Hilbert space,  $\mathcal{M}$  be a closed subspace of  $\mathcal{H}$ , and  $\mathbf{y} \in \mathcal{H}$ . Then there is a *unique*  $\hat{\mathbf{y}} \in \mathcal{M}$  which is closest to  $\mathbf{y}$ :

$$\|\mathbf{y} - \hat{\mathbf{y}}\| < \|\mathbf{y} - \tilde{\mathbf{y}}\| \quad \forall \tilde{\mathbf{y}} \in \mathcal{M}, \mathbf{y} \neq \hat{\mathbf{y}}.$$

Furthermore, a *necessary and sufficient* condition for  $\hat{\mathbf{y}}$  to be the closest point is that it satisfy the *Orthogonality Principle*:

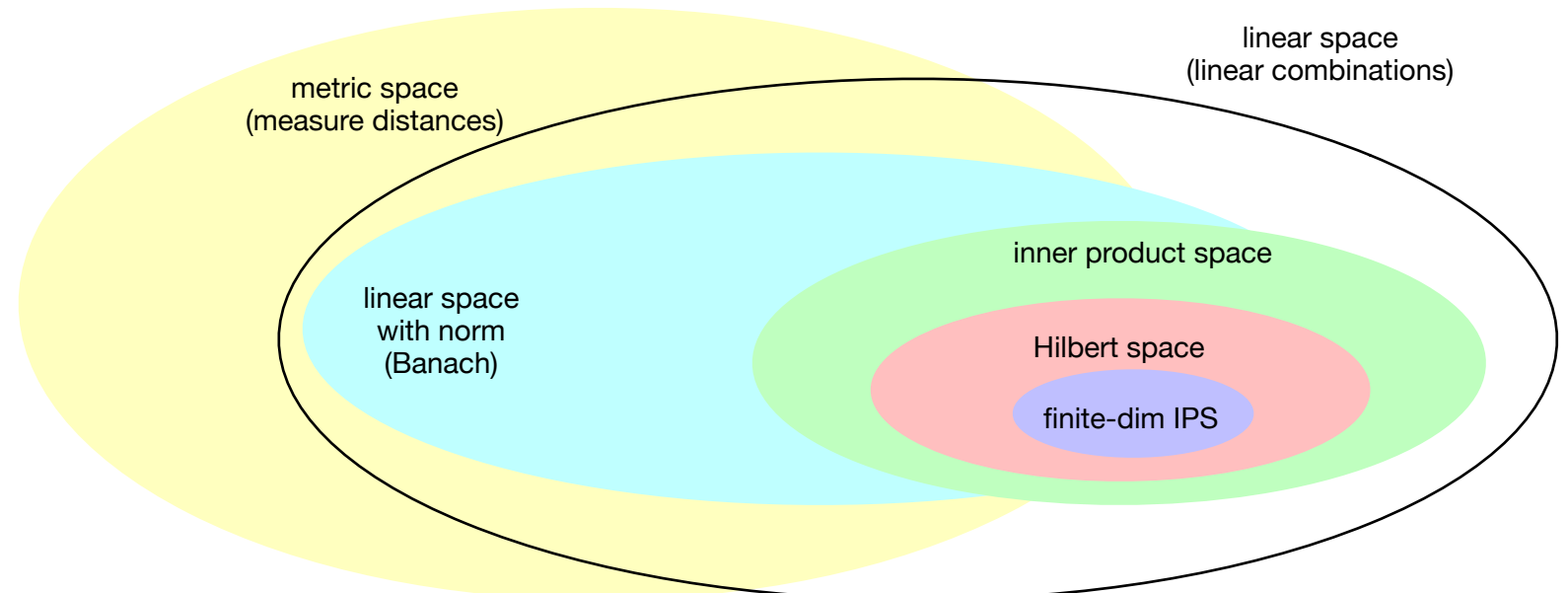
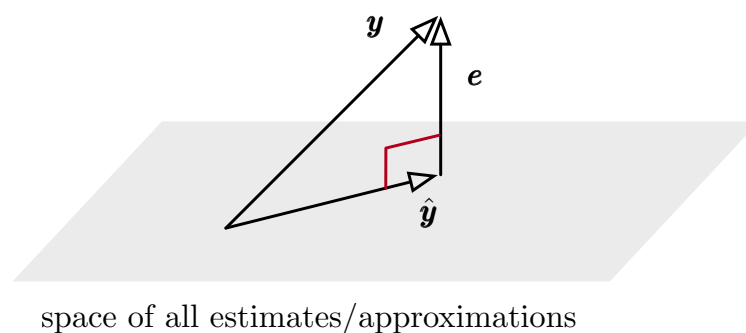
$$\langle \mathbf{y} - \hat{\mathbf{y}}, \tilde{\mathbf{y}} \rangle = 0 \quad \forall \tilde{\mathbf{y}} \in \mathcal{M}.$$

$\langle \mathbf{x}, \mathbf{y} \rangle$  is inner product here

A direct result of this orthogonality condition is

$$\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \|e\|^2 = \|\mathbf{y}\|^2 - \|\hat{\mathbf{y}}\|^2.$$

Here  $\langle \mathbf{x}, \mathbf{y} \rangle$  denotes the inner product defined on  $\mathcal{H}$  and  $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$  is the associated norm.

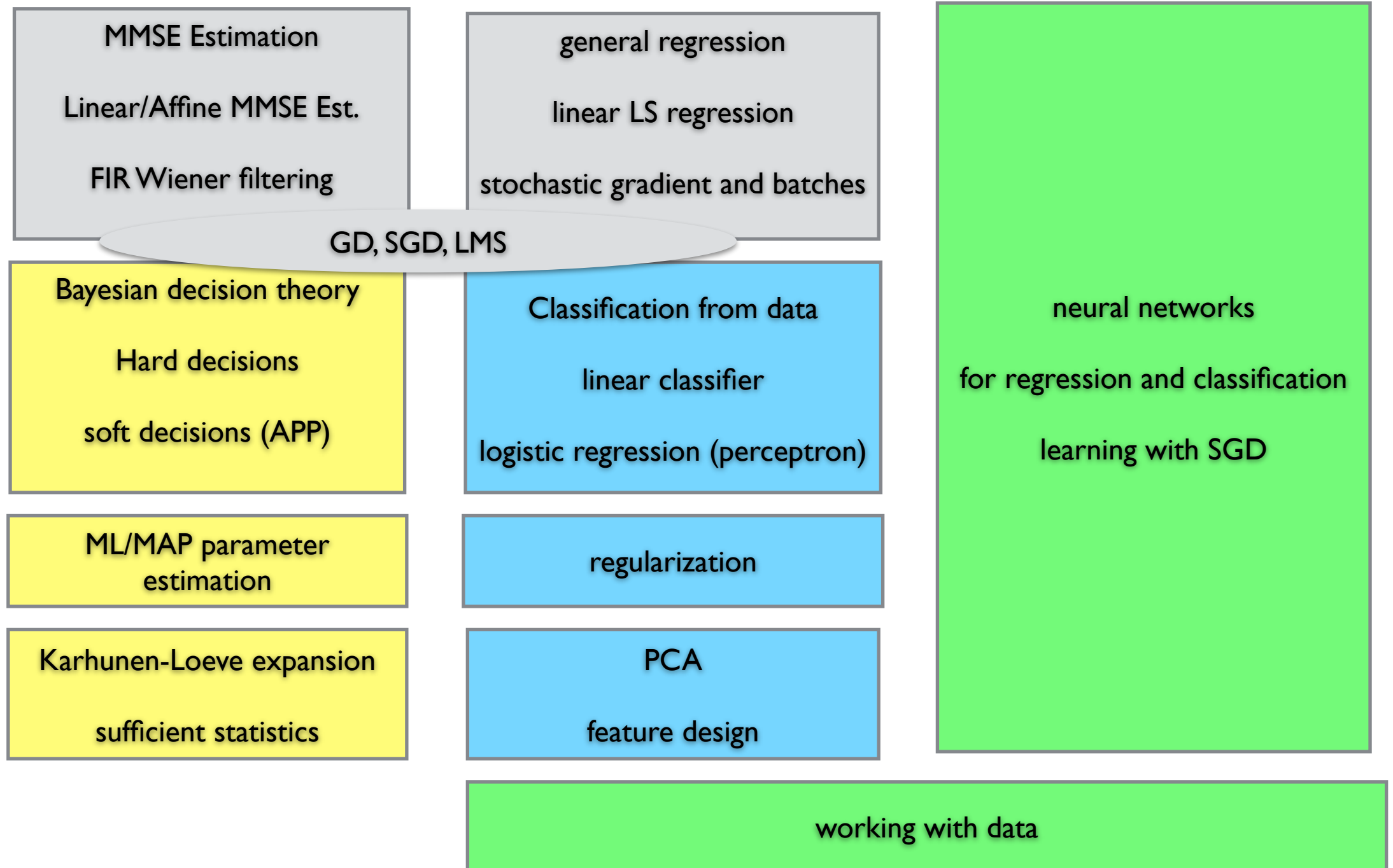




# Detection, Estimation, Regression

statistical models

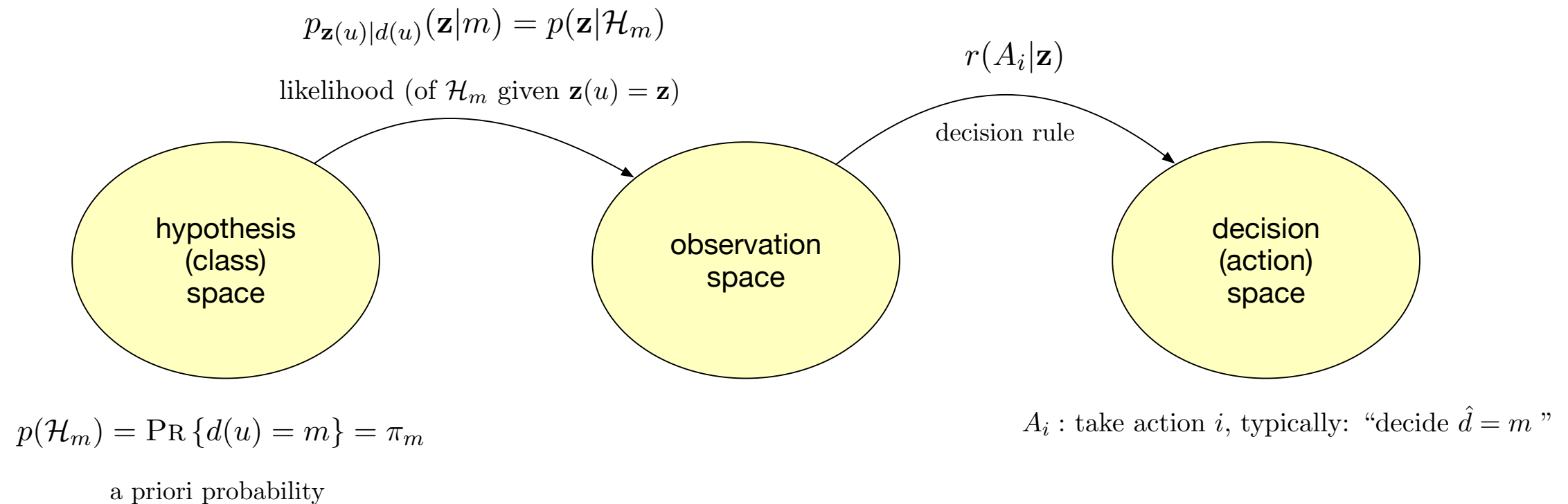
data driven



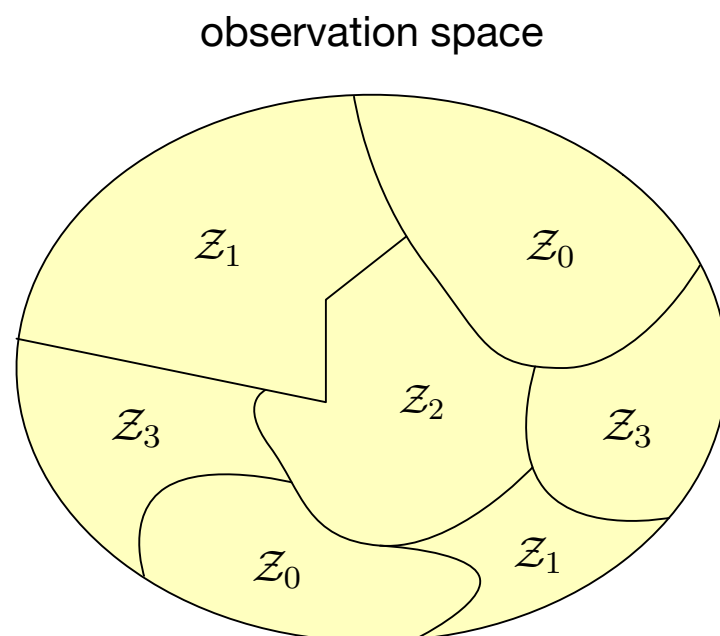
# Decision/Detection Theory

- **Bayesian Decision Theory**
  - Bayes decision rule
  - MAP rule - minimum error probability rule
- Maximum Likelihood
  - Likelihood, Negative-Log-Likelihood, Likelihood ratios
- Neyman-Pearson test and the ROC
  - Detection and False Alarm trade off

# Decision Theory Framework (Statistical Model Based)



**Goal:** design a good decision rule using the statistical model



typically try to implement the decision rule as a partitioning of the sample space

$$\mathcal{Z}_m = \text{decision region } m = \{\mathbf{z} \in \mathcal{Z} : r(A_m|\mathbf{z}) = 1\}$$

# Decision Theory Framework (Statistical Model Based)

Bayes risk for  
decision rule  $r$

$$\text{Risk}(r) = \int_{\mathbf{z}} p(\mathbf{z}) \left[ \sum_j r(A_j|\mathbf{z}) C(A_j|\mathbf{z}) \right] d\mathbf{z}$$

Cost for taking  
action  $m$  given  
observation  $\mathbf{z}$

$$C(A_j|\mathbf{z}) = \sum_i C(\mathcal{H}_i, A_j) p(\mathcal{H}_i|\mathbf{z})$$

$C_{i,j} = C(\mathcal{H}_i, A_j) = \text{Cost of deciding } \mathcal{H}_j \text{ when } \mathcal{H}_i \text{ is true}$

Bayes decision rule  
(minimizes Bayes risk)

$$r_{\text{bayes}}(A_m|\mathbf{z}) = \begin{cases} 1 & m = \arg \min_j C(A_j|\mathbf{z}) \\ 0 & \text{else} \end{cases}$$

APP factoring

$$p(\mathcal{H}_m|\mathbf{z}) = \frac{p(\mathbf{z}|\mathcal{H}_m)\pi_m}{p(\mathbf{z})}$$

a posteriori probability (APP)

$$\equiv p(\mathbf{z}|\mathcal{H}_m)\pi_m$$

equivalent for making decisions (  $p(\mathbf{z})$  not dependent on hypothesis)

# Maximum A Posteriori Probability (MAP) Rule

MAP is special case of Bayesian Decision Rule

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 1 & \cdots & 1 \\ 1 & 0 & 1 & \cdots & 1 \\ 1 & 1 & 0 & \cdots & 1 \\ 1 & 1 & 1 & \ddots & 1 \\ 1 & 1 & 1 & \cdots & 0 \end{bmatrix}$$

$$\begin{aligned} C(A_j|\mathbf{z}) &= \sum_i C_{i,j} p(\mathcal{H}_i|\mathbf{z}) \\ &= \sum_i (1 - \delta[i - j]) p(\mathcal{H}_i|\mathbf{z}) \\ &= \sum_{i \neq j} p(\mathcal{H}_i|\mathbf{z}) \\ &= 1 - p(\mathcal{H}_j|\mathbf{z}) \end{aligned}$$

For these costs, the Bayes risk is the probability of decision error

MAP rule


$$\max_m p(\mathcal{H}_m|\mathbf{z}) \iff \max_m p(\mathbf{z}|\mathcal{H}_m)\pi_m$$

MAP rule minimizes probability of decision error over finite number of hypotheses

## Aside: MAP and MMSE Estimation

Consider MMSE Estimation of a digital/discrete random variable

MMSE estimator:

$$\mathbb{E} \{d(u) | \mathbf{z}(u) = \mathbf{z}\} = \sum_m d_m p_{d(u) | \mathbf{z}(u)}(d_m | \mathbf{z})$$


APPs

the optimal MMSE estimator for a digital/discrete desirable requires APPs

## Aside: Hard Decisions and Soft Decisions

Consider MMSE Estimation of a digital/discrete random variable

Hard decision:  $\hat{d} = 3$

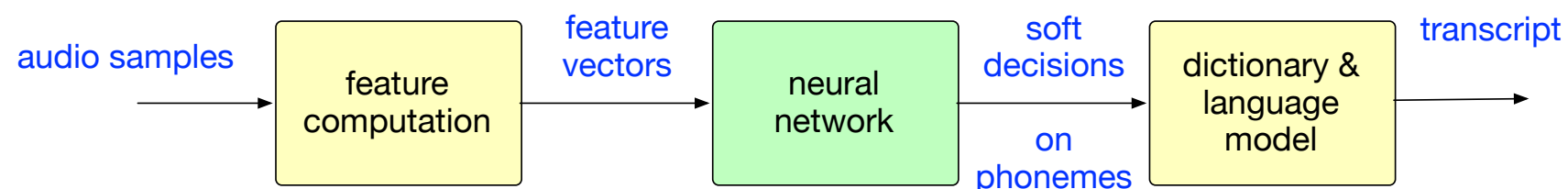
Soft Decision A:  $p(\tilde{d} = 0) = 0.099$     $p(\tilde{d} = 1) = 0.39$     $p(\tilde{d} = 2) = 0.1$     $p(\tilde{d} = 3) = 0.4$

Soft Decision B:  $p(\tilde{d} = 0) = 0.01$     $p(\tilde{d} = 1) = 0.01$     $p(\tilde{d} = 2) = 0.01$     $p(\tilde{d} = 3) = 0.97$

Both soft decisions A and B are consistent with the same hard decision, but B corresponds to much higher confidence

APPs are the ideal soft decisions (typically)

Soft decisions are often desired when the output of the classifier feeds additional processing — eg, Automatic Speech Recognition (ASR):



# Binary MAP Rule

M=2

$$P(\mathcal{E}) = P(\mathcal{E}|\mathcal{H}_0)\pi_0 + P(\mathcal{E}|\mathcal{H}_1)\pi_1$$

$$= \int_{\mathcal{Z}_1} f(\mathbf{z}|\mathcal{H}_0)\pi_0 d\mathbf{z} + \int_{\mathcal{Z}_0} f(\mathbf{z}|\mathcal{H}_1)\pi_1 d\mathbf{z}$$

$$f(\mathbf{z}|\mathcal{H}_1)\pi_1 \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} f(\mathbf{z}|\mathcal{H}_0)\pi_0$$

$$\Lambda(\mathbf{z}) = \frac{f(\mathbf{z}|\mathcal{H}_1)}{f(\mathbf{z}|\mathcal{H}_0)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \frac{\pi_0}{\pi_1} = T$$

Likelihood Ratio Test

likelihood ratio

when  $M > 2$ , still work with  $p(\mathbf{z}|\mathcal{H}_m) / p(\mathbf{z}|\mathcal{H}_0)$



# MAP Rule for the Binary AWGN Channel

M=2

$$\mathcal{H}_m : \quad \mathbf{z}(u) = \mathbf{s}_m + \mathbf{w}(u) \quad (D \times 1)$$

$$P(\mathcal{H}_m|\mathbf{z}) = \frac{f_{\mathbf{z}(u)}(\mathbf{z}|\mathcal{H}_m)\pi_m}{f_{\mathbf{z}(u)}(\mathbf{z})}$$

$$\equiv f_{\mathbf{z}(u)}(\mathbf{z}|\mathcal{H}_m)\pi_m$$

$$= \mathcal{N}_D(\mathbf{z}; \mathbf{s}_m; (N_0/2)\mathbf{I})\pi_m$$

$$= \frac{\pi_m}{(\pi N_0)^{D/2}} \exp \left[ \frac{-1}{N_0} \|\mathbf{z} - \mathbf{s}_m\|^2 \right]$$

$$\equiv \pi_m \exp \left[ \frac{-1}{N_0} \|\mathbf{z} - \mathbf{s}_m\|^2 \right]$$

$$\max_m P(\mathcal{H}_m|\mathbf{z}) \iff \min_m -\ln(P(\mathcal{H}_m|\mathbf{z}))$$

$$\iff \min_m \left[ -\ln(\pi_m) + \frac{1}{N_0} \|\mathbf{z} - \mathbf{s}_m\|^2 \right]$$

$$\iff \min_m \|\mathbf{z} - \mathbf{s}_m\|^2 \quad \left( \text{when } \pi_m = \frac{1}{M} \right)$$

# Other Rules (MAP Special Cases)

Maximum Likelihood (ML):  $\max_m f(\mathbf{z}|\mathcal{H}_m)$

Minimum Distance:  $\min_m d(\mathbf{z}, \mathbf{s}_m)$

Min. Euclidean (squared) distance:  $\min_m \|\mathbf{z} - \mathbf{s}_m\|^2$

**M=2**

MAP reduces to ML when a priori probabilities are uniform

Maximum Likelihood (ML):

$$f(\mathbf{z}|\mathcal{H}_1) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} f(\mathbf{z}|\mathcal{H}_0)$$

Minimum Distance:

$$d(\mathbf{z}, \mathbf{s}_0) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} d(\mathbf{z}, \mathbf{s}_1)$$

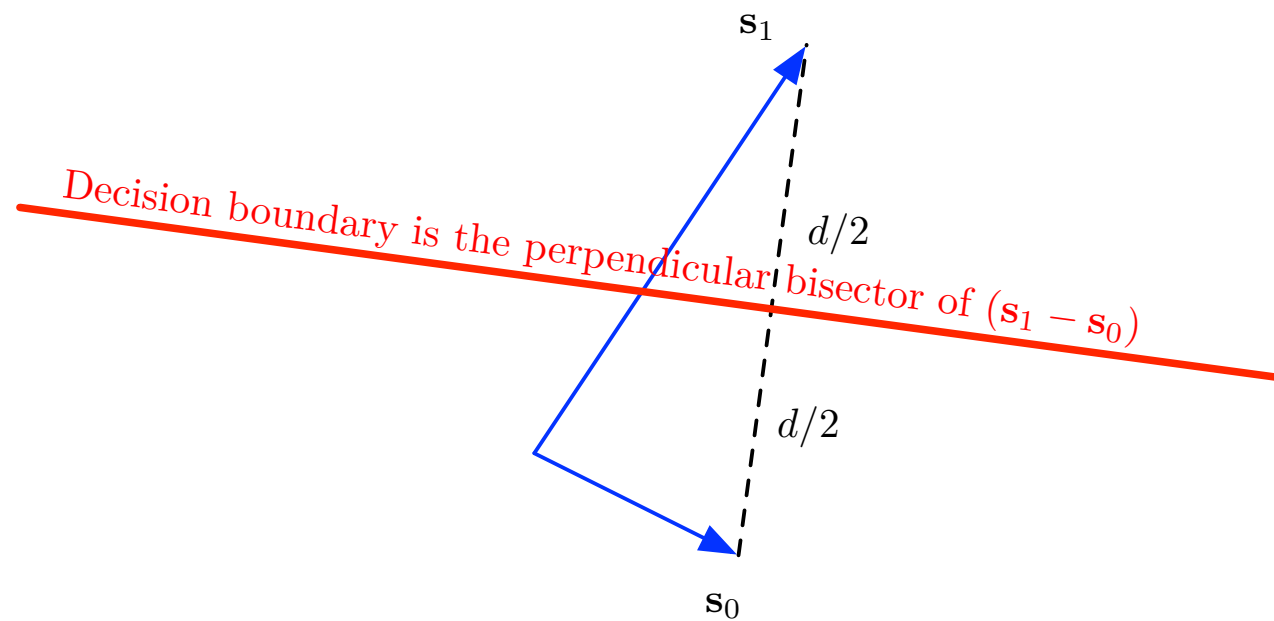
Min. Euclidean (squared) distance:

$$\|\mathbf{z} - \mathbf{s}_0\|^2 \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} \|\mathbf{z} - \mathbf{s}_1\|^2$$

## Other Rules (MAP Special Cases)

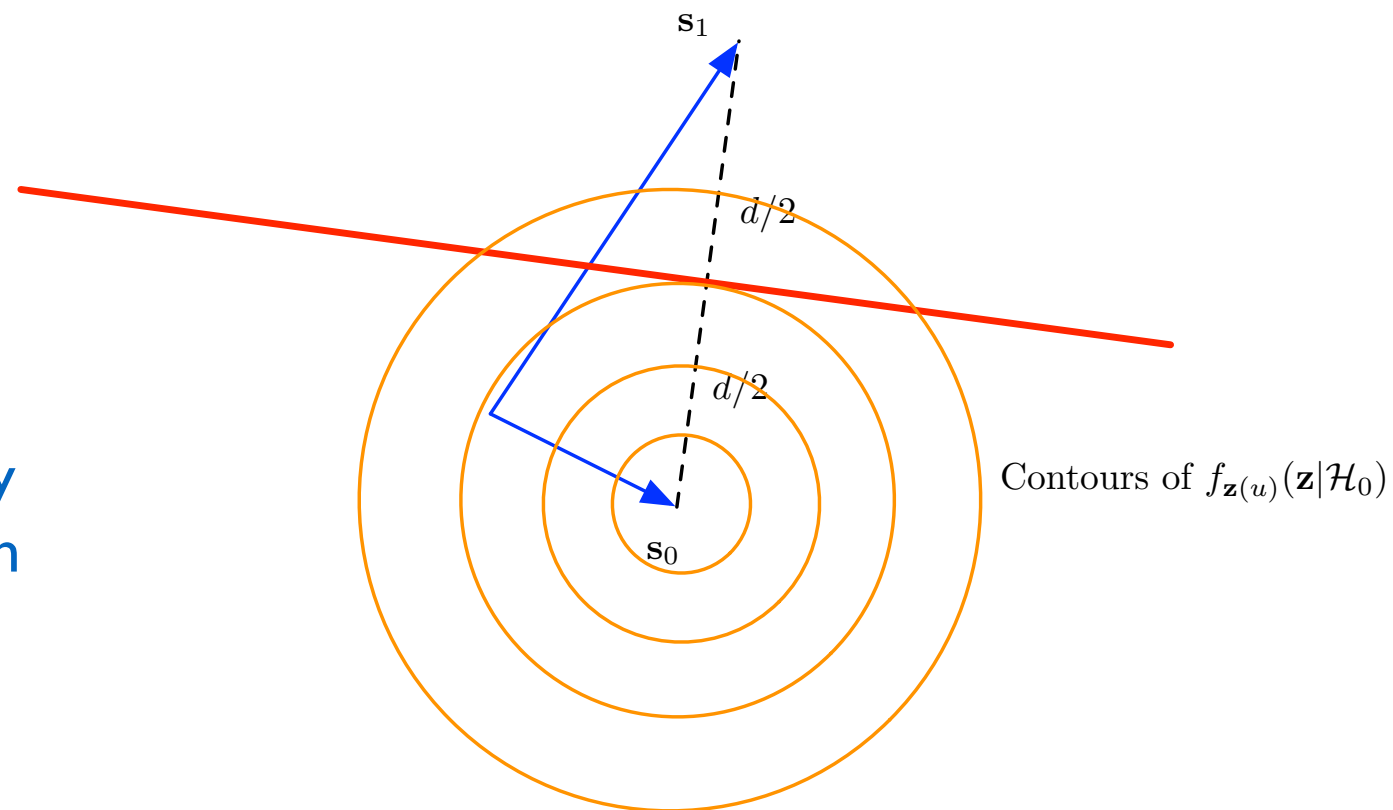


# Binary MAP Decisions (equal priors)



$$(\mathbf{s}_1 - \mathbf{s}_0)^T \mathbf{z} \begin{cases} \geq & \mathcal{H}_1 \\ < & \mathcal{H}_0 \end{cases} \frac{\|\mathbf{s}_1\|^2 - \|\mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)}{2}$$

Error probability given hypothesis 0 is the probability that noise throws observation over decision boundary



# Performance of Binary MAP Decisions (equal priors)

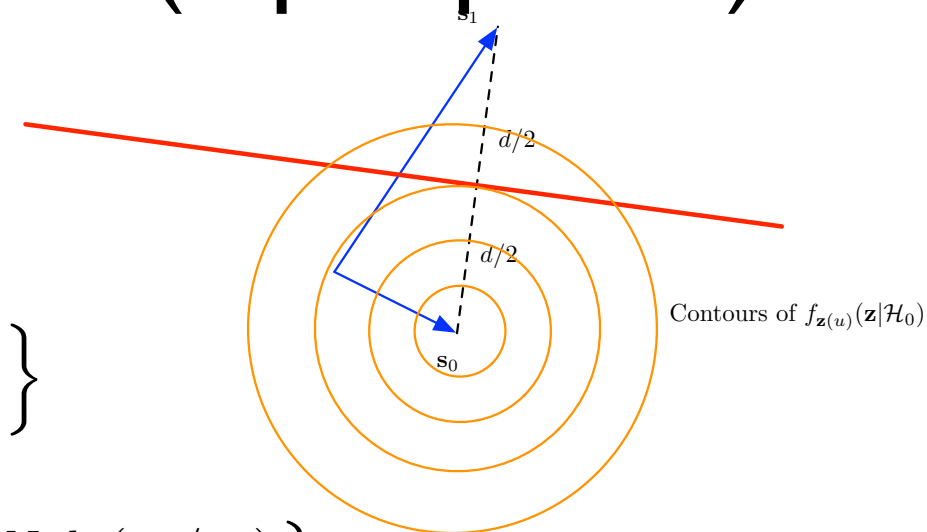
$$P(\mathcal{E}|\mathcal{H}_0) = \text{PR} \left\{ (\mathbf{s}_1 - \mathbf{s}_0)^t \mathbf{z}(u) > \frac{\|\mathbf{s}_1\|^2 - \|\mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)}{2} \mid \mathcal{H}_0 \right\}$$

$$= \text{PR} \left\{ (\mathbf{s}_1 - \mathbf{s}_0)^t (\mathbf{s}_0 + \mathbf{w}(u)) > \frac{\|\mathbf{s}_1\|^2 - \|\mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)}{2} \right\}$$

$$= \text{PR} \left\{ (\mathbf{s}_1^t \mathbf{s}_0 - \|\mathbf{s}_0\|^2) + (\mathbf{s}_1 - \mathbf{s}_0)^t \mathbf{w}(u) > \frac{\|\mathbf{s}_1\|^2 - \|\mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)}{2} \right\}$$

$$= \text{PR} \left\{ (\mathbf{s}_1 - \mathbf{s}_0)^t \mathbf{w}(u) > \frac{\|\mathbf{s}_1\|^2 + \|\mathbf{s}_0\|^2 - 2\mathbf{s}_1^t \mathbf{s}_0 - N_0 \ln(\pi_1/\pi_0)}{2} \right\}$$

$$= \text{PR} \left\{ V(u) > \frac{1}{2} [\|\mathbf{s}_1 - \mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)] \right\}$$



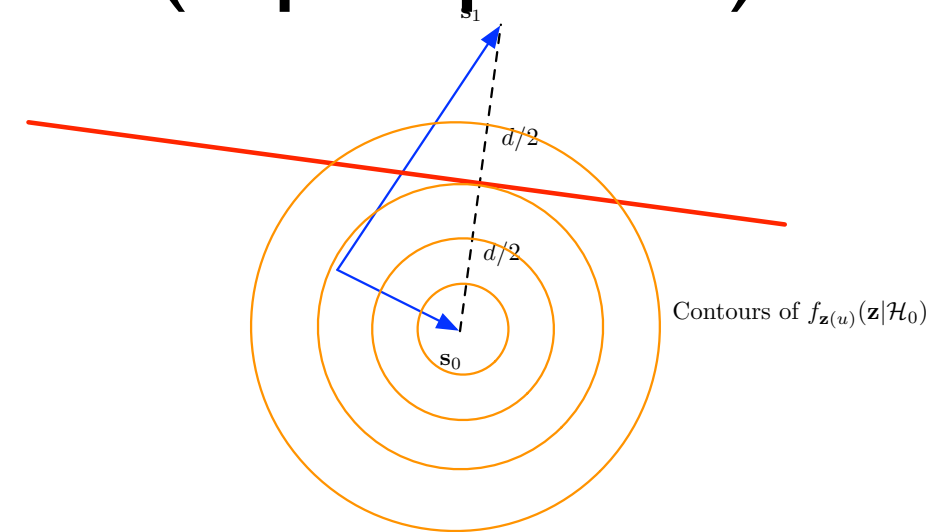
$$\mathbb{E} \{V(u)\} = 0$$

$$\sigma_V^2 = \frac{N_0}{2} \|\mathbf{s}_1 - \mathbf{s}_0\|^2$$

# Performance of Binary MAP Decisions (equal priors)

$$P(\mathcal{E}|\mathcal{H}_0) = Q\left(\sqrt{\frac{d^2}{2N_0}}\right)$$

$$d^2 = \|\mathbf{s}_1 - \mathbf{s}_0\|^2 \quad (\pi_1 = \pi_0)$$



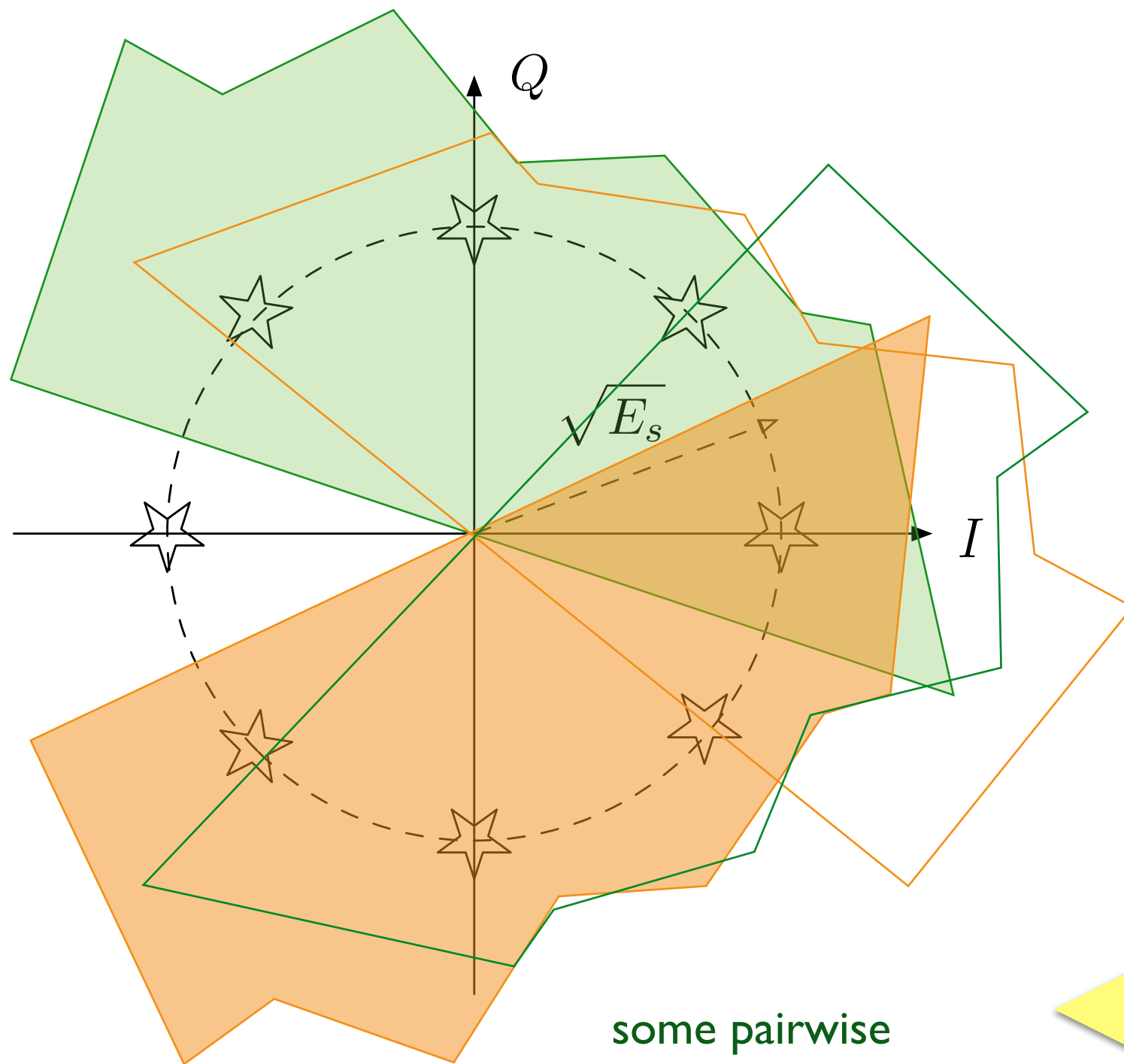
$$P(\mathcal{E}) = P(\mathcal{E}|\mathcal{H}_0)\pi_0 + P(\mathcal{E}|\mathcal{H}_1)\pi_1$$

$$= P(\mathcal{E}|\mathcal{H}_0)(1/2) + P(\mathcal{E}|\mathcal{H}_1)(1/2)$$

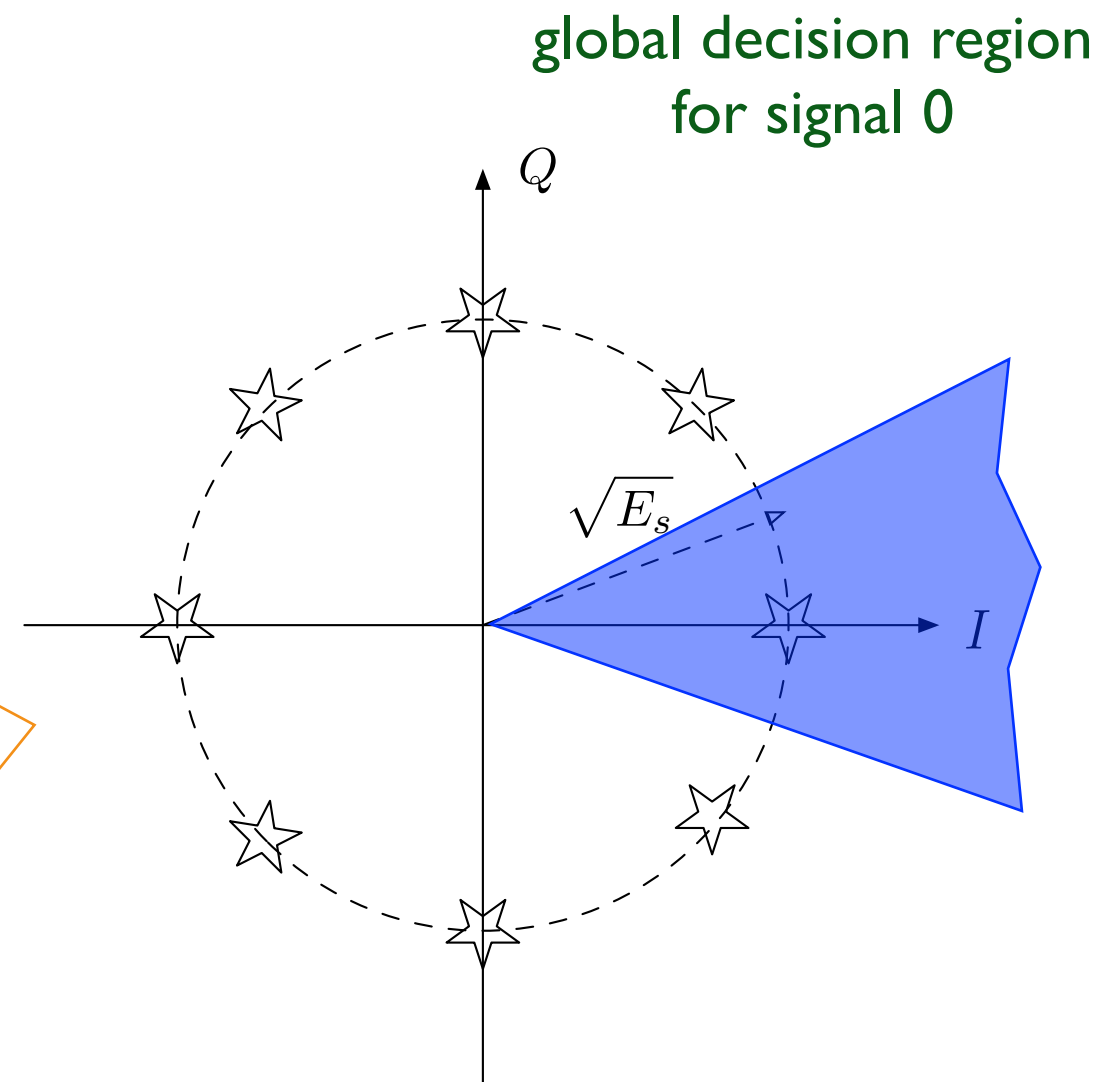
$$P(\mathcal{E}) = Q\left(\sqrt{\frac{d^2}{2N_0}}\right) = Q\left(\sqrt{\frac{\|\mathbf{s}_1 - \mathbf{s}_0\|^2}{2N_0}}\right)$$

Note: not a function of dimension

# M=8 (8-PSK) Example Min. Distance Rule



some pairwise  
decision regions for  
signal 0



global decision region  
for signal 0

the global decision region for  
 $H_m$  is the intersection of all  
pairwise decision regions for  
 $H_m$

# Bounds on Error Probability

Error probability for M-ary decisions can be found using the exact error probability of error in a pairwise test & the union bound

a simple resulting form for M-ary signals in AWGN:

$$\frac{1}{M} Q \left( \sqrt{\frac{d_{\min}^2}{2N_0}} \right) \leq P(\mathcal{E}) \leq (M-1) Q \left( \sqrt{\frac{d_{\min}^2}{2N_0}} \right)$$

dominated by the minimum distance between two signals



## Other Decision Criterion (non-Bayes)

**minimax rule:** the Bayes full for the worst case a priori probabilities

### **Neyman-Pearson rule:**

maximize detection probability for a given false alarm probability

$$P_D = P(\text{decide } \mathcal{H}_1 | \mathcal{H}_1)$$

Detection Probability

$$P_{FA} = P(\text{decide } \mathcal{H}_1 | \mathcal{H}_0)$$

False Alarm Probability

### **NP rule example:**

Given  $P_{fa} < 0.1$ , maximize  $P_d$

Can always maximize detection probability by always deciding  $\mathcal{H}_1$ , but this will have high false alarm probability.

**Note:** N-P and minimax rules do not need knowledge of priors

# Neyman-Pearson Theorem

## Neyman-Pearson Theorem:

The rule that maximizes detection probability under a limit on the false alarm probability is a likelihood ratio test

example: detect a signal in AWGN

$$\mathcal{H}_0 : z(u) = n(u)$$

$$\mathcal{H}_1 : z(u) = S + n(u) \quad S > 0$$

$$\frac{p(z|\mathcal{H}_1)}{p(z|\mathcal{H}_0)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} T$$

$$\iff \exp \left[ \frac{1}{2\sigma_n^2} (z^2 - [z - S]^2) \right] \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} T$$

$$\iff \exp \left[ \frac{1}{2\sigma_n^2} (2Sz - S^2) \right] \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} T$$

$$\iff z \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{>}} T'$$

$$P_{FA} = Q \left( \frac{T'}{\sigma_n} \right)$$

$$T' = \sigma_n Q^{-1}(P_{FA})$$

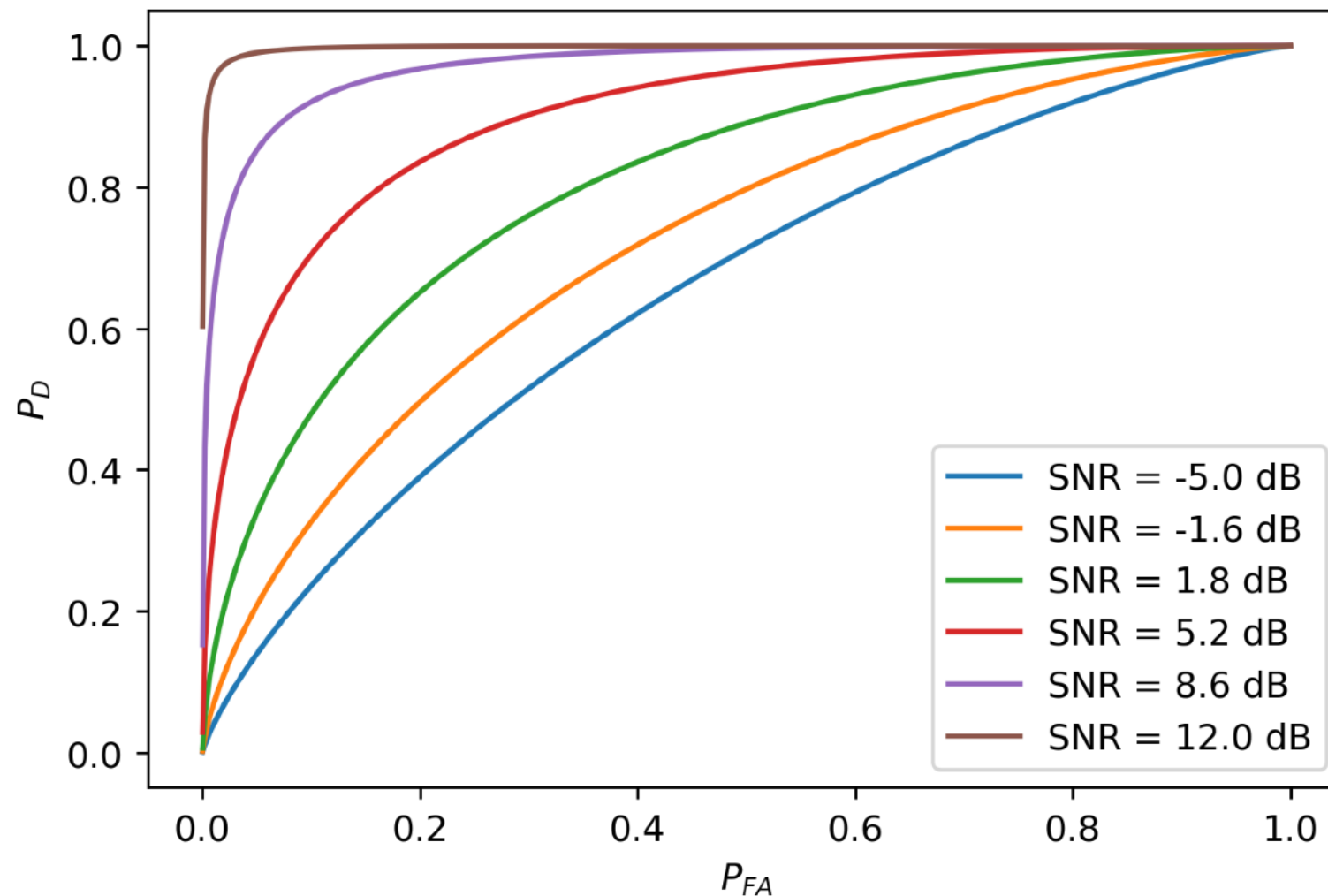
$$P_D = Q \left( \frac{T' - S}{\sigma_n} \right)$$

$$= Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{S^2}{\sigma_n^2}} \right)$$

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{S^2}{\sigma_n^2}} \right)$$

# Neyman-Pearson Example: ROC

$$P_D = Q \left( Q^{-1}(P_{FA}) - \sqrt{\frac{S^2}{\sigma_n^2}} \right)$$



## Receiver Operating Characteristic (ROC):

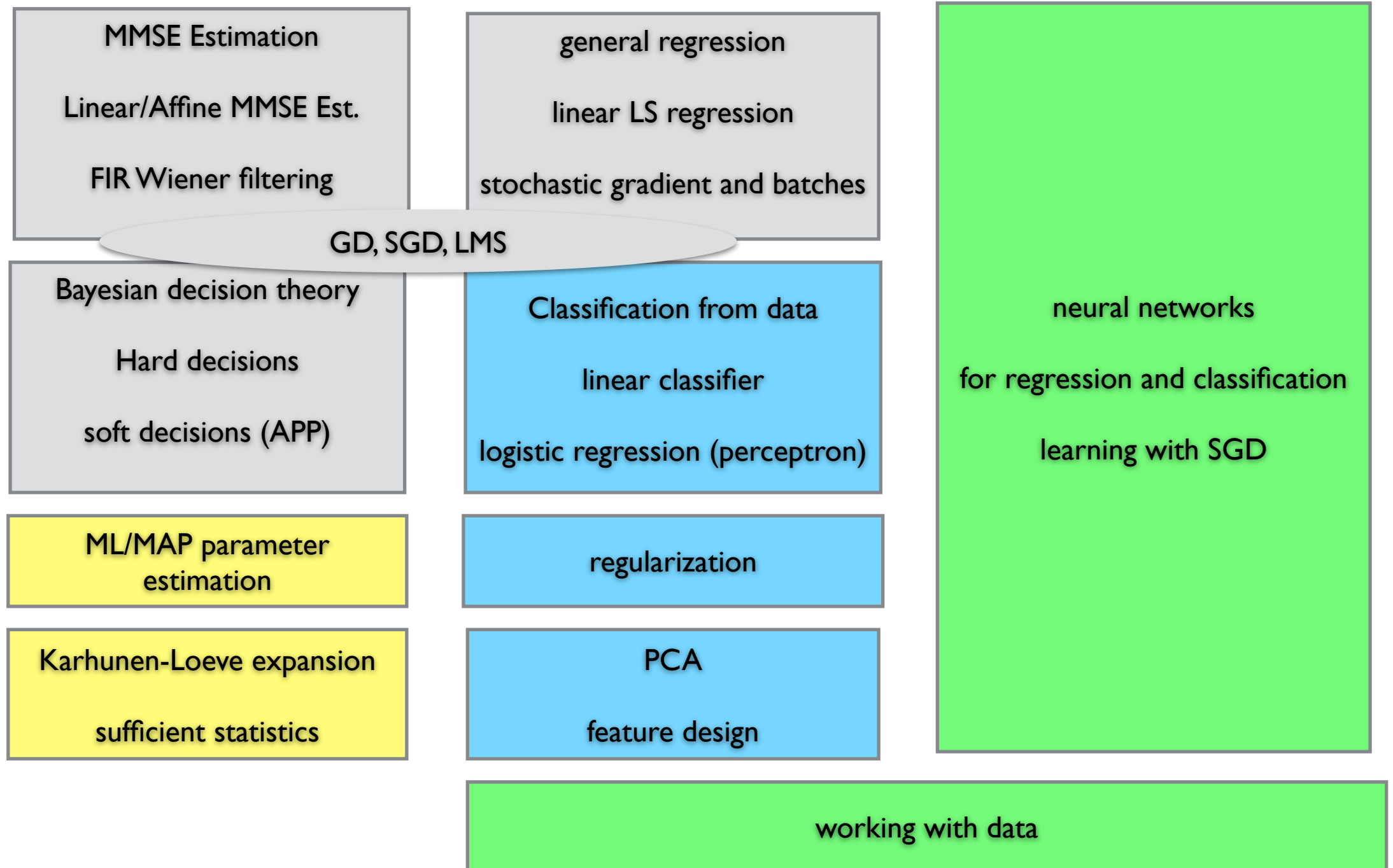
characterizes the trade-off between detection and false alarm probabilities

changes with the problem, but is another way to characterize classification performance

# Detection, Estimation, Regression

statistical models

data driven



# Linear Classifier

perform linear regression and then threshold to get a hard decision

## Example:

$$y \in \{-1, +1\}$$

$$\hat{y} = \text{sign}(\mathbf{w}^t \mathbf{x})$$

$$\text{sign}(v) = \begin{cases} +1 & v \geq 0 \\ -1 & v < 0 \end{cases}$$

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=0}^{N-1} (y_n - \mathbf{w}^t \mathbf{x}_n)^2$$

standard LLSE regression with thresholding of the prediction

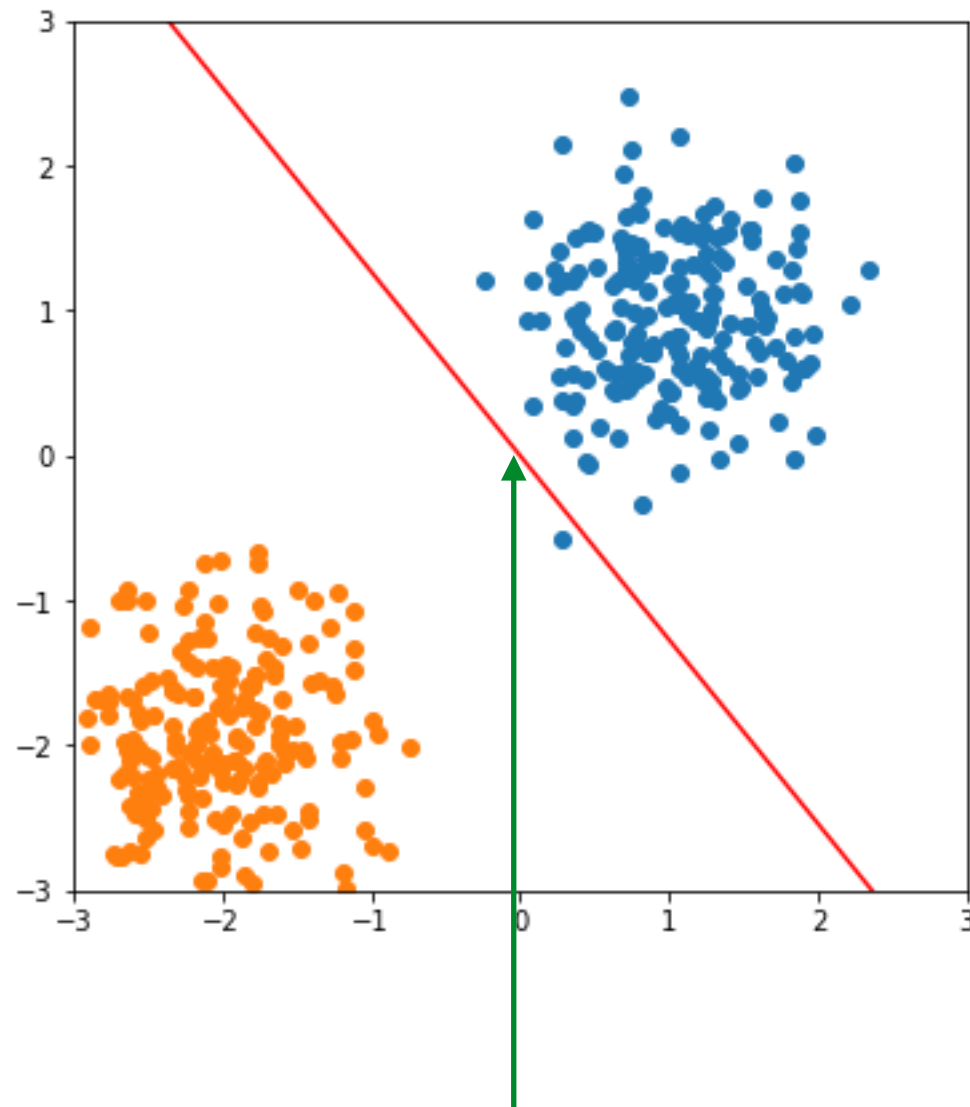
## Review:

[linear\\_classifier\\_examples.ipynb](#)

# Example: Linear and Affine Regression

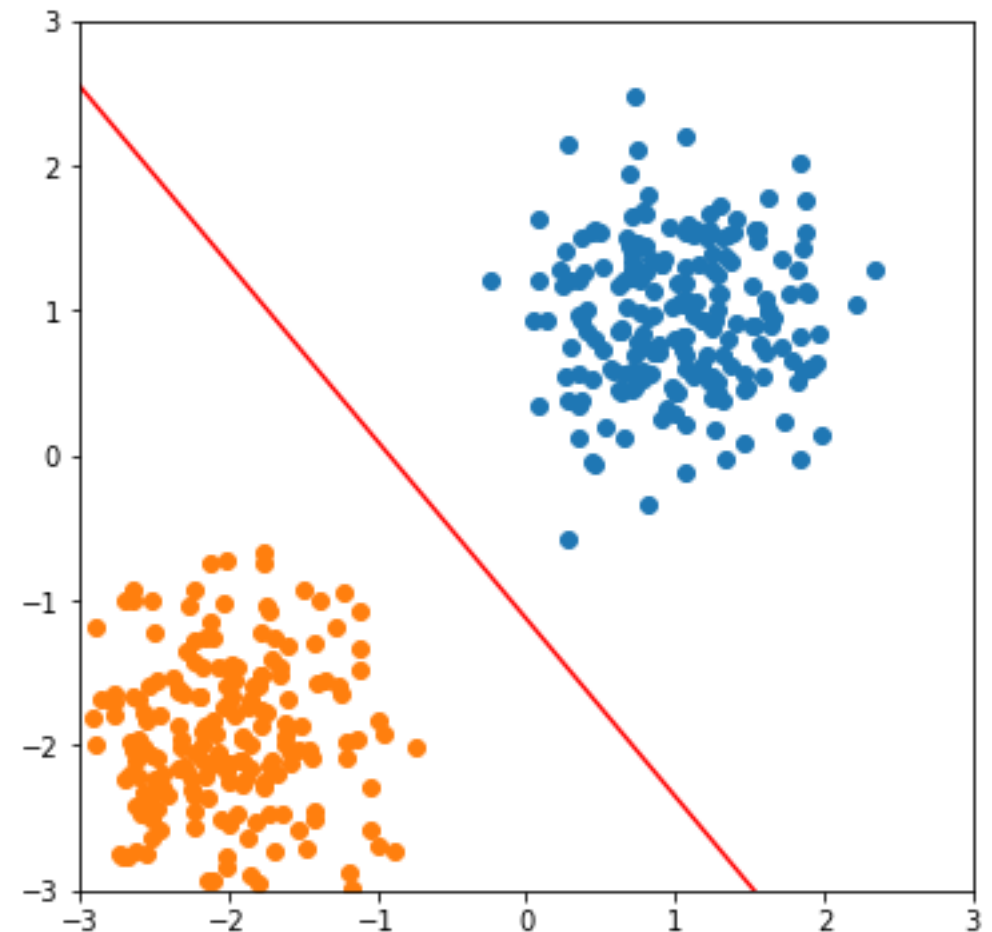
$$\hat{y} = \mathbf{x}^t \mathbf{w}$$

$$\hat{y} = \begin{bmatrix} \mathbf{x}^t & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}$$



for the case with no bias term, the decision threshold has to pass through the origin

$$\mathbf{s}_0 = \begin{bmatrix} +1 \\ +1 \end{bmatrix}$$
$$\mathbf{s}_1 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$



adding the bias term allows for offset from the origin

# Maximum Likelihood Parameter Estimation

Theta is a (non-random) parameter set

$$\hat{\Theta}_{ML} = \hat{\Theta}_{ML}(y) = \arg \max_{\Theta} p_{y(u)}(y; \Theta)$$

Alternate notation:  $p_{y(u)}(y|\Theta)$  **recall:** likelihood of Theta (given y)

The ML estimate is the parameters values that best explain the observation y under the statistical model for y(u)

May be based on a conditional or joint distribution:

$$\hat{\Theta}_{ML} = \hat{\Theta}_{ML}(y, \mathbf{x}) = \arg \max_{\Theta} p_{y(u)|\mathbf{x}(u)}(y|\mathbf{x}; \Theta)$$

$$\hat{\Theta}_{ML} = \hat{\Theta}_{ML}(y, \mathbf{x}) = \arg \max_{\Theta} p_{\mathbf{x}(u), y(u)}(\mathbf{x}, y; \Theta)$$

# ML Estimation Example

this is a model for the data  $\{ (\mathbf{x}_n, y_n) \}$ :

$$y_n = \mathbf{w}^t \mathbf{x}_n + v_n, \quad n = 0, 1, \dots, N-1$$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{v}(u)$$

$$p_{\mathbf{v}(u)}(\mathbf{v}) = \mathcal{N}_N(\mathbf{v}; \mathbf{0}; \sigma_v^2 \mathbf{I})$$

$$p_{\mathbf{y}(u)|\mathbf{X}(u)}(\mathbf{y}|\mathbf{X}; \mathbf{w}) = p_{\mathbf{v}(u)}(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathcal{N}_N(\mathbf{y}; \mathbf{X}\mathbf{w}; \sigma_v^2 \mathbf{I})$$

$$\begin{aligned} \text{NLL}(\mathbf{w}) &= -\ln(p_{\mathbf{y}(u)}(\mathbf{y}|\mathbf{X}; \mathbf{w})) \\ &= -\ln\left(\frac{1}{(2\pi\sigma_v^2)^{N/2}} \exp\left[\frac{-1}{2\sigma_v^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2\right]\right) \\ &= \frac{1}{2\sigma_v^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{N}{2} \ln(2\pi\sigma_v^2) \end{aligned}$$

$$\max_{\mathbf{w}} p_{\mathbf{y}(u)|\mathbf{X}(u)}(\mathbf{y}|\mathbf{X}; \mathbf{w}) \iff \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

Maximum Likelihood  $\iff$  min Neg-Log-Likelihood  $\iff$  LLSE regression  
(under this model for the data)



# Properties of ML Estimators

- Asymptotically Gaussian:
  - For large amounts of data, the ML estimate is Gaussian with mean equal to the true parameter (models matched)
- Consistent:
  - The limit in probability of the ML estimate is the true parameter (model matched)
- The ML estimate minimizes the KL Divergence between the model distribution and the empirical data distribution. KL divergence measures the difference between two distribution (Info.Theory).
- Minimizing KL divergence in this case also corresponds to minimizing the cross entropy

# Logistic Regression Motivation

Note that a linear classifier has hard decision:

$$\hat{y} = \text{sign}(\mathbf{w}^t \mathbf{x}) \quad y \in \{-1, +1\}$$

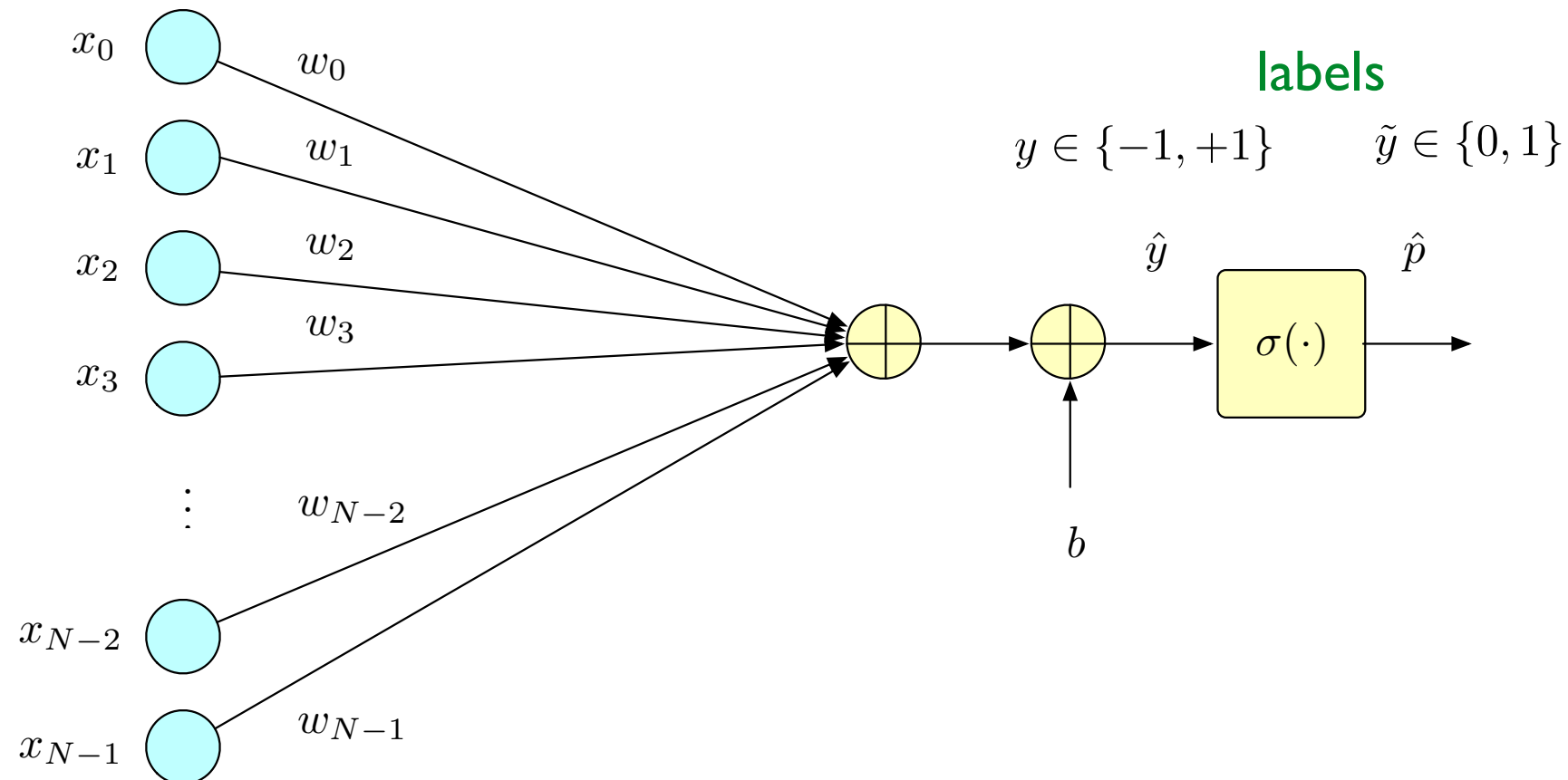
with corresponding soft decision:

$$\hat{y} = \mathbf{x}^t \mathbf{w}$$

Note that this is a real number — the magnitude is the “confidence” and the hard decision is the sign

How can we convert this to a soft decision that is a probability?

# Logistic Regression



Two problems to address:

1. What is a good “sigma” function to map from reals targeting +/- 1 to a probability of a 1?
2. What is a good loss function between the binary labels  $\{0, 1\}$  and the regressor output  $\hat{p} \sim P(1)$ ?

# Recall from the ML interpretation of LLSE Regression

model for ML estimation of  $\mathbf{w}$ :

$$y = \mathbf{w}^t \mathbf{x} + v(u)$$

$$p(v) = \mathcal{N}(v; 0; \sigma_v^2)$$

if we adopt the convention that:

$$y = +1 \quad \Longleftrightarrow \quad \tilde{y} = 1$$

$$y = -1 \quad \Longleftrightarrow \quad \tilde{y} = 0$$

Likelihood ratio for  $y$  (binary classification):

$$\begin{aligned} \frac{p(y = +1 | \mathbf{x}; \mathbf{w})}{p(y = -1 | \mathbf{x}; \mathbf{w})} &= \frac{\mathcal{N}(+1; \mathbf{w}^t \mathbf{x}; \sigma_v^2)}{\mathcal{N}(-1; \mathbf{w}^t \mathbf{x}; \sigma_v^2)} \\ &= \exp \left[ \frac{2}{\sigma_v^2} \mathbf{w}^t \mathbf{x} \right] \end{aligned}$$

Log-likelihood ratio:

$$L = \frac{2}{\sigma_v^2} \mathbf{w}^t \mathbf{x}$$

**Takeaway:  $\mathbf{w}$**   
dotted with  $\mathbf{x}$  can be viewed  
as a log-likelihood ratio or log  
ration of (scaled)  
probabilities

# Logistic Regression Motivation

First, suppose we have the log ratio of two probability-like measure (maybe not normalized)

$$L = \ln \left( \frac{p_1}{p_0} \right)$$

$$= \ln p_1 - \ln p_0$$

$$= \ell_1 - \ell_0$$

$$p_0 = \frac{e^{\ell_0}}{e^{\ell_0} + e^{\ell_1}} \quad p_1 = \frac{e^{\ell_1}}{e^{\ell_0} + e^{\ell_1}}$$

**FIX THESE TYPOS**

$$= \frac{1}{1 + e^L} \quad = \frac{1}{1 + e^{-L}}$$

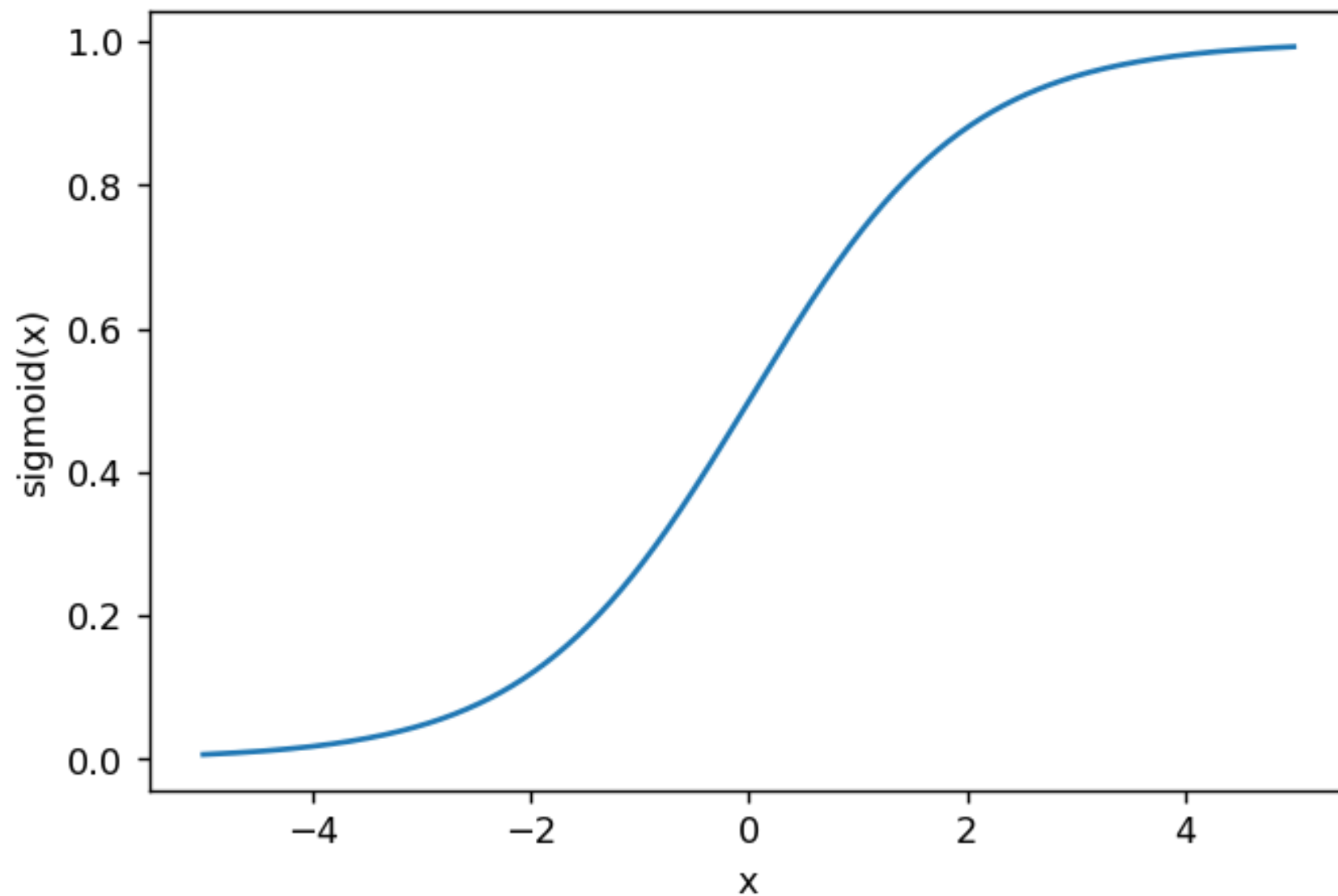
logistic function maps a log-likelihood ratio to the “probability of a 1”

$$p_1 = 1 - p_0 = \sigma(L) \triangleq \frac{e^L}{1 + e^L} = \frac{1}{1 + e^{-L}}$$

sigma is called the Logistic function or sigmoid function (sigmoid is overloaded term)

# Logistic Regression Motivation

maps a log-likelihood ratio to a probability ( $p$  or  $p_{\text{numerator}}$ )



**Note:**  $\text{sigmoid}(0) = 0.5$

# Logistic Regression Motivation

## Other useful properties

$$\begin{aligned}\dot{\sigma}(s) &= \frac{d}{ds} \left[ \frac{e^s}{1 + e^s} \right] \\ &= \frac{e^s}{1 + e^s} - \frac{(e^s)^2}{(1 + e^s)^2} \\ &= \left[ \frac{e^s}{1 + e^s} \right] \left[ \frac{1}{1 + e^s} \right] \\ &= \sigma(s)(1 - \sigma(s))\end{aligned}$$

$$\sigma(s) = \frac{1}{2} \left[ 1 + \tanh \left( \frac{s}{2} \right) \right]$$

# Logistic Regression Motivation

This motivates a model for a binary variable  $y$ :

$$\tilde{y}(u) \sim \text{Bernoulli}(p)$$

$$p = p_1 = \sigma(\mathbf{w}^t \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^t \mathbf{x})}$$

$$(1 - p) = p_0 = \frac{1}{1 + \exp(+\mathbf{w}^t \mathbf{x})}$$

Model the values of the binary targets

$$\tilde{y}_n \sim \text{Bernoulli}(\sigma(\mathbf{w}^t \mathbf{x}_n)) \quad \text{iid}$$



# Logistic Regression

Take the ML approach to determining  $\mathbf{w}$  for this model:

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}; \mathbf{w}) &= \prod_{n=0}^{N-1} p_n^{\tilde{y}_n} [1 - p_n]^{(1-\tilde{y}_n)} \\ &= \prod_{n=0}^{N-1} p_n^{\mathbb{I}[y_n=+1]} [1 - p_n]^{\mathbb{I}[y_n=-1]} \end{aligned}$$

$$p_n = \sigma(\mathbf{w}^t \mathbf{x}_n)$$

$$p_n^{\tilde{y}_n} [1 - p_n]^{(1-\tilde{y}_n)} = \begin{cases} p_n & \tilde{y}_n = 1 \ (y_n = +1) \\ 1 - p_n & \tilde{y}_n = 0 \ (y_n = -1) \end{cases}$$

The negative log-likelihood is....

**Example:**

**Labels ( $\mathbf{y}$ -tilde):** 1, 0, 1

**Output ( $\sigma(\mathbf{w} \cdot \mathbf{x})$ ):** 0.9, 0.1, 0.2

**$p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ :** 0.9 \* 0.9 \* 0.2

# Logistic Regression

The negative log-likelihood is:

$$\begin{aligned}\text{NLL}(\mathbf{w}) &= - \left( \sum_{n=0}^{N-1} \tilde{y}_n \log(p_n) + (1 - \tilde{y}_n) \log(1 - p_n) \right) \\ &= - \left( \sum_{n=0}^{N-1} \tilde{y}_n \log(\sigma(\mathbf{w}^t \mathbf{x}_n)) + (1 - \tilde{y}_n) \log((1 - \sigma(\mathbf{w}^t \mathbf{x}_n))) \right) \\ &= \sum_{n=0}^{N-1} \log(1 + \exp[-y_n \mathbf{w}^t \mathbf{x}_n])\end{aligned}$$

$p_n = \sigma(\mathbf{w}^t \mathbf{x}_n)$

This is called the binary cross-entropy loss. We will see that this can be viewed as a distance between two distributions

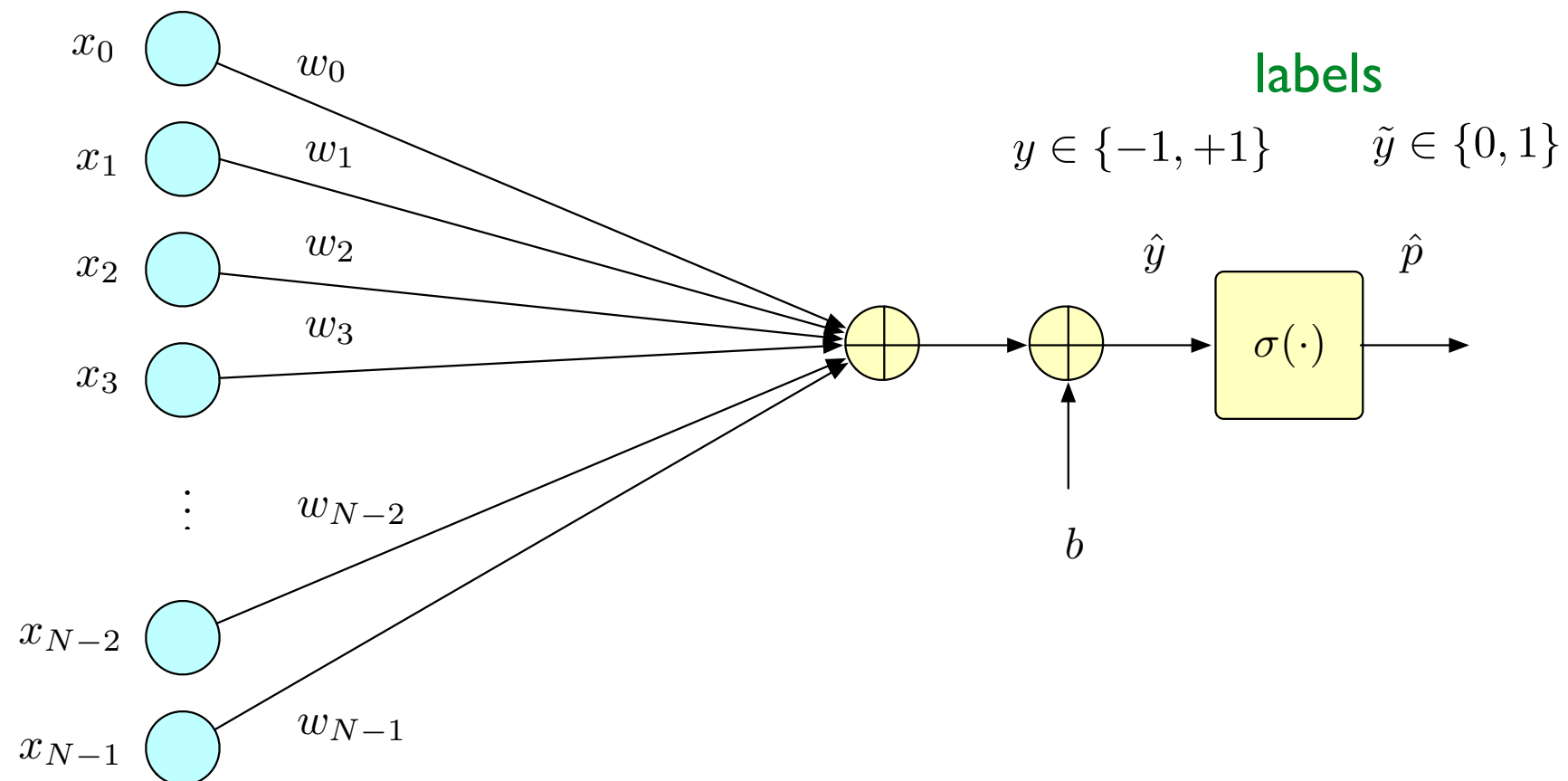
**Example:**

**Labels ( $\tilde{y}$ ):** 1, 0, 1

**Output ( $\sigma(\mathbf{w} \cdot \mathbf{x})$ ):** 0.9, 0.1, 0.2

**NLL( $\mathbf{w}$ ):** 0.11 + 0.11 + 1.6

# Logistic Regression



Two problems addressed:

1. Logistic function maps a LLR to  $P(I)$
2. Binary cross-entropy is the loss that arises from a Bernoulli model and maximum likelihood parameter estimation.

# Logistic Regression

## Summary:

Logistic regression is ML estimation of  $\mathbf{w}$  for an iid Bernoulli model with

$$p_n = \sigma(\mathbf{w}^t \mathbf{x}_n)$$

which can be viewed as regression with the (empirical) binary cross-entropy cost function

no closed form, usually use SGD to perform the regression

We will see that this is a special case of two concepts:

1. It is a single-perceptron and MLP (neural networks) are many of these combined (with slight modification)
2. The loss function derived is the binary cross-entropy between the output probability mass function  $(p, 1-p)$  and the “one-hot” encoded label pmf  $y$ -tilde.

# Single Perceptron History

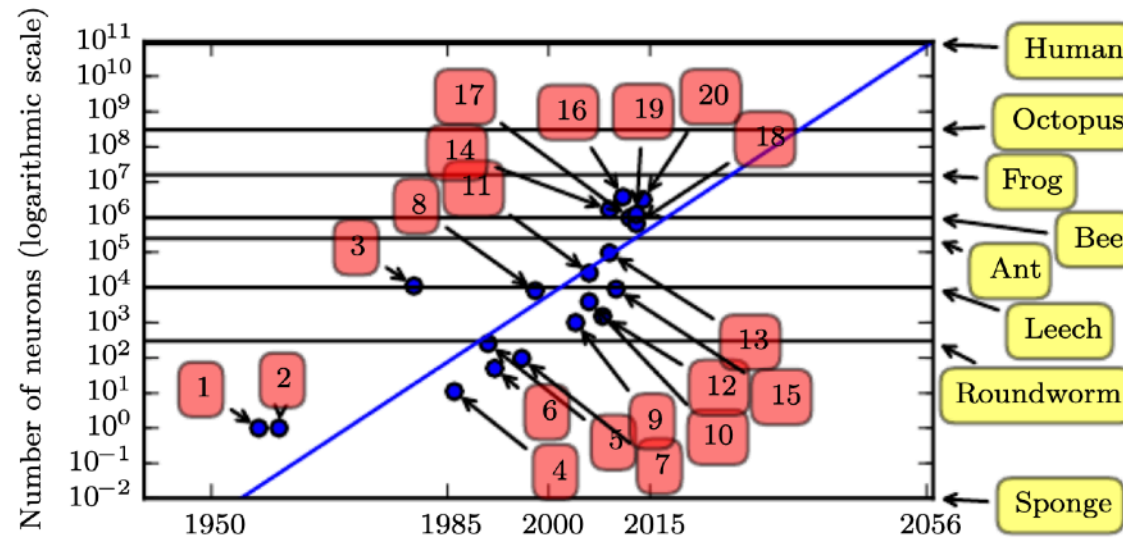


Figure 1.11: Increasing neural network size over time. Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from [Wikipedia \(2015\)](#).

1. Perceptron ([Rosenblatt, 1958, 1962](#))
2. Adaptive linear element ([Widrow and Hoff, 1960](#))
3. Neocognitron ([Fukushima, 1980](#))
4. Early back-propagation network ([Rumelhart et al., 1986b](#))
5. Recurrent neural network for speech recognition ([Robinson and Fallside, 1991](#))
6. Multilayer perceptron for speech recognition ([Bengio et al., 1991](#))
7. Mean field sigmoid belief network ([Saul et al., 1996](#))
8. LeNet-5 ([LeCun et al., 1998b](#))
9. Echo state network ([Jaeger and Haas, 2004](#))
10. Deep belief network ([Hinton et al., 2006](#))
11. GPU-accelerated convolutional network ([Chellapilla et al., 2006](#))
12. Deep Boltzmann machine ([Salakhutdinov and Hinton, 2009a](#))
13. GPU-accelerated deep belief network ([Raina et al., 2009](#))
14. Unsupervised convolutional network ([Jarrett et al., 2009](#))
15. GPU-accelerated multilayer perceptron ([Ciresan et al., 2010](#))
16. OMP-1 network ([Coates and Ng, 2011](#))
17. Distributed autoencoder ([Le et al., 2012](#))
18. Multi-GPU convolutional network ([Krizhevsky et al., 2012](#))
19. COTS HPC unsupervised convolutional network ([Coates et al., 2013](#))
20. GoogLeNet ([Szegedy et al., 2014a](#))

this model was proposed with a simple learning algorithm (special case of SGD)

# Notes on Logistic Regression / Single Perceptron

- In the view of  $\mathbf{w} \cdot \mathbf{x}$  as a log-likelihood, the value of  $\sigma_v$  is absorbed into learning the coefficients  $\mathbf{w}$
- Labeling convention  $0, 1 \longleftrightarrow -1, +1$  can be altered as long as you keep the training labels consistent. The convention I followed is the most common
- BCE minimization is a special case for Maximum Likelihood Estimation — i.e., it always minimizes a Cross-Entropy measure
- Finds the parameter that minimizes the Kullback–Leibler divergence (KL divergence) between the model distribution and the empirical data distribution
- Delay the details of SGD for logistic regression since it is a special case of back-propagation for a Multi-layer Perceptron (MLP)

# ML Estimation and Information Theory

Entropy:

$$\begin{aligned} H(X(u)) &= \mathbb{E} \left\{ \log_2 \left( \frac{1}{p_{X(u)}(X(u))} \right) \right\} \\ &= \sum_k p_{X(u)}(k) \log_2 \left( \frac{1}{p_{X(u)}(k)} \right) \\ &= \sum_k p_k \log_2 \left( \frac{1}{p_k} \right) \end{aligned}$$

Intuition:

events with low probability have large information  
— e.g., “it will snow in Phoenix tomorrow”

the entropy is the average information learned when  
the value of  $X(u)$  is revealed.

weather report in Phoenix has low entropy (almost  
always the same), whereas in Sioux City, SD it has  
high entropy (highly variant weather)

Examples:

fair die:

$$H(X(u)) = \log_2(1/6) = 2.58 \text{ bits/roll}$$

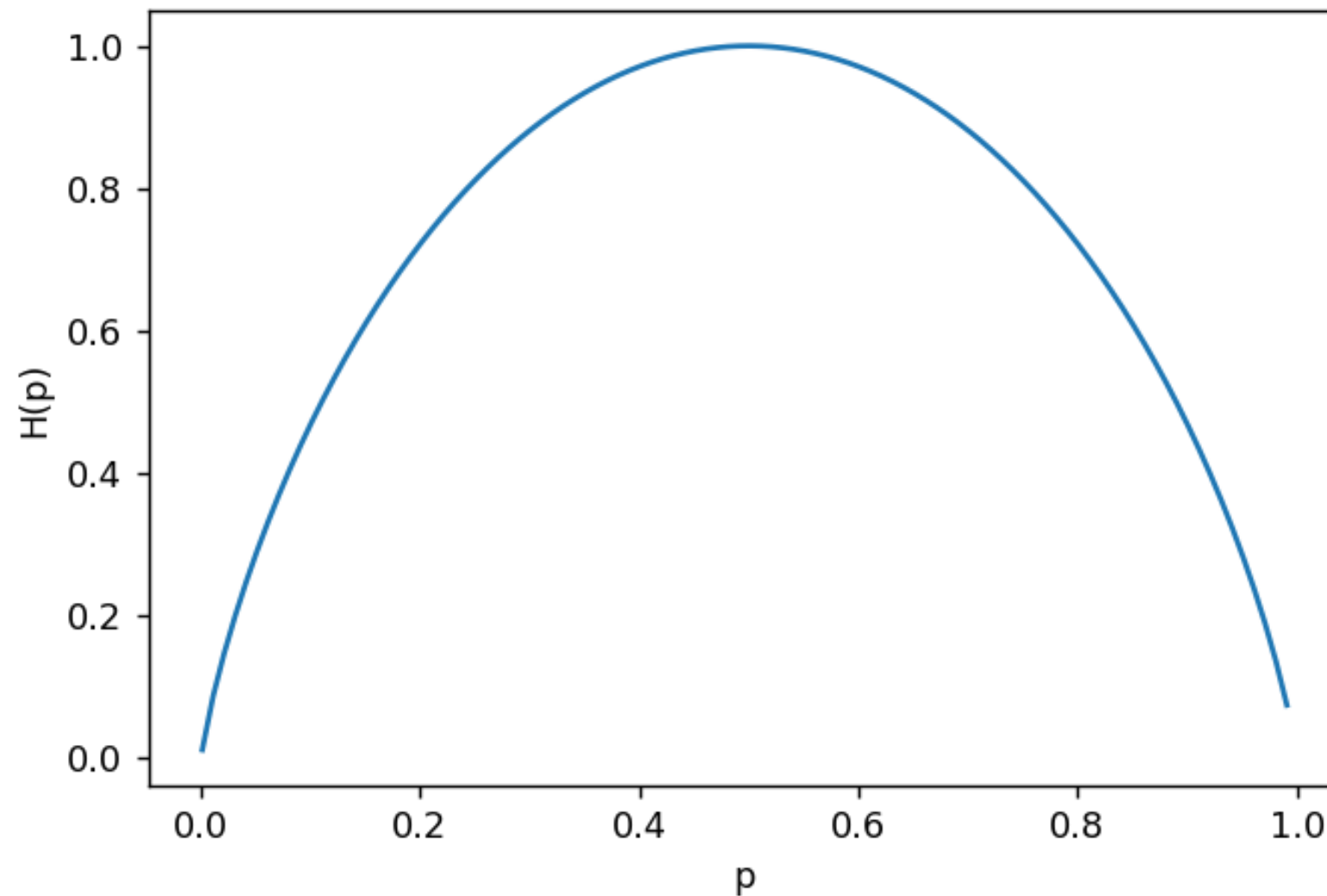
loaded die:

$$\begin{aligned} H(X(u)) &= -0.4 \log_2(0.4) - 0.1 \log_2(0.1) - 0.01 \log_2(0.01) \\ &\quad - 0.09 \log_2(0.09) - 0.25 \log_2(0.25) - 0.15 \log_2(0.15) \\ &= 2.15 \text{ bits/roll} \end{aligned}$$

# ML Estimation and Information Theory

Entropy of iid Bernoulli Source (with success probability  $p$ )

$$H(p) = -p \log_2(p) - (1 - p) \log_2(1 - p)$$





# ML Estimation and Information Theory

## KL-Divergence

$$\begin{aligned} D(p\|\tilde{p}) &= \mathbb{E}_p \left\{ \log \left( \frac{p_x(X(u))}{\tilde{p}_x(X(u))} \right) \right\} \\ &= \sum_k p_k \log \left( \frac{p_k}{\tilde{p}_k} \right) \\ &= \sum_k p_k \log(p_k) - \sum_k p_k \log(\tilde{p}_k) \\ &= \text{CE}(p, \tilde{p}) - H(p) \\ \text{CE}(p, \tilde{p}) &= \mathbb{E}_p \left\{ \log \left( \frac{1}{\tilde{p}_x(X(u))} \right) \right\} \end{aligned}$$

## Cross-Entropy

$$\text{CE}(p, \tilde{p}) = \mathbb{E}_p \left\{ \log \left( \frac{1}{\tilde{p}_x(X(u))} \right) \right\}$$

# ML Estimation and Information Theory

ML parameter estimation minimizes empirical CE (and KL divergence)

$p_{\text{data}}(y|\mathbf{x})$  = data distribution of the data (typically unknown)

$p_{\text{model}}(y|\mathbf{x}; \Theta)$  = modeled distribution of the data (function of parameters)

$$\begin{aligned}\text{CE}(p_{\text{data}}, p_{\text{model}}(\Theta)) &= \mathbb{E}_{p_{\text{data}}(y|x)} \left\{ \log \left( \frac{1}{p_{\text{model}}(y(u)|\mathbf{x}(u); \Theta)} \right) \right\} \\ &\approx \langle -\log(p_{\text{model}}(y|\mathbf{x}; \Theta)) \rangle_{\mathcal{D}} \\ &= \frac{-1}{N} \sum_{n=0}^{N-1} \log(p_{\text{model}}(y_n|\mathbf{x}_n; \Theta))\end{aligned}$$

**Max-Likelihood Estimation of neural network weights is always minimizing the empirical cross entropy between data distribution and the modeled distribution**

$$\begin{aligned}\max_{\Theta} p_{\text{model}}(\mathbf{y}|\mathbf{X}; \Theta) &\iff \min_{\theta} (-\log(p_{\text{model}}(\mathbf{y}|\mathbf{X}; \Theta))) \\ &\iff \min_{\theta} \left( -\sum_{n=0}^{N-1} \log(p_{\text{model}}(y_n|\mathbf{x}_n; \Theta)) \right) \quad (\text{iid } y_n \text{ assumed}) \\ &\iff \min_{\theta} \left( \frac{-1}{N} \sum_{n=0}^{N-1} \log(p_{\text{model}}(y_n|\mathbf{x}_n; \Theta)) \right) \quad (\text{empirical Cross-Entropy})\end{aligned}$$

# Logistic Regression — We found BCE

The negative log-likelihood is:

$$\begin{aligned}\text{NLL}(\mathbf{w}) &= - \left( \sum_{n=0}^{N-1} \tilde{y}_n \log(p_n) + (1 - \tilde{y}_n) \log(1 - p_n) \right) \\ &= - \left( \sum_{n=0}^{N-1} \tilde{y}_n \log(\sigma(\mathbf{w}^t \mathbf{x}_n)) + (1 - \tilde{y}_n) \log((1 - \sigma(\mathbf{w}^t \mathbf{x}_n))) \right) \\ &= \sum_{n=0}^{N-1} \log(1 + \exp[-y_n \mathbf{w}^t \mathbf{x}_n])\end{aligned}$$
$$p_n = \sigma(\mathbf{w}^t \mathbf{x}_n)$$

Define the binary cross entropy function:

$$\text{BCE}(p, \tilde{p}) \triangleq \mathbb{E}_p \left\{ \log \left( \frac{1}{\tilde{p}(y)} \right) \right\}$$

special case of the general  
derivation

$$= p \log \frac{1}{\tilde{p}} + (1 - p) \log \frac{1}{(1 - \tilde{p})}$$

$$= - [p \log(\tilde{p}) + (1 - p) \log(1 - \tilde{p})]$$

# ML Estimation and Information Theory

Only this binary cross-entropy and the multi-class cross-entropy (next slide) are called “cross entropy” typically, the first example is called MSE loss in the deep learning frameworks/literature

Example:

$$y_n = \mathbf{w}^t \mathbf{x}_n + v_n, \quad n = 0, 1, \dots, N-1$$

$$p_{\mathbf{v}(u)}(\mathbf{v}) = \mathcal{N}_N(\mathbf{v}; \mathbf{0}; \sigma_v^2 \mathbf{I})$$

Gaussian noise  
model, CE is LSE

$$\max_{\mathbf{w}} p_{\mathbf{y}(u)|\mathbf{X}(u)}(\mathbf{y}|\mathbf{X}; \mathbf{w}) \iff \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$

Example:

$$\tilde{y}_n \sim \text{Bernoulli}(\sigma(\mathbf{w}^t \mathbf{x}_n)) \quad \text{iid}$$

here, our p-data was a  
point-mass pmf

Bernoulli model,  
CE is empirical  
BCE

$$\max_{\mathbf{w}} p_{\mathbf{y}(u)|\mathbf{X}(u)}(\mathbf{y}|\mathbf{X}; \mathbf{w}) \iff \min_{\mathbf{w}} \overline{\text{BCE}}(p_{\text{data}}, p_{\text{model}}; \mathbf{w})$$

# ML Estimation and Information Theory

**Example:**  $y_n = m$ , with probability  $p_m = \text{softmax}_m(\mathbf{W}\mathbf{x}_n)$ ,  $m = 0, 1, 2, \dots, M-1$ , i.i.d.

$$\text{softmax}_m(\mathbf{v}) = \frac{e^{v_m}}{\sum_{j=0}^{M-1} e^{v_j}}$$

direct generalization of logistic function  
— maps log-likelihoods to pmf

M-value model,  
CE is multi  
category  
empirical CE

$$\max_{\mathbf{w}} p_{\mathbf{y}(u)|\mathbf{X}(u)}(\mathbf{y}|\mathbf{X}; \mathbf{w}) \iff \min_{\mathbf{w}} \overline{\text{MCE}}(p_{\text{data}}, p_{\text{model}}; \mathbf{w})$$

- Labels are usually “one-hot” ( $p_{\text{data}}$  is a point mass) — “hard labels”
- Like this:  $p(\text{cat})=1$ ,  $p(\text{dog})=0$ ,  $p(\text{bird})=0$
- Not this:  $p(\text{cat})=0.9$ ,  $p(\text{dog})=0.15$ ,  $p(\text{bird})=0.05$

$$\overline{\text{MCE}}(p_{\text{data}}, p_{\text{model}}(\mathbf{w})) = \frac{-1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} I[y_n = m] \log(p_{\text{model}}(y_n = m; \mathbf{w}))$$

# Multi-class Cross Entropy Example

One hot encoding:

cat: 0  
dog: 1  
bird: 2

Sample data labels:


n=0: y=1 (dog)  
n=1: y=2 (bird)  
n=2: y=0 (cat)

Classifier Output:

n=0: [ 0.3, 0.5, 0.2]  
n=1: [ 0, 0, 1]  
n=2: [ 0.4, 0.5, 0.1]

[ p(cat), p(dog), p(bird) ]

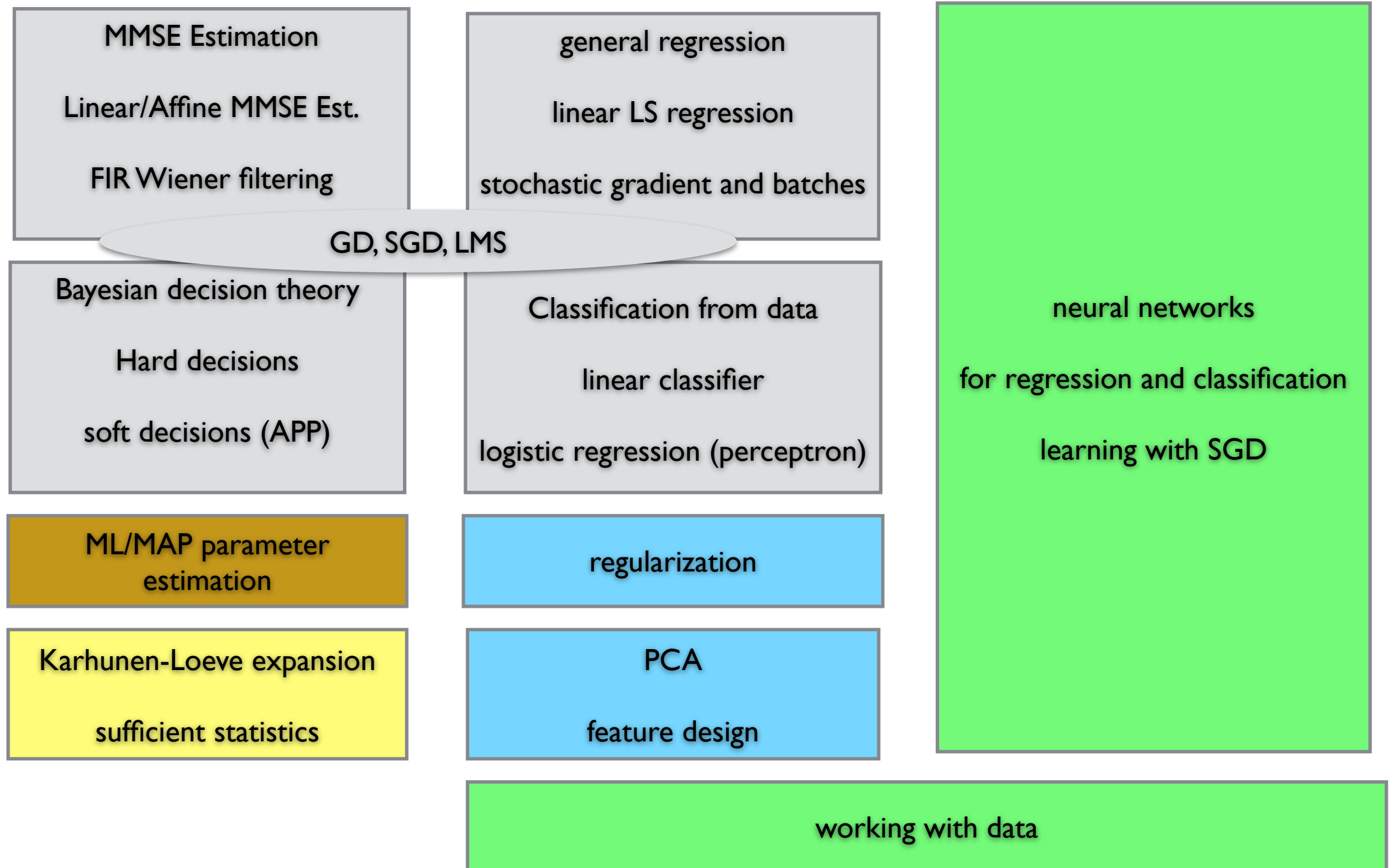
$$\text{Loss} = \frac{-1}{3} [\log(0.5) + \log(1) + \log(0.4)]$$


$$\overline{\text{MCE}}(p_{\text{data}}, p_{\text{model}}(\mathbf{w})) = \frac{-1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} I[y_n = m] \log(p_{\text{model}}(y_n = m; \mathbf{w}))$$

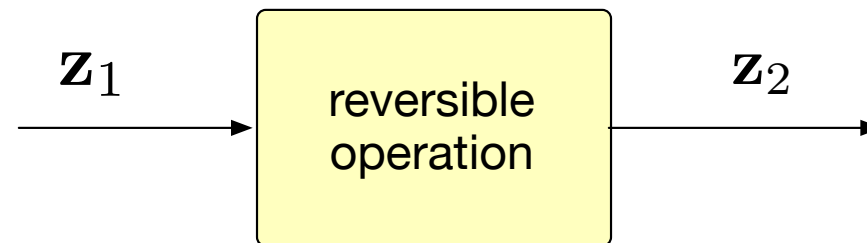
# Detection, Estimation, Regression

statistical models

data driven



# Theorem of Reversibility



$z_2$  can be used to perform inference on  $d$  without a loss of information

This may complicate or simplify, but, for example, the Max-Likelihood processing will yield same results



# Theorem of Irrelevance

Suppose we have two observations

$$p(\mathbf{z}_1, \mathbf{z}_2 | d) = p(\mathbf{z}_1 | \mathbf{z}_2, d) p(\mathbf{z}_2 | d)$$

If the following holds

$$p(\mathbf{z}_1 | \mathbf{z}_2, d) = p(\mathbf{z}_1 | \mathbf{z}_2)$$

Then we say that  $\mathbf{z}_1$  is irrelevant given  $\mathbf{z}_2$  for the purposes of inferring  $d$  (i.e., detection/estimation)

This means that  $\mathbf{z}_1$  can be thrown away if we have  $\mathbf{z}_2$  without loss of information for the purposes of inferring  $d$

**some irreversible operation are ok!**

# Set of Sufficient Statistics

A set of sufficient statistics for inferring  $d$  is a function of the observation that makes the observation irrelevant

$$\begin{aligned} p(\mathbf{z}, \mathbf{g}(\mathbf{z})|d) &= p(\mathbf{z}|\mathbf{g}(\mathbf{z}), d)p(\mathbf{g}(\mathbf{z})|d) \\ &= p(\mathbf{z}|\mathbf{g}(\mathbf{z}))p(\mathbf{g}(\mathbf{z})|d) \\ &\equiv p(\mathbf{g}(\mathbf{z})|d) \end{aligned}$$

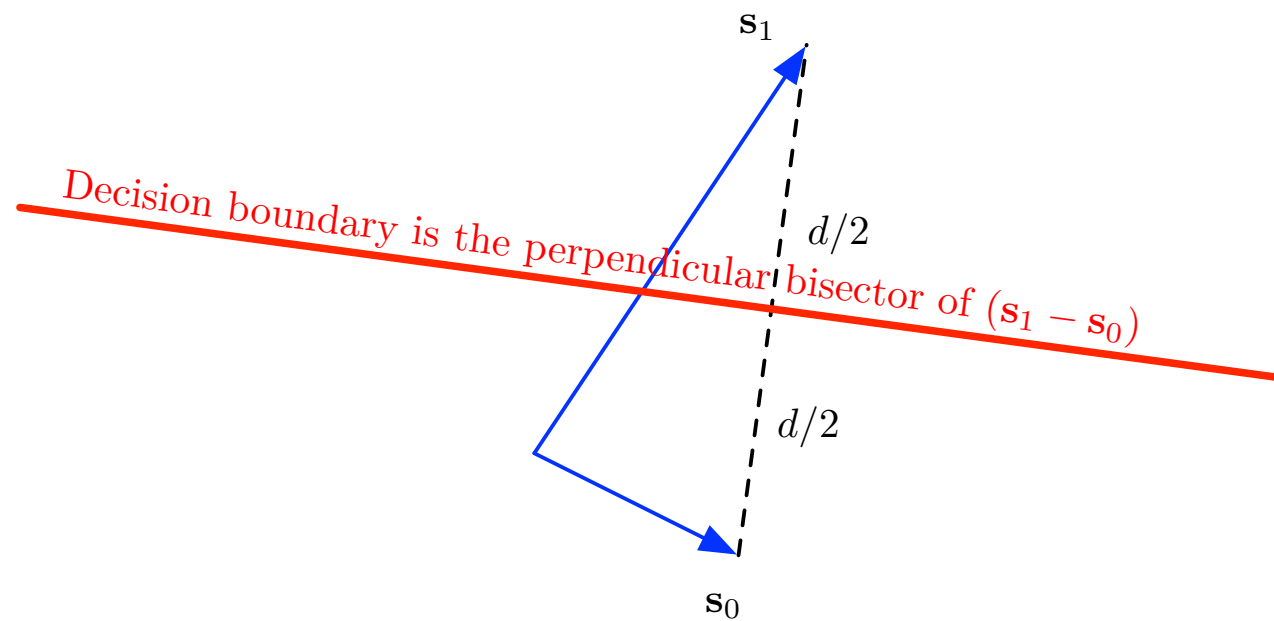
Example:

$\{\mathbf{z}^t \mathbf{s}_m\}_{m=0}^{M-1}$  is a set of sufficient stats for the vector AWGN channel

**Note:** this can be a large reduction in the number of dimensions without any loss of optimality

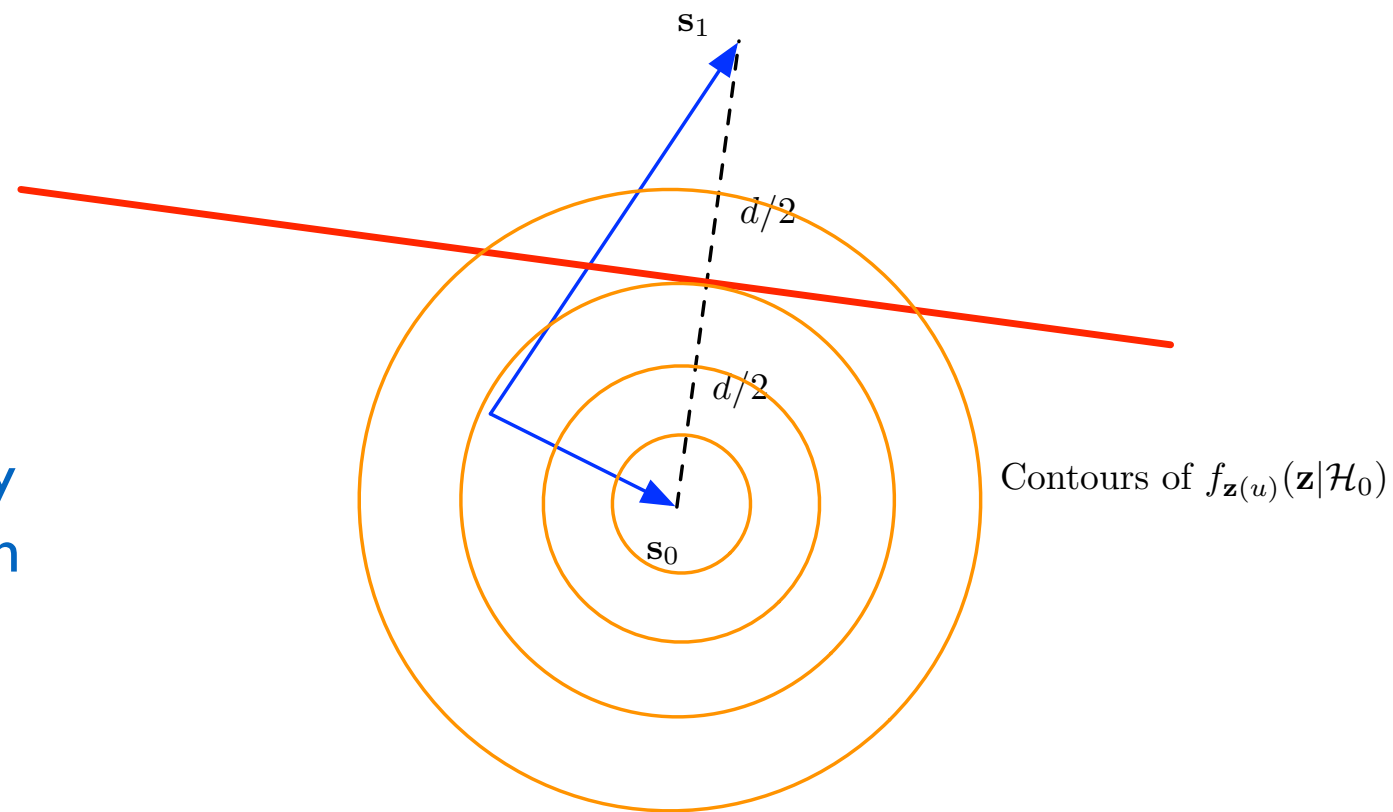
**sufficient statistics are the ideal features**

# EXAMPLE: Binary MAP Decisions (equal priors)



$$(\mathbf{s}_1 - \mathbf{s}_0)^T \mathbf{z} \begin{cases} \geq & \mathcal{H}_1 \\ < & \mathcal{H}_0 \end{cases} \frac{\|\mathbf{s}_1\|^2 - \|\mathbf{s}_0\|^2 - N_0 \ln(\pi_1/\pi_0)}{2}$$

Error probability given hypothesis 0 is the probability that noise throws observation over decision boundary



# Sufficient Stat for this Problem...

$$g(\mathbf{z}) = \mathbf{z}^t (\mathbf{s}_1 - \mathbf{s}_0)$$

definitely not reversible!

**Note:** the original dimension may be 2 or 1,000,000 and the sufficient stat still has only dimension 1.

this reduces the observation to the essential information relevant to inferring between the two possibilities from the observation

(aside: the performance is not a function of dimension either in this case)

**sufficient statistics are the ideal features**

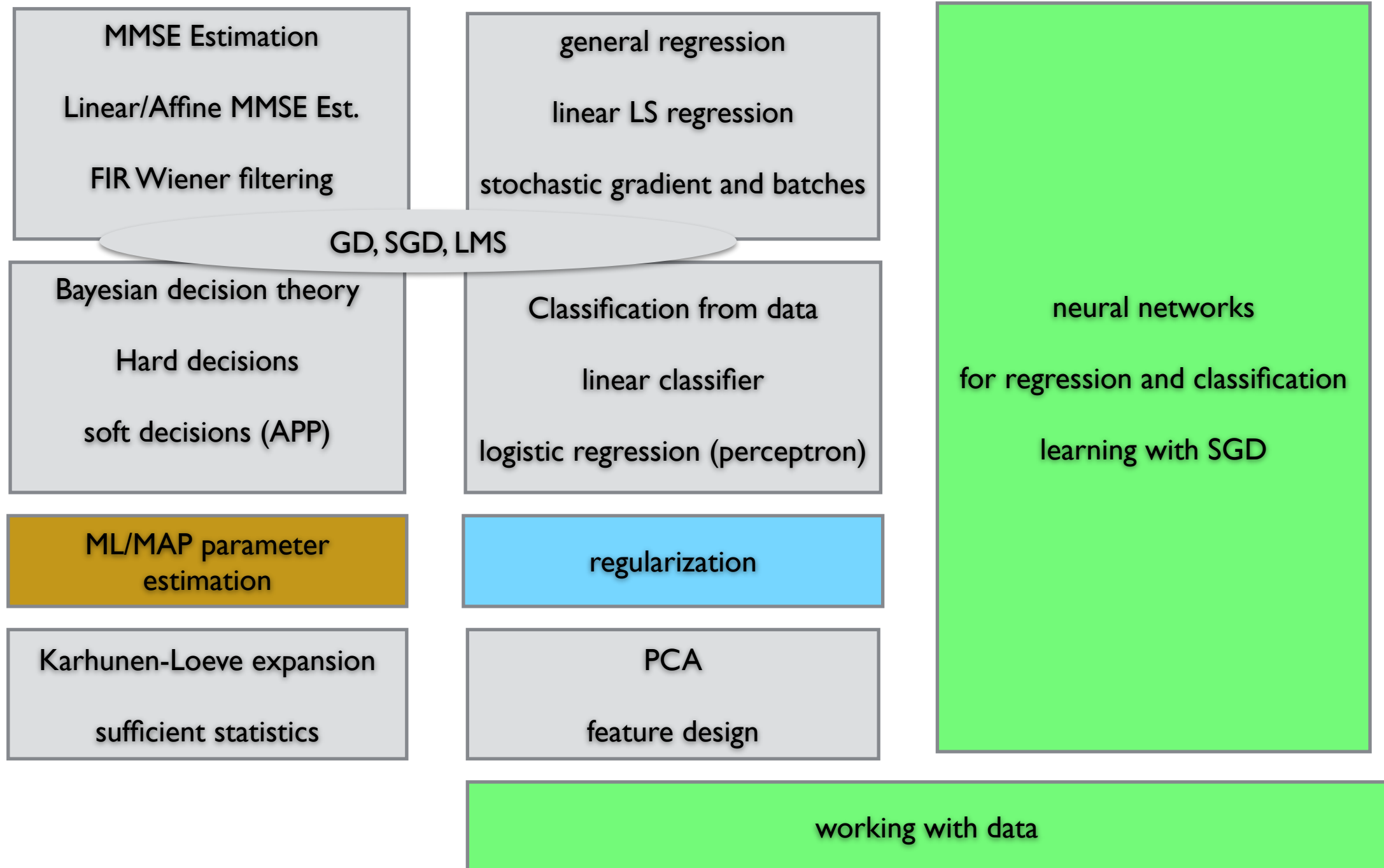
# More on PCA and LDA

I am going to delay this a bit...

# Detection, Estimation, Regression

statistical models

data driven



# MAP Estimation (parameter estimation)

Theta is a parameter set with a known/modeled a priori distribution:  
like ML, but with a prior distribution on Theta

$$\hat{\Theta}_{MAP} = \hat{\Theta}_{MAP}(y) = \arg \max_{\theta} p_{y(u)|\Theta(u)}(y|\theta) p_{\Theta(u)}(\theta)$$

May be based on a conditional or joint distribution:

$$\hat{\Theta}_{MAP} = \hat{\Theta}_{MAP}(y, \mathbf{x}) = \arg \max_{\theta} p_{y(u)|\mathbf{x}(u), \Theta(u)}(y|\mathbf{x}, \theta) p_{\Theta(u)|\mathbf{x}(u)}(\theta|\mathbf{x})$$

$$\hat{\Theta}_{MAP} = \hat{\Theta}_{MAP}(y, \mathbf{x}) = \arg \max_{\theta} p_{y(u), \mathbf{x}(u), \Theta(u)}(y, \mathbf{x}, \theta) p_{\Theta(u)}(\theta)$$

# Example of MAP estimation

## Assumed model:

$$y_n = \mathbf{w}^t \mathbf{x}_n + v_n, \quad n = 0, 1, \dots, N-1$$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{v}(u)$$

$$p_{\mathbf{v}(u)}(\mathbf{v}) = \mathcal{N}_N(\mathbf{v}; \mathbf{0}; \sigma_v^2 \mathbf{I})$$

$$p_{\mathbf{w}(u)}(\mathbf{w}) = \mathcal{N}_D(\mathbf{w}; \mathbf{0}; \sigma_w^2 \mathbf{I})$$

assume that  $\mathbf{w}$  and  $\mathbf{v}$   
are independent

now, we model  $\mathbf{w}$  as  
 $\mathbf{w}(u)$  — i.e., model the weights  
as random with some known  
distribution

$$\begin{aligned} p_{\mathbf{y}(u)|\mathbf{X}(u),\mathbf{w}(u)}(\mathbf{y}|\mathbf{X}, \mathbf{w}) &= p_{\mathbf{v}(u)|\mathbf{w}(u)}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= p_{\mathbf{v}(u)}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathcal{N}_N(\mathbf{y}; \mathbf{X}\mathbf{w}; \sigma_v^2 \mathbf{I}) \end{aligned}$$

Find the negative log of this...

$$\begin{aligned} -\ln [p_{\mathbf{y}(u)|\mathbf{X}(u),\mathbf{w}(u)}(\mathbf{y}|\mathbf{x}, \mathbf{w}) p_{\mathbf{w}(u)}(\mathbf{w})] &= -\ln [\mathcal{N}_N(\mathbf{y}; \mathbf{X}\mathbf{w}; \sigma_v^2 \mathbf{I}) \mathcal{N}_D(\mathbf{w}; \mathbf{0}; \sigma_w^2 \mathbf{I})] \\ &= \left[ \frac{1}{2\sigma_v^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{D}{2} \ln(2\pi\sigma_v^2) \right] \\ &\quad + \left[ \frac{1}{2\sigma_w^2} \|\mathbf{w}\|^2 + \frac{N}{2} \ln(2\pi\sigma_w^2) \right] \\ &\equiv \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \end{aligned}$$



# Example of MAP estimation

Assumed model:  $y_n = \mathbf{w}^t \mathbf{x}_n + v_n, \quad n = 0, 1, \dots, N-1$

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \mathbf{v}(u)$$

$$p_{\mathbf{v}(u)}(\mathbf{v}) = \mathcal{N}_N(\mathbf{v}; \mathbf{0}; \sigma_v^2 \mathbf{I})$$

$$p_{\mathbf{w}(u)}(\mathbf{w}) = \mathcal{N}_D(\mathbf{w}; \mathbf{0}; \sigma_w^2 \mathbf{I})$$

assume that  $\mathbf{w}$  and  $\mathbf{v}$   
are independent

Find the negative log of this...

$$\begin{aligned} -\ln [p_{\mathbf{y}(u)|\mathbf{X}(u),\mathbf{w}(u)}(\mathbf{y}|\mathbf{x}, \mathbf{w})p_{\mathbf{w}(u)}(\mathbf{w})] &= -\ln [\mathcal{N}_N(\mathbf{y}; \mathbf{X}\mathbf{w}; \sigma_v^2 \mathbf{I})\mathcal{N}_D(\mathbf{w}; \mathbf{0}; \sigma_w^2 \mathbf{I})] \\ &= \left[ \frac{1}{2\sigma_v^2} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \frac{D}{2} \ln(2\pi\sigma_v^2) \right] \\ &\quad + \left[ \frac{1}{2\sigma_w^2} \|\mathbf{w}\|^2 + \frac{N}{2} \ln(2\pi\sigma_w^2) \right] \\ &\equiv \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \\ \lambda &= \frac{\sigma_v^2}{\sigma_w^2} \end{aligned}$$

# Regularization Interpretation...

$$\max_{\theta} p_{y(u)|\mathbf{x}(u),\Theta(u)}(y|\mathbf{x},\theta) p_{\Theta(u)|\mathbf{x}(u)}(\theta|\mathbf{x}) \iff \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2$$

The a-priori Gaussian distribution on the weights leads to “L2 regularization”

penalizes large  $\mathbf{w}$  — even if large  $\mathbf{w}$  cause smaller squared error

this can be viewed a method to combat **over-fitting**

*lambda* is called the regularization coefficient in this context

*Larger lambda ==> penalize larger weights more aggressively (at expense of SE)*

# Regularization Interpretation...

Another popular type of regularizer is “L1 regularizer”

for example, for squared-error cost function with L1 regularization:

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|_1$$

$$\|\mathbf{w}\|_1 = \sum_i |w_i|$$

## Questions:

- does this correspond to an a-priori distribution on the weights?
- If so, which one?
- Qualitatively, what is the difference between L1 and L2 regularization?

# Regularization

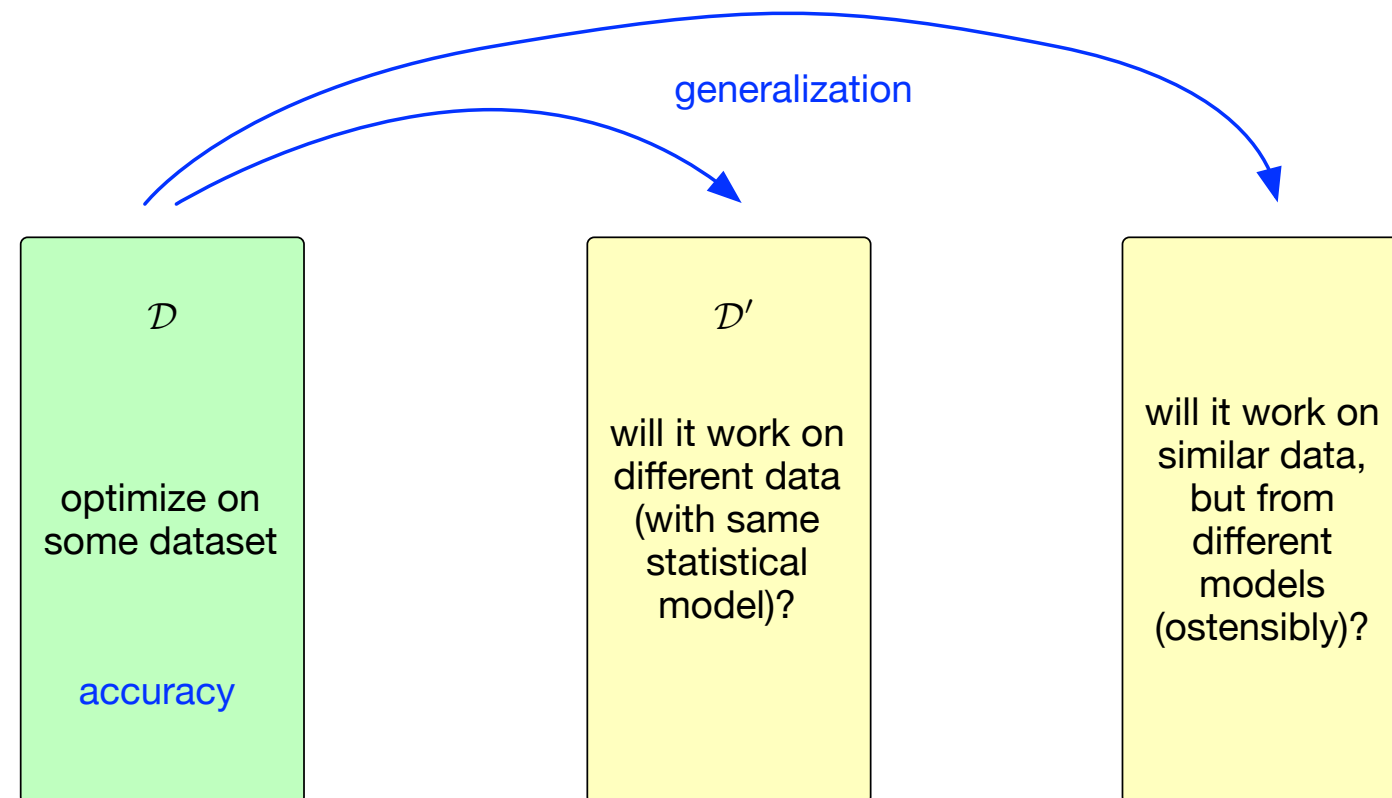
What is regularization and why do it?

We have seen an example: enforce penalty on weights to bias toward a prior distribution.  
effect is to reduce over-fitting (get smaller weights)

Not all regularization methods can be viewed this way  
in some cases, intuitive, empirical penalty enforcing functions are used

What is a more general definition of regularization?

# Machine Learning Goal



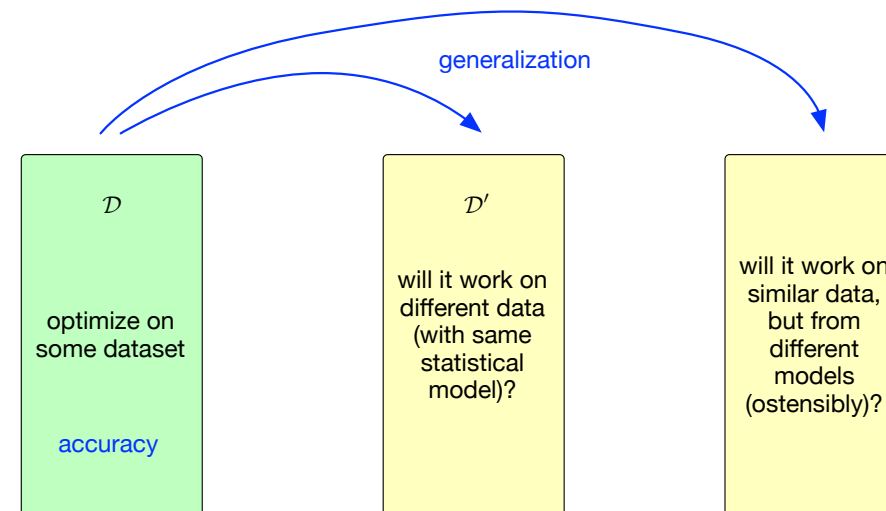
in the end, we do not really care about the performance on the dataset we have  
(it is labeled, after all)

we care about performance on similar data that has no labels

Accuracy/Generalization trade-off (aka bias-variance trade):

generally, optimizing accuracy to the extreme will cause reduced generalization  
capability

# Machine Learning Goal



**Accuracy/Generalization trade-off (aka bias-variance trade):**

generally, optimizing accuracy to the extreme will cause reduced generalization capability

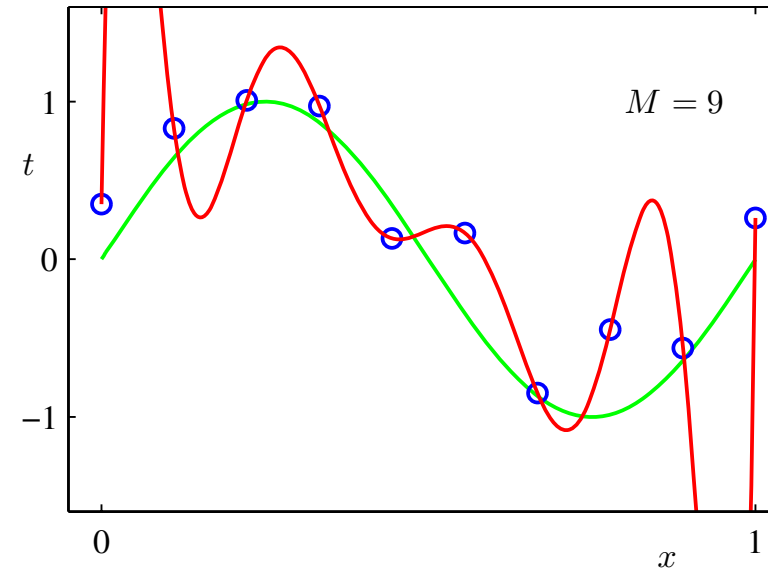
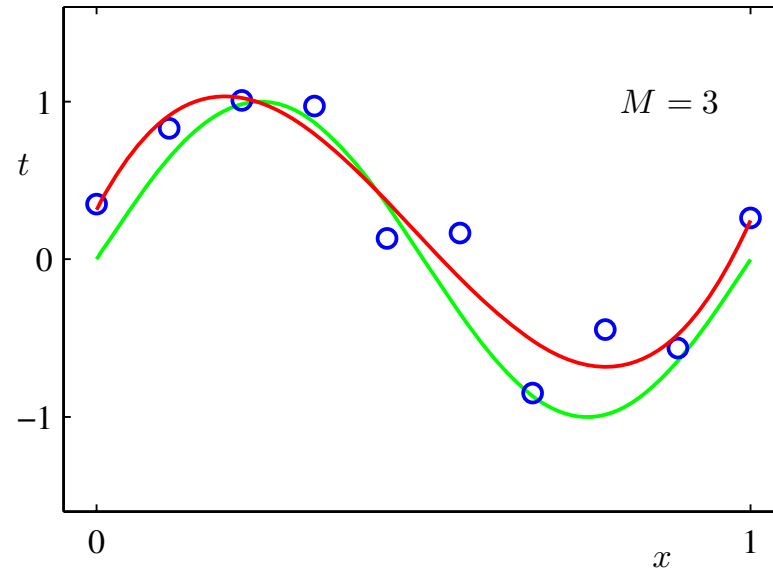
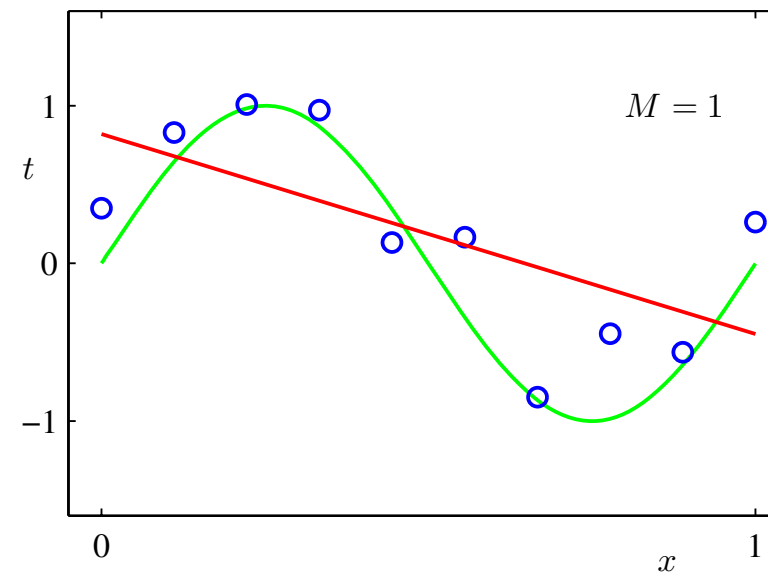
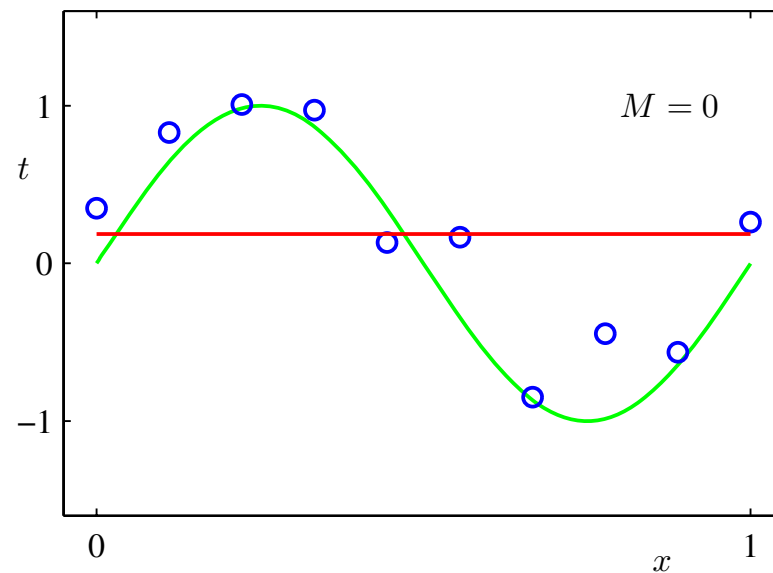
**extreme ML case:**

just make a table of  $(x_n, y_n)$

perfect accuracy, but no ability to generalize!

# Regression from Data

under-fitting

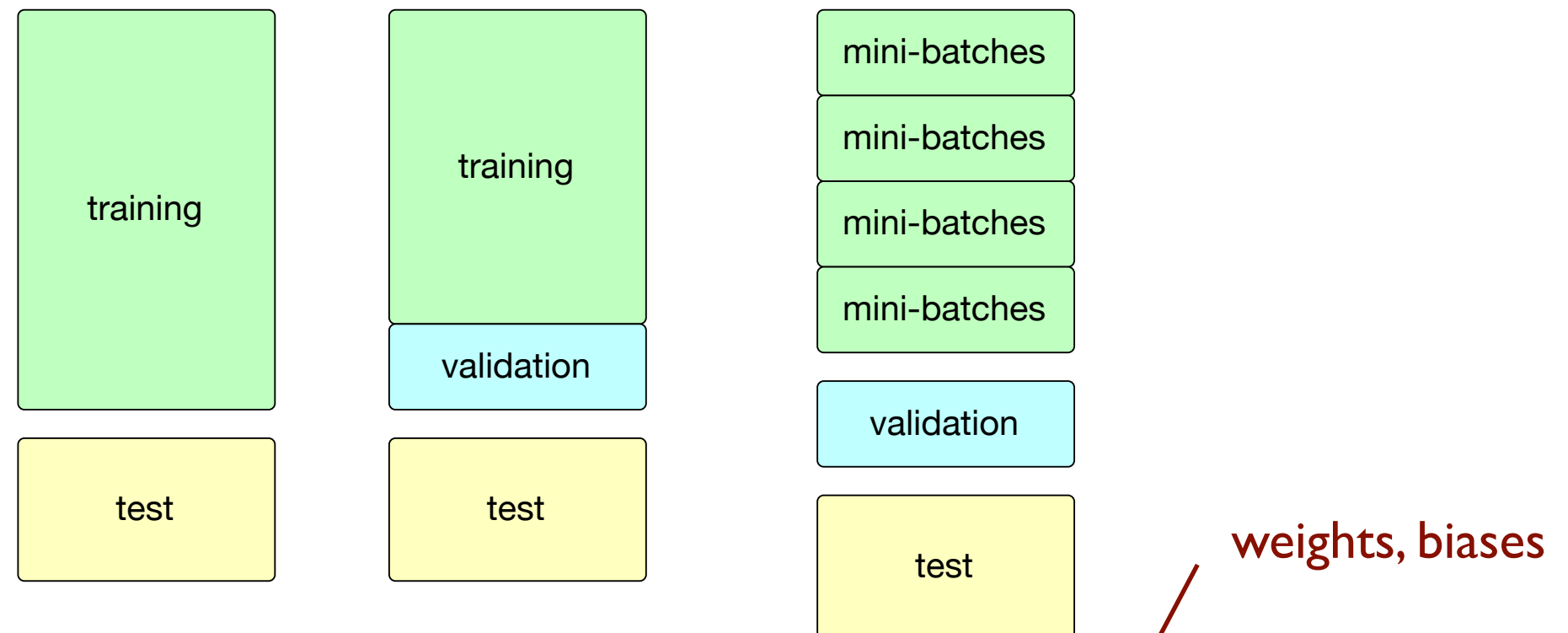


over-fitting

**Figure 1.4** Plots of polynomials having various orders  $M$ , shown as red curves, fitted to the data set shown in Figure 1.2.

*Choosing the right model (complexity) is challenging given a finite data set and no good model for what generated it!!!*

# Training (+ Val)/Test Split



**(training-)training:** use this for SGD learning — trainable parameters

**(training-)validation:** use this for SGD learning — hyper-parameters

**test:** only use this when you are done to verify

learning rate, batch size,  
etc.



# Training (+ Val)/Test Split

Typical Train/Val/Test split:

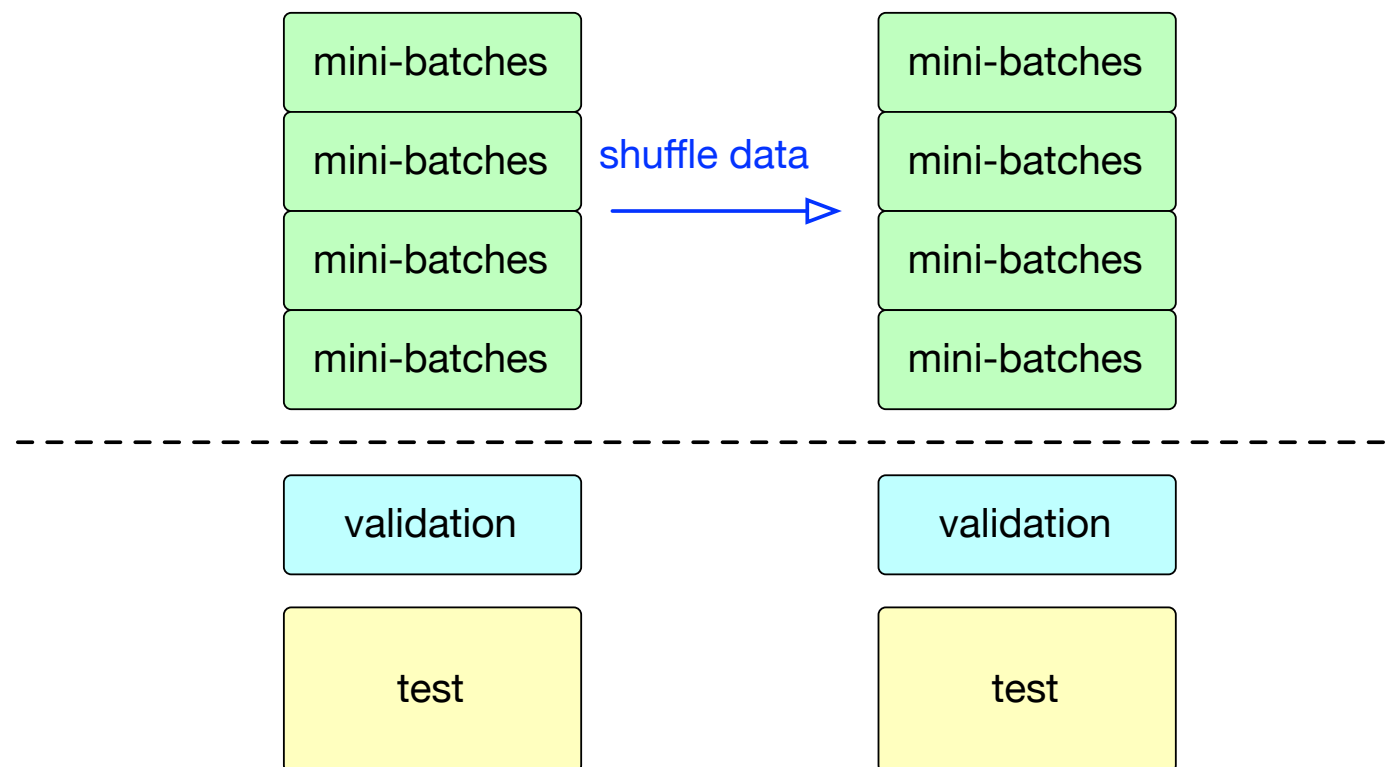
70/15/15

need your val and test splits to be large enough to capture natural variation in the data

need your val and test splits to be large enough to allow reliable classification error estimation

want lots of data

# Training (+ Val)/Test Split



shuffle all data in training (not including validation) after each epoch

```
perm = np.random.permutation(N_train)
x_train = x_train[perm]
y_train = y_train[perm]
```

good practice to shuffle all of the data once before the train/val/test split

# Training (+ Val)/Test Split

**(mini)-batch:** do one SGD update (averaging) per mini-batch

**(mini)-batch size:** number of data examples per mini-batch

**epoch:** one training run through all of the training data

**iteration:** number of mini-batches per epoch

typically, we “test” the model on the validation data at the end of each epoch

# Training (+ Val)/Test Split

## Example

100,000 total ( $x_n, y_n$ ) (shuffle it all once)

70,000 train

15,000 val

15,000 test

batch size = 70:

1000 mini-batches in the training data (1000 iterations per epoch)

1000 gradient updates in an epoch, each averaged over 70 samples

these are typically processed serially: batch 1, batch 2, etc.

gradient updates are serial

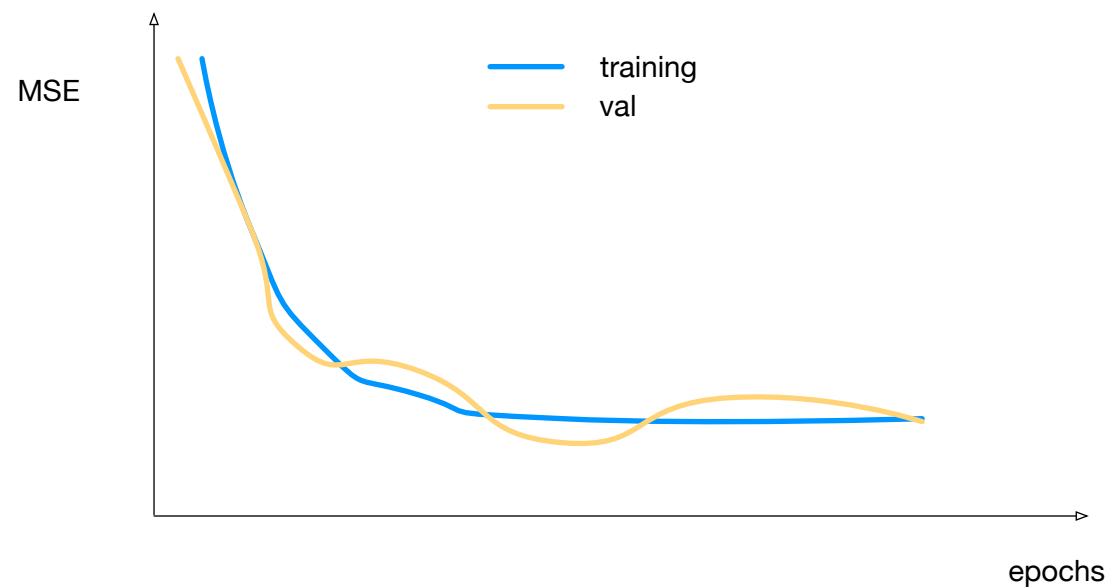
(can change with many parallel compute nodes)

run inference (forward only) on val data after each epoch

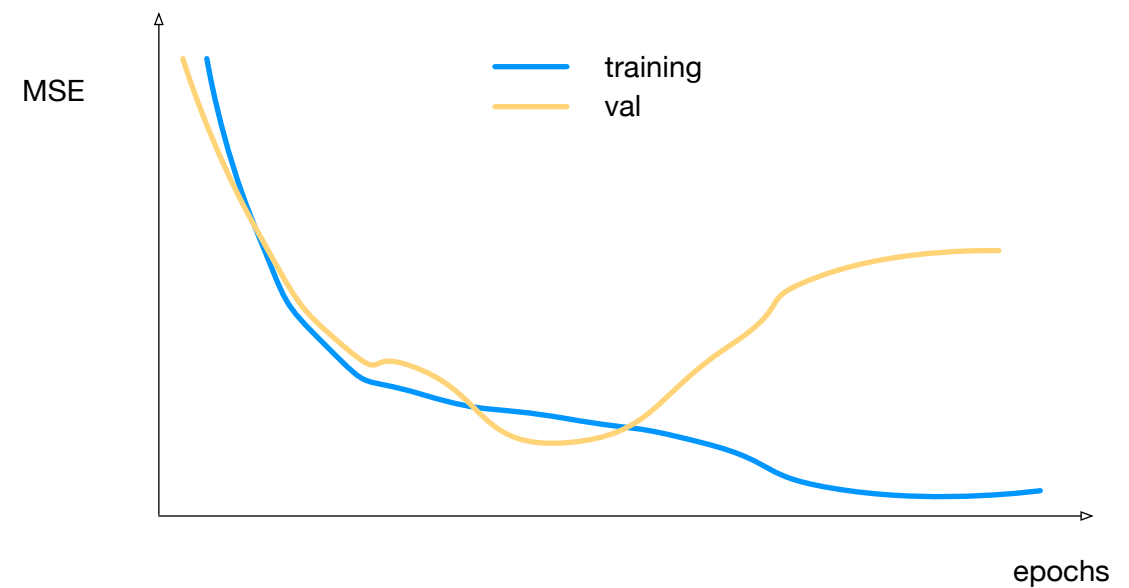
monitor learning curve, iteration hyper-hyper-parameter search...

when done, run on test

# Over-fitting



desired behavior



typical over-fitting

Better performance on training and worse or non-improving  
on val (for Prob Correct classification, it gets higher)

# Regularization

What is regularization and why do it?

We have seen an example: enforce penalty on weights to bias toward a prior distribution.

effect is to reduce over-fitting (get smaller weights)

Not all regularization methods can be viewed this way

in some cases, intuitive, empirical penalty enforcing functions are used

What is a more general definition of regularization?

**regularization is anything you do in training that is aimed at improving generalization over accuracy — i.e., anything that does not optimize the cost on the training data**

we will see very different versions of this — e.g., drop-out

# Main Ideas from Background

- Random vectors
  - Eigenvalues of covariance matrix provides information regarding direction preferences (principle components)
  - May drop directions with very little energy/power
- Estimation
  - MMSE estimator is conditional expectation — difficult to find
  - Linear/Affine MMSE is simple and only depends on second moments
  - For jointly-Gaussian observed/desired, affine is optimal
- Detection
  - MAP rule is minimum error probability.
  - Requires complete statistical description

# Main Ideas from Background

- Regression (from data)
  - Linear regression is same as affine/linear MMSE estimation, but with data averaging replacing ensemble averaging
  - Stacking interpretation
  - ML parameter interpretation
  - MAP parameter interpretation for regularization
- Classification (from data)
  - Linear classifier: linear regression with  $\pm 1$  target and “slicer”
  - Logistic regression
- Information Theory:
  - ML parameter estimation  $\Rightarrow$  Empirical Cross-entropy loss function
  - Only called CE for classification tasks



# Learning More...

**ECE 562:** Covariance/Correlation, KLT, random vectors, sequences, waveforms, MMSE estimation

**ECE 563:** ML, MAP estimation, Kalman Filters, Least Squares, Recursive Least Squares, Consistency, Unbiased, etc., Decision theory

**ECE 565:** Information theory, limits of information storage and transmission

**ECE 517:** Statistics, regression, logistic regression, EM algorithm, Monte Carlo methods

**ECE 583:** LMS algorithm and spectral estimation

**ECE 588:** Gradient descent, SGD, more advanced numerical optimization, convex optimization, constrained optimization

**ECE 564:** decision theory and (loopy) belief propagation, digital comm/coding

**ECE 559:** most overlap with this class, less neural networks, more detail on material up until this point + SVMs

**ECE 660:** also overlap with this class, unsupervised learning, decision trees, boot-strap, SVMs, etc.

**ECE 500:** neural networks + related topics in statistics and inference including fuzzy logic

**all of the above have less applied work, less python, etc.**