Introduction to Machine Learning:

Lecture 1 – Machine Learning Fundamentals

Michael Kagan

SLAC NATIONAL ACCELERATOR LABORATORY

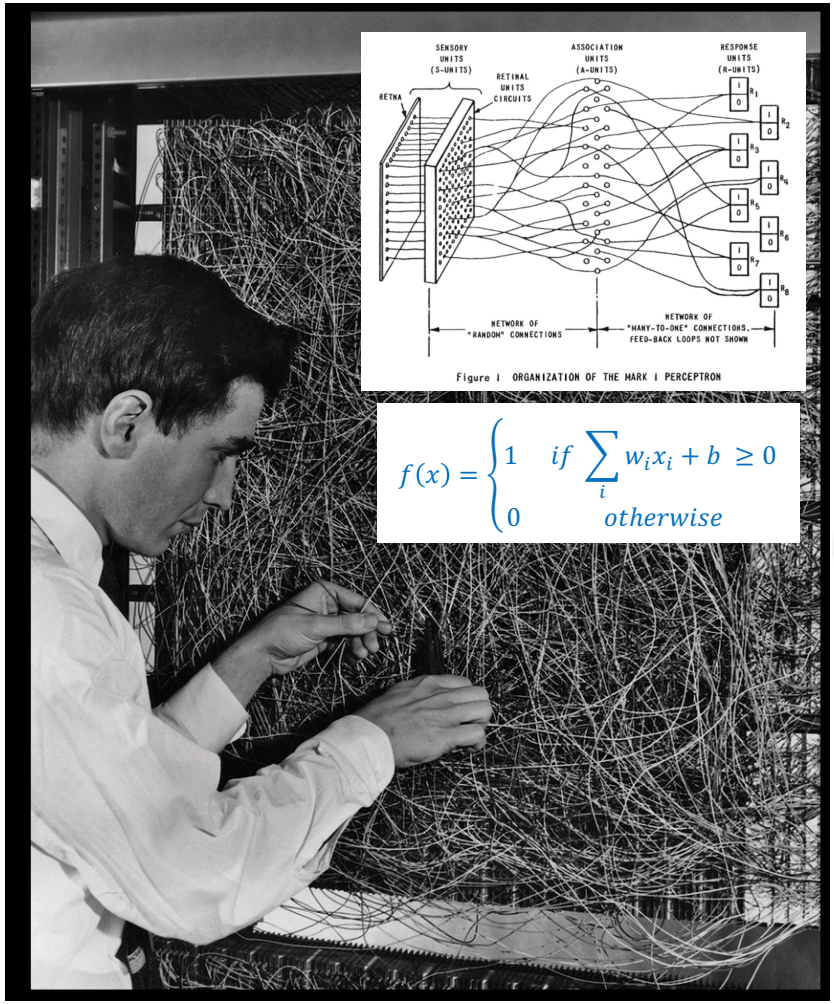
TRISEP Summer School

July 8-12, 2024

The Plan

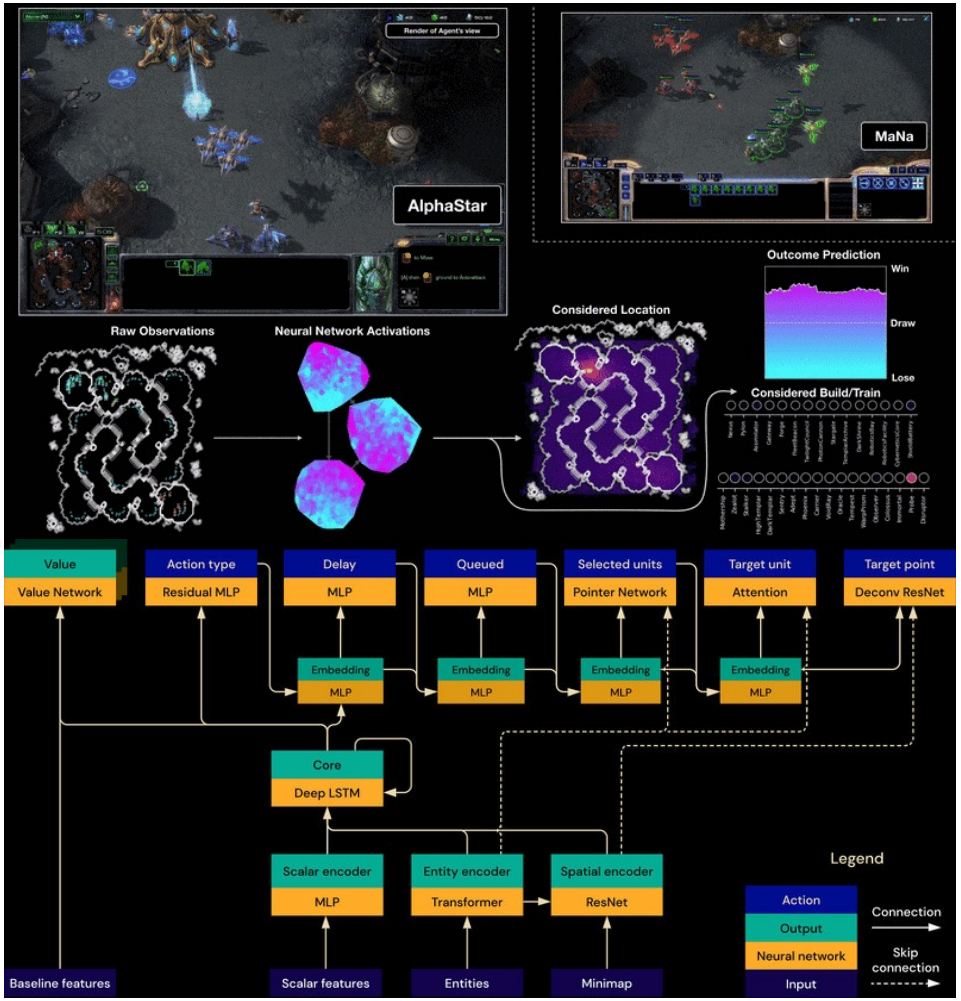
- Lecture 1 – Machine Learning Fundamentals
- Lecture 2 – Intro to Neural Networks
- Lecture 3 – Intro to Deep Learning
- Lecture 4 – Intro to Unsupervised Learning
- Lecture 5 – Intro to Deep Generative Models

Long History of Machine Learning



$$f(x) = \begin{cases} 1 & if \sum_i w_i x_i + b \geq 0 \\ 0 & otherwise \end{cases}$$

Perceptron



AlphaStar

Rosenblatt 1958, 1960

Vinyals et. al. 2019

The Power of ML

street style photo of a woman selling pho at a Vietnamese street market, sunset, shot on fujifilm

generate low-level, high-dim data from high-level concepts

High-Level Concept

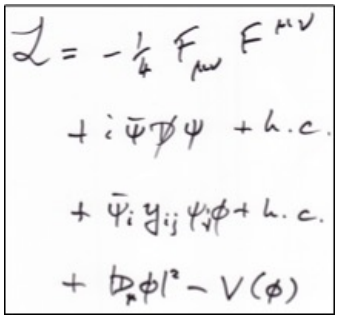
Low-Level Data

This is a picture of Barack Obama. His foot is positioned on the right side of the scale. The scale will show a higher weight.

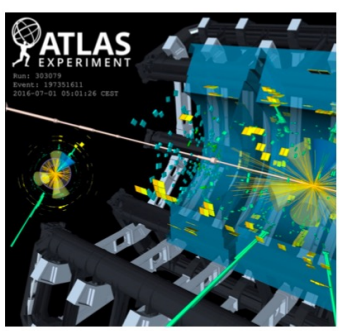
reconstruct high level concepts from low-level, high-dim data

Slide credit: L. Heinrich

Particle Physics Has Similar Goals!



generate low-level, high-dim data from high-level concepts

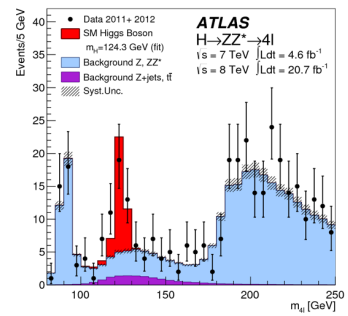


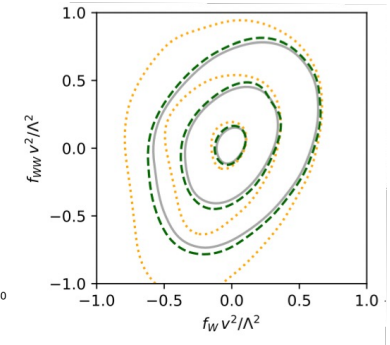
Simulation

High-Level Concept

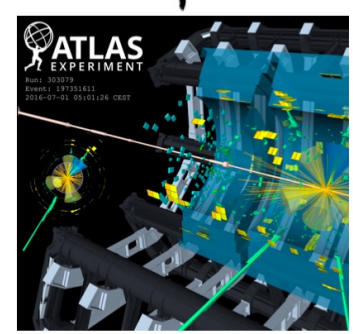
Low-Level Data

Data Analysis



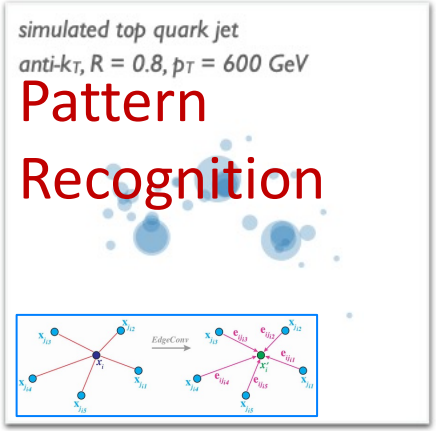


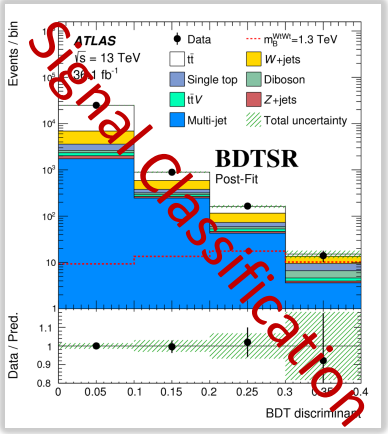
reconstruct high level concepts from low-level, high-dim data

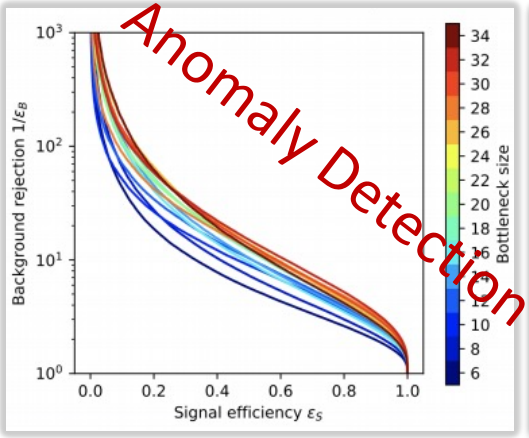


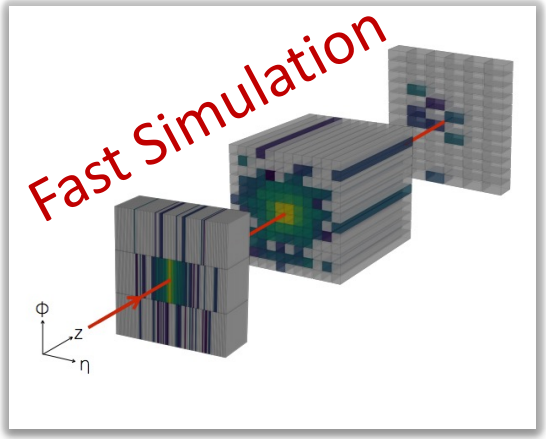
Slide credit: L. Heinrich

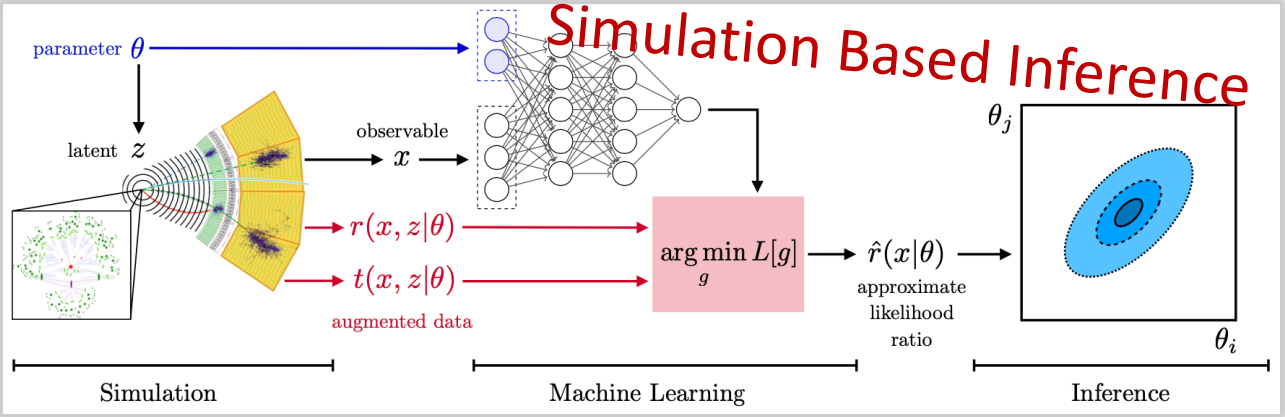
Machine Learning in HEP

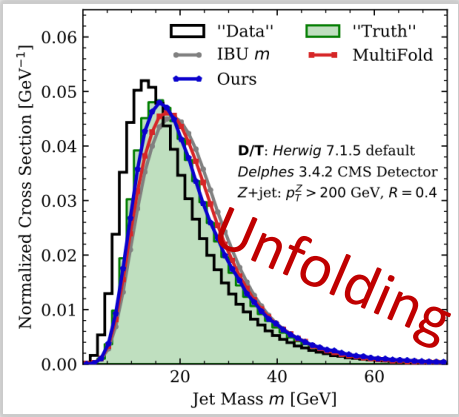


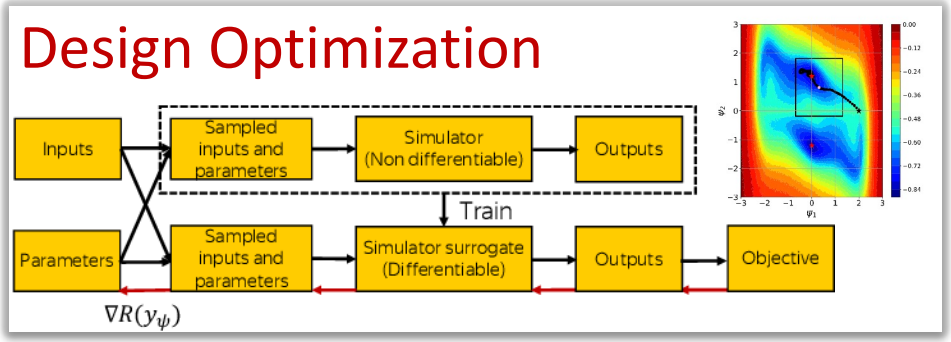


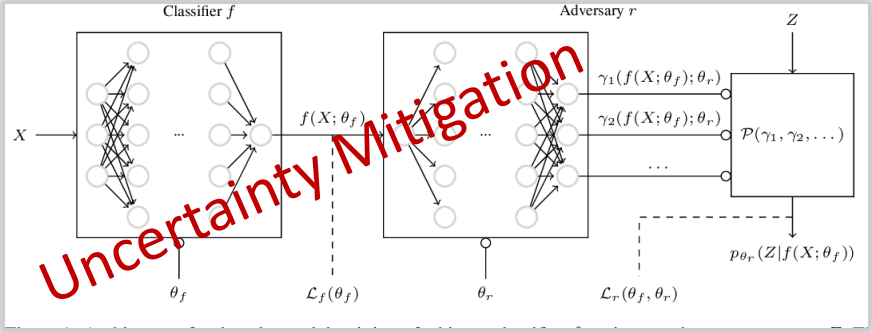










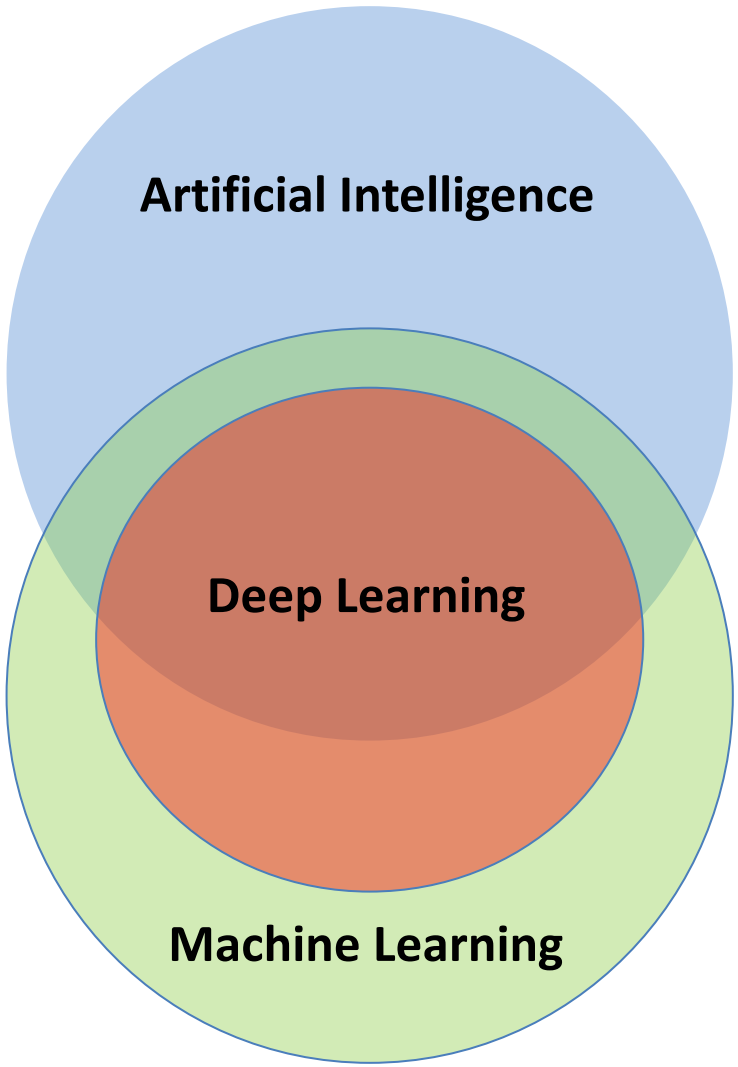


+ More!

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

Artificial Intelligence, Machine Learning, Deep Learning



- **AI:** make computers act in an intelligent way
  - Rules, reasoning, symbol manipulation
- **ML:** Uses data to learn "intelligent" algorithms
- **Deep Learning**: Approach to ML that (often) uses complex pipelines to process low level data (e.g. pixels)

Slide credit: L. Heinrich

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:

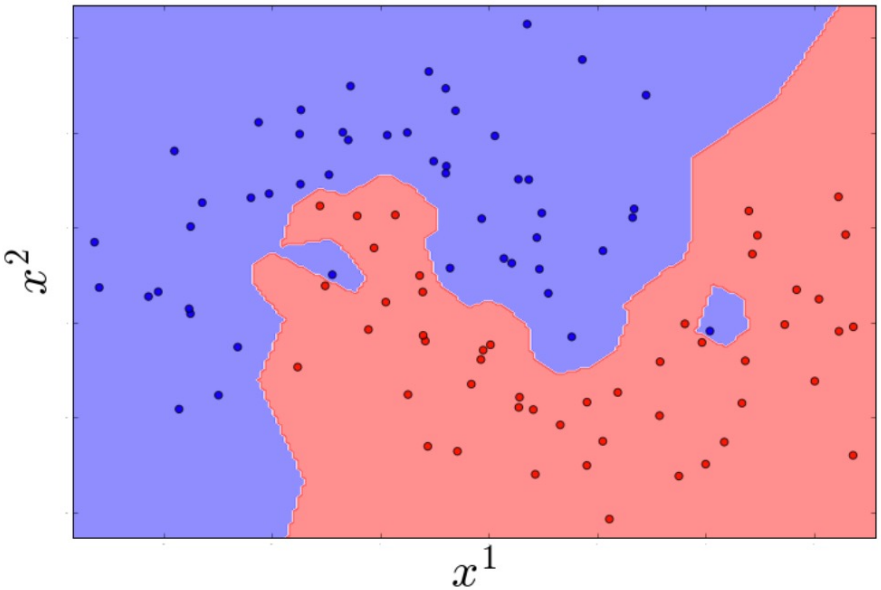
  $\{x_i\}$ – N examples of ***observed features***

  $\{y_i\}$ – N prediction ***targets*** or ***labels***

- Learn function mapping $\boldsymbol{h(x) = y}$

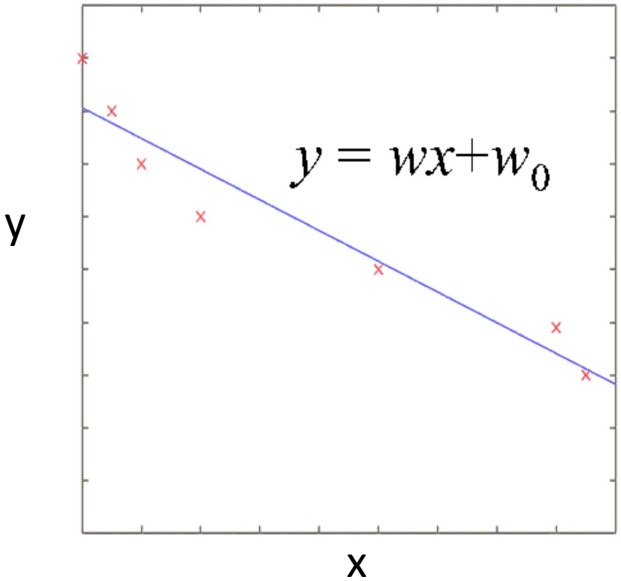
Classification:

Y is a finite set of **labels** (i.e. classes) denoted with integers



Regression:

Y is a real number

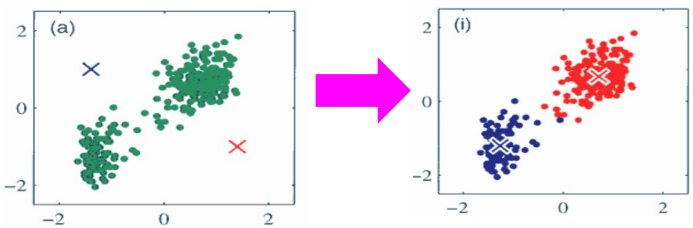


Unsupervised Learning

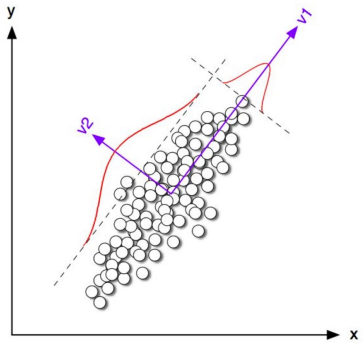
Given data $D = \{x_i\}$, but no labels, find structure in data

[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 density $p(x)$, and/or learn to draw new samples of x

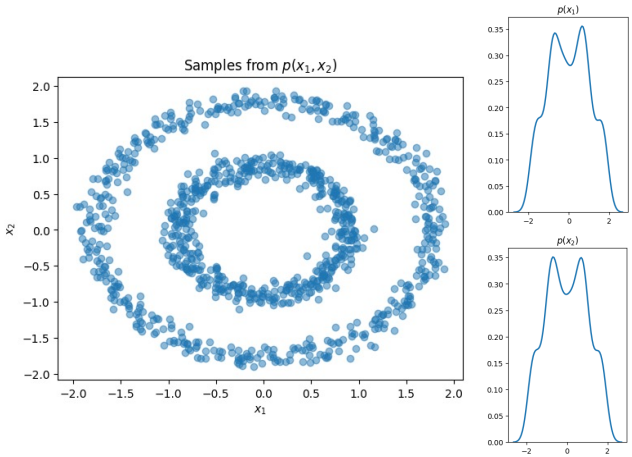
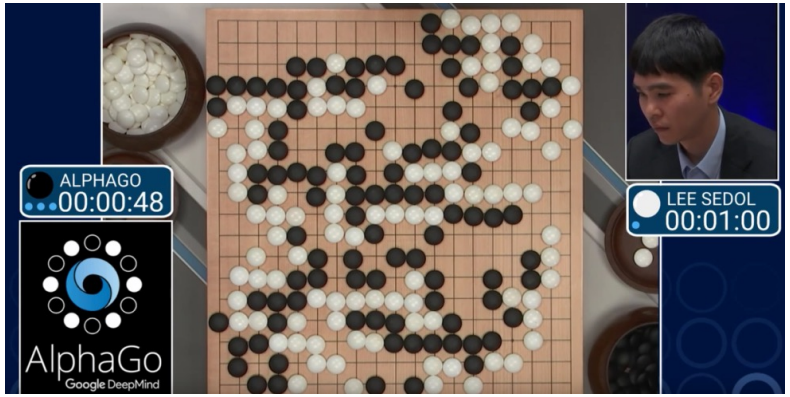


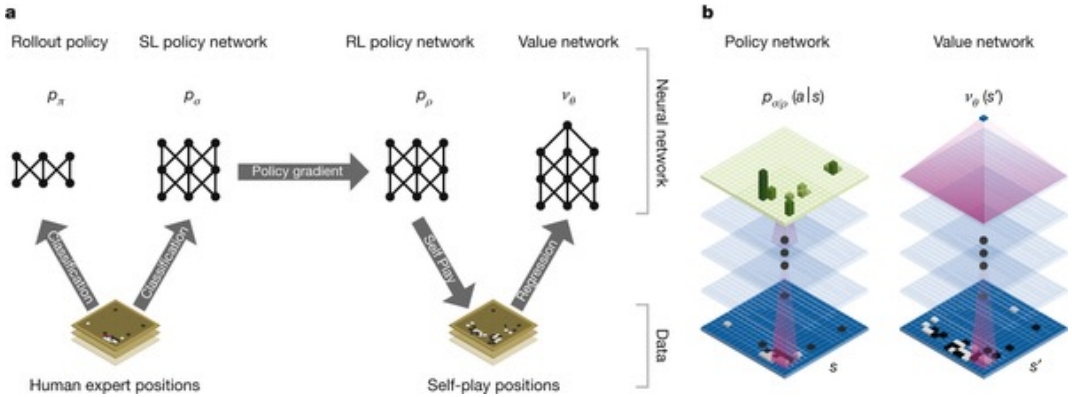
Image Credit - Link

Reinforcement Learning

Image credit: Ravikumar

- Learn to make the best sequence of decisions to achieve a given goal when feedback is often delayed until you reach the goal





Nature 529, 484–489 (28 January 2016)

Brief Review of Probability and Statistics

Probability Mass Function

Probability Mass Function of Discrete random variables (r.v.)

$$P(x_i) = p_i$$

- Prob. of ith 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. $p \in [0,1]$

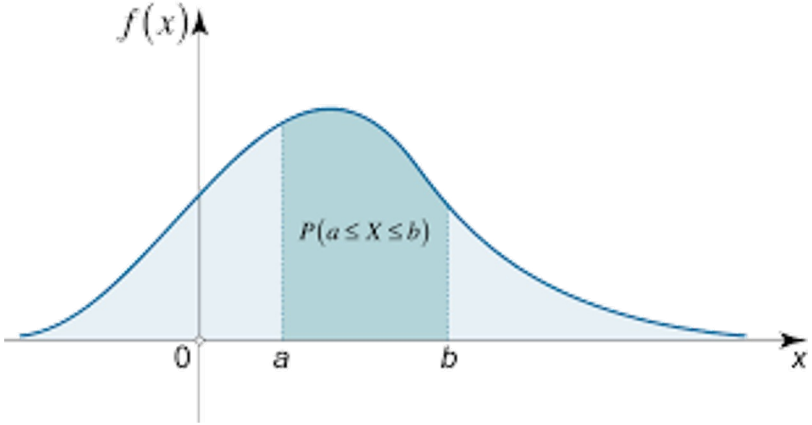
Slide Credit: K. Cranmer, Intro to Stats.

Probability Mass and Density Functions

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

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

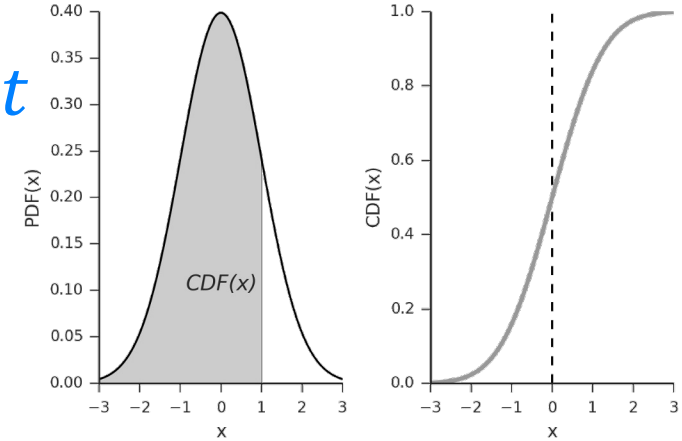
– Normalized: $\int_{-\infty}^{\infty} f(x)dx = 1$



Cumulative Distribution Function

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

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



Slide Credit: K. Cranmer, Intro to Stats.

Expected Values

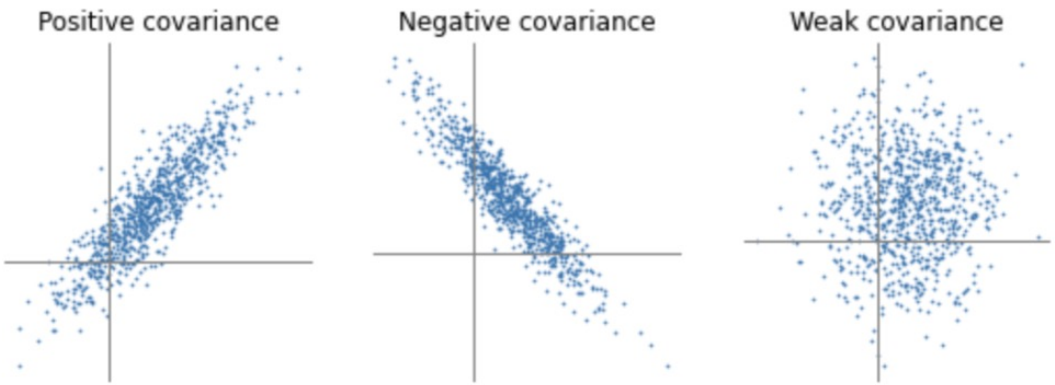
- Expected value of a function of random variables

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

- 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: $Cov(x, y) = \mathrm{E}[(x - \mathrm{E}[x])(y - \mathrm{E}[y])]$



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 $p(x)$

Expected Values

- Expected value of a function of random variables

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

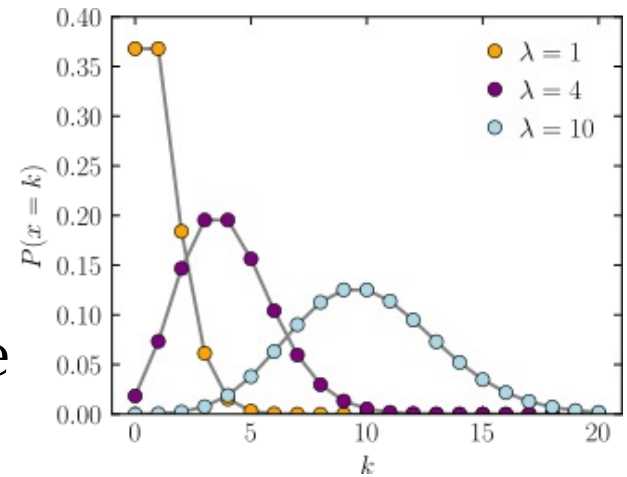
- Often we can't compute this integral
- Or often in Machine Learning we don't know $p(x)$

- With set of N repeated observations $\{x_i\}$ that are independent and identically distributed, can approximate with Empirical Estimator… i.e. **Monte Carlo** estimate

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

Parametric Models

- PDF often depends on parameters θ 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
- λ = average number of events

Continuous: Gaussian Distribution:

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

- μ is the average value
- σ^2 is the variance

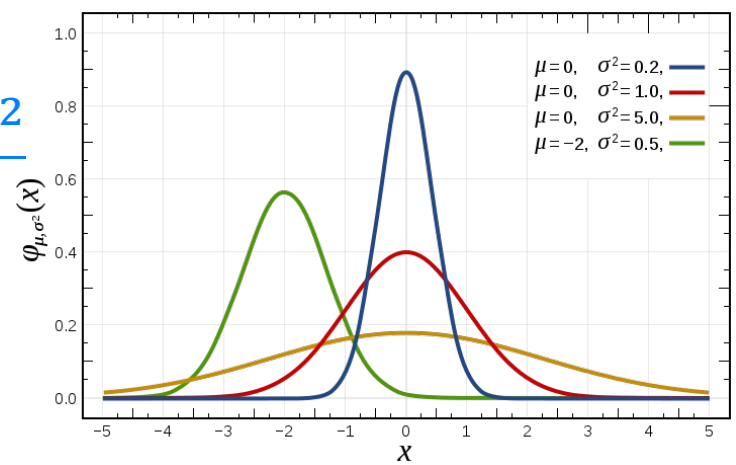


Image source: Wikipedia

Likelihood Function

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

Poisson Example: Likelihood of μ for a given n

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

– Continuous function of μ
– NOTE: not a PDF

– Common to examine: $-\ln L$

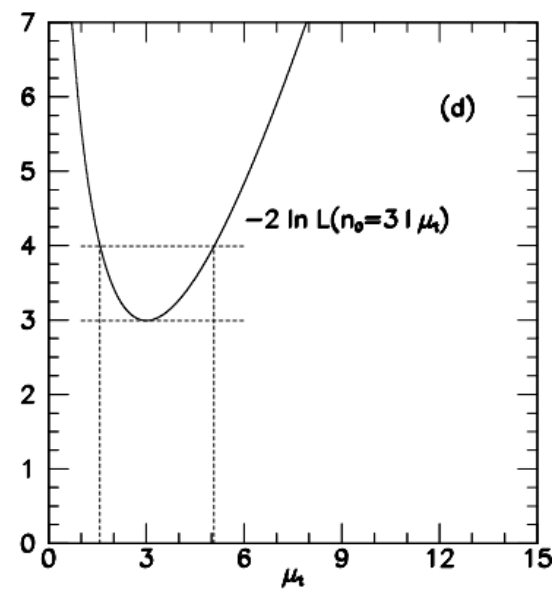


Figure from R. Cousins, Am. J. Phys. 63 398 (1995)

Slide Credit: K. Cranmer: Intro to Stats.

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

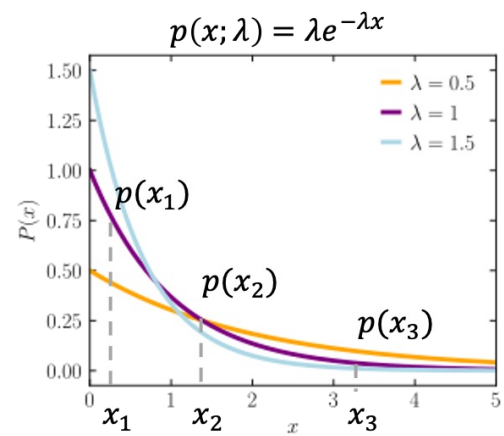
- Given observations $\{x_i\}$ and model PDF $f(x|\theta)$ the maximum likelihood estimator for θ is:

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

Maximum Likelihood

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

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



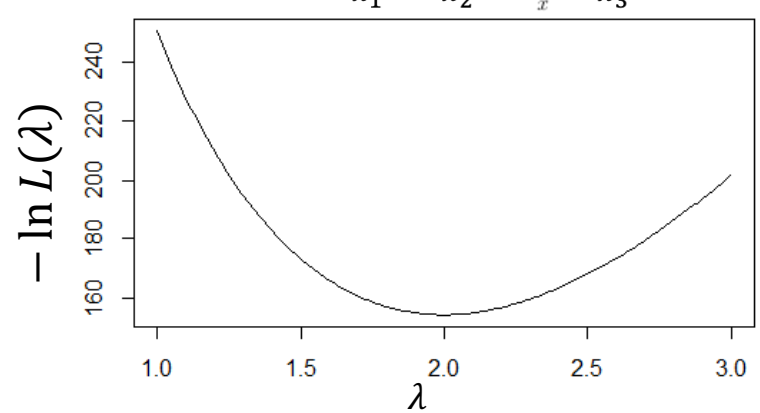
Example: Exponential $p(x;\lambda) = \lambda e^{-\lambda x}$

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

Finding Minimum:

$$0 = \frac{\partial(-\ln L(\lambda))}{\partial \lambda} = \frac{-n}{\lambda} + \sum_i x_i$$

$$\rightarrow \lambda^*(\{x_i\}) = \frac{n}{\sum_i x\_i}$$



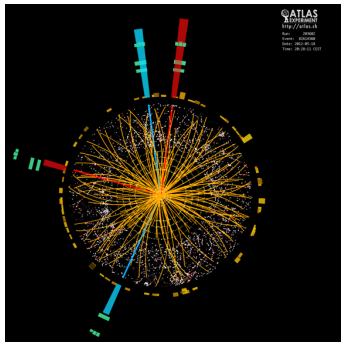
Bayes Rule

- 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 y
  - $p(x)$ is the "evidence", acts as 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.

Supervised Learning: How does it work?

Supervised Learning: How does it work?



$h(\boldsymbol{x};\ \boldsymbol{w})$
Function with adjustable parameters

Loss Function

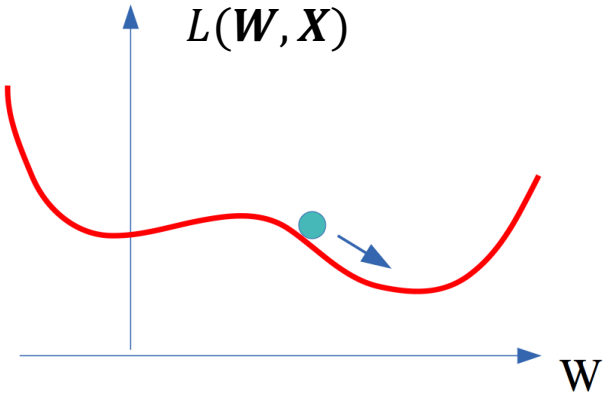
Compare prediction with true label

Loss

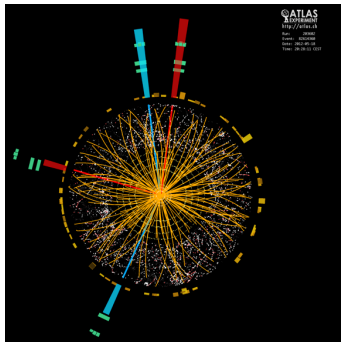
True labels:
Higgs = 1
Bkg = 0

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

Y. Le Cun



Supervised Learning: How does it work?



$h(\boldsymbol{x};\ \boldsymbol{w})$
Function with adjustable parameters

Loss Function

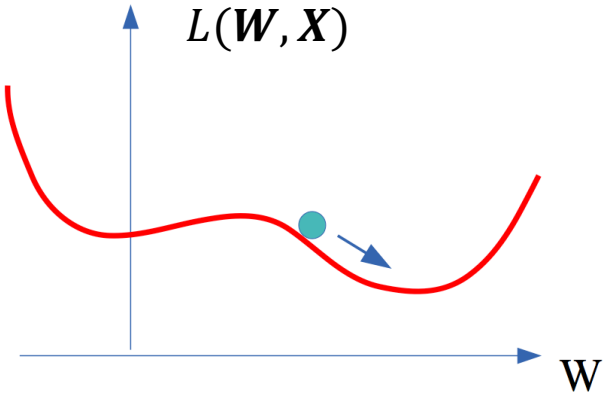
Compare prediction with true label

Loss

True labels:
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

$$\arg\min_{\mathbf{w}} \underbrace{\frac{1}{N}\sum_{i=1}^{N} L(h(\mathbf{x}_i;\mathbf{w}), y_i)}_{\text{Average expected loss}} + \underbrace{\lambda\Omega(\mathbf{w})}_{\text{Model regularization}}$$

- Framework to design learning algorithms
- $\boldsymbol{L}$ is loss function: compare prediction $h(\cdot)$ to label y
- $\Omega(\boldsymbol{w})$ is a regularizer, penalizing certain values of $\boldsymbol{w}$
  - λ controls how much penalty. Hyperparameter we tune
- Learning is cast as an optimization problem

Example Loss Functions

- Square Error Loss:
  - Often used in regression

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

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

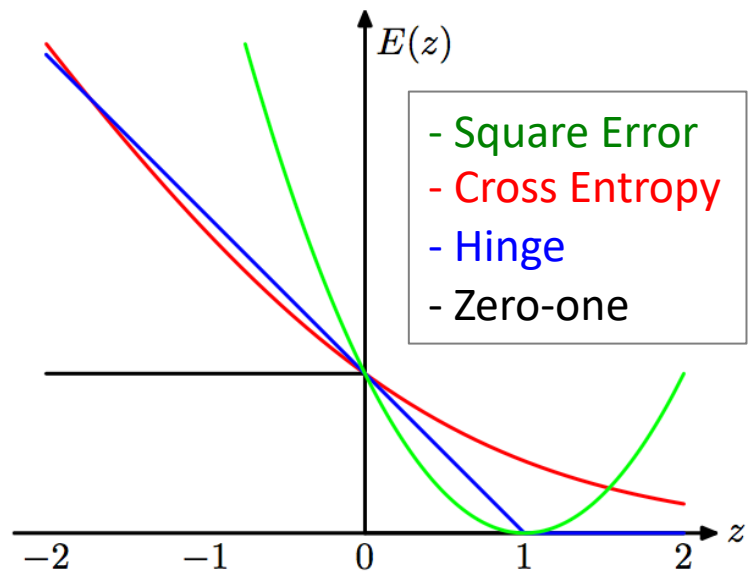
$$L(h(\mathbf{x};\mathbf{w}), y) = -\,y \log h(\mathbf{x};\mathbf{w}) - (1-y)\log(1-h(\mathbf{x};\mathbf{w}))$$

- Hinge Loss:
  - With y ∈ {-1,1}

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

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

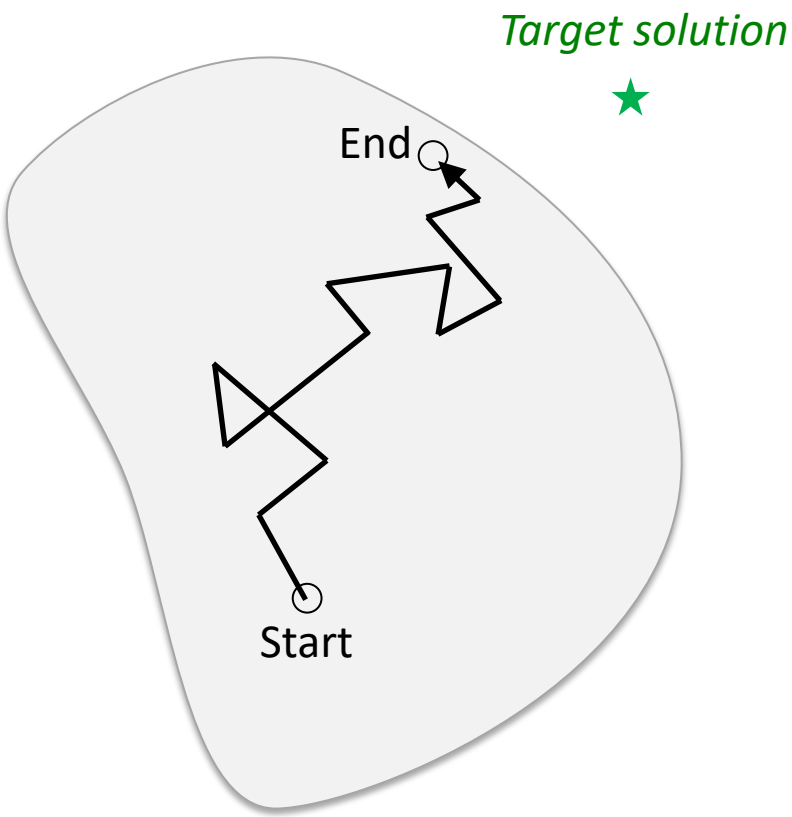
$$L(h(\mathbf{x};\mathbf{w}), y) = 1_{y \neq h(\mathbf{x};\mathbf{w})}$$



[Bishop]

Model Space and Learning Algorithms

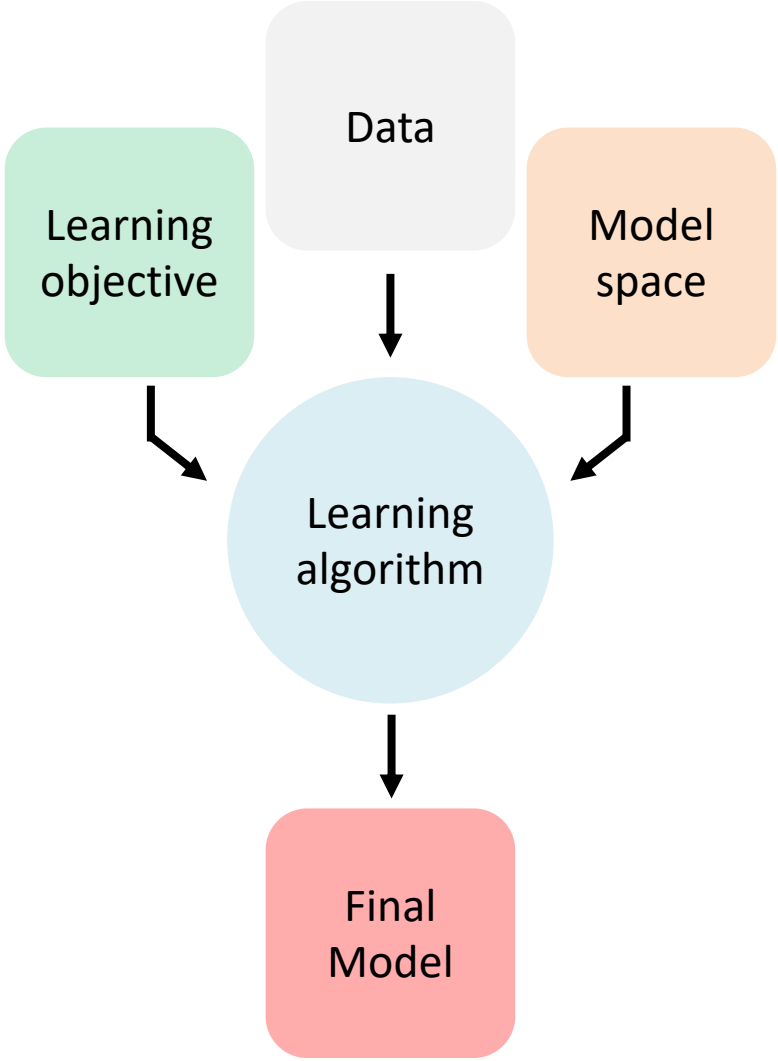
- Choose type of model
  - Each set of parameters is a point in space of models
- Need to find the best model parameters for loss
- **Learning is like a search** through space of models, **guided by the data**
- Various possibilities
  - Exhaustive search
  - Closed form solutions (rare)
  - Iterative optimization



Space of Possible Models

Putting It All Together

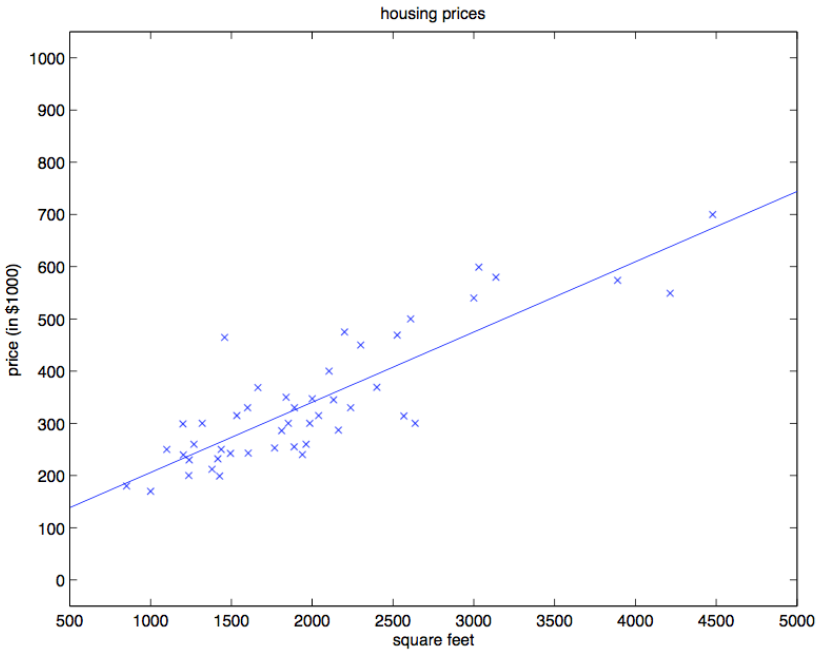
- Gather data to be used
- Propose a space of possible models
- Define what "good" means with loss function / learning objective
- Use learning algorithm to find best model



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

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

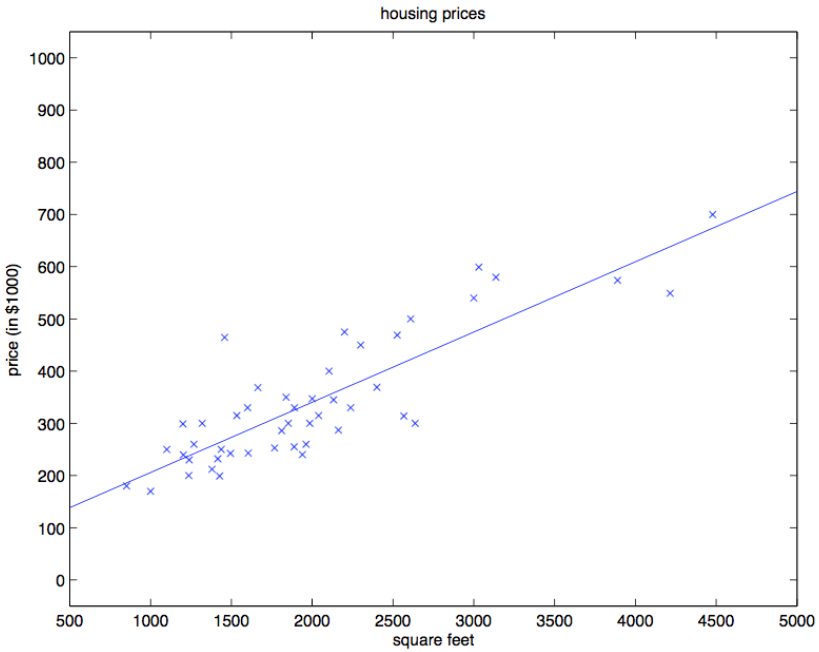
- Squared Loss function:

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

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

Least Squares Linear Regression

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

NOTE: Often use affine coordinates:

$$y = \boldsymbol{w}^T\boldsymbol{x} + w_0 \rightarrow \boldsymbol{w}^T\boldsymbol{x}$$

where

$$\boldsymbol{w} = \{w_0, w_1, \ldots, wn\}$$
$$\boldsymbol{x} = \{1, x_1, \ldots, xn\}$$

Least Squares Linear Regression: Matrix Form

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

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \arg\min_{\mathbf{w}} L(\mathbf{w})$$

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:

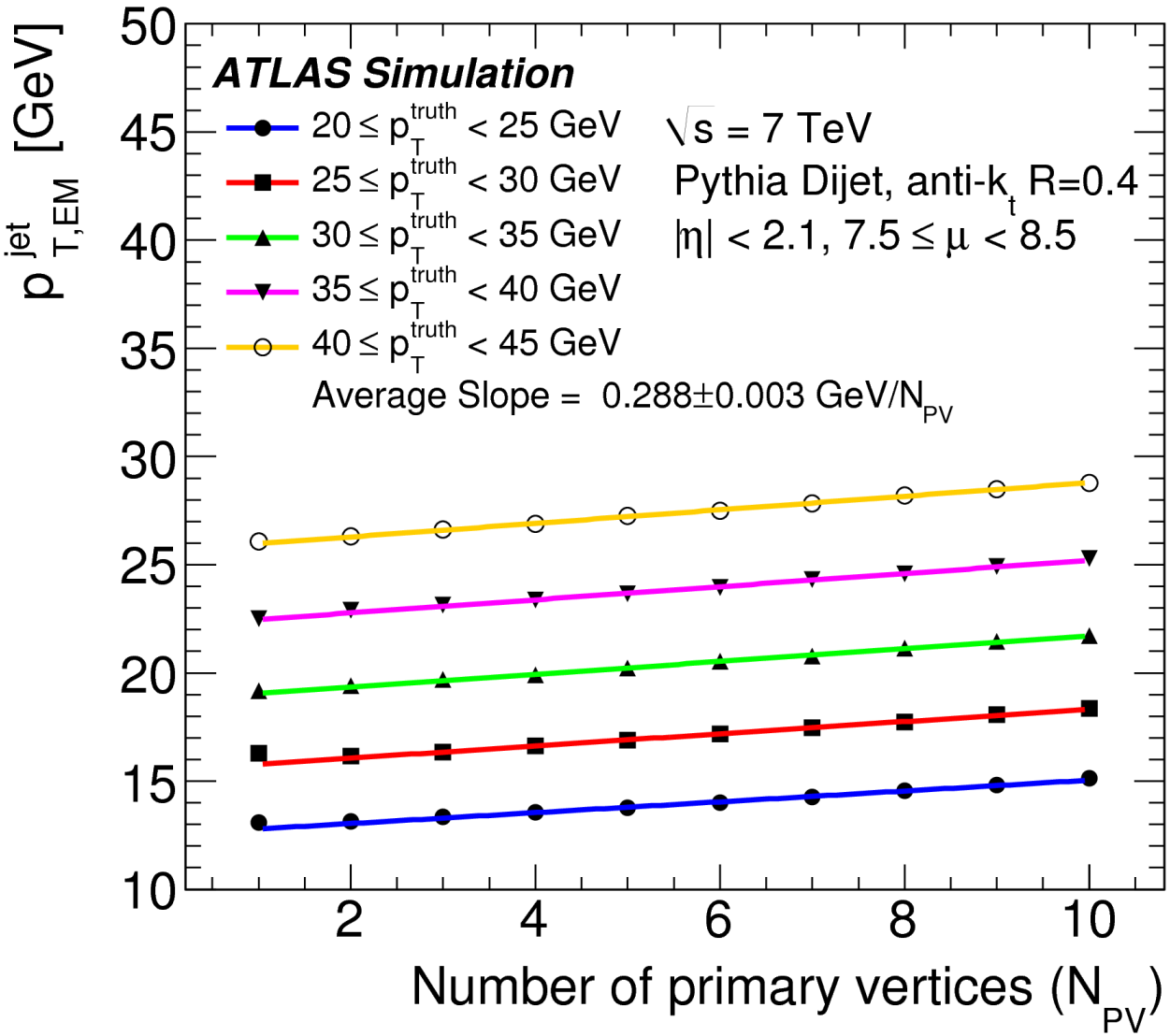
$$L(m) = p(\mathbf{y}|\mathbf{X}; m) = \prod_i p(y_i|x_i; m)$$

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

Squared loss function!

Linear Regression Example

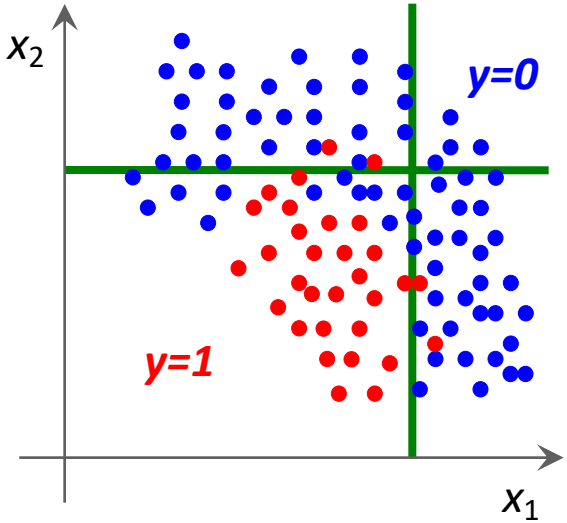
Eur. Phys. J. C (2015) 75:17



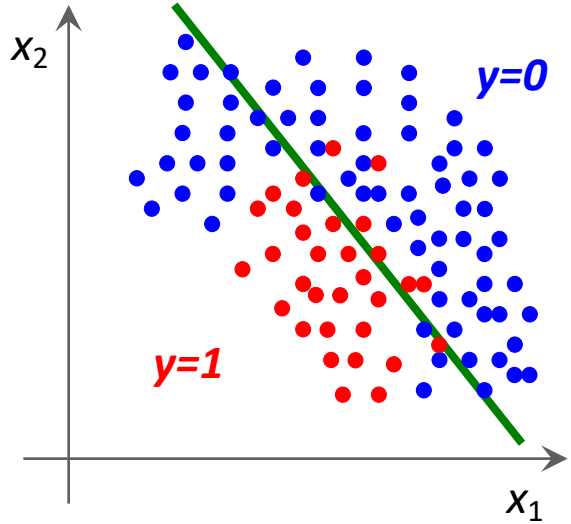
- Reconstructed Jet energy vs. Number of primary vertices

Linear Classification

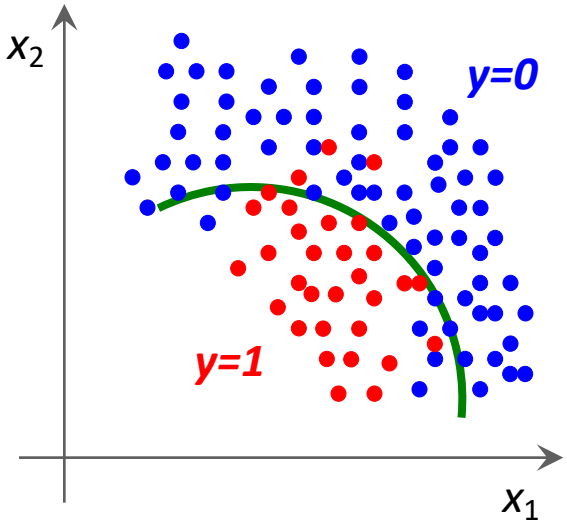
Classification



Rectangular cuts

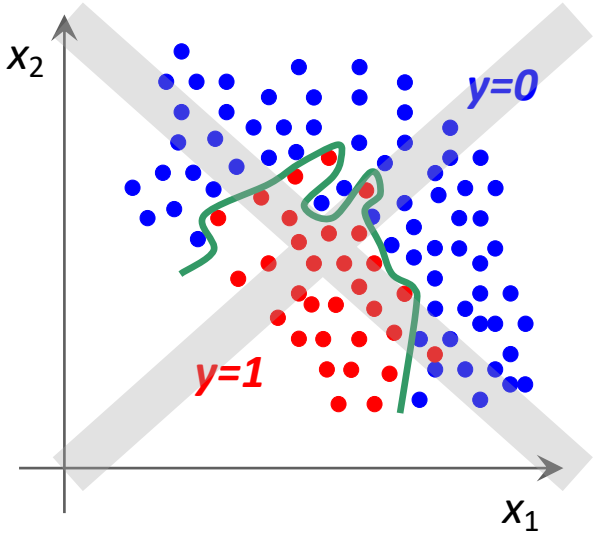


Linear discriminant



Nonlinear discriminant

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



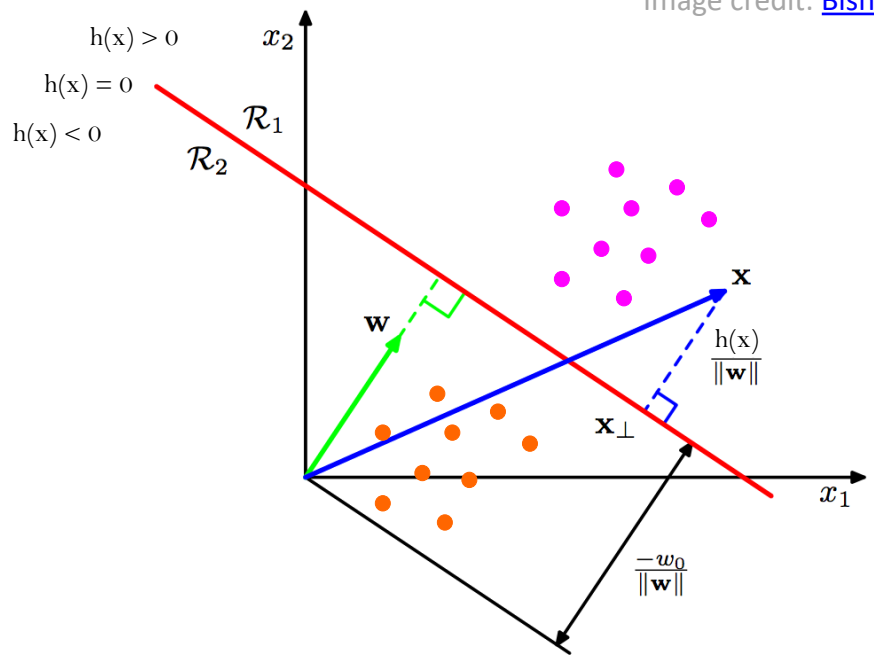
Linear Decision Boundaries

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

- Linear discriminant model

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

Image credit: Bishop



[Bishop]

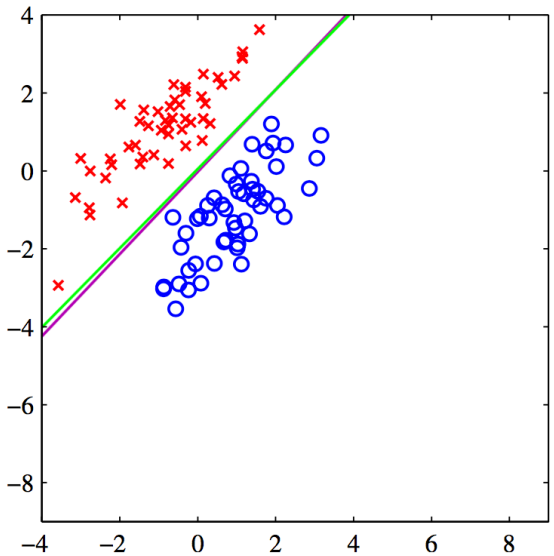
- **Decision boundary** defined by hyperplane

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

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

Linear Classifier with Least Squares?

Image credit: Bishop

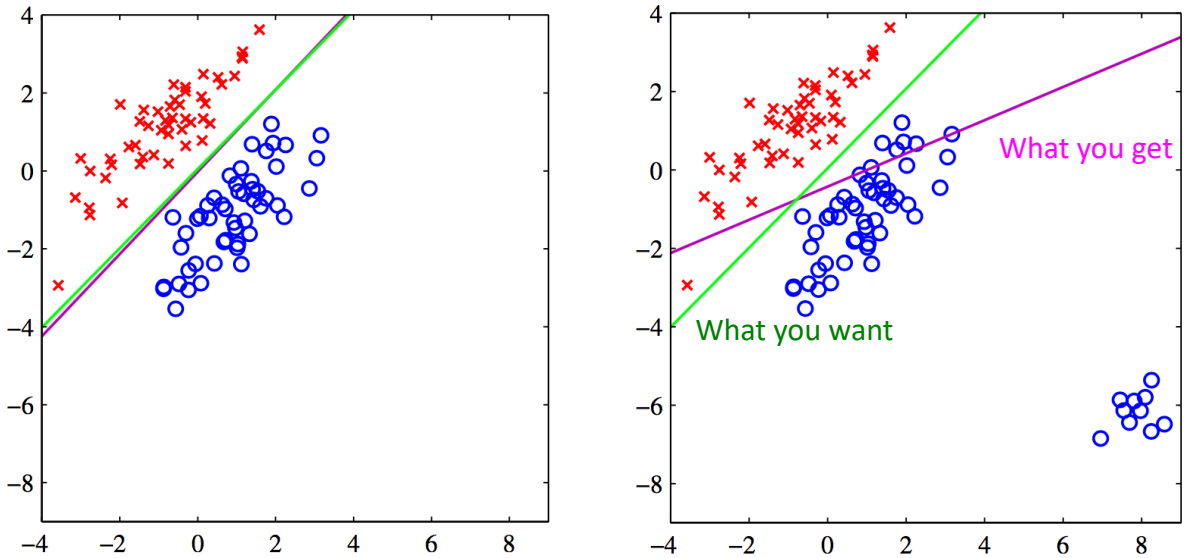


$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

- Why not use least squares loss with binary targets?

Linear Classifier with Least Squares?

Image credit: Bishop



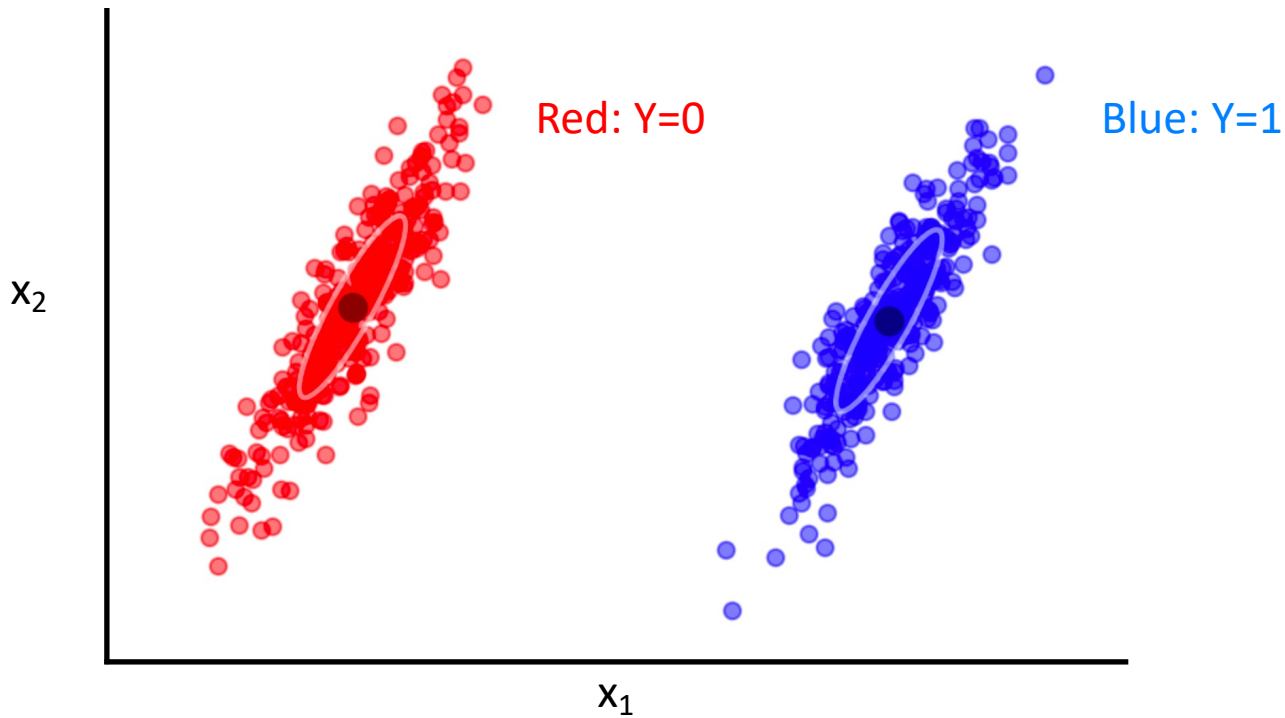
$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

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



Linear Discriminant Analysis

- Goal: Separate data from two classes / populations

- Data from joint distribution $(\boldsymbol{x}, y) \sim p(\boldsymbol{X}, Y)$
  - Features: $\boldsymbol{x} \in \mathbb{R}^m$
  - 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 $(\boldsymbol{x}, y) \sim p(\boldsymbol{X}, Y)$
  - Features: $\boldsymbol{x} \in \mathbb{R}^m$
  - 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 of a point **x**

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

Want to build classifier to predict label y given input **x**

Predicting the Class

- Separating classes → Predict the class of a point **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 **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 **x**

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

Bayes Rule

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

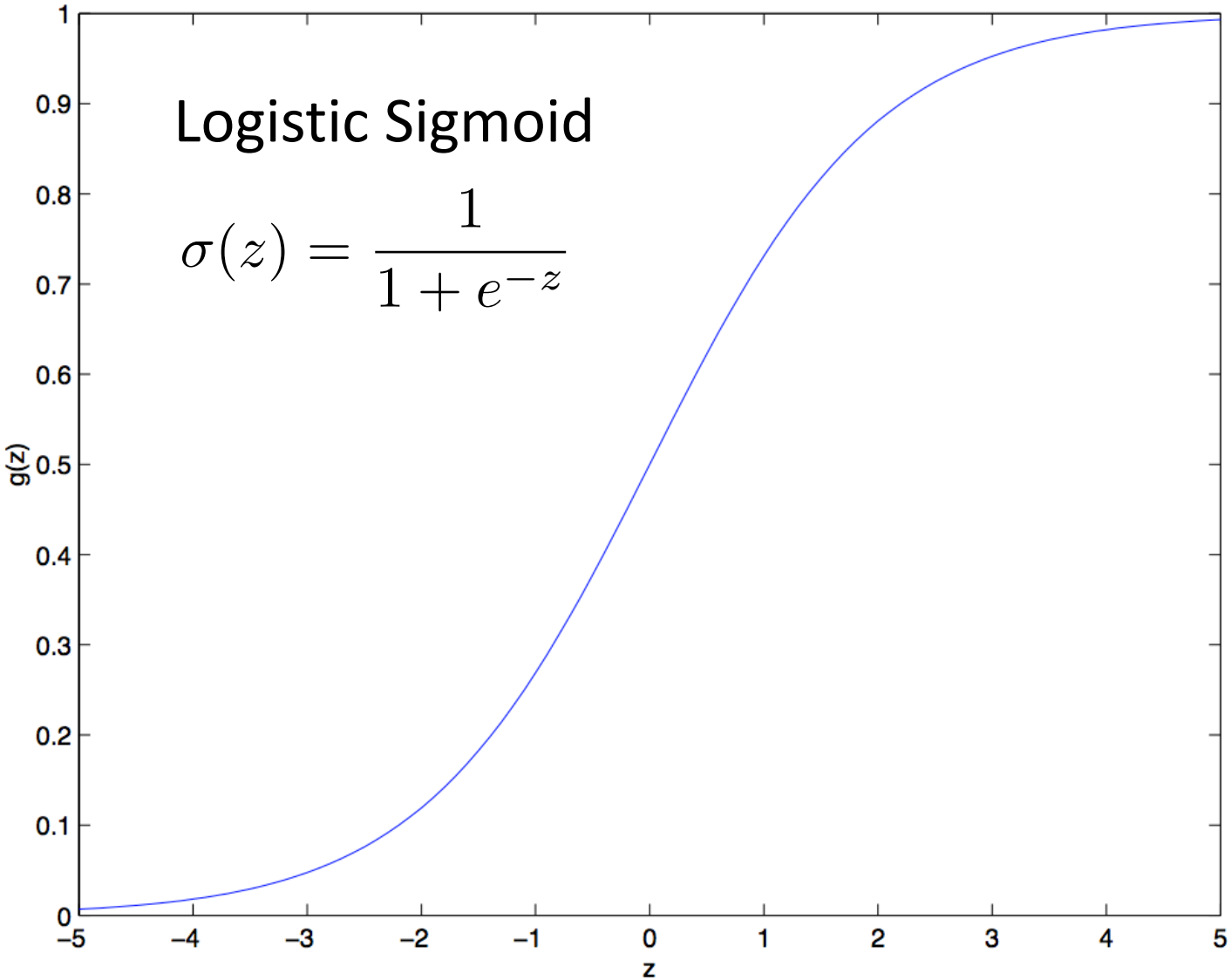
Marginal definition

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

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

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:

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

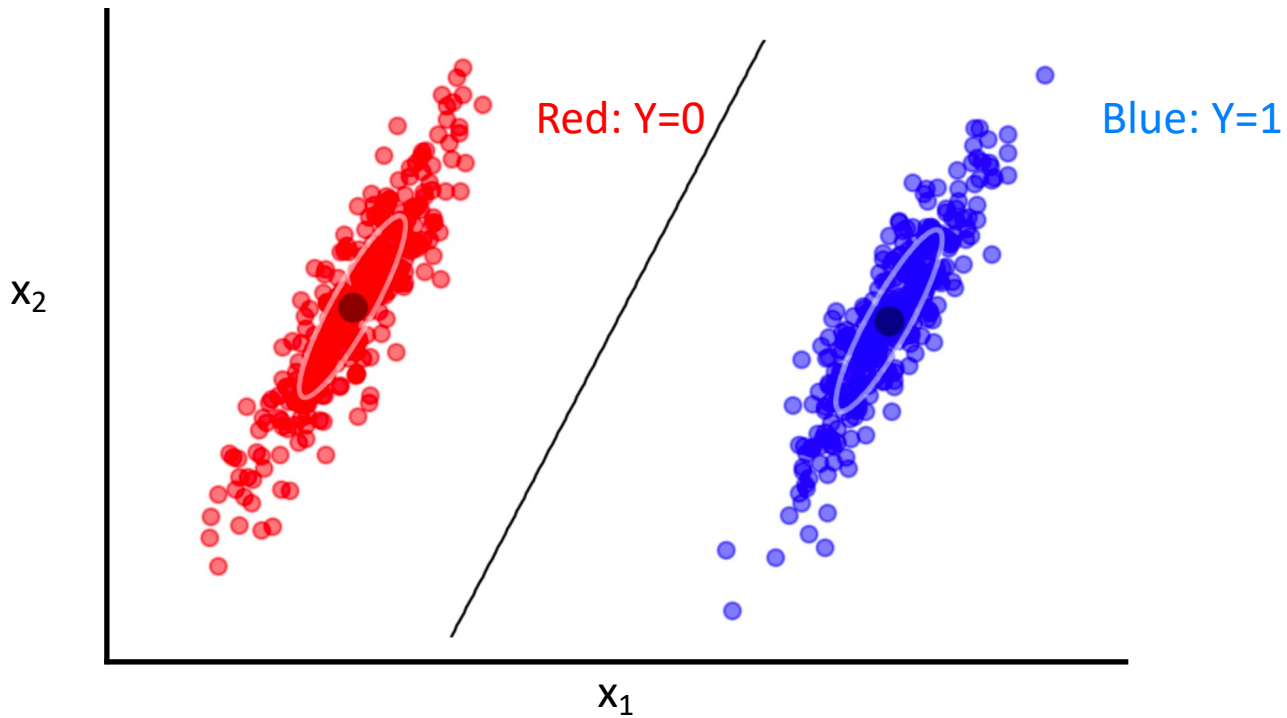
$$= \sigma\Big(-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1) + \frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0) + const.\Big)$$

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

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

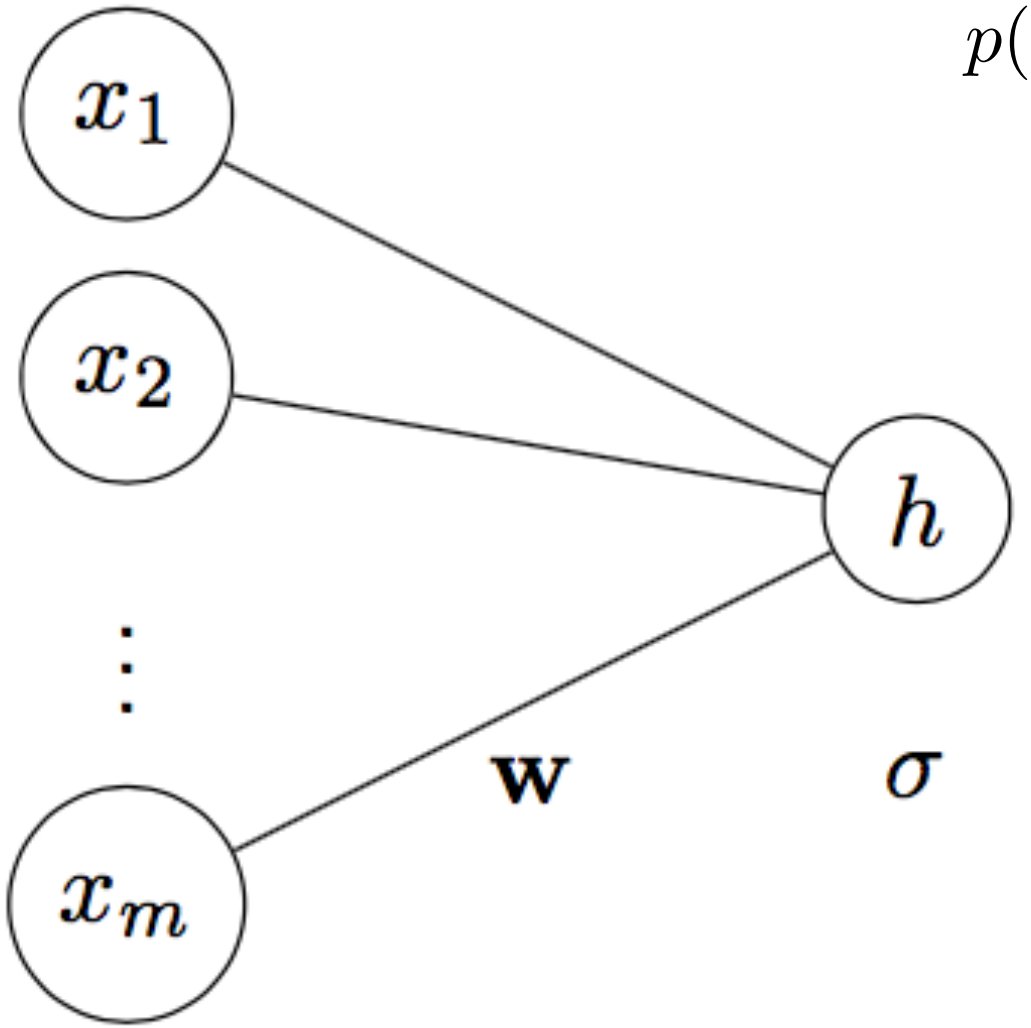
- What if we ignore Gaussian assumption on data?

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

- Farther from boundary $\boldsymbol{w}^T\boldsymbol{x} + b = 0$, more certain about class

- Sigmoid converts distance to class 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

- What if we ignore Gaussian assumption on data?

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

- With $p_i \equiv p(y_i = y|\boldsymbol{x}_i)$

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

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

Logistic Regression

- Negative log-likelihood

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

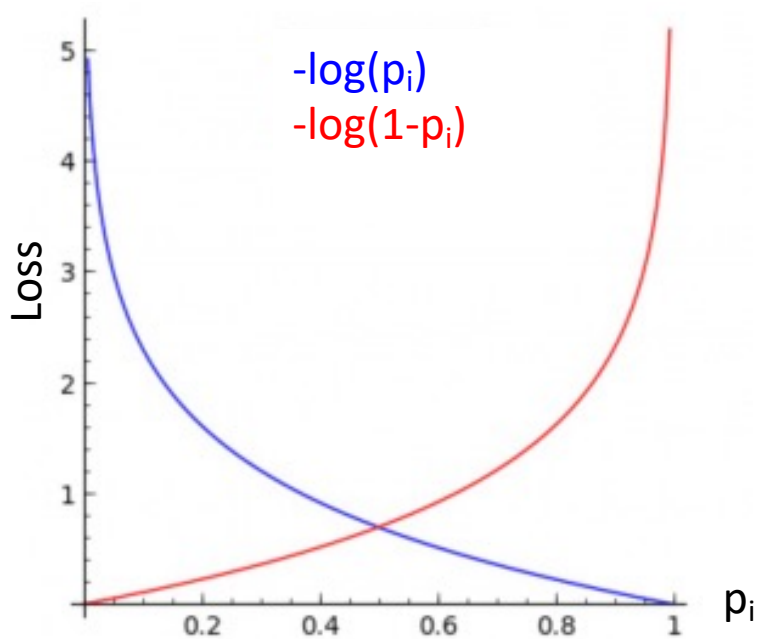
Logistic Regression

- 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

$$
\begin{aligned}
-\ln \mathcal{L} &= -\ln \prod_i (p_i)^{y_i} (1-p_i)^{1-y_i} \\
&= -\sum_i y_i \ln(p_i) + (1-y_i)\ln(1-p_i) \\
&= \sum_i y_i \ln(1+e^{-\mathbf{w}^T\mathbf{x}}) + (1-y_i)\ln(1+e^{\mathbf{w}^T\mathbf{x}})
\end{aligned}
$$

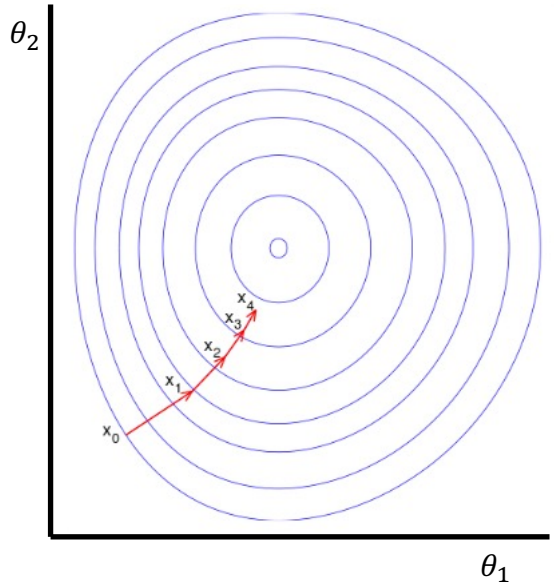
binary cross entropy loss function!

- 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



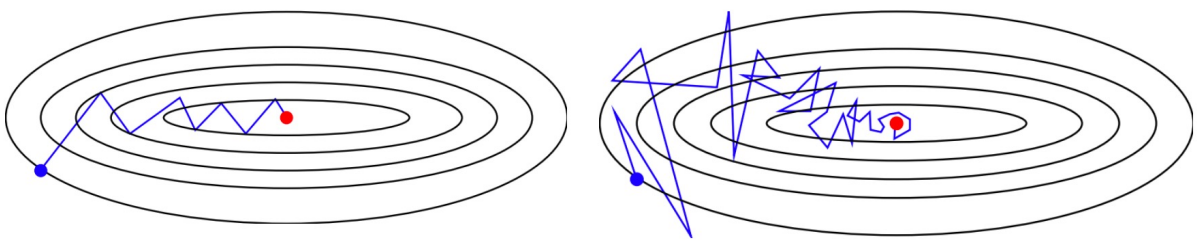
Stochastic Gradient Descent

- 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



Batch gradient descent *Stochastic gradient descent*

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

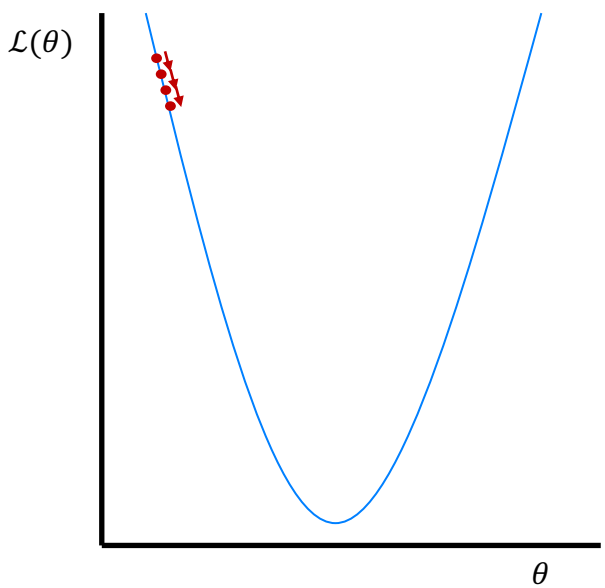
- Several updates to SGD, like momentum, ADAM, RMSprop to
  - Help to speed up optimization in flat regions of loss
  - Have adaptive learning rate
  - Learning rate adapted for each parameter
  - …

Step Sizes

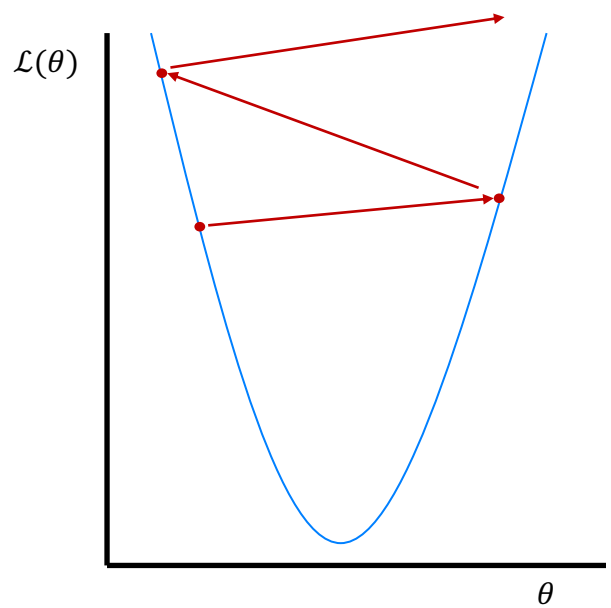
- Too small a learning rate, convergence very slow

- Too large a learning rate, algorithm diverges

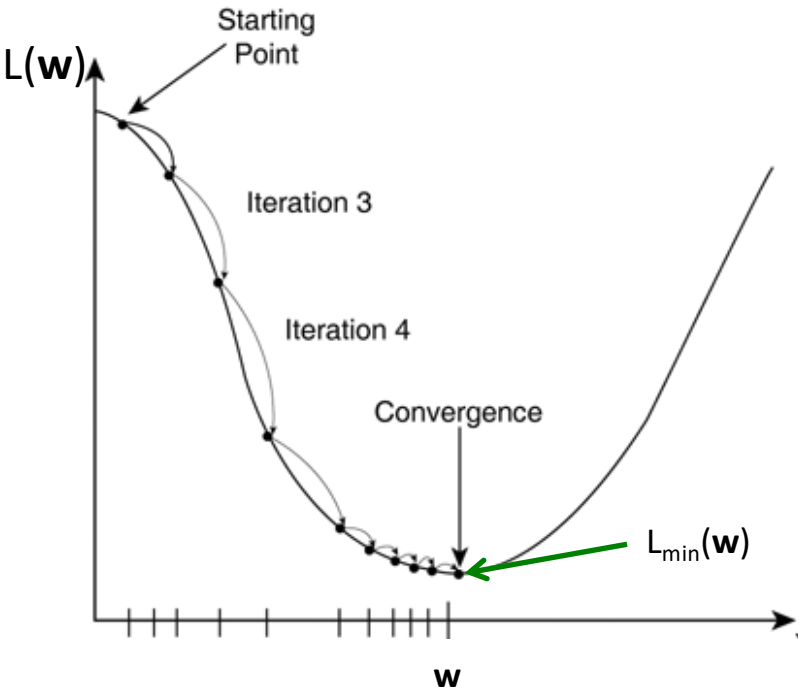
Small Learning rate

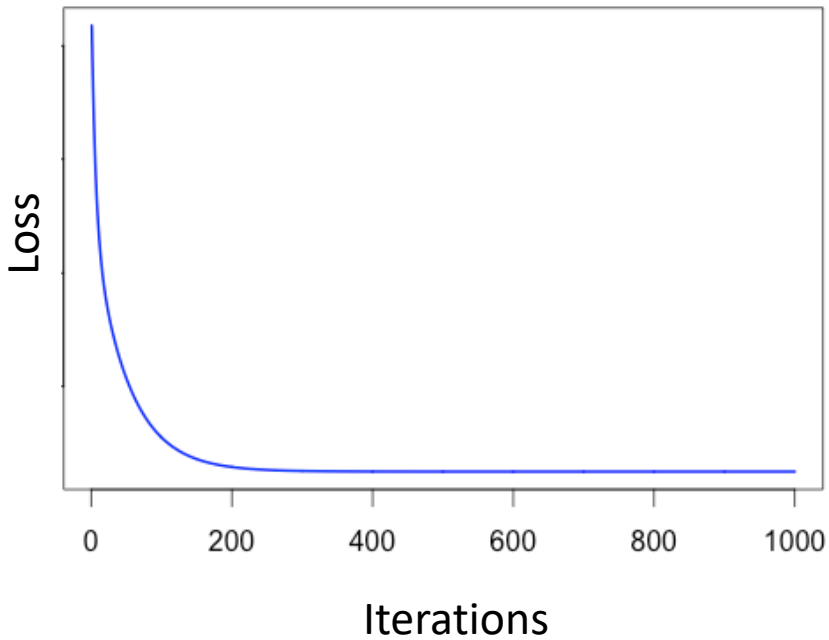


Large Learning rate



Gradient Descent

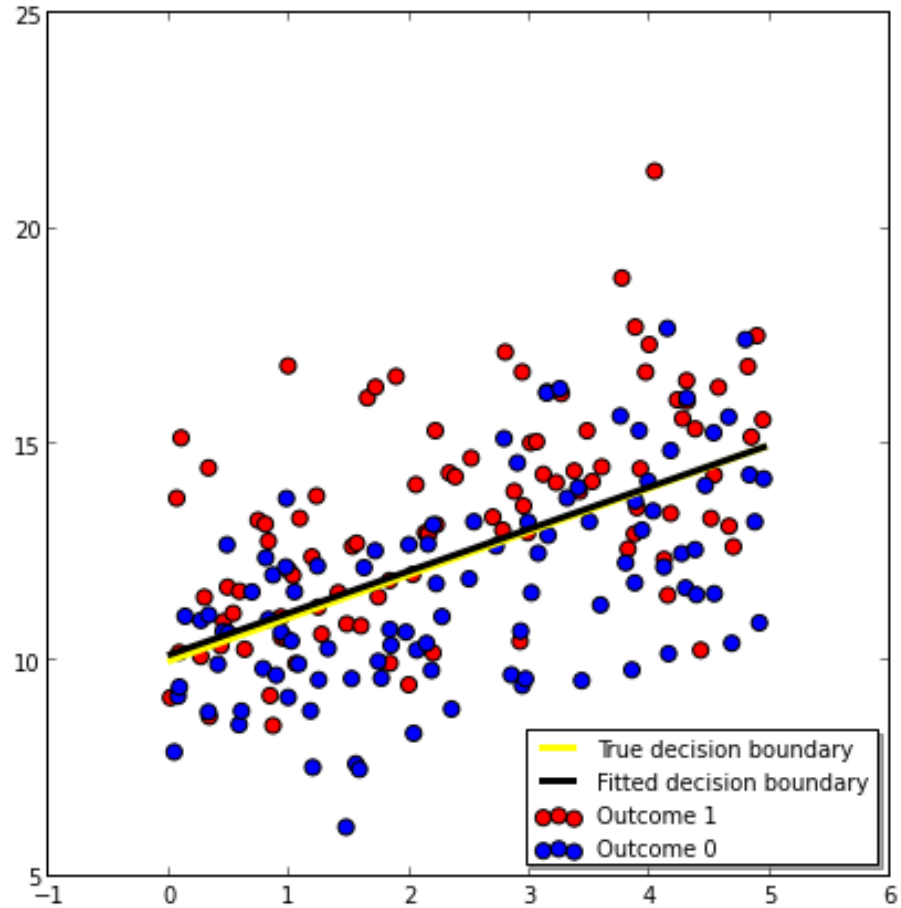




- Logistic Regression Loss is convex
  - Single global minimum

- Iterations lower loss and move toward minimum

Logistic Regression Example



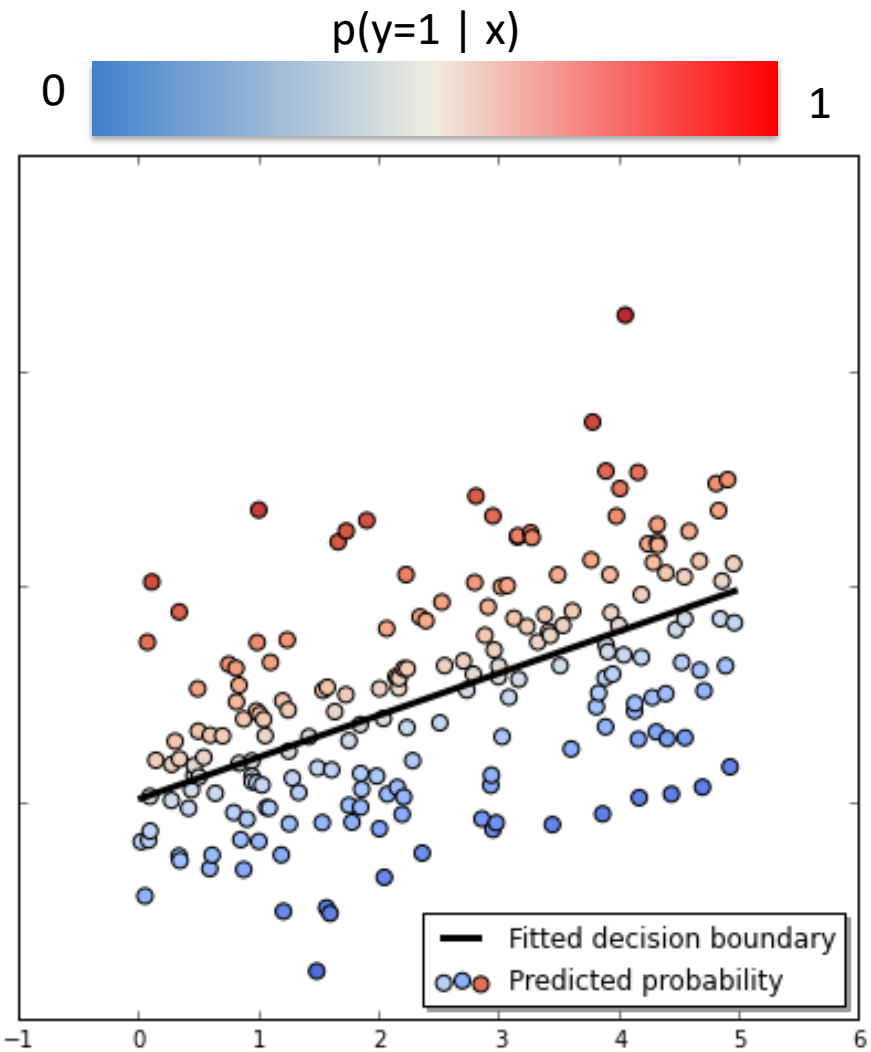
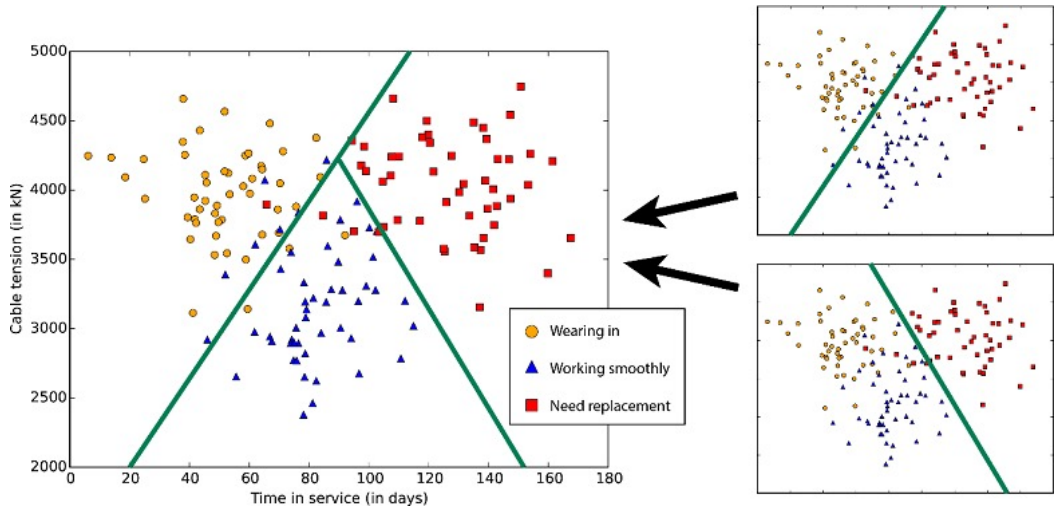


Image source

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 with "*one-hot*" vector $\mathbf{y}_k = (0, \ldots, 1, \ldots 0)$

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 $\mathbf{w}_k$

Summary

- Machine learning uses mathematical & 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 is a powerful toolkit to analyze data
  - Linear methods can help greatly in understanding data
  - What about bias and variance of models?

Backup

Notation

- $\mathbf{X} \in \mathbb{R}^{mxn}$ Matrices in bold upper case:
- $\mathbf{x} \in \mathbb{R}^{n(x1)}$ Vectors in bold lower case
- $x \in \mathbb{R}$ Scalars in lower case, non-bold
- $\mathcal{X}$ Sets are script
- $\{\mathbf{x}_i\}_1^m$ Sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$
- $y \in \mathbb{I}^{(k)} / \mathbb{R}^{(k)}$ Labels represented as

  - Integer for classes, often {0,1}. E.g. {Higgs, Z}
  - Real number. E.g electron energy

- Variables = features = inputs
- Data point $\mathbf{x} = \{x_1, \ldots, x_n\}$ has n-features

- Typically use affine coordinates:

$$
\begin{aligned}
y = \mathbf{w}^T\mathbf{x} + w_0 &\rightarrow \quad \mathbf{w}^T\mathbf{x} \\
&\rightarrow \quad \mathbf{w} = \{w_0, w_1, \ldots, w_n\} \\
&\rightarrow \quad \mathbf{x} = \{1, \ x_1, \ldots, \ x_n\}
\end{aligned}
$$

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*** η to reduce loss

- How does loss change in different directions?

  Let λ 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: $v = -\nabla_\theta \mathcal{L}(\theta)$