# Introduction to Machine Learning
## Lecture 1

Michael Kagan

SLAC

Hadron Collider Physics Summer School
August 23, 2021

# Outline

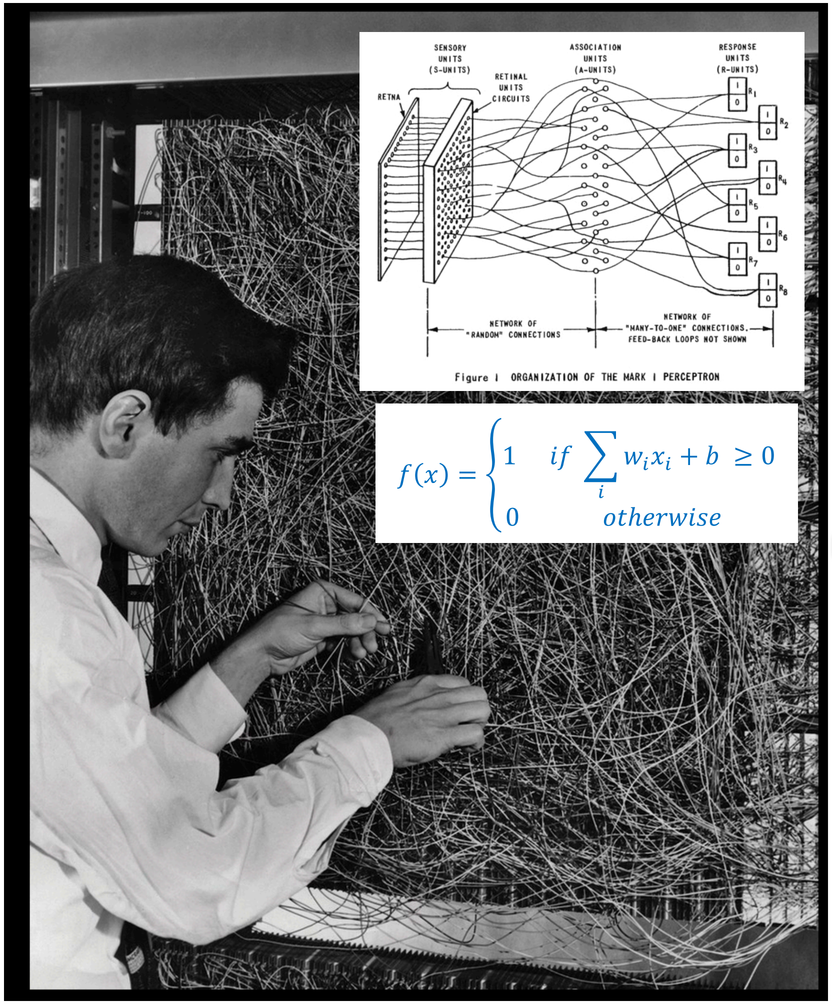
- Lecture 1
  - Brief introduction to probability and statistics
  - Introduction to Machine Learning fundamentals
  - Linear Models

- Lecture 2
  - Neural Networks
  - Deep Neural Networks
  - Convolutional, Recurrent, and Graph Neural Networks

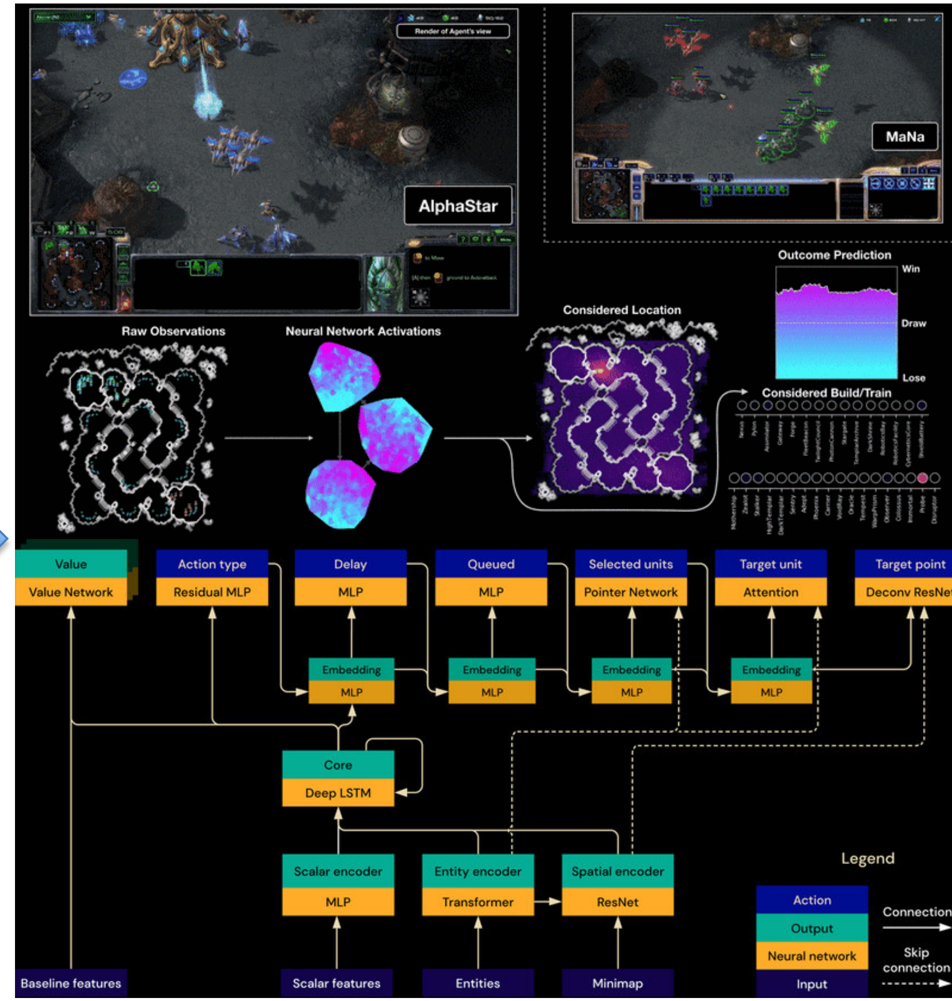
- Lecture 3
  - Unsupervised Learning
  - Autoencoders
  - Generative Adversarial Networks and Normalizing Flows

# Long History of Machine Learning






Perceptron

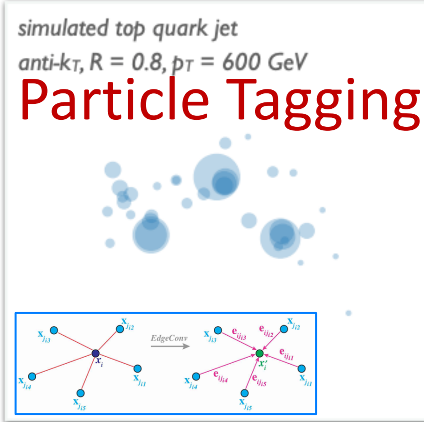


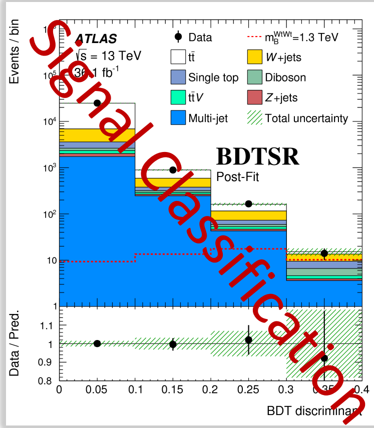

AlphaStar

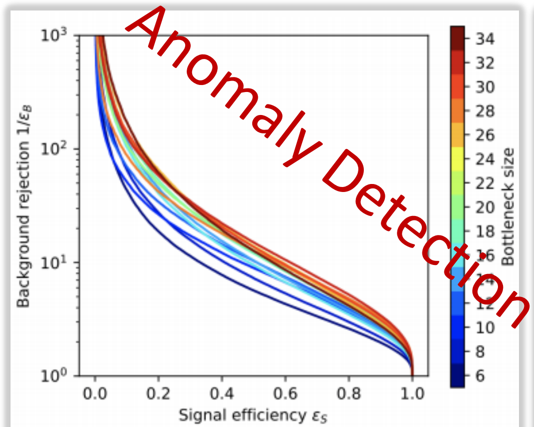
Rosenblatt 1958, 1960

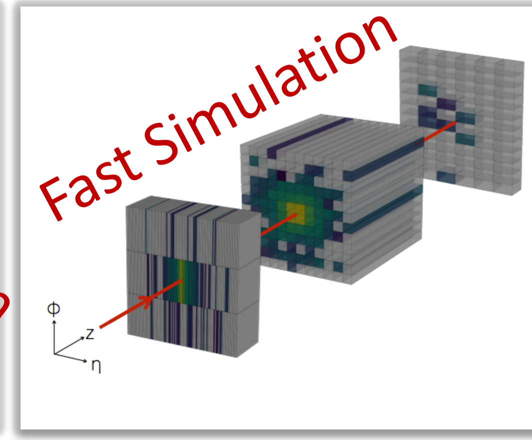
Vinyals et. al. 2019

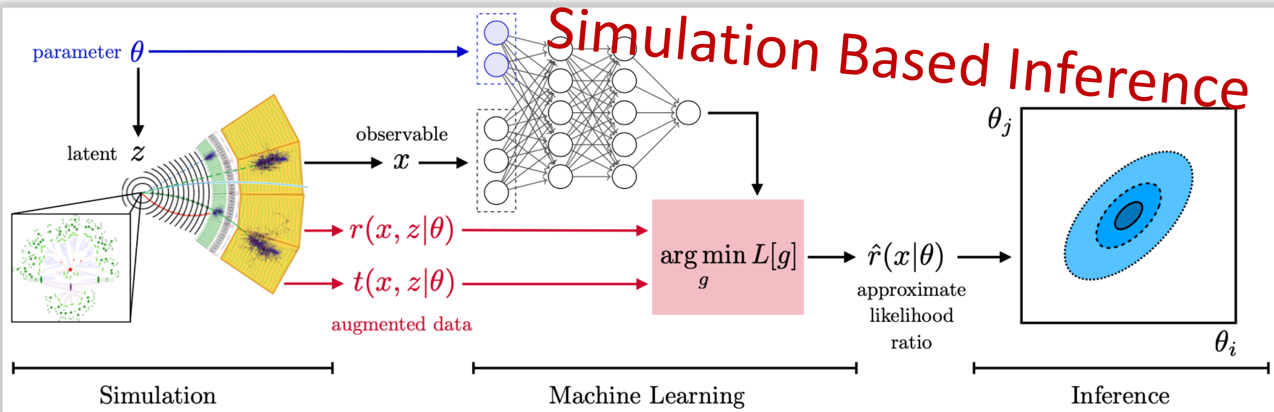
# Machine Learning in HEP

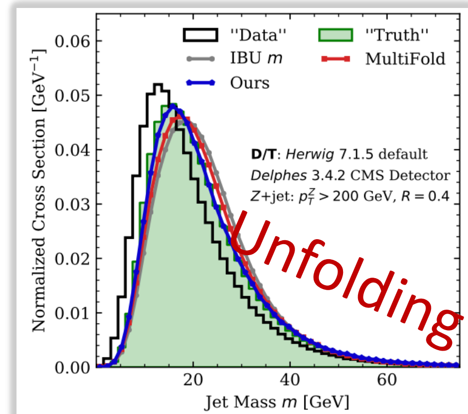


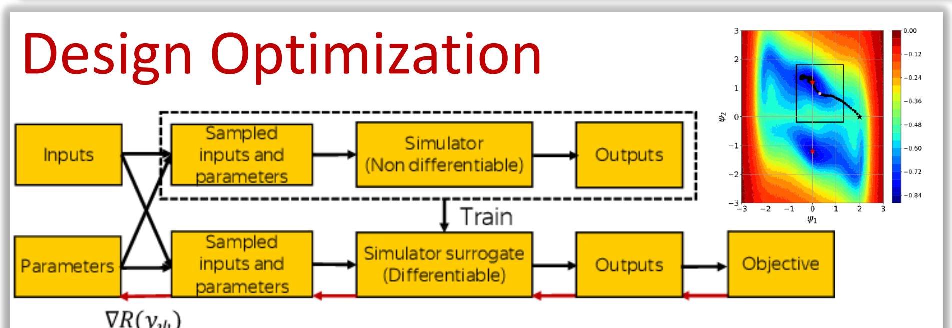



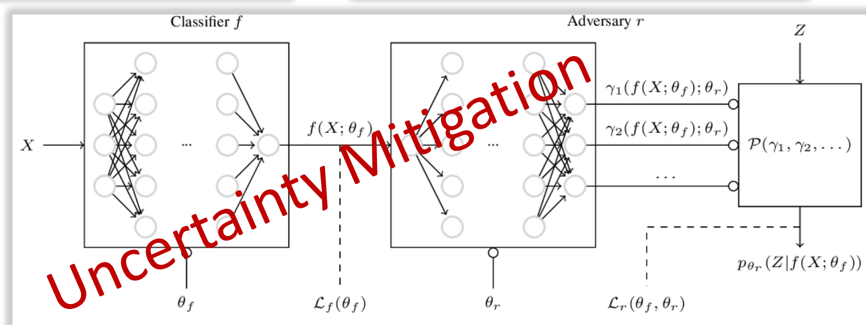




**+ More! Check out** [The Living Review of ML in HEP]

# What is Machine Learning?



- Giving computers the ability to learn without explicitly programming them (Arthur Samuel, 1959)
- Statistics + Algorithms
- Computer Science + Probability + Optimization Techniques
- **Fitting data with complex functions**
- **Mathematical models** learnt from data that characterize the patterns, regularities, and relationships amongst variables in the system

# Machine Learning: Models

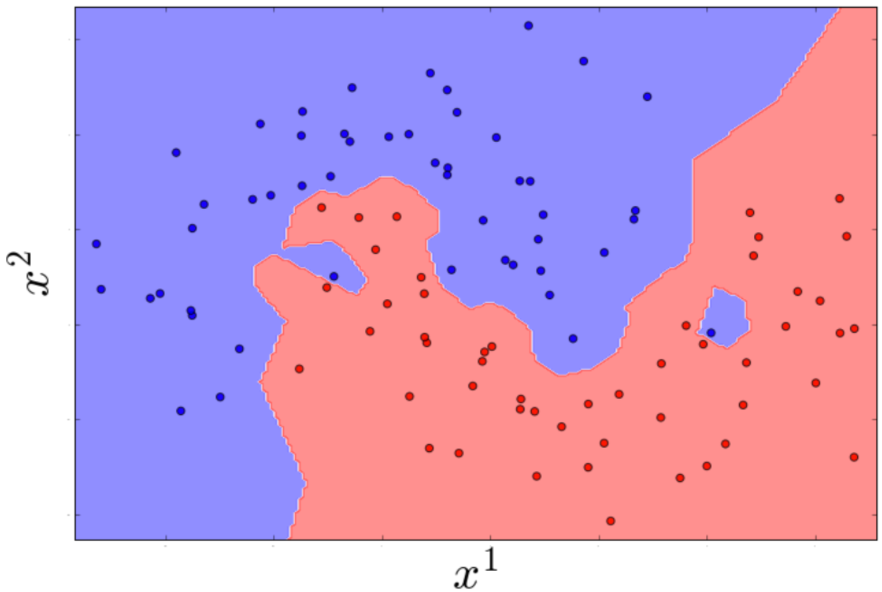
- Key element is a **mathematical model**
  - A mathematical characterization of system(s) of interest, typically via random variables
  - Chosen model depends on the task / available data
- **Learning**: estimate statistical model from data
  - Supervised learning
  - Unsupervised Learning
  - Reinforcement Learning
  - …
- **Prediction and Inference:** using statistical model to make predictions on new data points and infer properties of system(s)

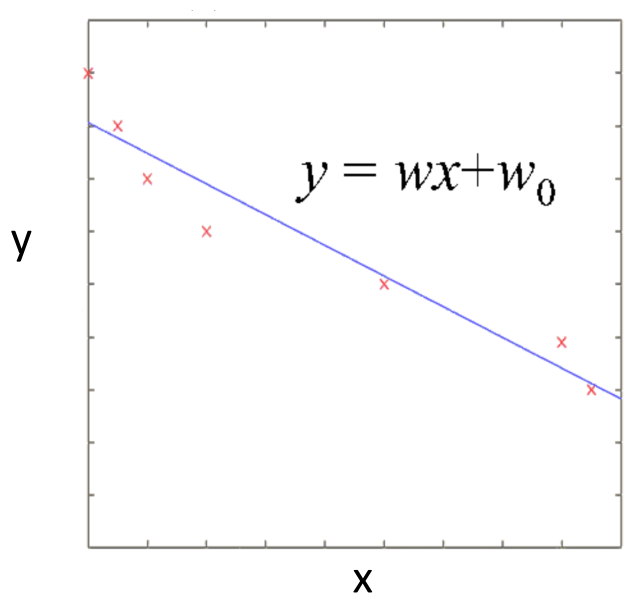
# Supervised Learning

- Given N examples with observable features $\{x_i \in \mathcal{X}\}$ and prediction **targets** $\{y_i \in \mathcal{Y}\}$, learn function mapping **h(x)=y**

**Classification:**
$\mathcal{Y}$ is a finite set of **labels** (i.e. classes) denoted with integers

**Regression:**
$\mathcal{Y}$ is a real number



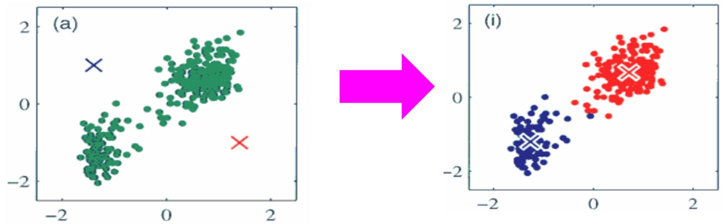



# Unsupervised Learning

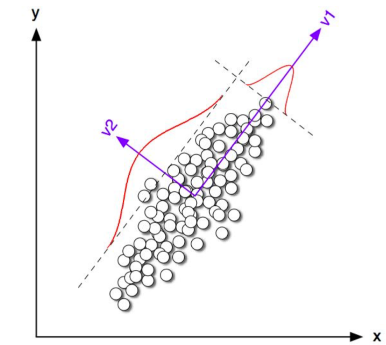
Given some data D={$x_i$}, but no labels, find structure in data

[Bishop]

**Clustering**: partition the data into groups D={$D_1 \cup D_2 \cup D_3 \ldots \cup D_k$}



**Dimensionality reduction**: find a low dimensional (less complex) representation of the data with a mapping Z=h(X)




**Density estimation and sampling**: estimate the PDF p(x), and/or learn to draw plausible new samples of x

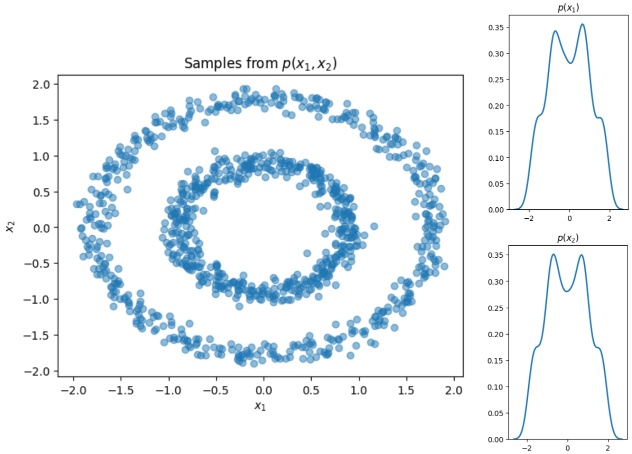



Image Credit - Link

# Reinforcement Learning

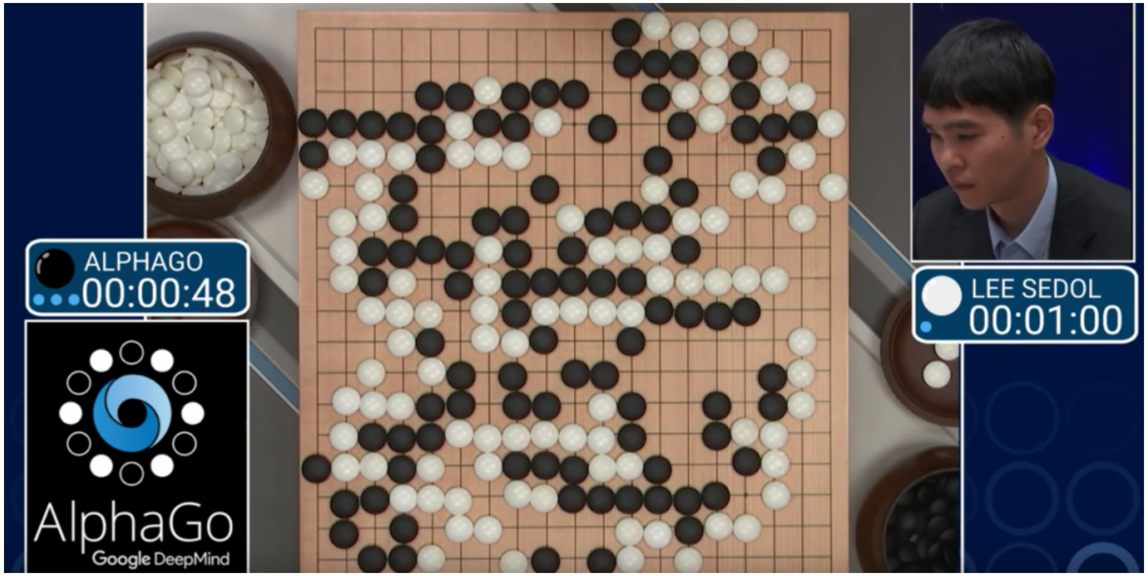


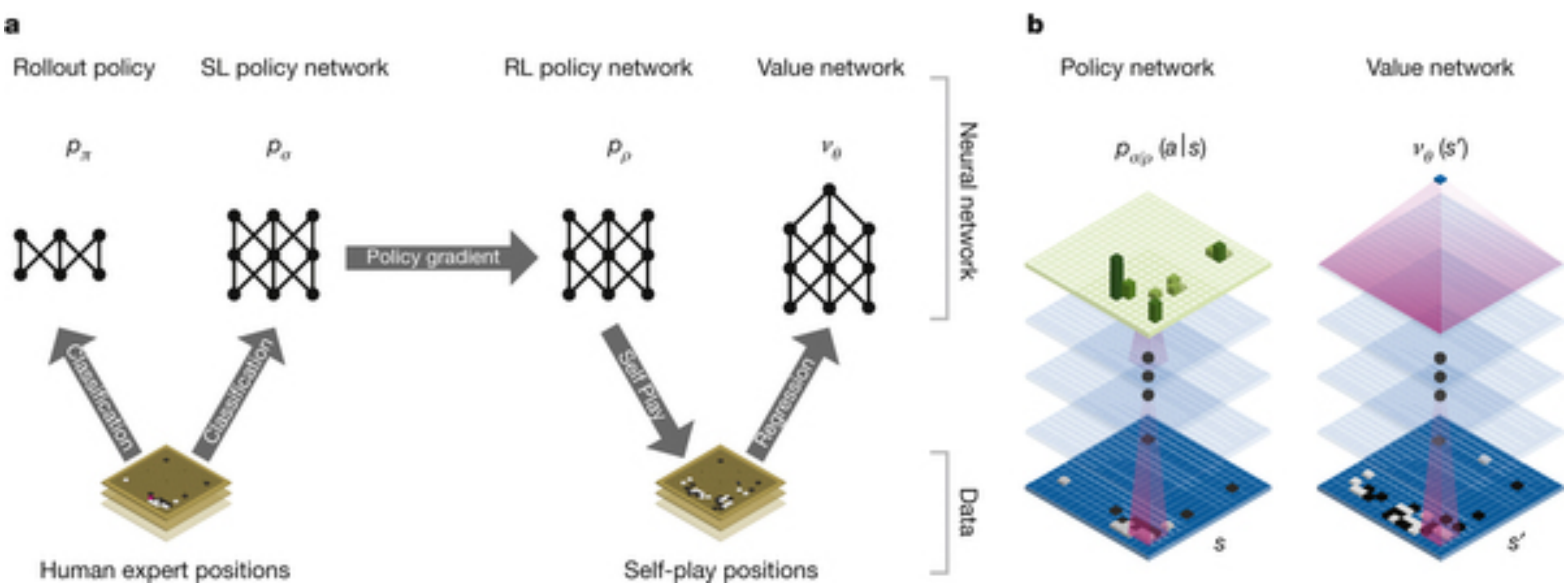

[Ravikumar]

- Models for agents that take actions depending on current state
  - Actions incur rewards, and affect future states ("feedback")
- Learn to make the best sequence of decisions to achieve a given goal when feedback is often delayed until you reach the goal

# Deep Reinforcement Learning with AlphaGo






Nature 529, 484–489 (28 January 2016)

# Brief Review of Probability and Statistics

# Probability Mass Function



**Probability Mass Function** for Discrete random variables (r.v.)

$$P(x_i) = p_i$$

- Prob. of $i^{th}$ outcome: limit of long term frequency $\lim_{N \to \infty} \frac{\# x_i}{N\, trials}$
- Normalized: $\sum_i P(x_i) = 1$

Bernoulli Distribution: $\mathrm{P}(x) = p^x (1-p)^{1-x}$

- $x \in \{0,1\}$ 1≡HEADS, 0≡ TAILS
- Biased coin with heads prob. $p \in [0,1]$

Slide Credit: K. Cranmer: Intro to Stats.

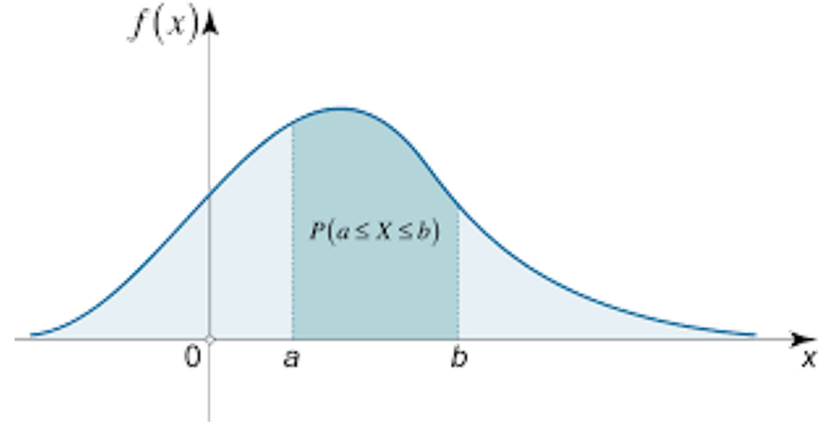
# Probability Mass and Density Functions



## Probability Density Function (PDF) for Continuous r.v.

$$P(x \in [x, x + dx]) = f(x)dx$$

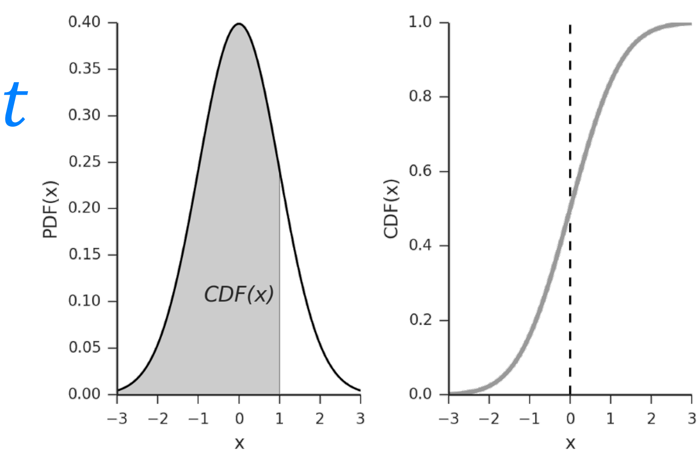
– Normalized: $\int_{-\infty}^{\infty} f(x)dx = 1$




## Cumulative Distribution Function

$$F_X(x) = P(X < x) = \int_{-\infty}^{x} f(t)dt$$

– Density defined as: $f(x) = \frac{\partial F_X(x)}{\partial x}$




Slide Credit: K. Cranmer: Intro to Stats.

# Expected Values

- Expected value of a function of random variables

$$\mathrm{E}[g(x)] = \int_{-\infty}^{\infty} g(x)p(x)dx$$

- Mean of a r.v. : $\mathrm{E}[x] = \bar{x} = \int_{-\infty}^{\infty} x\, p(x)dx$

- Variance: $Var(X) = \mathrm{E}[(x - \mathrm{E}[x])^2] = \mathrm{E}[x^2] - \mathrm{E}[x]^2$

- Covariance of two r.v.'s: $Cov(x, y) = \mathrm{E}[(x - \mathrm{E}[x])(y - \mathrm{E}[y])]$

# Expected Values



- Expected value of a function of random variables

$$\mathrm{E}[g(x)] = \int_{-\infty}^{\infty} g(x)p(x)dx$$

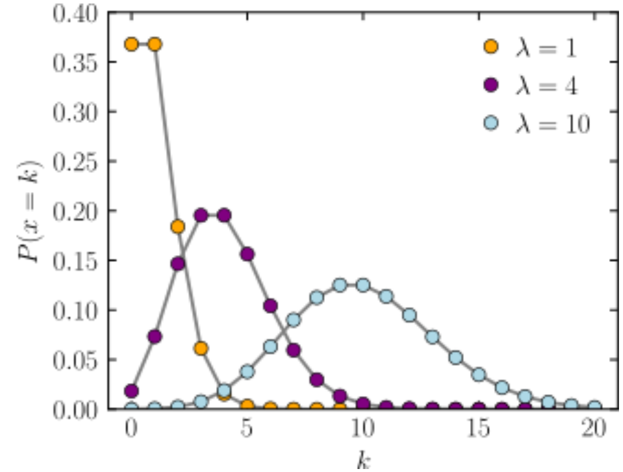
- Often we can't compute this integral
- Or often in Machine Learning we don't know $p(x)$

- With set of N repeated observations $\{x_i\}$ that are independent and identically distributed, can approximate with Empirical Estimator

$$\mathrm{E}[g(x)] \approx \frac{1}{N}\sum_{i=1}^{N} g(x_i)$$

# Parametric Models

- PDF often depends on parameters $\theta$ we are interested in
  - Write the density as $f(x|\theta)$ or $f(x;\theta)$




**Discrete**: Poisson Distribution:

$$Poiss(k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- Prob. of $k$ events in fixed interval of time
- $\lambda$ = average number of events

**Continuous**: Gaussian Distribution:

$$G(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- $\mu$ is the average value
- $\sigma^2$ is the variance

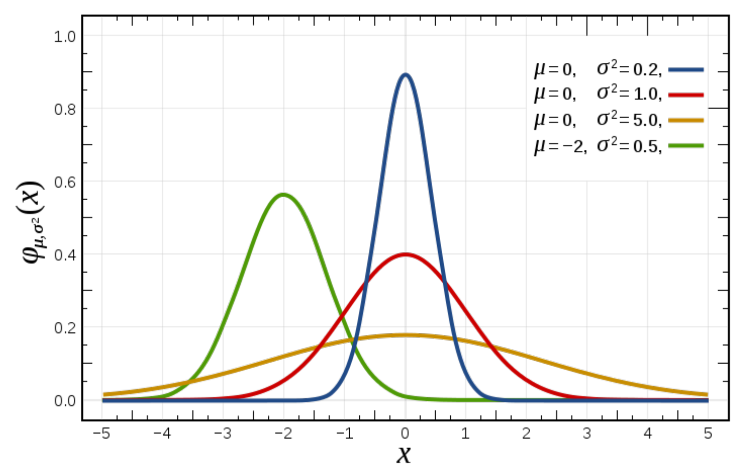



Image source: Wikipedia

# Likelihood Function



- Given value $x = x'$ to evaluate PDF, can consider it as a continuous function of the parameters $\theta$

Poisson Example: Likelihood of $\mu$ given $n$

$$L(\mu) = Poiss(n|\mu)$$

– Continuous function of $\mu$

– NOTE: not a PDF

– Common to examine $-\ln L$

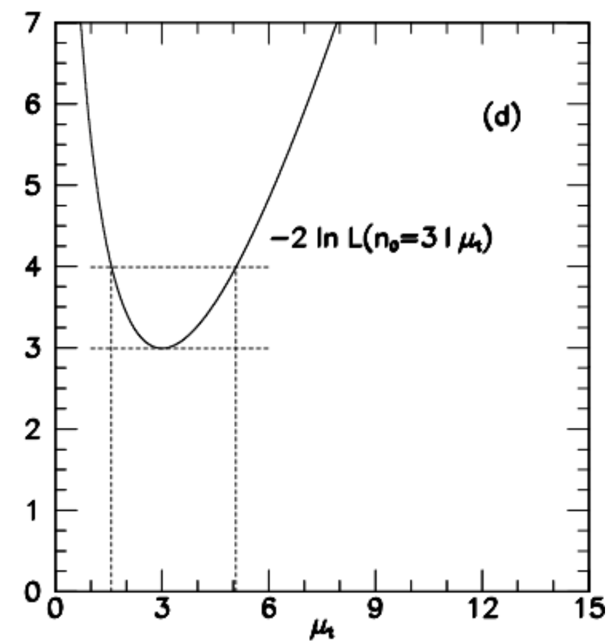



Figure from R. Cousins, Am. J. Phys. 63 398 (1995)

# Likelihood with Repeated Observations



- Given a set of repeated observations of $x$ that are independent and identically distributed
  - Repeated observations written $\{x_i\}$
  - $x \sim f(x|\theta)$ means the $x$ follows distribution $f(x|\theta)$

- Likelihood

$$L(\theta) = \prod_i f(x_i|\theta)$$

- Log-likelihood

$$\ln L(\theta) = \sum_i \ln f(x_i|\theta)$$

Slide Credit: K. Cranmer: Intro to Stats.

# Maximum Likelihood

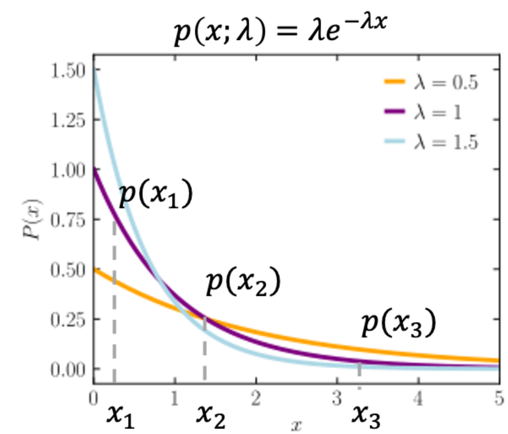
- Given observations $\{x_i\}$ and model PDF $f(x|\theta)$ the maximum likelihood estimator for $\theta$ is:

$$\theta^*(x) = \arg\max_\theta L(\theta) = \arg\min_\theta -\ln L(\theta)$$

# Maximum Likelihood

- Given observations $\{x_i\}$ and model PDF $f(x|\theta)$ the maximum likelihood estimator for $\theta$ is:

$$\theta^*(x) = \arg\max_\theta L(\theta) = \arg\min_\theta - \ln L(\theta)$$

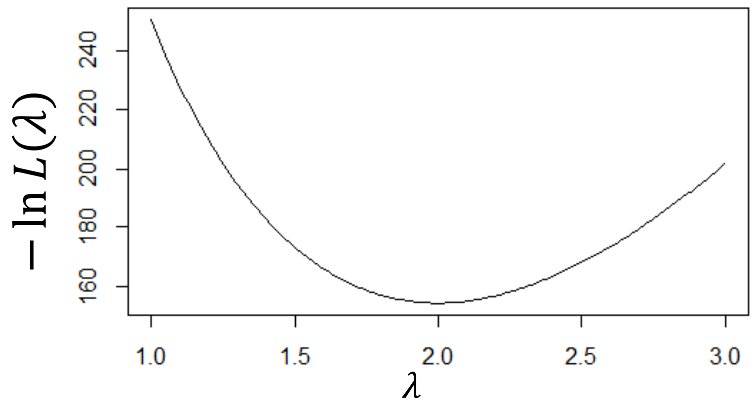



Example: Exponential $p(x;\lambda) = \lambda e^{-\lambda x}$

$$-\ln L(\lambda) = \sum_{i=1}^n -\ln\lambda + \lambda x_i$$
$$= -n\ln\lambda + \lambda\sum_i x_i$$

Finding Minimum:

$$0 = \frac{\partial(-\ln L(\lambda))}{\partial\lambda} = \frac{-n}{\lambda} + \sum_i x_i$$
$$\rightarrow \lambda^*(\{x_i\}) = \frac{n}{\sum_i x\_i}$$



# Bayes Rule



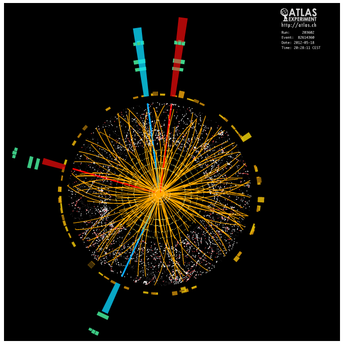
- Given two r.v. with join density $p(x, y)$
- Marginal distribution: $p(x) = \int_{-\infty}^{\infty} p(x, y)dy$
- Conditional distribution: $p(x|y) = \frac{p(x,y)}{p(y)}$
- Bayes Rule: $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$
  - $p(y)$ is the "prior" in that is doesn't account for $x$
  - $p(x|y)$ is the "likelihood" of observing $x$ given knowledge of $y$
  - $p(x)$ acts as the normalizing constant
  - $p(y|x)$ is often denoted the "posterior" because it is derived from knowledge of $x$

Slide Credit: K. Cranmer: Intro to Stats.

See Backup for visual proof of Bayes Theorem

# Supervised Learning: How does it work?

# Supervised Learning: How does it work?



h(**x**; **w**)
Function with adjustable parameters

Loss Function

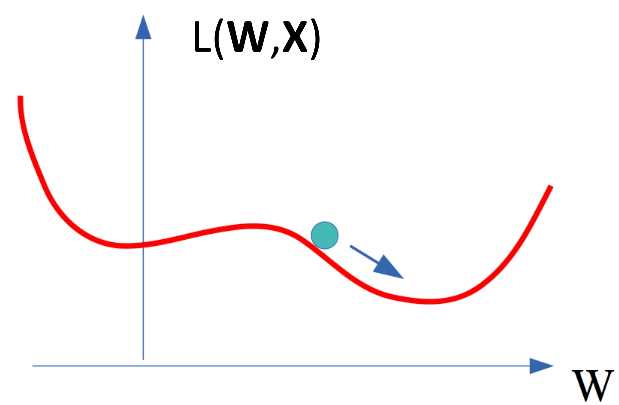
Compare prediction with true label

→ Loss

<u>True labels:</u>
Higgs = 1
Bkg = 0

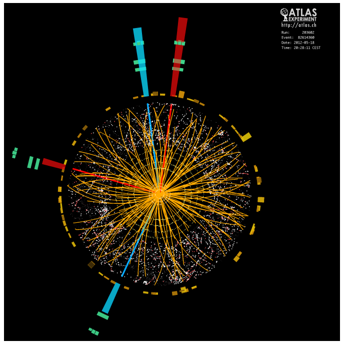
- Design function with adjustable parameters
- Design a Loss function
- Find best parameters which minimize loss

Y. Le Cun



# Supervised Learning: How does it work?






h(**x**; **w**)
Function with adjustable parameters

Loss Function

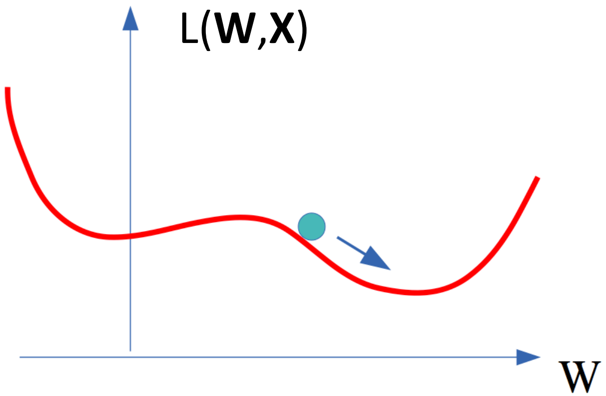
Compare prediction with true label

→ Loss

<u>True labels:</u>
Higgs = 1
Bkg = 0

- Design function with adjustable parameters
- Design a Loss function
- Find best parameters which minimize loss
  - Use a labeled *training-set* to compute loss
  - Adjust parameters to reduce loss function
  - Repeat until parameters stabilize

Y. Le Cun



# Empirical Risk Minimization

$$\min_{w} \frac{1}{N} \sum_{i}^{N} L(h(x_i;\ w), y_i) + \lambda \Omega(w)$$

Empirical expected loss

Model regularization

- Find best weights $w$ to minimizes the expected loss
  - $L \equiv$ Loss to compare predictions $h(x)$ with target $y$
  - $h(x; w) \equiv$ parameterized family of functions
  - $\Omega(w) \equiv$ regularization to penalize certain values of $w$
  - $\lambda \equiv$ Hyperparameter to control penalty
  - Use empirical estimate of expected loss over data $\{x_i, y_i\}$
- Framework to design learning algorithms
- Learning is cast as an optimization problem
  - Searching over parameter space

# Example Loss Functions

- Square Error Loss:
  - Often used in regression

$$L(h(\mathbf{x};\mathbf{w}),y) = \left(h(\mathbf{x};\mathbf{w}) - y\right)^2$$

- Cross entropy:
  - With y ∈ {0,1}
  - Often used in classification

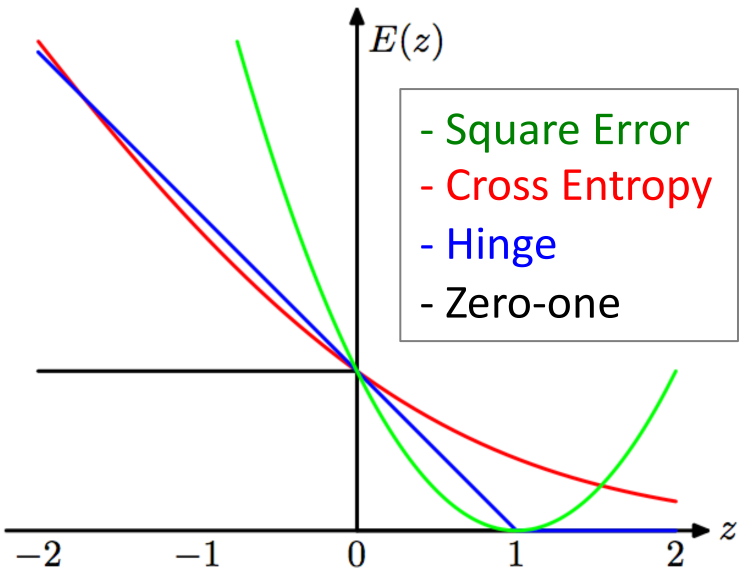
$$L(h(\mathbf{x};\mathbf{w}),y) = -\,y\log h(\mathbf{x};\mathbf{w}) - (1-y)\log(1-h(\mathbf{x};\mathbf{w}))$$

- Hinge Loss:
  - With y ∈ {-1,1}

$$L(h(\mathbf{x};\mathbf{w}),y) = \max(0, 1 - yh(\mathbf{x};\mathbf{w}))$$

- Zero-One loss
  - With h(**x**; **w**) predicting label

$$L(h(\mathbf{x};\mathbf{w}),y) = 1_{y \neq h(\mathbf{x};\mathbf{w})}$$



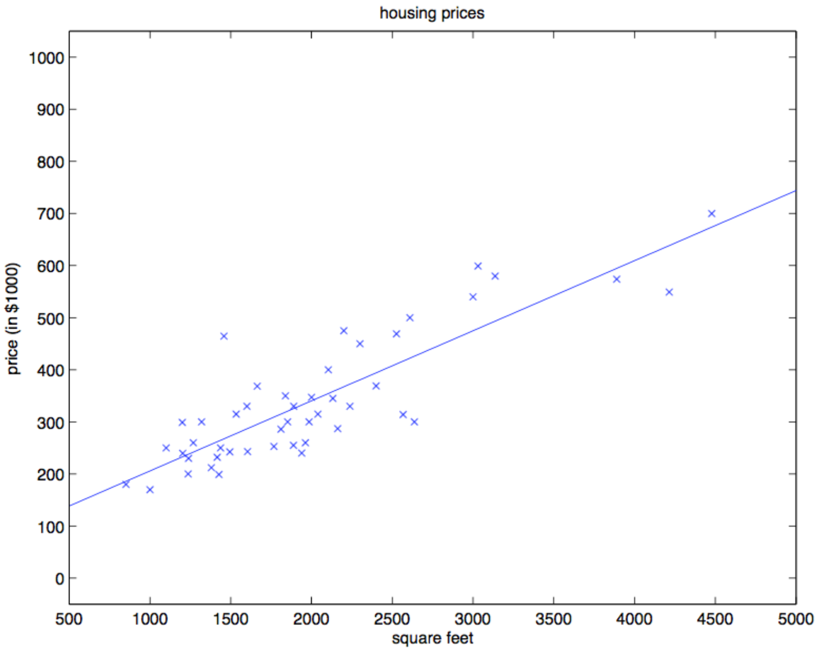

[Bishop]

# Least Squares Linear Regression

# Least Squares Linear Regression



- Set of input / output pairs $D = \{\mathbf{x}_i, y_i\}_{i=1\ldots n}$
  - $\mathbf{x}_i \in \mathbb{R}^m$
  - $y_i \in \mathbb{R}$




- Assume a linear model

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}$$

- Squared Loss function:

$$L(\mathbf{w}) = \frac{1}{2}\sum_i \left(y_i - h(\mathbf{x}_i; \mathbf{w})\right)^2$$

- Find $\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\mathbf{w})$

# Least Squares Linear Regression: Matrix Form



- Set of input / output pairs D = $\{\mathbf{x}_i , y_i\}_{i=1\ldots n}$
  - Design matrix $\mathbf{X} \in \mathbb{R}^{n\text{x}m}$
  - Target vector $\mathbf{y} \in \mathbb{R}^n$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,m} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

# Least Squares Linear Regression: Matrix Form

- Set of input / output pairs $D = \{\mathbf{x}_i , y_i\}_{i=1\ldots n}$
  - Design matrix $\mathbf{X} \in \mathbb{R}^{n\mathrm{x}m}$
  - Target vector $\mathbf{y} \in \mathbb{R}^{n}$

- Rewrite loss: $$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- Minimize w.r.t. **w**: $$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \arg\min_{\mathbf{w}} L(\mathbf{w})$$

# Linear Regression – Probabilistic Interpretation

- Assume $y_i = mx_i + e_i$
- Random error: $e_i \sim \mathcal{N}(0, \sigma) \ \rightarrow \ p(e_i) \propto \exp\left(\frac{1}{2}\frac{e_i^2}{\sigma^2}\right)$
  - Noisy measurements, unmeasured variables, …

# Linear Regression – Probabilistic Interpretation

- Assume $y_i = mx_i + e_i$
- Random error: $e_i \sim \mathcal{N}(0, \sigma) \;\rightarrow\; p(e_i) \propto \exp\left(\frac{1}{2}\frac{e_i^2}{\sigma^2}\right)$
  - Noisy measurements, unmeasured variables, …
- Then $y_i \sim \mathcal{N}(mx_i, \sigma) \;\rightarrow\; p(y_i | x_i; m) \propto \exp\left(\frac{1}{2}\frac{(y_i - mx_i)^2}{\sigma^2}\right)$

# Linear Regression – Probabilistic Interpretation

- Assume $y_i = mx_i + e_i$

- Random error: $e_i \sim \mathcal{N}(0, \sigma) \;\rightarrow\; p(e_i) \propto \exp\left(\frac{1}{2}\frac{e_i^2}{\sigma^2}\right)$
  - Noisy measurements, unmeasured variables, …

- Then $y_i \sim \mathcal{N}(mx_i, \sigma) \;\rightarrow\; p(y_i|x_i;m) \propto \exp\left(\frac{1}{2}\frac{(y_i - mx_i)^2}{\sigma^2}\right)$

- Likelihood function:

$$L(m) = p(\mathbf{y}|\mathbf{X};m) = \prod_i p(y_i|x_i;m)$$

$$\rightarrow -\log L(m) \sim \sum_i (y_i - mx_i)^2$$

# Linear Regression – Probabilistic Interpretation

- Assume $y_i = mx_i + e_i$
- Random error: $e_i \sim \mathcal{N}(0, \sigma) \;\rightarrow\; p(e_i) \propto \exp\left(\frac{1}{2}\frac{e_i^2}{\sigma^2}\right)$
  - Noisy measurements, unmeasured variables, …
- Then $y_i \sim \mathcal{N}(mx_i, \sigma) \;\rightarrow\; p(y_i|x_i;m) \propto \exp\left(\frac{1}{2}\frac{(y_i - mx_i)^2}{\sigma^2}\right)$
- Likelihood function:

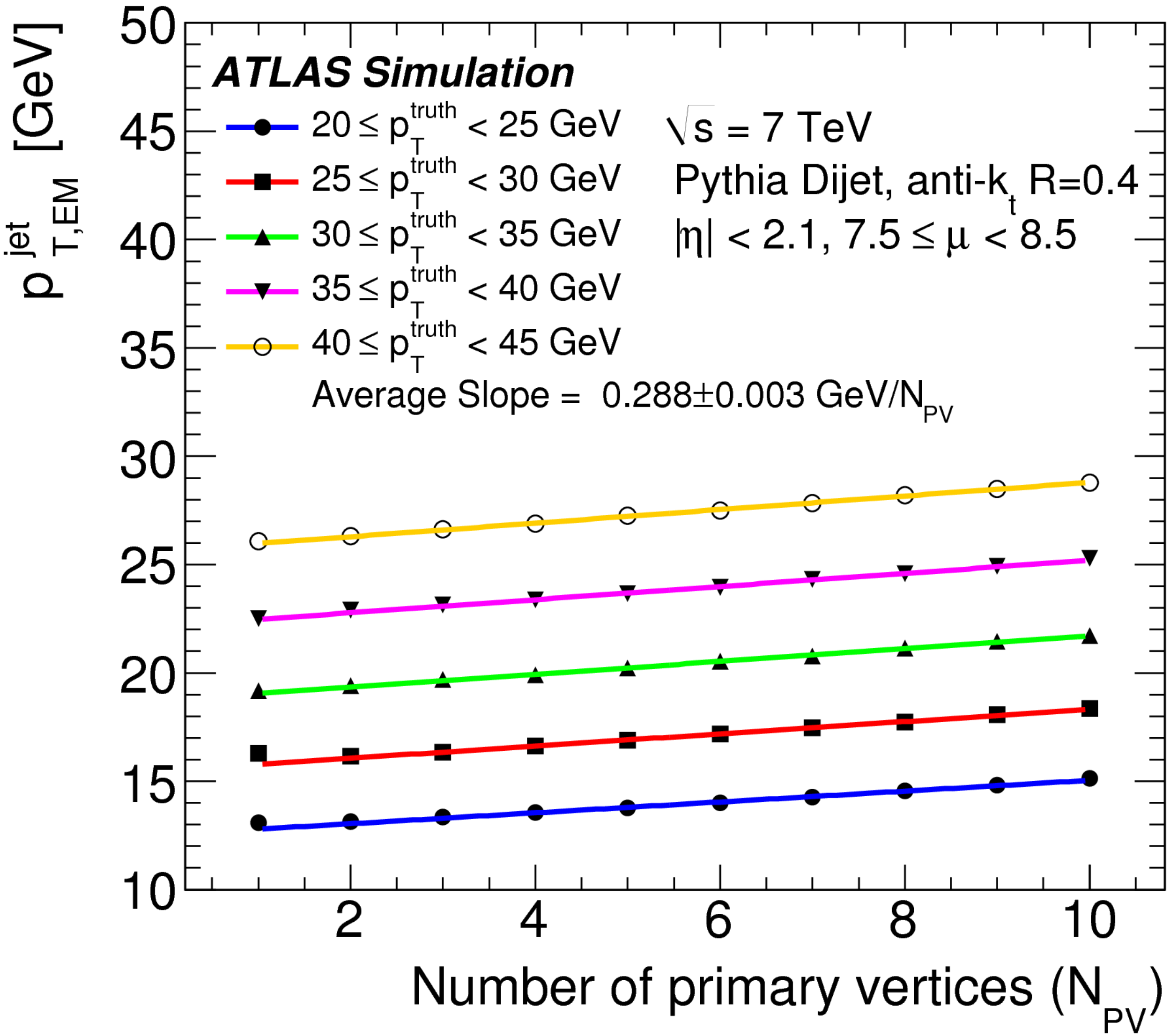
$$L(m) = p(\mathbf{y}|\mathbf{X};m) = \prod_i p(y_i|x_i;m)$$

$$\rightarrow -\log L(m) \sim \sum_i (y_i - mx_i)^2$$

Squared loss function!

# Linear Regression Example



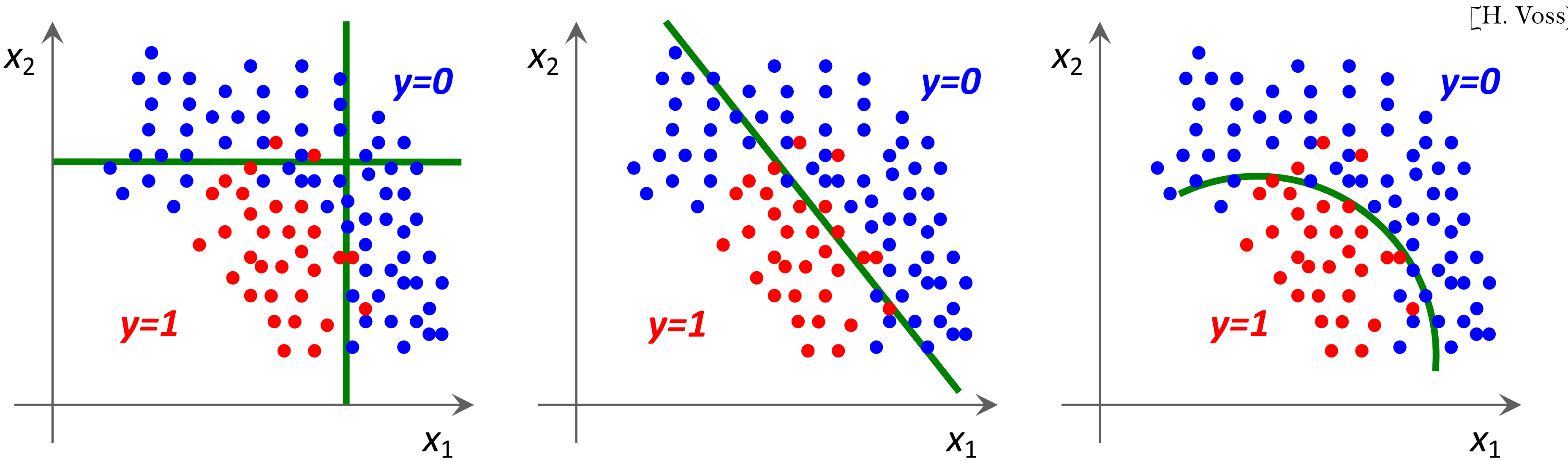



Eur. Phys. J. C (2015) 75:17

- Reconstructed Jet energy vs. Number of primary vertices

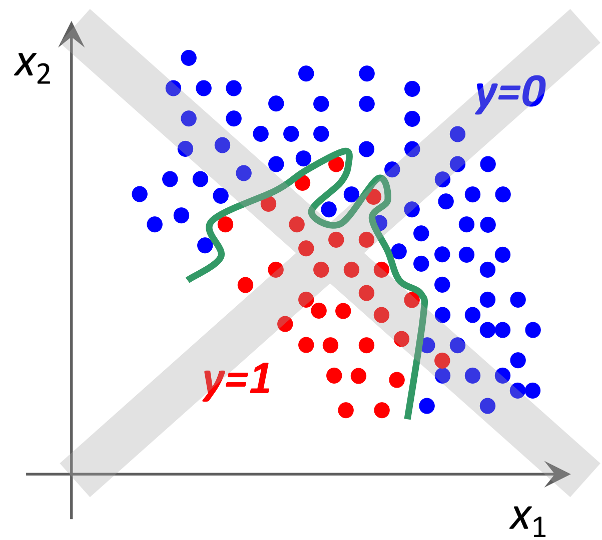
# Linear Classification

# Classification



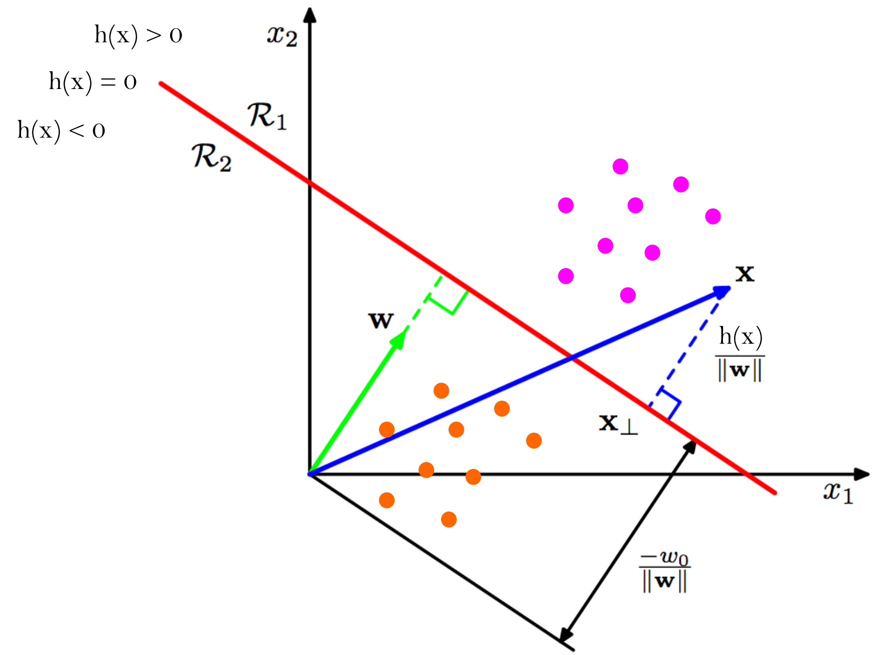

Rectangular cuts Linear discriminant Nonlinear discriminant

- Learn a function to separate different classes of data
- Avoid over-fitting:
  - Learning too fined details about your training sample that will not generalize to unseen data



# Linear Decision Boundaries

- Separate two classes:
  - $\mathbf{x}_i \in \mathbb{R}^m$
  - $y_i \in \{-1,1\}$




[Bishop]

- Linear discriminant model

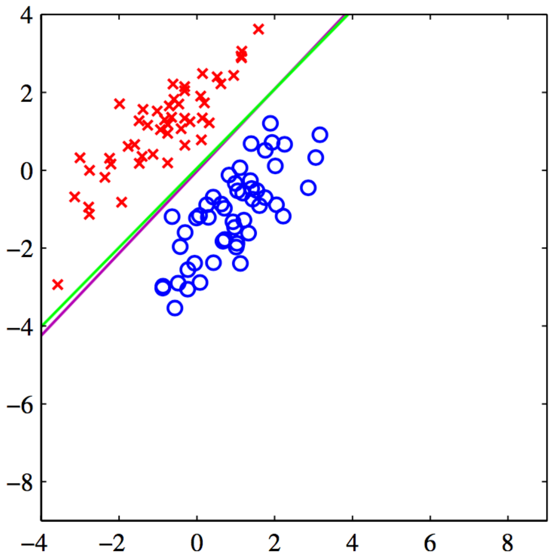
$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}+b$$

- **Decision boundary** defined by hyperplane

$$h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}+b = 0$$

- Class predictions: Predict class 0 if $h(\mathbf{x}_i ; \mathbf{w}) < 0$, else class 1

# Linear Classifier with Least Squares?

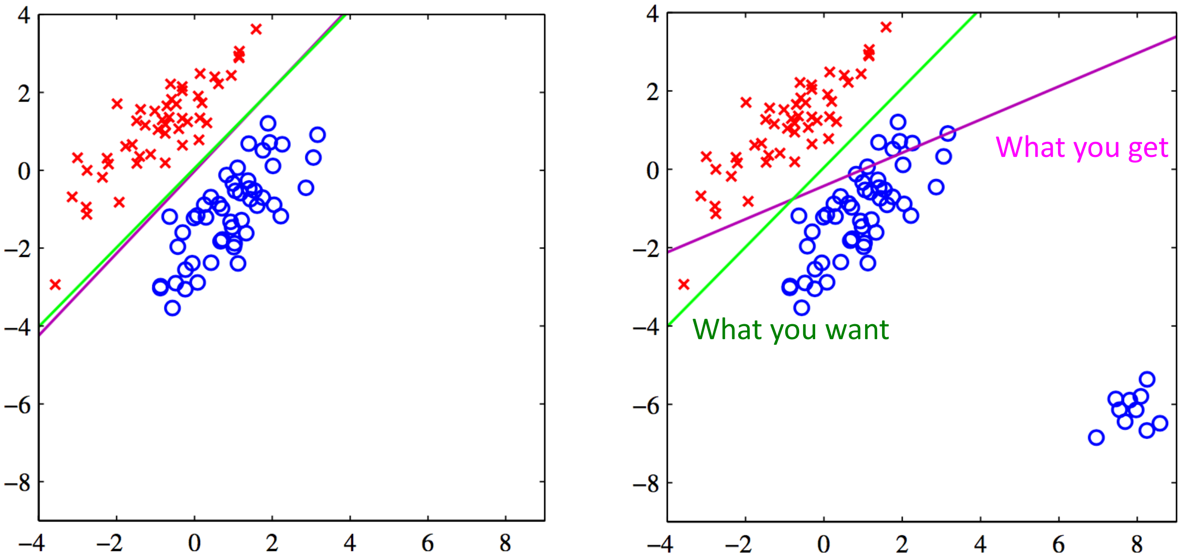



$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

[Bishop]

- Why not use least squares loss with binary targets?

# Linear Classifier with Least Squares?




$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

[Bishop]

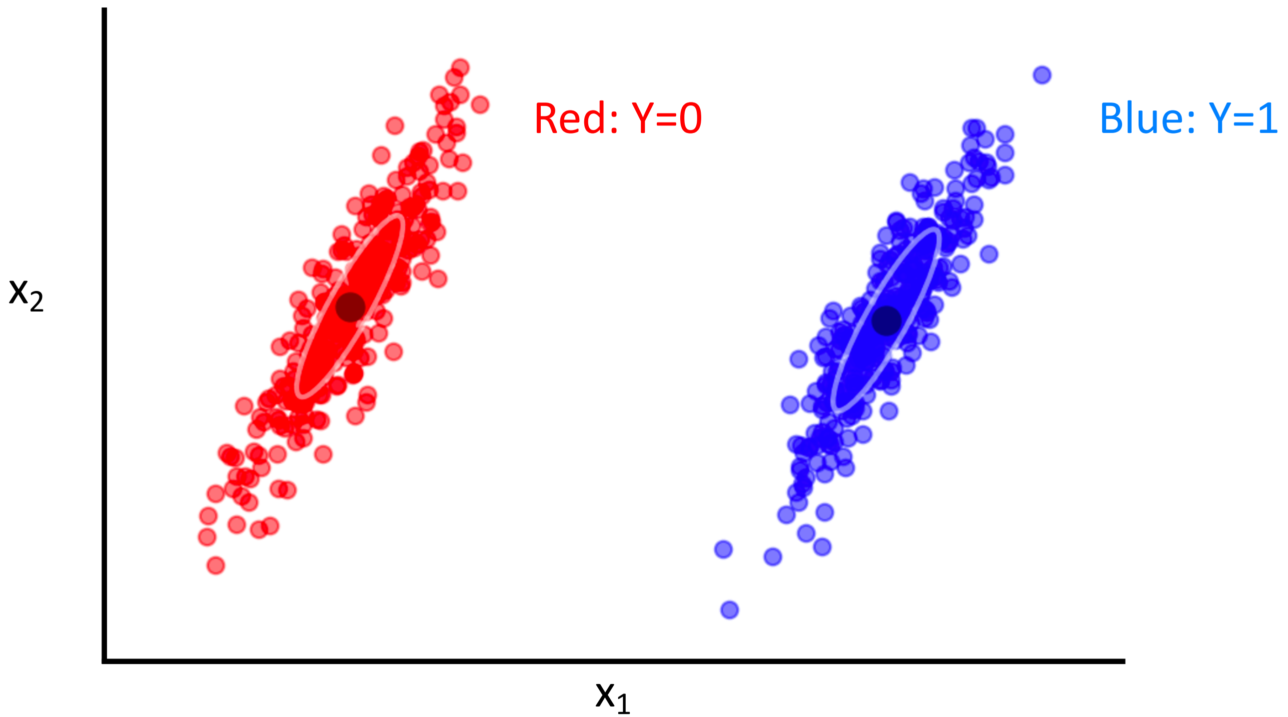
- Why not use least squares loss with binary targets?
  - Penalized even when predict class correctly
  - Least squares is very sensitive to outliers

# Linear Discriminant Analysis



- Goal: Separate data from two classes / populations

- Data from joint distribution $(\mathbf{x}, y) \sim p(\mathbf{X}, Y)$
  - Features: $\mathbf{x} \in \mathbb{R}^m$
  - Labels: $y \in \{0,1\}$



# Linear Discriminant Analysis

- Goal: Separate data from two classes / populations

- Data from joint distribution $(\mathbf{x}, y) \sim p(\mathbf{X}, Y)$
  - Features: $\mathbf{x} \in \mathbb{R}^m$
  - Labels: $y \in \{0,1\}$

- Breakdown the joint distribution:

$$p(x, y) = p(x|y)p(y)$$

**Likelihood:**
Distribution of features for a given class

**Prior:**
Probability of each class

# Linear Discriminant Analysis



- Goal: Separate data from two classes / populations

- Data from joint distribution $(\mathbf{x}, y) \sim p(\mathbf{X}, Y)$
  - Features: $\mathbf{x} \in \mathbb{R}^m$
  - Labels: $y \in \{0,1\}$

- Breakdown the joint distribution:

$$p(x, y) = p(x|y)p(y)$$

- Assume likelihoods are Gaussian

$$p(x|y) = \frac{1}{\sqrt{(2\pi)^m|\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_y)^T \Sigma^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_y)\right)$$

# Predicting the Class

- Separating classes → Predict the class $y$ of a point $\mathbf{x}$

$$p(y = 1|\mathbf{x})$$

# Predicting the Class



- Separating classes → Predict the class of a point $\mathbf{x}$

$$p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x})}$$

Bayes Rule

# Predicting the Class



- Separating classes → Predict the class of a point $\mathbf{x}$

$$p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x})}$$

Bayes Rule

$$= \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=0)p(y=0)+p(\mathbf{x}|y=1)p(y=1)}$$

Marginal definition

# Predicting the Class



- Separating classes → Predict the class of a point $\mathbf{x}$

$$p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x})}$$

Bayes Rule

$$= \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=0)p(y=0) + p(\mathbf{x}|y=1)p(y=1)}$$

Marginal definition

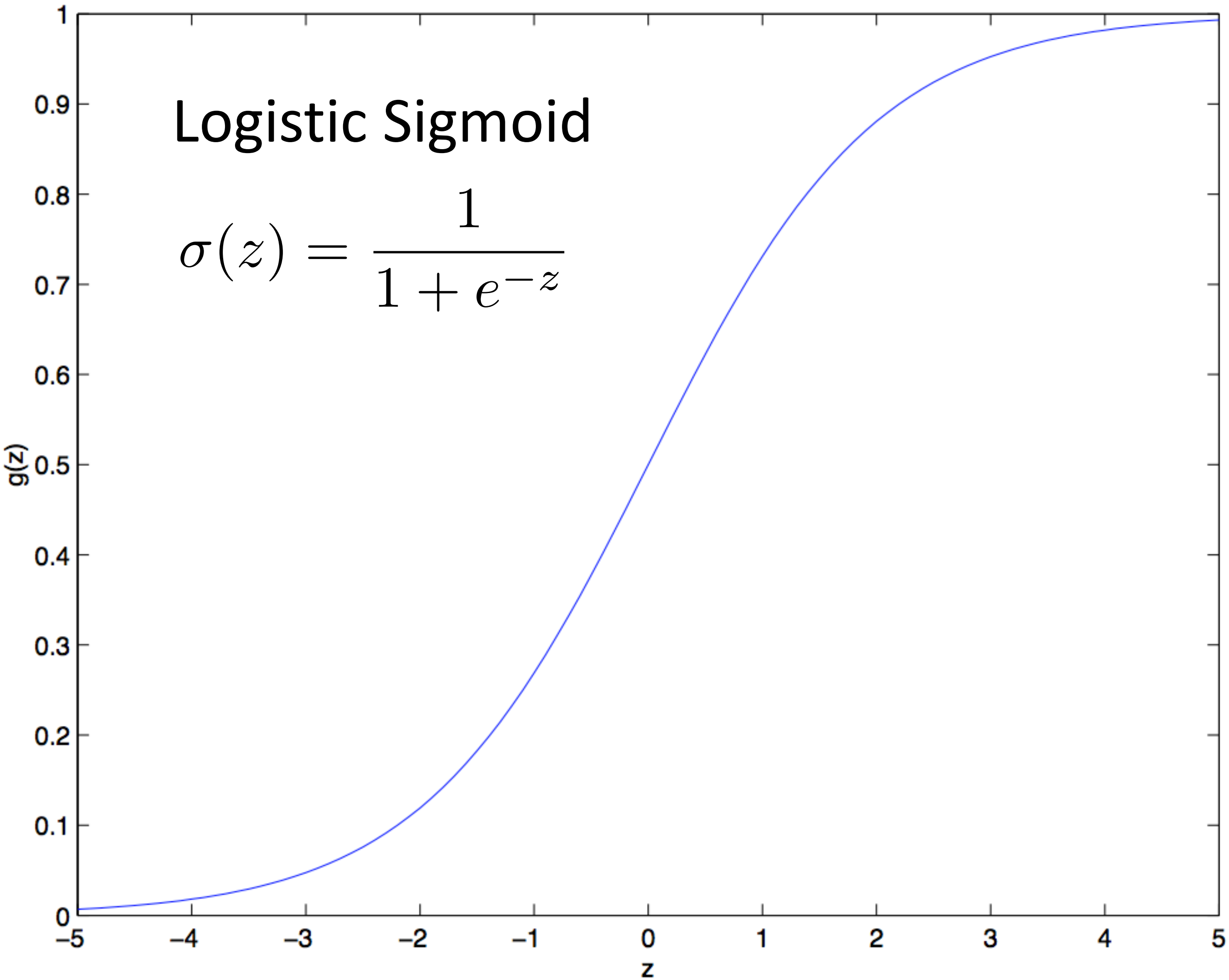
$$= \frac{1}{1 + \frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x}|y=1)p(y=1)}}$$

$$= \frac{1}{1 + \exp\left(\log \frac{p(\mathbf{x}|y=0)p(y=0)}{p(\mathbf{x}|y=1)p(y=1)}\right)}$$

Why?

# Logistic Sigmoid Function





# Predicting Classes with Gaussian Likelihoods



$$p(y=1|\mathbf{x}) = \sigma\Big( \log \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} + \log \frac{p(y=1)}{p(y=0)} \Big)$$

Log-likelihood ratio

Constant w.r.t. **x**

# Predicting Classes with Gaussian Likelihoods



$$p(y=1|\mathbf{x}) = \sigma\Big( \log \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)} + \log \frac{p(y=1)}{p(y=0)} \Big)$$

- For our Gaussian data:

$$= \sigma\Big( \log p(\mathbf{x}|y=1) - \log p(\mathbf{x}|y=0) + const. \Big)$$

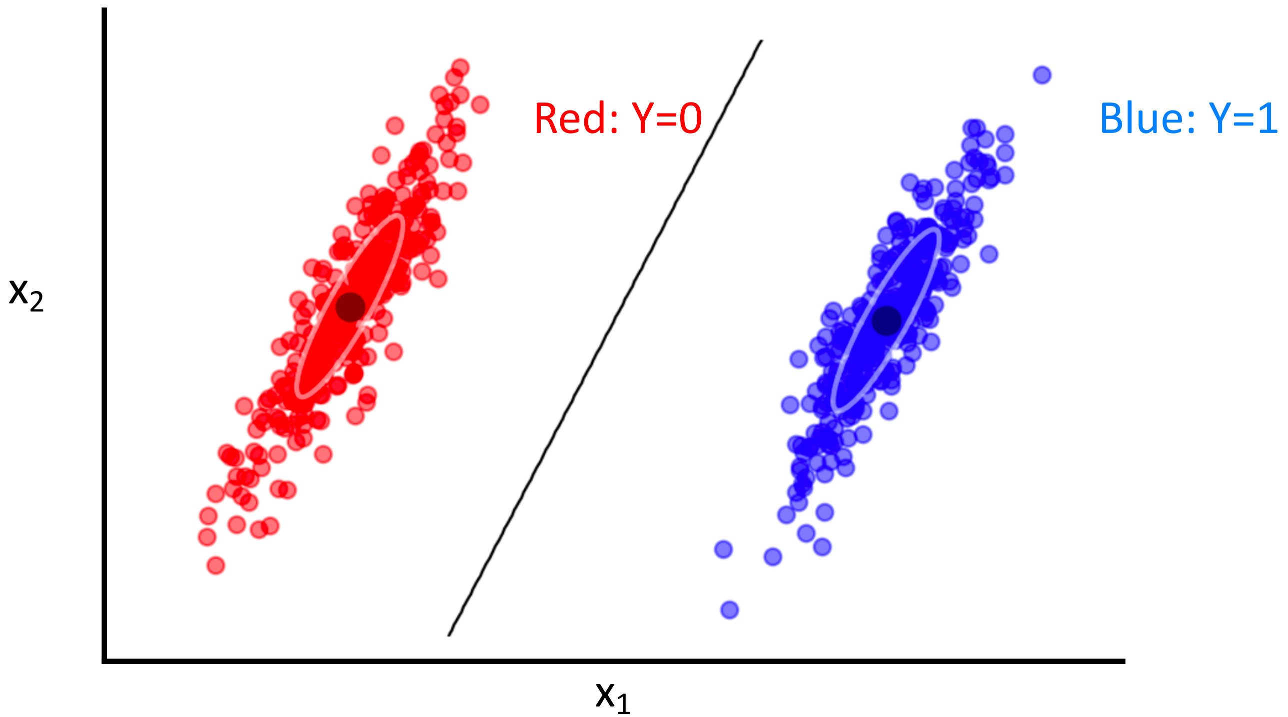
$$= \sigma\Big( -\frac{1}{2}(\mathbf{x}-\mu_1)^T \Sigma^{-1} (\mathbf{x}-\mu_1) + \frac{1}{2}(\mathbf{x}-\mu_0)^T \Sigma^{-1} (\mathbf{x}-\mu_0) + const. \Big)$$

$$= \sigma\Big( \mathbf{w}^T \mathbf{x} + b \Big)$$

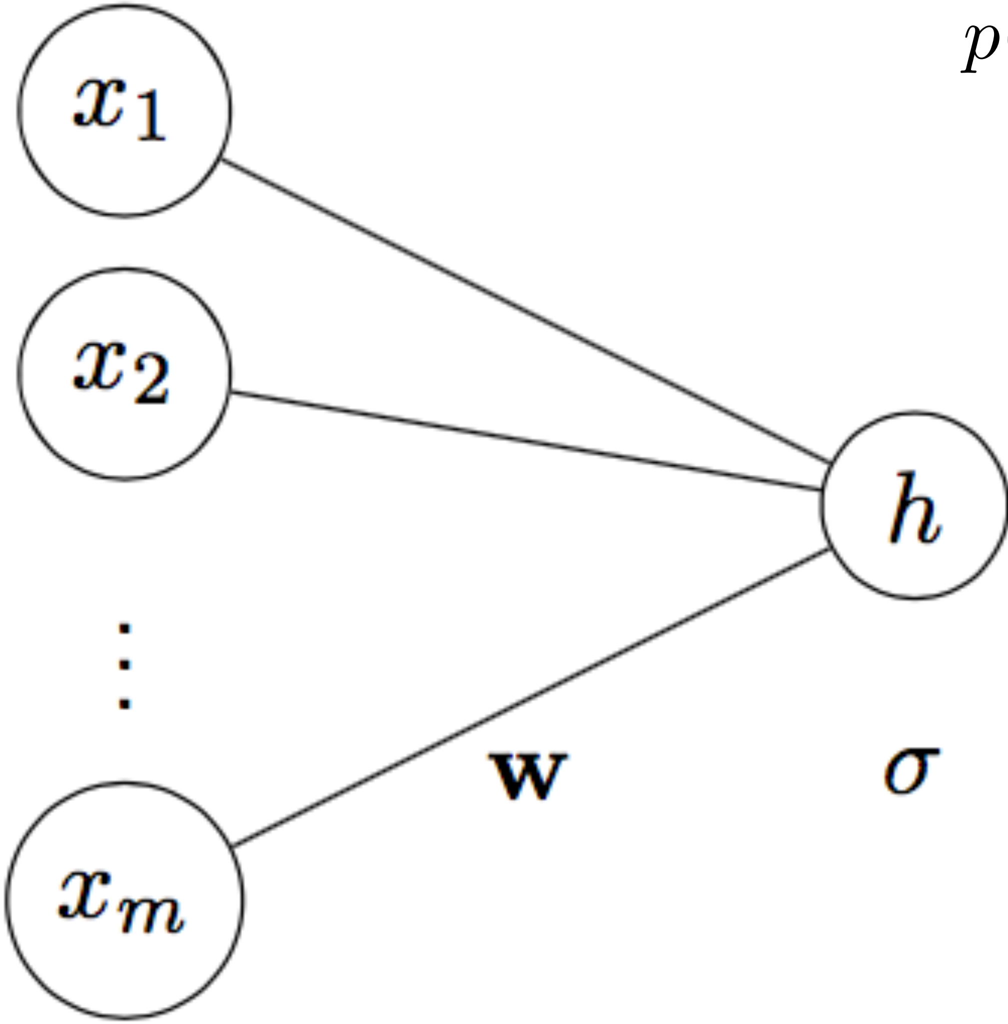
Collect terms

# What did we learn?

- For this data, the log-likelihood ratio is linear!
  - Line defines boundary to separate the classes
  - Sigmoid turns distance from boundary to probability



# Logistic Regression




$$p(y=1|\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + b\right) = \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}\text{-b}}}$$

This unit is the main building block of Neural Networks!

# Logistic Regression



- Even without Gaussian assumption on data, can still use model as classifier:

$$p(y=1|\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x}+b\right) \equiv h(\mathbf{x};\mathbf{w})$$

- How to train model? Use Maximum Likelihood
  - Define: $p_i \equiv p(y_i = y|\boldsymbol{x}_i)$

$$P(y_i = y|x_i) = \text{Bernoulli}(p_i) = (p_i)^{y_i}(1-p_i)^{1-y_i} = \begin{cases} p_i & \text{if } y_i=1 \\ 1-p_i & \text{if } y_i=0 \end{cases}$$

- **Goal**:
  - Given i.i.d. dataset of pairs $(\mathbf{x}_i, y_i)$ find **w** and b that maximize likelihood of data

# Logistic Regression

- Negative log-likelihood

$$-\ln \mathcal{L} = -\ln \prod_i (p_i)^{y_i} (1 - p_i)^{1-y_i}$$

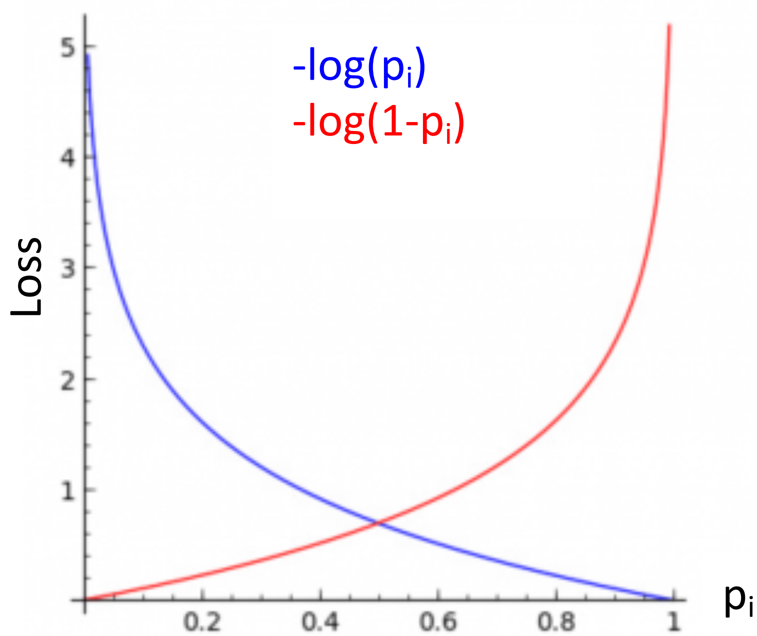
# Logistic Regression

- Negative log-likelihood

$$-\ln \mathcal{L} = -\ln \prod_i (p_i)^{y_i} (1-p_i)^{1-y_i}$$

$$= -\sum_i y_i \ln(p_i) + (1-y_i) \ln(1-p_i)$$

binary cross entropy loss function!



# Logistic Regression



- Negative log-likelihood

$$-\ln \mathcal{L} = -\ln \prod_i (p_i)^{y_i} (1-p_i)^{1-y_i}$$

$$= -\sum_i y_i \ln(p_i) + (1-y_i) \ln(1-p_i)$$

binary cross entropy loss function!

$$= \sum_i y_i \ln(1+e^{-\mathbf{w}^T\mathbf{x}}) + (1-y_i) \ln(1+e^{\mathbf{w}^T\mathbf{x}})$$

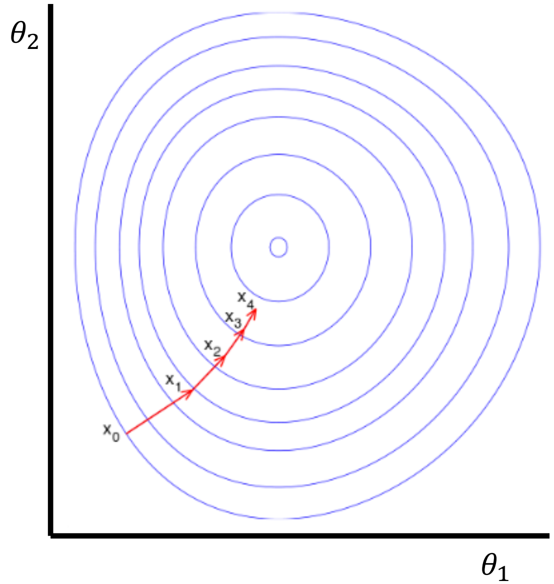
- No closed form solution to $w^* = \arg\min_w - \ln \mathcal{L}(w)$

- How to solve for $\mathbf{w}$?

# Gradient Descent



- Minimize loss by repeated gradient steps
  - Compute gradient w.r.t. current parameters: $\nabla_{\theta_i}\mathcal{L}(\theta_i)$
  - Update parameters: $\theta_{i+1} \leftarrow \theta_i - \eta\nabla_{\theta_i}\mathcal{L}(\theta_i)$
  - η is the *learning rate*, controls how big of a step to take

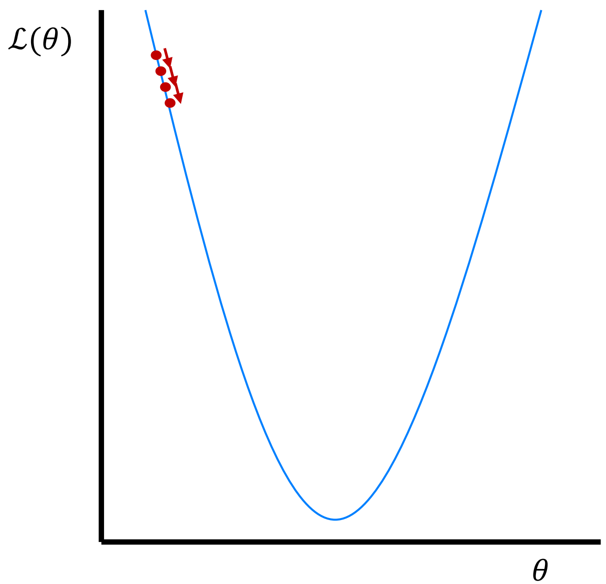


# Step Sizes

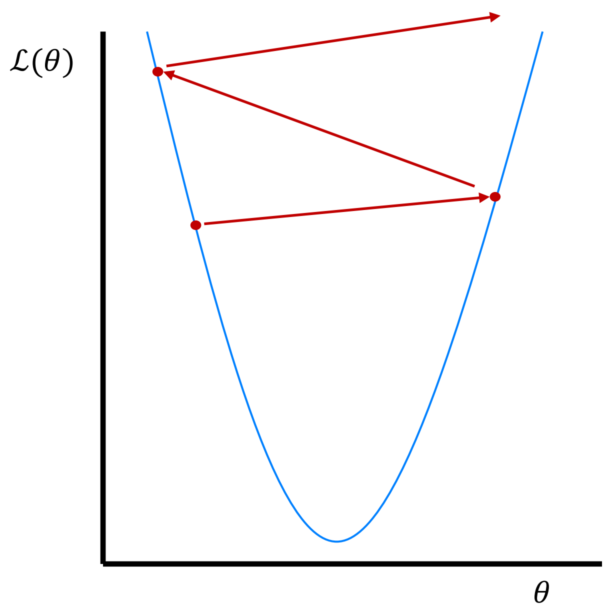
- Too small a learning rate, convergence very slow

- Too large a learning rate, algorithm diverges

Small Learning rate




Large Learning rate



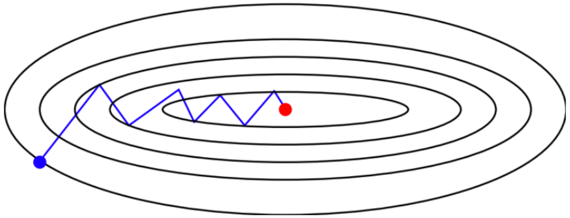
# Stochastic Gradient Descent

- Loss is composed of a sum over samples:

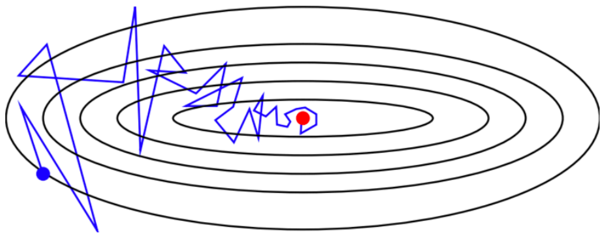
$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \mathcal{L}\big(y_i, h(x_i;\theta)\big)$$

  – Computing gradient grows linearly with N!

- **(Mini-Batch) Stochastic Gradient Descent**
  – Compute gradient update using 1 random sample (small size batch)
  – Gradient is unbiased → on average it moves in correct direction
  – Tends to be much faster the full gradient descent
  – Several updates to SGD, like momentum, ADAM, RMSprop



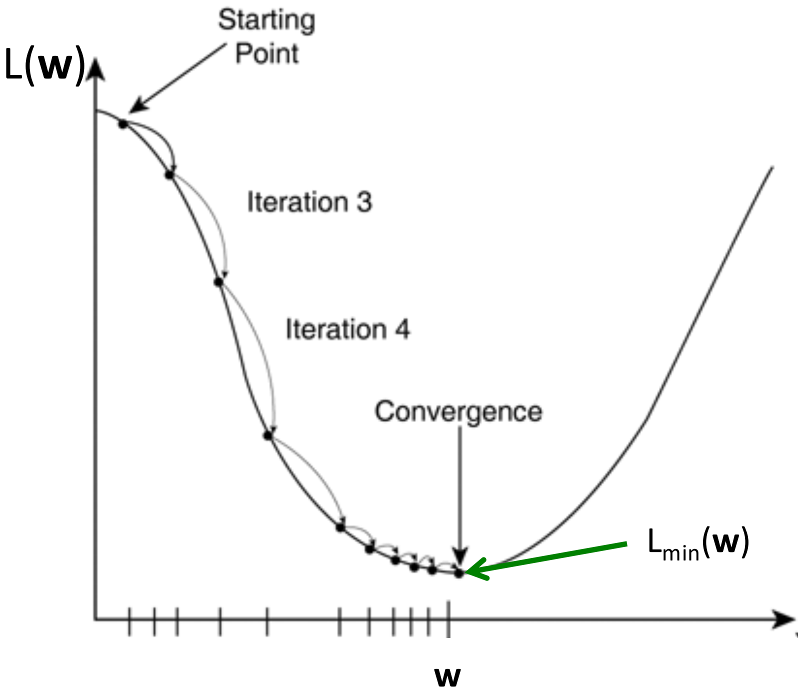
*Batch gradient descent*

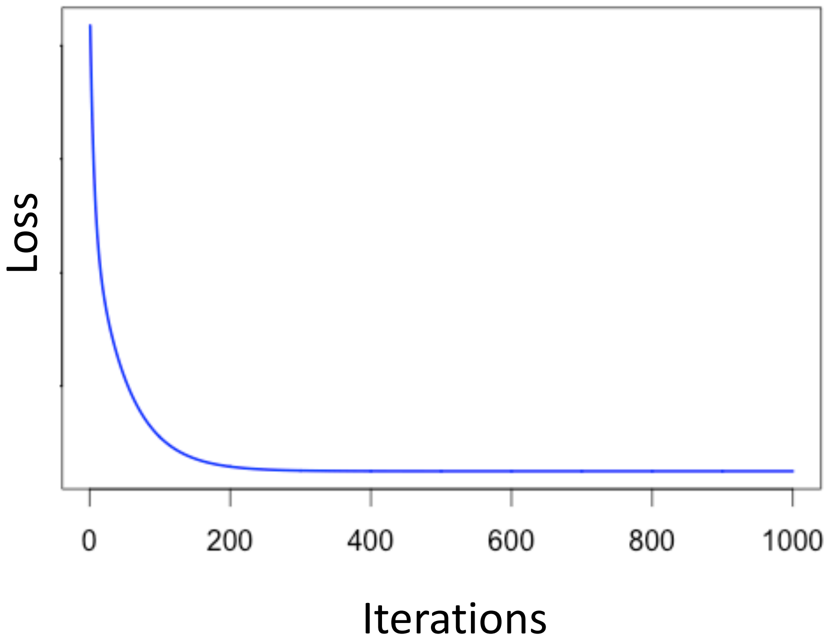


*Stochastic gradient descent*

# Gradient Descent






- Logistic Regression Loss is convex
  - Single global minimum
- Iterations lower loss and move toward minimum

# Logistic Regression Example

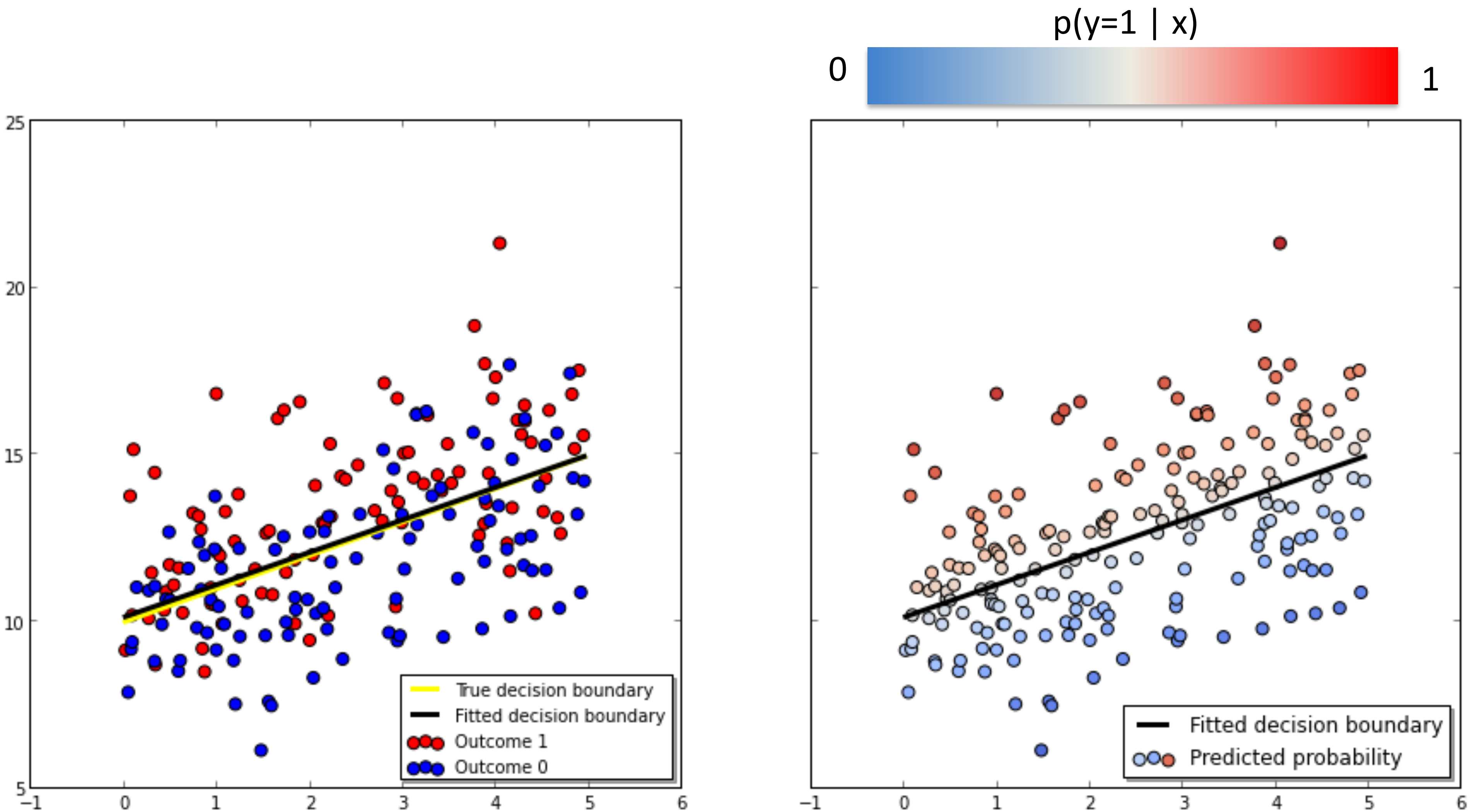
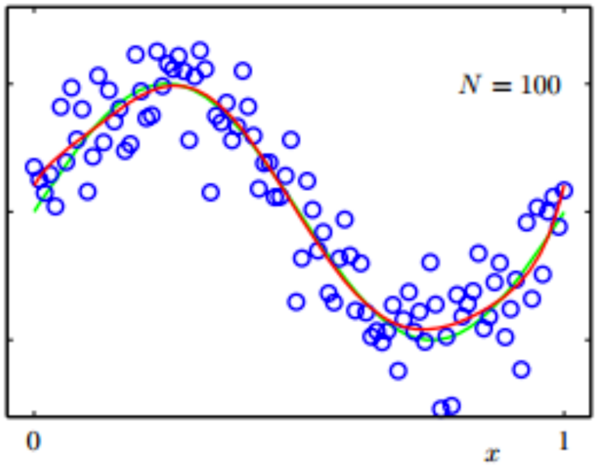


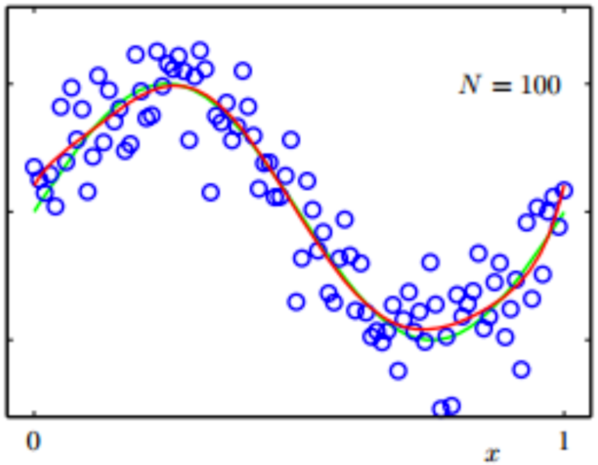

Image source

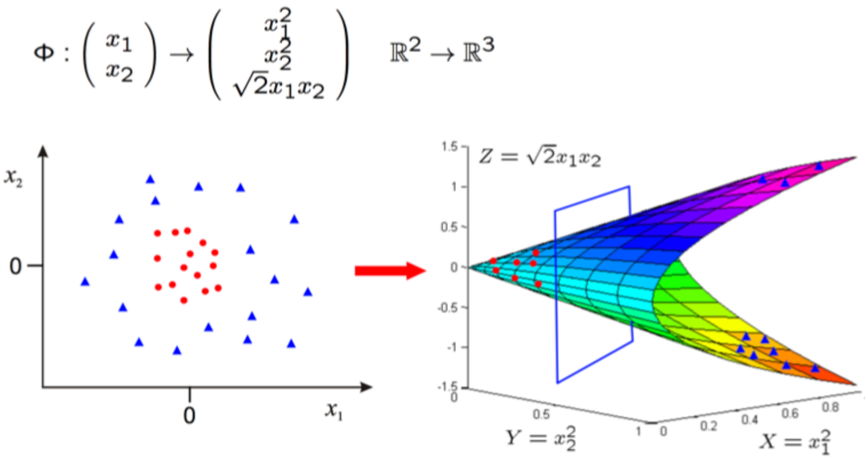
# Basis Functions




- What if non-linear relationship between **y** and **x**?

# Basis Functions



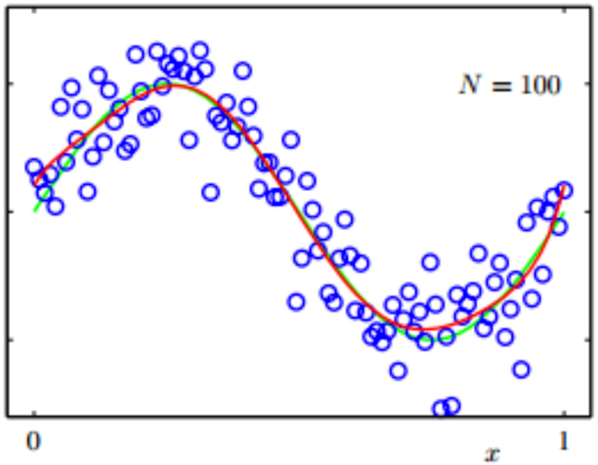



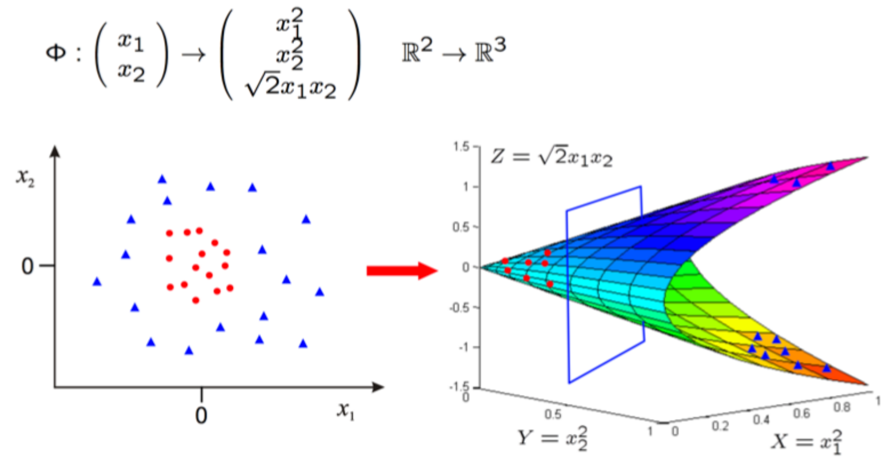

- What if non-linear relationship between **y** and **x**?
- Can choose basis functions ϕ(x) to form new features

$$h(x;w) = \sigma\left(w^T \phi(x)\right)$$

  - Polynomial basis ϕ(x) ~ $\{1, x, x^2, x^3, \ldots\}$, Gaussian basis, …
  - **Logistic regression on new features ϕ(x)**

# Basis Functions



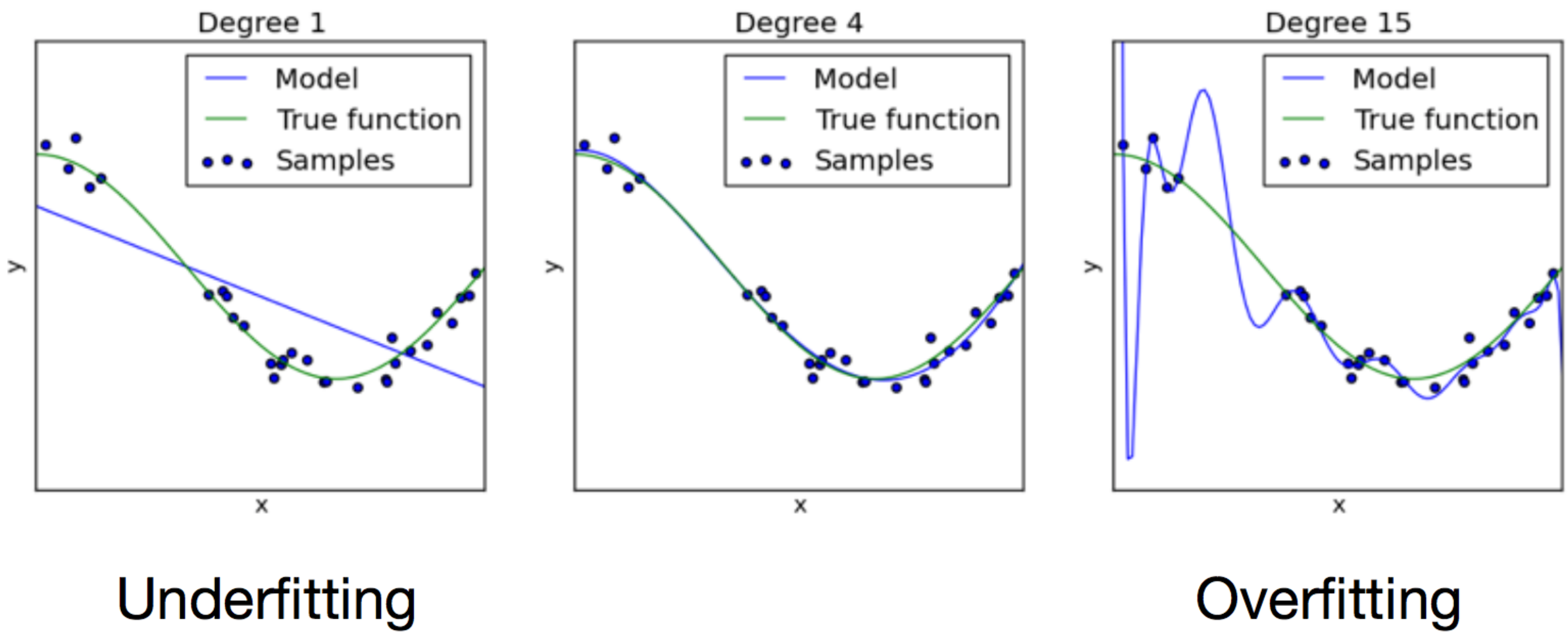

- What if  non-linear relationship between **y** and **x**?
- Can choose basis functions ϕ(x) to form new features

$$h(x; w) = \sigma\left(w^T \phi(x)\right)$$

  - Polynomial basis ϕ(x) ~ $\{1, x, x^2, x^3, \ldots\}$, Gaussian basis, …
  - Logistic regression on new features ϕ(x)
- What basis functions to choose? *Overfit* with too much flexibility?

# What is Overfitting






http://scikit-learn.org/

- What models allow us to do is **generalize** from data
- Different models generalize in different ways

# Bias Variance Tradeoff



- generalization error = systematic error + sensitivity of prediction
  (bias) (variance)

# Bias Variance Tradeoff



- generalization error = systematic error + sensitivity of prediction
  (bias) (variance)

- Simple models under-fit: will deviate from data (high bias) but will not be influenced by peculiarities of data (low variance).

# Bias Variance Tradeoff



- generalization error = systematic error + sensitivity of prediction
  (bias) (variance)

- Simple models under-fit: will deviate from data (high bias) but will not be influenced by peculiarities of data (low variance).

- Complex models over-fit: will not deviate systematically from data (low bias) but will be very sensitive to data (high variance).

# Bias Variance Tradeoff

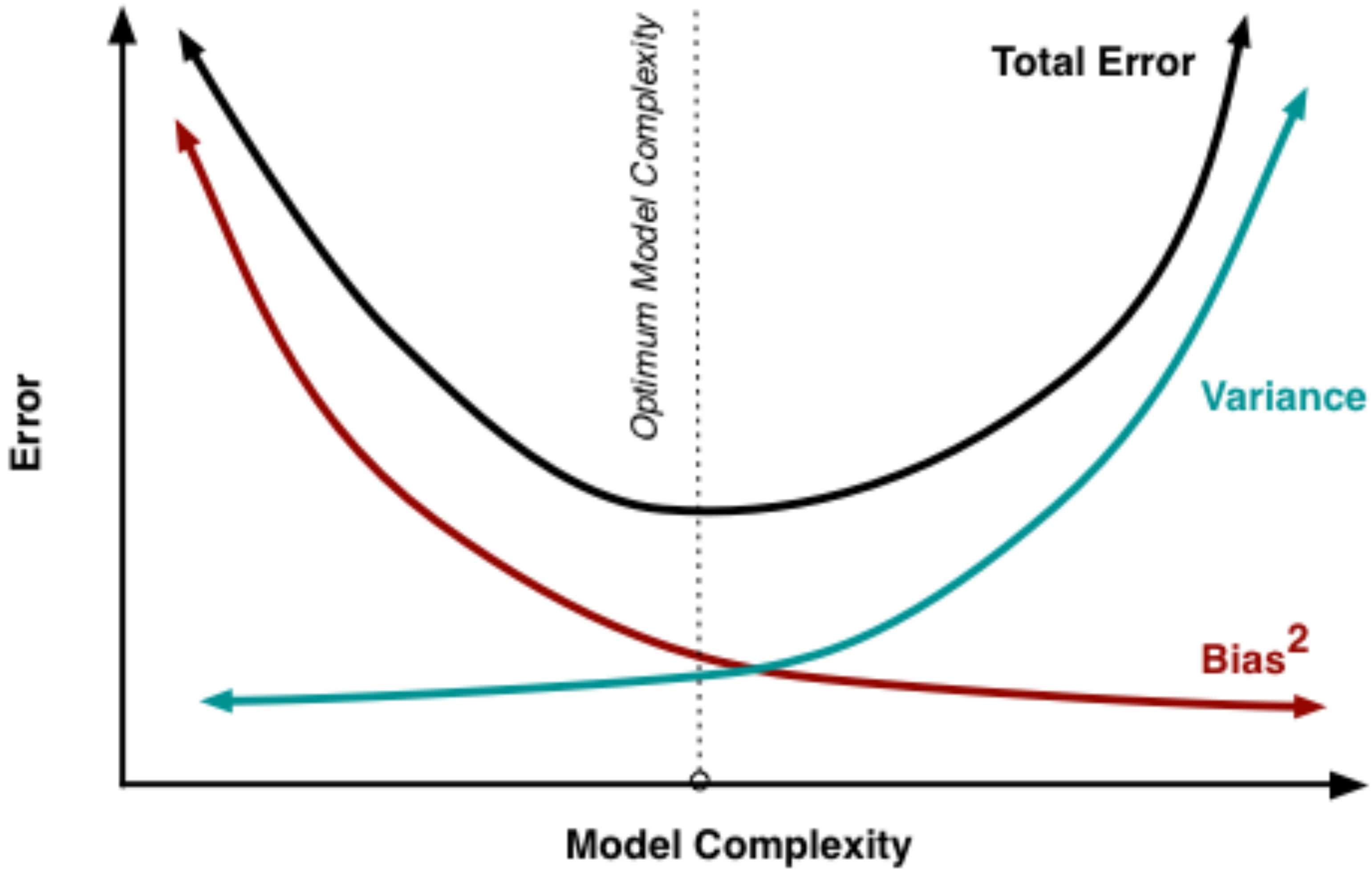


- generalization error = systematic error + sensitivity of prediction
  (bias) (variance)

- Simple models under-fit: will deviate from data (high bias) but will not be influenced by peculiarities of data (low variance).

- Complex models over-fit: will not deviate systematically from data (low bias) but will be very sensitive to data (high variance).
  - **As dataset size grows, can reduce variance! Can use more complex model**

# Bias Variance Tradeoff

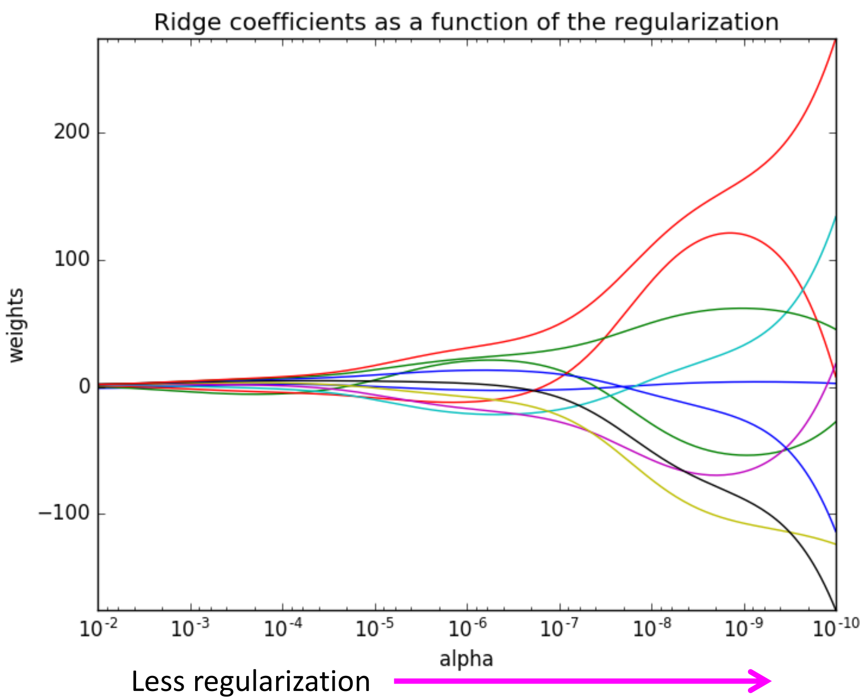


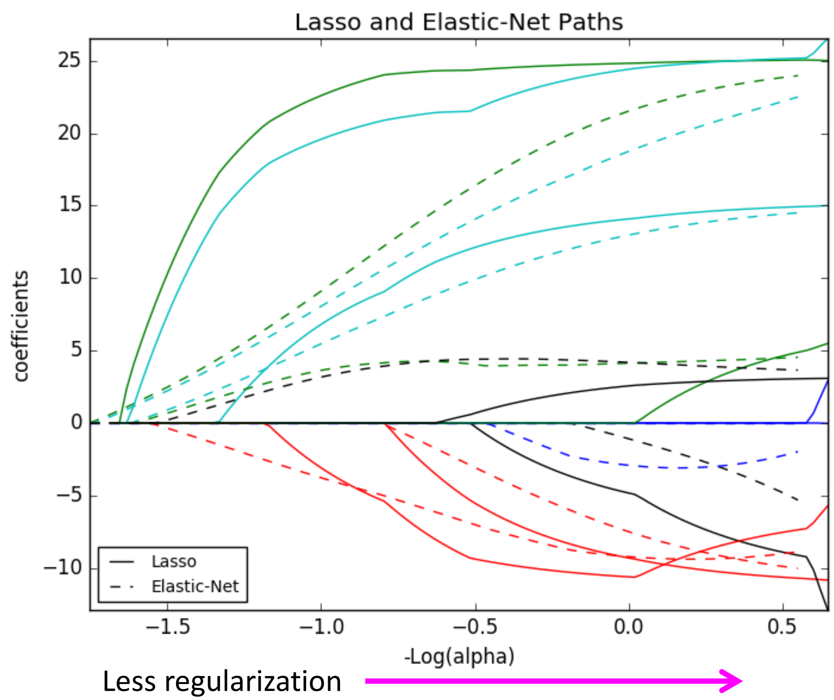
# Regularization – Control Complexity

$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^2 + \alpha\Omega(\mathbf{w})$$

$$L2: \quad \Omega(\mathbf{w}) = ||\mathbf{w}||^2$$

$$L1: \quad \Omega(\mathbf{w}) = ||\mathbf{w}||$$



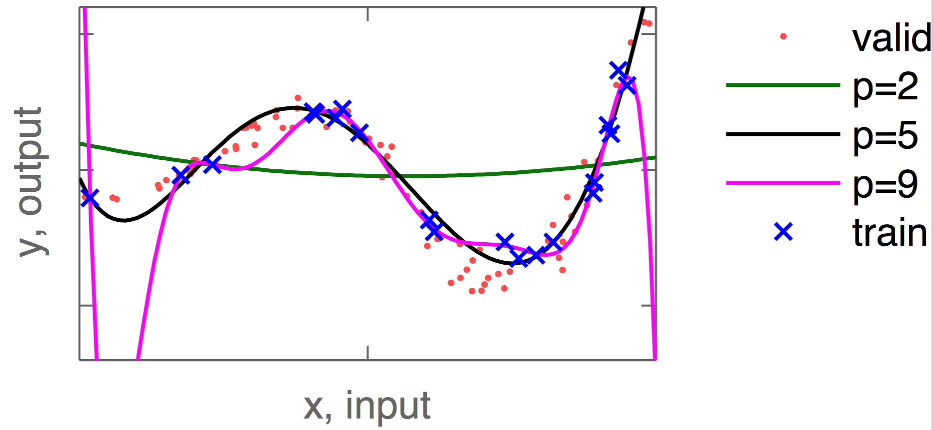

- L2 keeps weights small, L1 keeps weights sparse!
- But how to choose hyperparameter α?

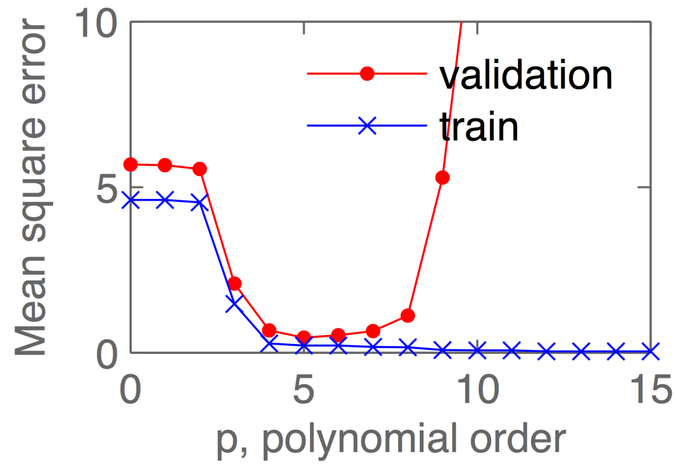
http://scikit-learn.org/

# How to Measure Generalization Error?

Training set | Validation set | Test set

- Split dataset into multiple parts
- **Training set**
  - Used to fit model parameters
- **Validation set**
  - Used to check performance on independent data and tune hyper parameters
- **Test set**
  - final evaluation of performance after all hyper-parameters fixed
  - Needed since we tune, or "peek", performance with validation set

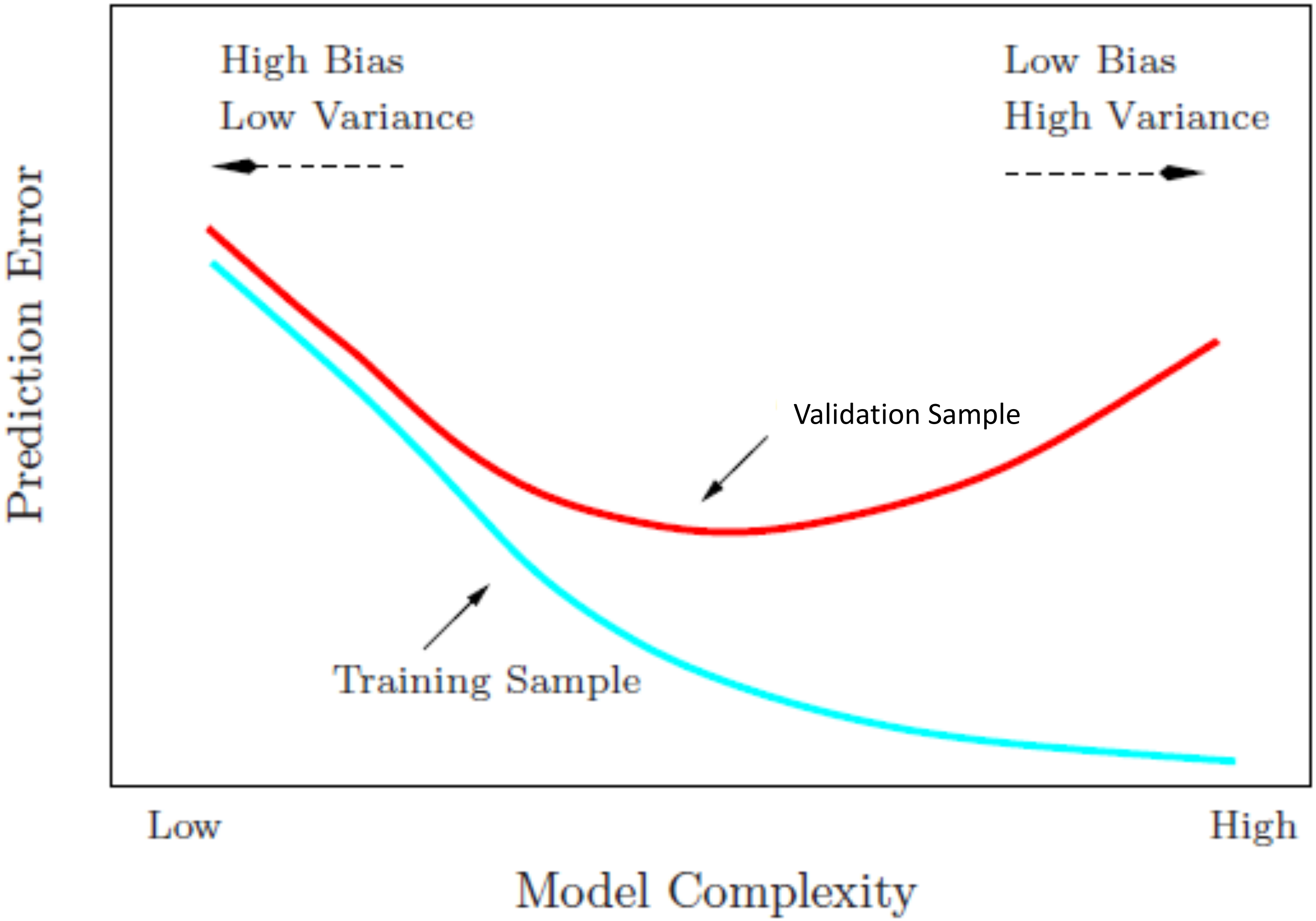



[Murray]

# How to Measure Generalization Error?





# Summary

- Machine learning uses mathematical and statistical models learned from data to characterize patterns and relations between inputs, and use this for inference / prediction

- Machine learning comes in many forms, much of which has probabilistic and statistical foundations and interpretations (i.e. *Statistical Machine Learning*)

- Machine learning provides a powerful toolkit to analyze data
  - Linear methods can help greatly in understanding data
  - Choosing a model for a given problem is difficult, keep in mind the bias-variance tradeoff when building an ML model

# Recommended Materials



- Many excellent books (many available free online)
  - Introduction to Statistical Learning
  - Elements of Statistical Learning
  - Pattern Recognition and Machine learning (Bishop)
  - …

- Many excellent courses and documentation available online
  - Andre Ng's machine learning course on Coursera
  - University course material online: Stanford CS229, Harvard CS181, …
  - Lectures from Machine Learning Summer School (MLSS)
  - Lectures from Yandex Machine learning in HEP summer schools
  - Scikit Learn documentation
  - Francois Fleuret course at University of Geneva
  - Gilles Louppe course at University of Liege
  - Yann LeCun & Alfredo Canziani course at NYU

- **References**:
  - I used / borrowed from many of these references to make these lectures!

# References


- http://scikit-learn.org/
- [Bishop] Pattern Recognition and Machine Learning, Bishop (2006)
- [ESL] Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani & Friedman 2009
- [Murray] Introduction to machine learning, Murray
  - http://videolectures.net/bootcamp2010_murray_iml/
- [Ravikumar] What is Machine Learning, Ravikumar and Stone
  - http://www.cs.utexas.edu/sites/default/files/legacy_files/research/documents/MLSS-Intro.pdf
- [Parkes] CS181, Parkes and Rush, Harvard University
  - http://cs181.fas.harvard.edu
- [Ng] CS229, Ng, Stanford University
  - http://cs229.stanford.edu/
- [Rogozhnikov] Machine learning in high energy physics, Alex Rogozhnikov
  - https://indico.cern.ch/event/497368/
- [Fleuret] Francois Fleuret, EE559 Deep Learning, EPFL, 2018
  - https://documents.epfl.ch/users/f/fl/fleuret/www/dlc/

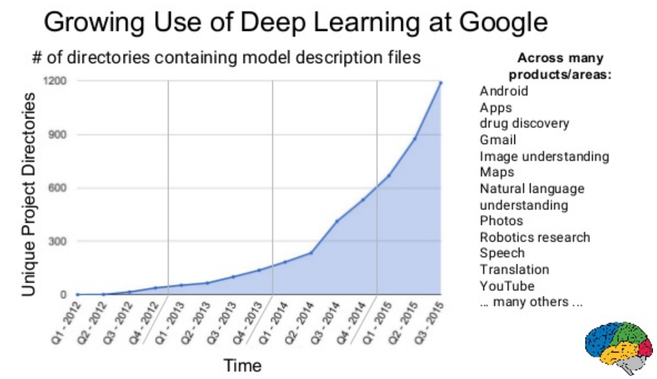
# References



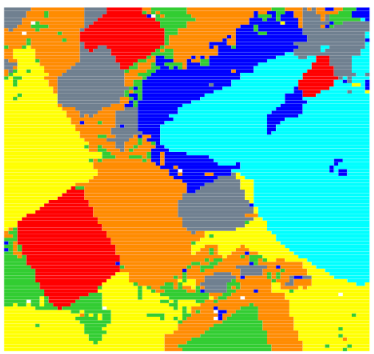

- http://scikit-learn.org/
- [Bishop] Pattern Recognition and Machine Learning, Bishop (2006)
- [ESL] Elements of Statistical Learning (2nd Ed.) Hastie, Tibshirani & Friedman 2009
- [Murray] Introduction to machine learning, Murray
  - http://videolectures.net/bootcamp2010_murray_iml/
- [Ravikumar] What is Machine Learning, Ravikumar and Stone
  - http://www.cs.utexas.edu/sites/default/files/legacy_files/research/documents/MLSS-Intro.pdf
- [Parkes] CS181, Parkes and Rush, Harvard University
  - http://cs181.fas.harvard.edu
- [Ng] CS229, Ng, Stanford University
  - http://cs229.stanford.edu/
- [Rogozhnikov] Machine learning in high energy physics, Alex Rogozhnikov
  - https://indico.cern.ch/event/497368/

# Where is ML Used, an Incomplete List

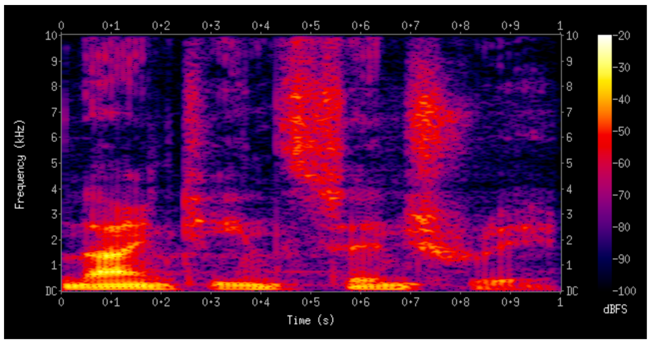
- Natural Language Processing
- Speech and handwriting recognition
- Object recognition and computer vision
- Fraud detection
- Financial market analysis
- Search engines
- Spam and virus detection
- Medical diagnosis
- Robotics control
- Automation: energy usage, systems control, video games, self-driving cars
- Advertising
- Data Science
- …

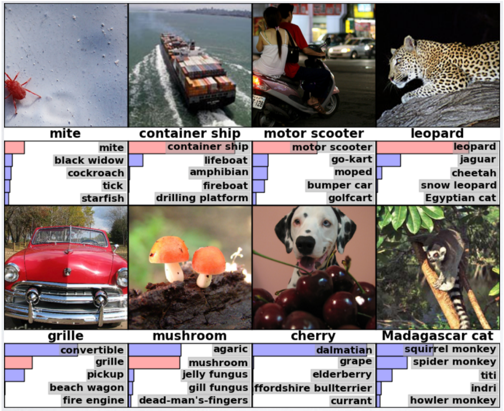


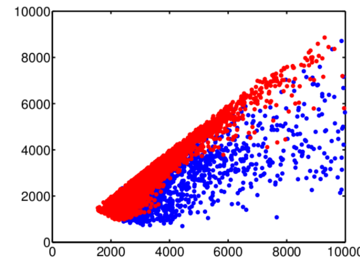

Predicted Land Usage



[ESL]



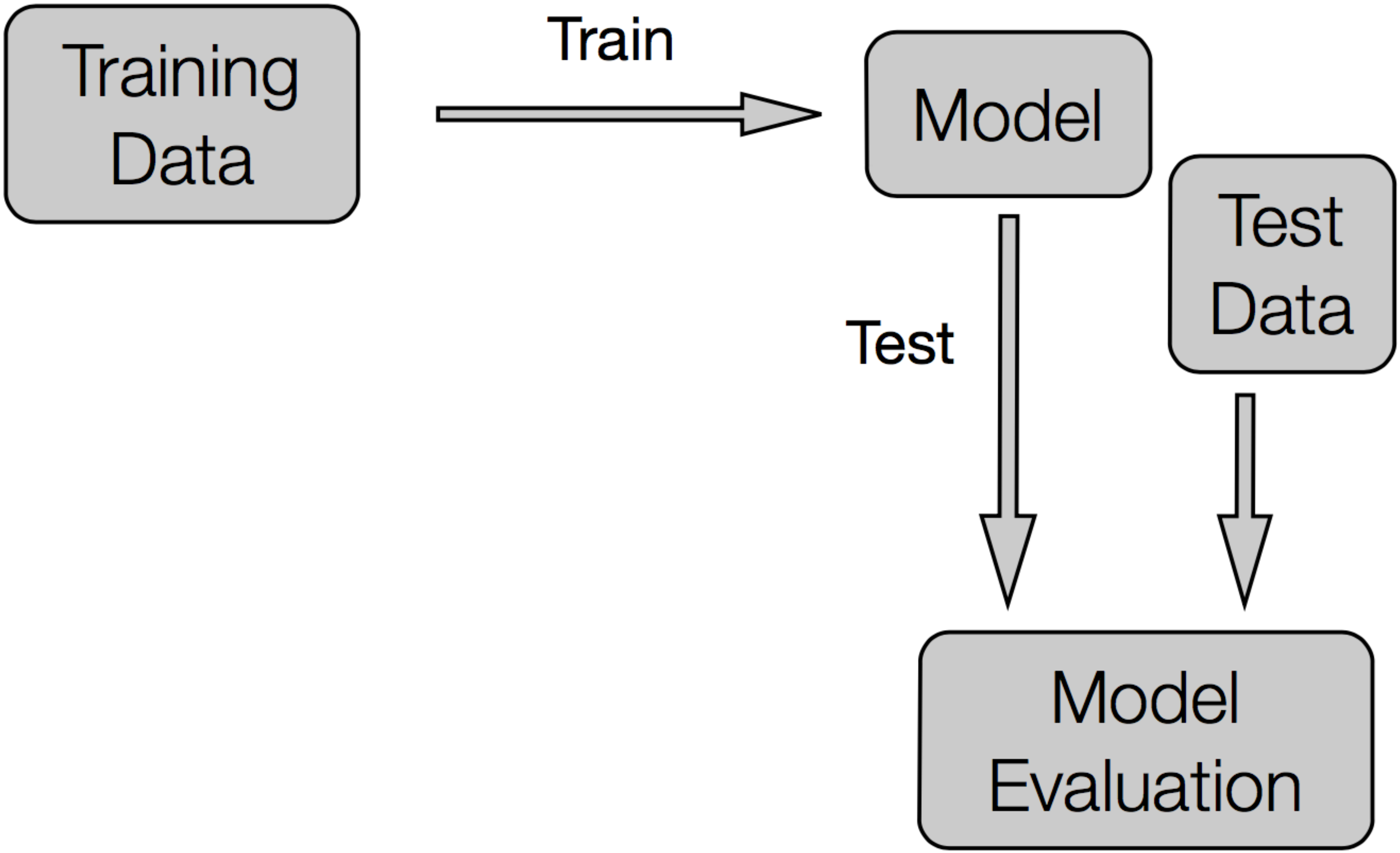



Minor elliptical axis (y) against Major elliptical axis (x) for stars (red) and galaxies (blue). (Amos Storkey) http://www-wfau.roe.ac.uk/sss/

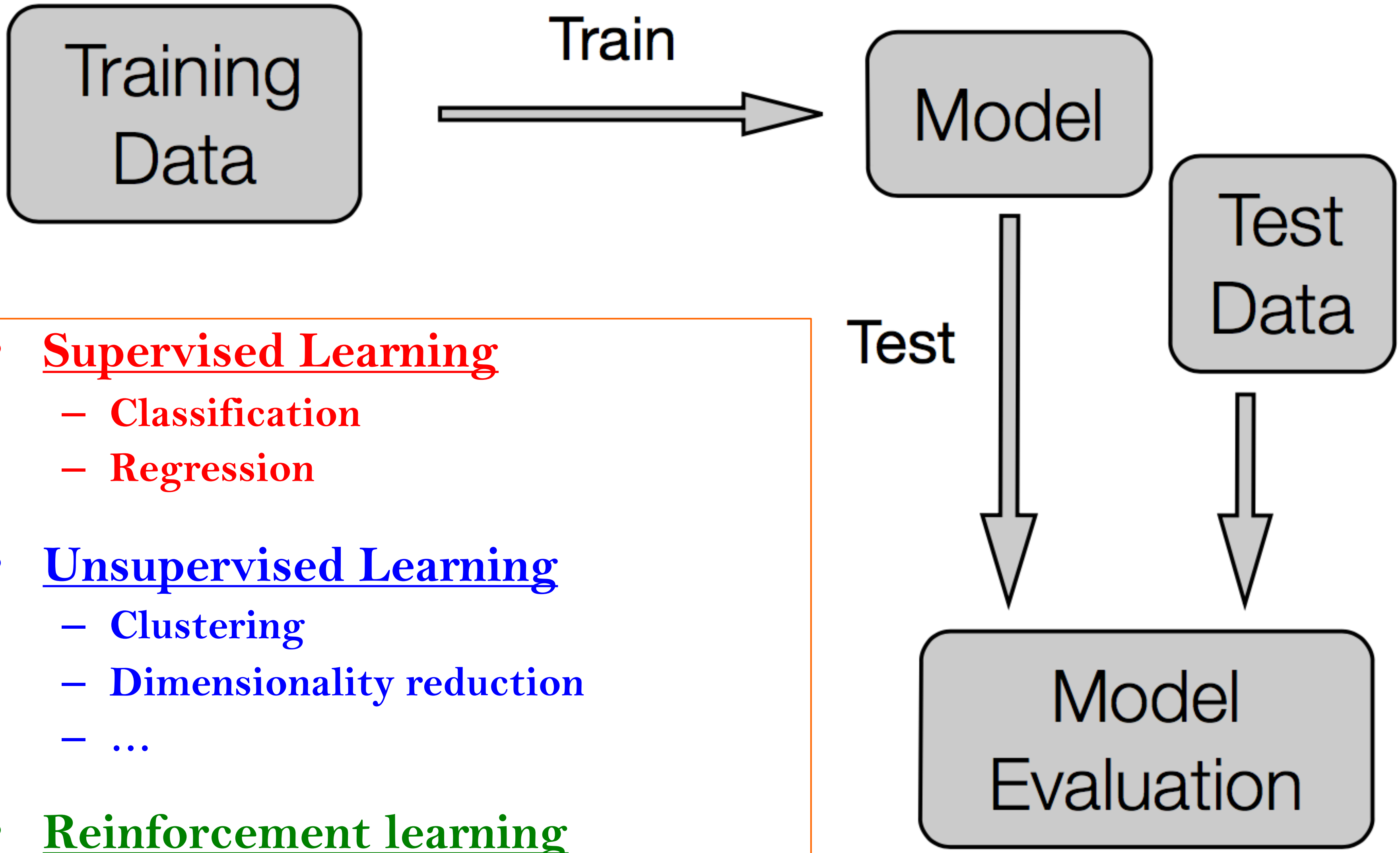
# Learning






[Ravikumar]

# Learning




- <u>Supervised Learning</u>
  - Classification
  - Regression
- <u>Unsupervised Learning</u>
  - Clustering
  - Dimensionality reduction
  - …
- <u>Reinforcement learning</u>

[Ravikumar]

# What we would want to do



$$\min_{h \in H} \int L(h(x), y) p(x, y) dx dy$$

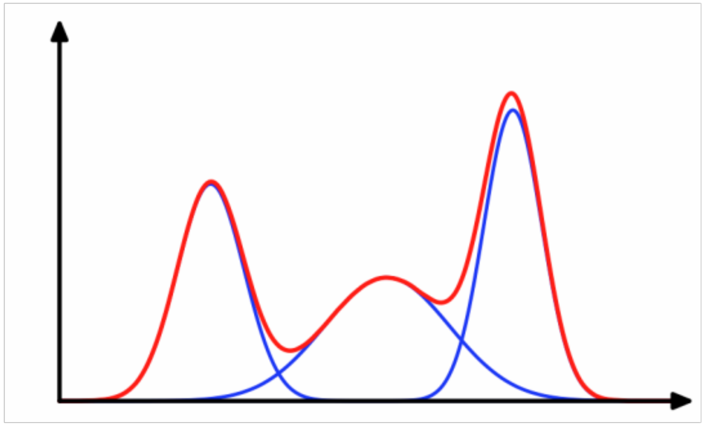
- Find best function $h$ to minimizes the expected loss
  - $L \equiv$ Loss to compare predictions $h(x)$ with target $y$
  - $H \equiv$ Set of functions to search over
  - $p(x, y) \equiv$ PDF of data

- But:
  - Don't know how to choose the set of functions $H$
  - Don't know how to search over all functions
  - Don't know true data distribution $p(x, y)$
  - Only have samples of data $\{x_i, y_i\}$

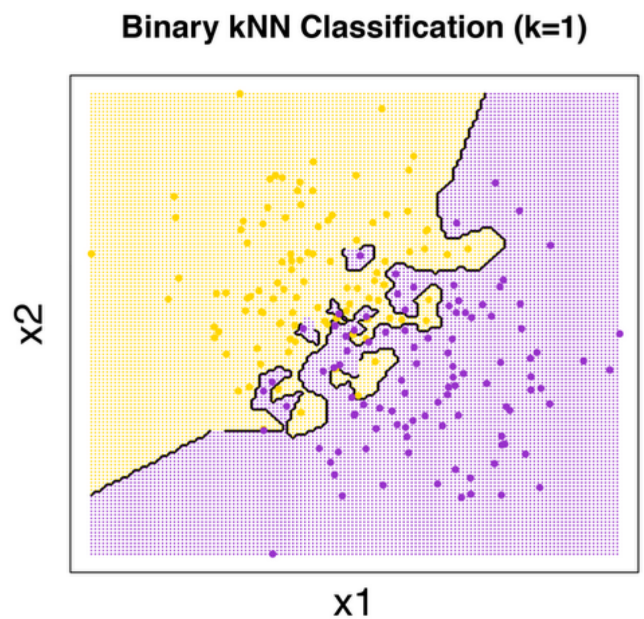
# Parametric vs. Non-parametric Models

# Parametric vs. Non-parametric Models

- **Parametric Models**:
  - Don't grow in complexity w/ dataset size.
  - Fixed set of parameters to learn
    - Example: sum of Gaussians, each with mean, variance, and normalization



- **Non-Parametric Models**:
  - Grow in complexity w/ more data
  - Don't have a fixed set of parameters,
    - Example: Nearest-Neighbors



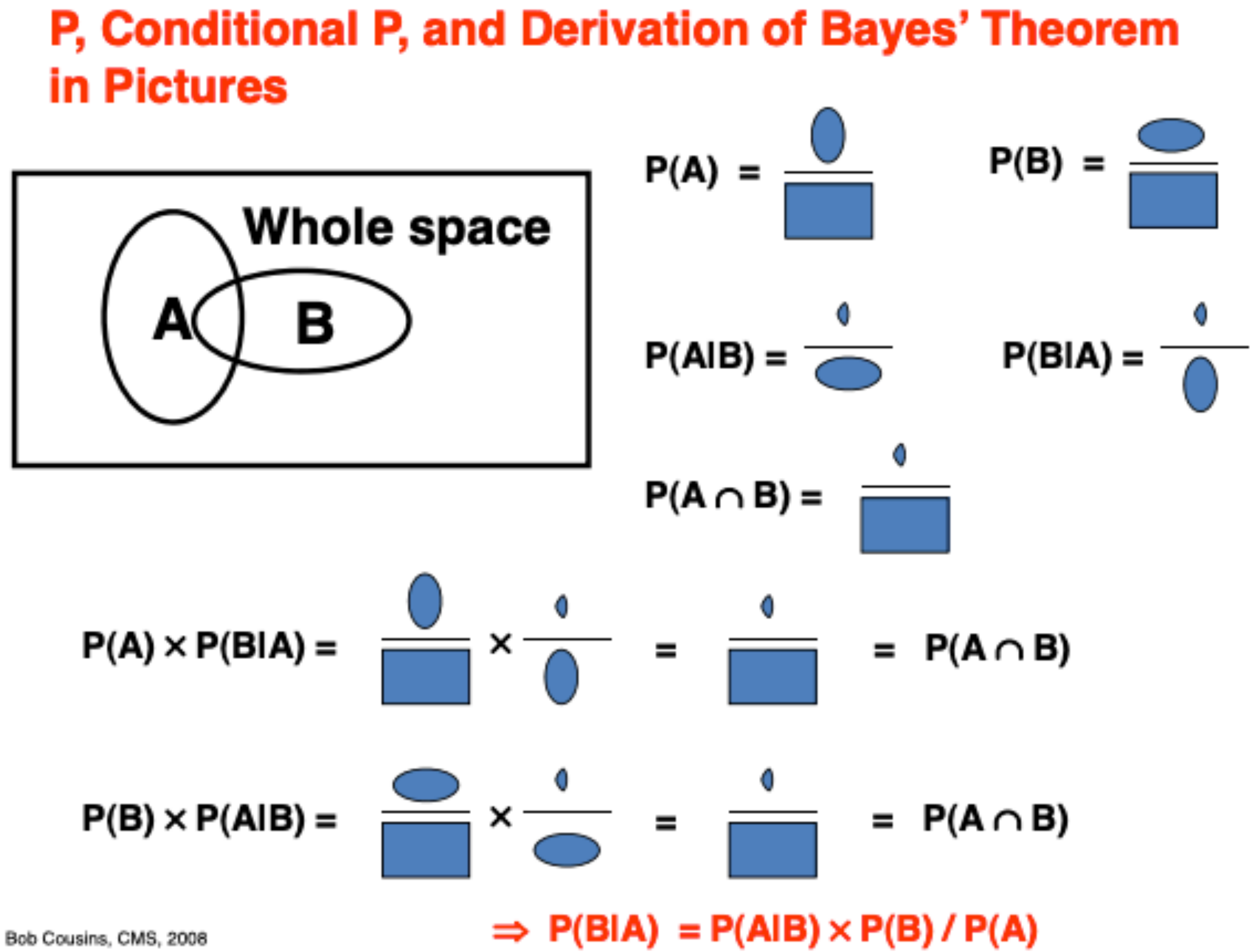

http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/

# Bayes Theorem in Pictures

# Bayes Theorem in Pictures

K. Cranmer: Intro to Stats.

… IN PICTURES (FROM BOB COUSINS)



# Gradient Descent

# How to Minimize Loss $\mathcal{L}(\theta)$? Gradient Descent

- **Gradient Descent**:

  Make a step $\theta \leftarrow \theta + \eta v$ in ***direction*** $v$ with ***step size*** $\boldsymbol{\eta}$ to reduce loss

- How does loss change in different directions?

  Let $\lambda$ be a perturbation along direction $v$

$$\frac{d}{d\lambda}\mathcal{L}(\theta + \lambda v)\bigg|_{\lambda=0} = v \cdot \nabla_\theta \mathcal{L}(\theta)$$

- Then Steepest Descent direction is: $v = -\nabla_\theta \mathcal{L}(\theta)$

# Stochastic Gradient Descent



- Loss is composed of a sum over samples:

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \mathcal{L}\big(y_i, h(x_i;\theta)\big)$$

  - Computing gradient grows linearly with N!

- **(Mini-Batch) Stochastic Gradient Descent**
  - Compute gradient update using 1 random sample (small size batch)
  - Gradient is unbiased → on average it moves in correct direction
  - Tends to be much faster the full gradient descent

- Several updates to SGD, like momentum, ADAM, RMSprop to
  - Help to speed up optimization in flat regions of loss
  - Have adaptive learning rate
  - Learning rate adapted for each parameter
  - …

# Bias Variance Tradeoff

# Bias Variance Tradeoff



- Model h(x), defined over dataset, modeling random variable output y

$$E[y] = \bar{y}$$
$$E[h(x)] = \bar{h}(x)$$

- Examining generalization error at x, w.r.t. possible training datasets

$$E[(y - h(x))^2] = E[(y - \bar{y})^2] \quad + \quad (\bar{y} - \bar{h}(x))^2 \quad + \quad E[(h(x) - \bar{h}(x))^2]$$
$$= \text{noise} \quad + \quad (\text{bias})^2 \quad + \quad \text{variance}$$

# Bias Variance Tradeoff

- Model h(x), defined over dataset, modeling random variable output y

$$E[y] = \bar{y}$$
$$E[h(x)] = \bar{h}(x)$$

- Examining generalization error at x, w.r.t. possible training datasets

$$E[(y - h(x))^2] = E[(y - \bar{y})^2] + (\bar{y} - \bar{h}(x))^2 + E[(h(x) - \bar{h}(x))^2]$$
$$= \text{noise} + (\text{bias})^2 + \text{variance}$$

Intrinsic noise in system or measurements
Can not be avoided or improved with modeling
Lower bound on possible noise

# Bias Variance Tradeoff



- Model h(x), defined over dataset, modeling random variable output y

$$E[y] = \bar{y}$$
$$E[h(x)] = \bar{h}(x)$$

- Examining generalization error at x, w.r.t. possible training datasets

$$E[(y - h(x))^2] = E[(y - \bar{y})^2] \quad + \quad (\bar{y} - \bar{h}(x))^2 \quad + \quad E[(h(x) - \bar{h}(x))^2]$$
$$= \text{noise} \quad + \quad (\text{bias})^2 \quad + \quad \text{variance}$$

- The **more complex the model** h(x) is, the more data points it will capture, and **the lower the bias** will be.

# Bias Variance Tradeoff



- Model h(x), defined over dataset, modeling random variable output y

$$E[y] = \bar{y}$$
$$E[h(x)] = \bar{h}(x)$$

- Examining generalization error at x, w.r.t. possible training datasets

$$E[(y - h(x))^2] = E[(y - \bar{y})^2] + (\bar{y} - \bar{h}(x))^2 + E[(h(x) - \bar{h}(x))^2]$$
$$= \text{noise} + (\text{bias})^2 + \text{variance}$$

- The **more complex the model** h(x) is, the more data points it will capture, and **the lower the bias** will be.

- **More Complexity** will make the model "move" more to capture the data points, and hence its **variance will be larger**.

# Bias Variance Tradeoff

- Model h(x), defined over dataset, modeling random variable output y

$$E[y] = \bar{y}$$
$$E[h(x)] = \bar{h}(x)$$

- Examining generalization error at x, w.r.t. possible training datasets

$$E[(y - h(x))^2] = E[(y - \bar{y})^2] + (\bar{y} - \bar{h}(x))^2 + E[(h(x) - \bar{h}(x))^2]$$
$$= \text{noise} + (\text{bias})^2 + \text{variance}$$

- The **more complex the model** h(x) is, the more data points it will capture, and **the lower the bias** will be.

- **More Complexity** will make the model "move" more to capture the data points, and hence its **variance will be larger**.
  - **As dataset size grows, can reduce variance! Can use more complex model**

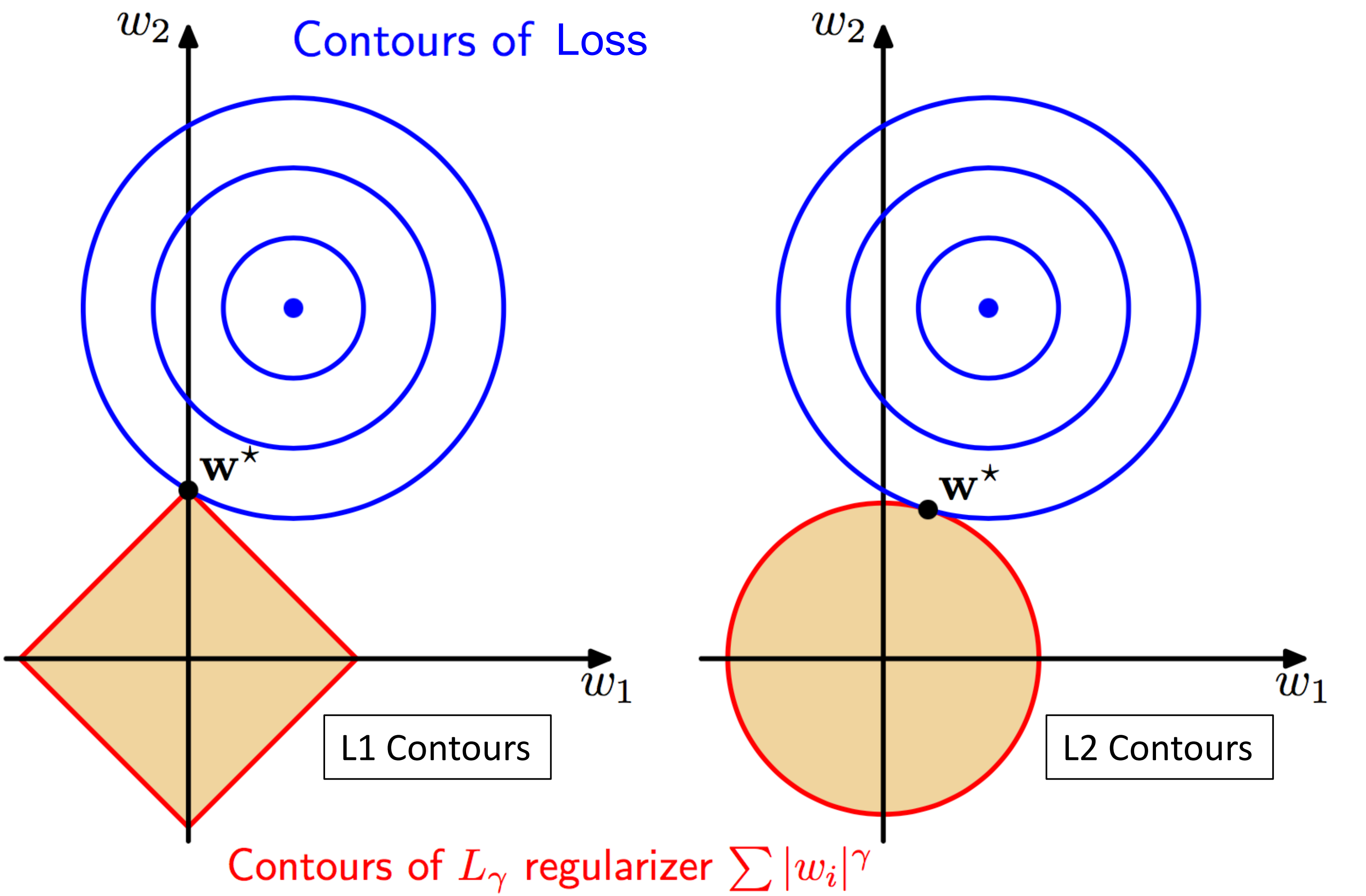
# Regularization and Cross Validation

# Regularization

- Can control the complexity of a model by placing **constraints on the model parameters**
  - Trading some bias to reduce model variance
- **L2 norm**: $\Omega(\mathbf{w}) = ||\mathbf{w}||^2 = \sum_i w_i^2$
  - "Ridge regression", enforcing weights not too large
  - Equivalent to Gaussian prior over weights
- **L1 norm**: $\Omega(\mathbf{w}) = ||\mathbf{w}|| = \sum_i |w_i|$
  - "Lasso regression", enforcing sparse weights
- Elastic net $\rightarrow$ L1 + L2 constraints

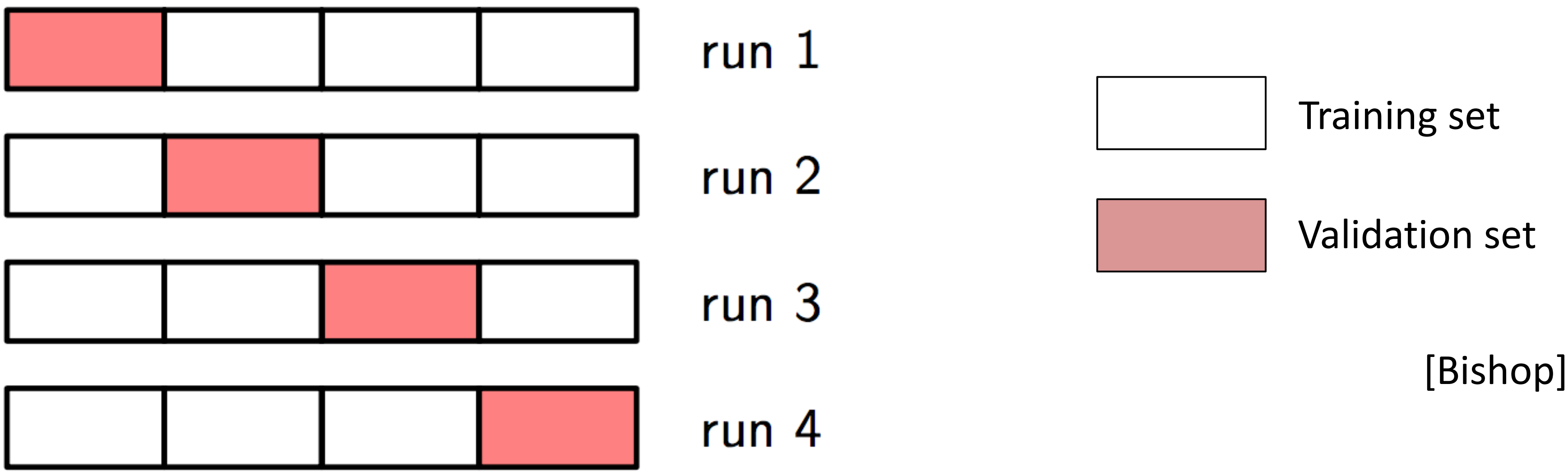
# Regularization






Pattern Recognition and Machine Learning C. M. Bishop (2006)

# Cross Validation



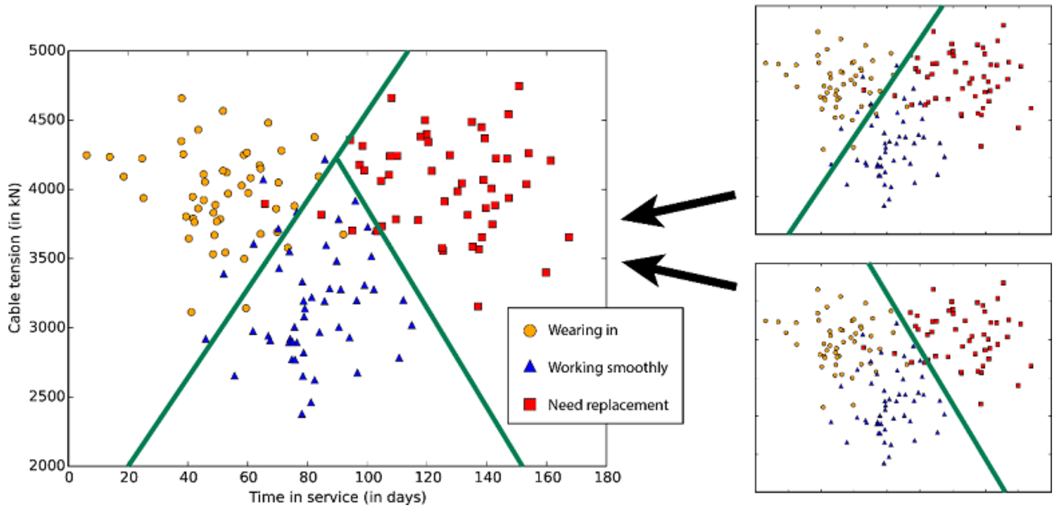

- Especially when dataset is small, split training set into K-folds
  - Train on (K-1) folds, validate on 1 fold, then iterate
  - Use average estimated performance on K-folds
  - Allows for estimate of performance RMS

- Even when dataset not small, useful technique to estimate variance of expected performance, and for comparing different models / hyperparameters

# Multiclass Classification

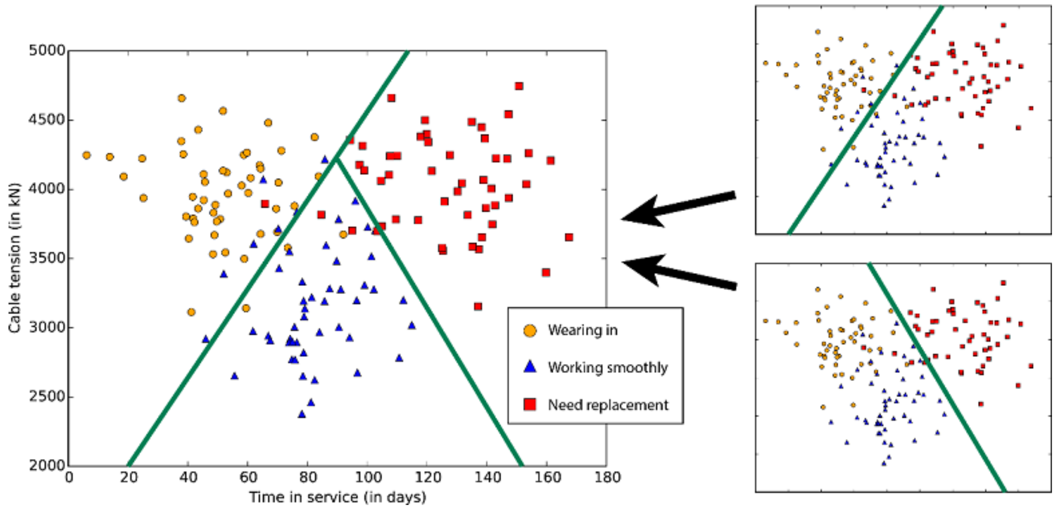
# Multiclass Classification?

- What if there is more than two classes?



# Multiclass Classification?

- What if there is more than two classes?




- Softmax → multi-class generalization of logistic loss
  - Have N classes $\{c_1, \ldots, c_N\}$
  - Model target $\mathbf{y}_k = (0, \ldots, 1, \ldots 0)$

$k^{th}$ element in vector

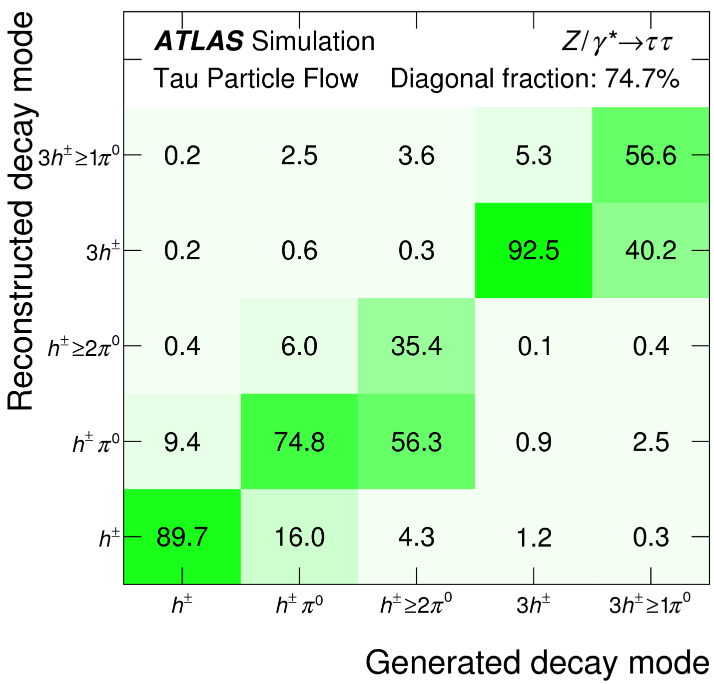
$$p(c_k|x) = \frac{\exp(\mathbf{w}_k x)}{\sum_j \exp(\mathbf{w}_j x)}$$

  - Gradient descent for each of the weights $\mathbf{w}_k$

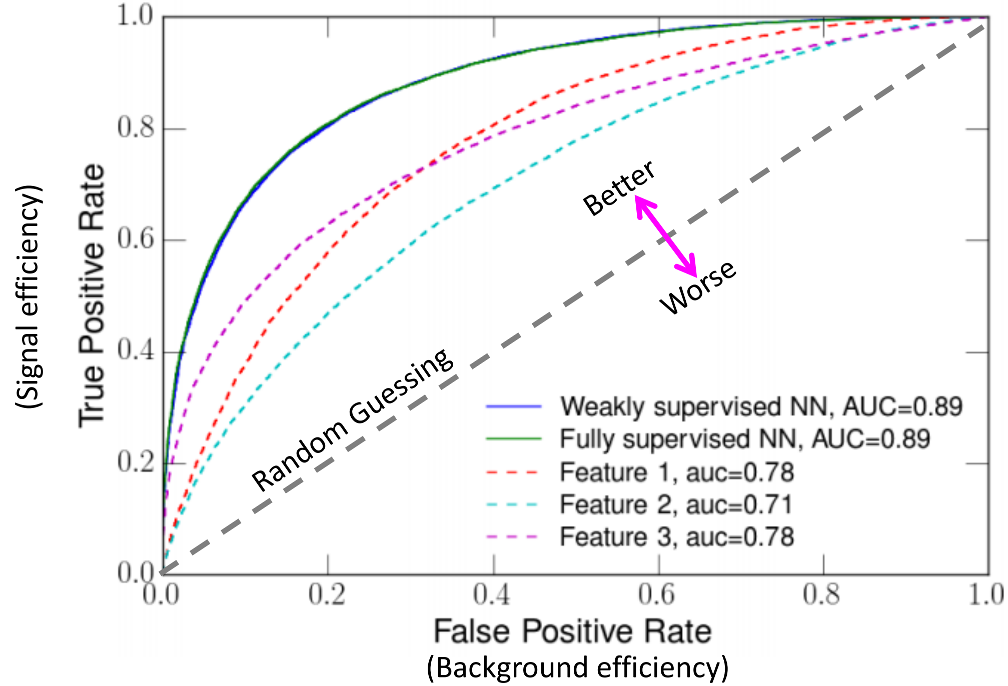
# Estimating a Classifier Performance

| | | Predicted | |
|---|---|---|---|
| | | Positive | Negative |
| True | Positive | True Positives (TP) | False Negatives (FN) |
| | Negative | False Positives (FP) | True Negatives (TN) |

Confusion Matrix
Classifying tau decays




Receiver Operating Characteristic (ROC) Curve
classifying quarks vs. gluons




arXiv:1512.05955

arXiv:1702.00414