Introduction to Machine Learning: Lecture I

Michael Kagan

SLAC

INFN School of Statistics 2022

May 19, 2022

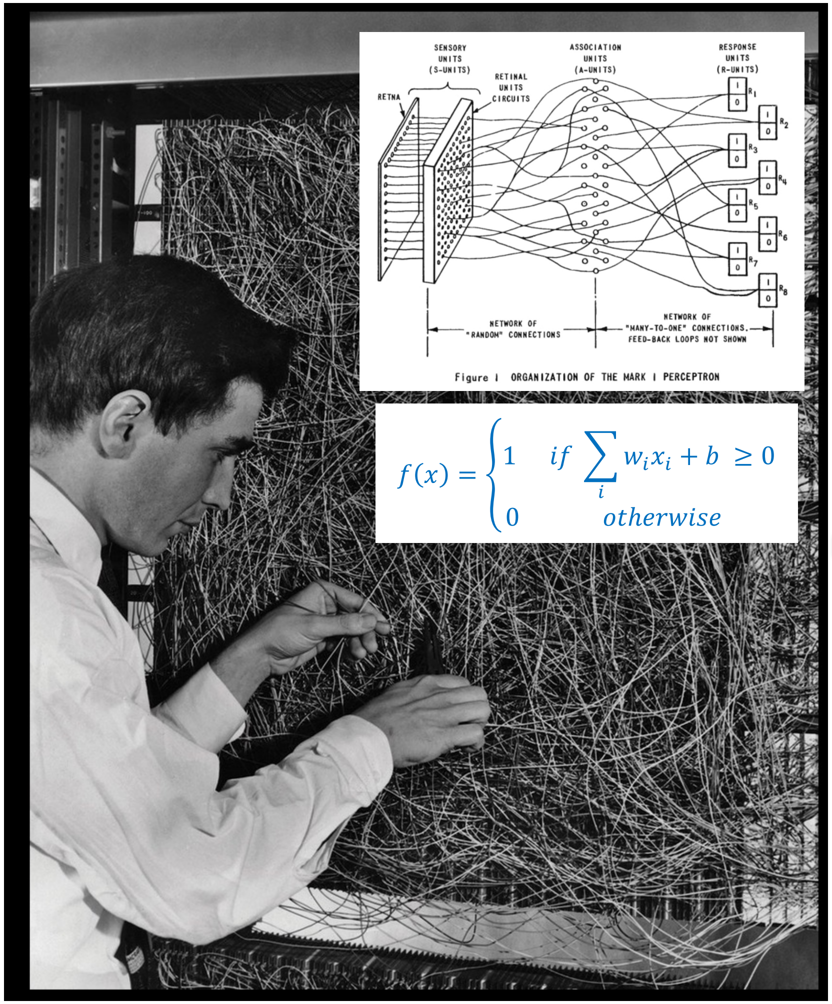
The Plan

- Lecture 1
  - Introduction to Machine Learning fundamentals
  - Linear Models

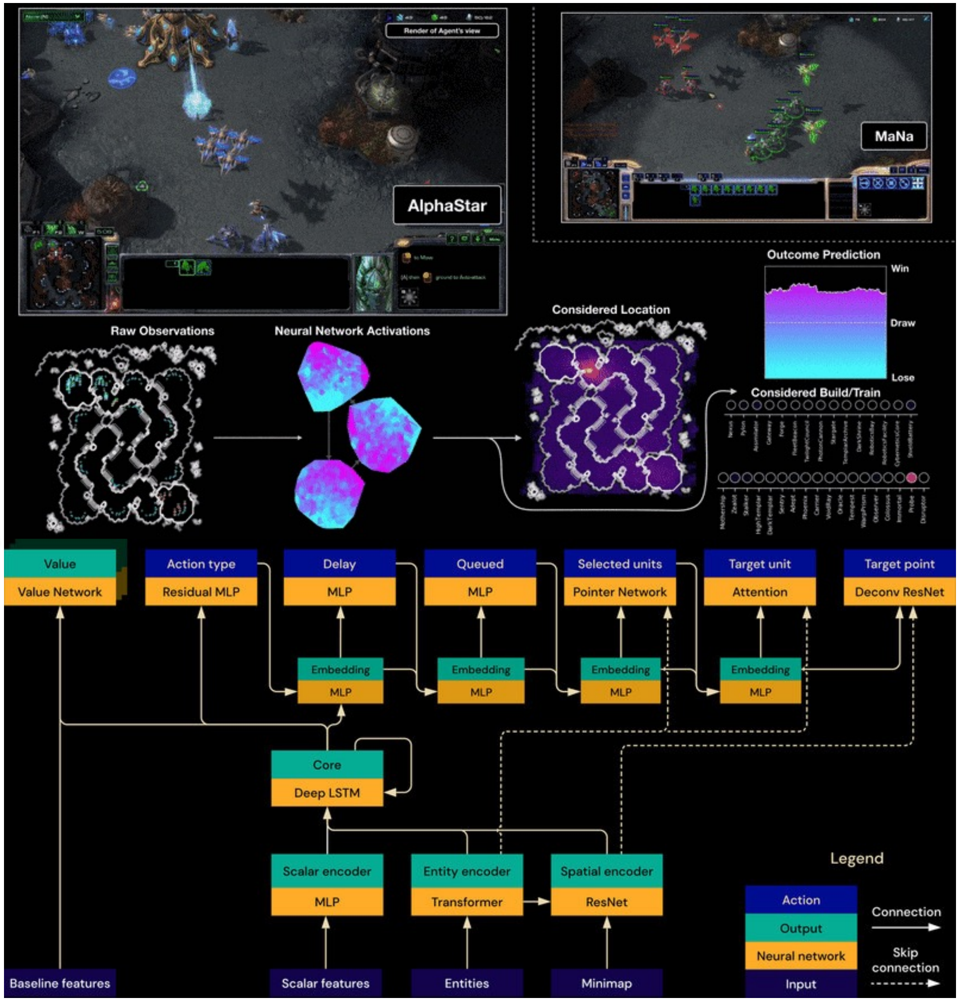
- Lecture 2
  - Neural Networks
  - Deep Neural Networks
  - Convolutional, Recurrent, and Graph Neural Networks

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

Long History of Machine Learning


SENSORY UNITS (S-UNITS)
RETINA
RETINAL UNITS CIRCUITS
ASSOCIATION UNITS (A-UNITS)
RESPONSE UNITS (R-UNITS)
R_1
R_2
R_3
R_4
R_5
R_6
R_7
R_8
NETWORK OF "RANDOM" CONNECTIONS
NETWORK OF "MANY-TO-ONE" CONNECTIONS, FEED-BACK LOOPS NOT SHOWN
Figure 1 ORGANIZATION OF THE MARK I PERCEPTRON

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



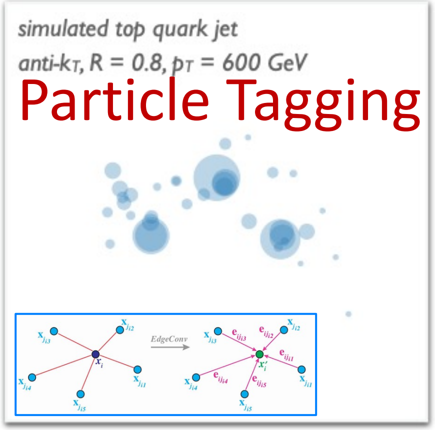
Perceptron

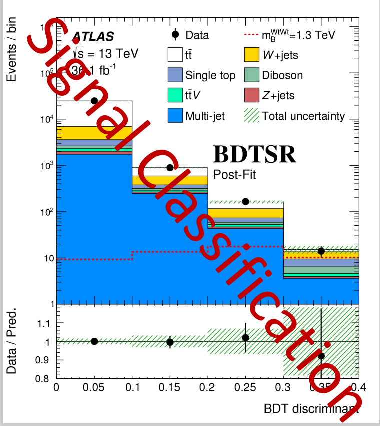
AlphaStar

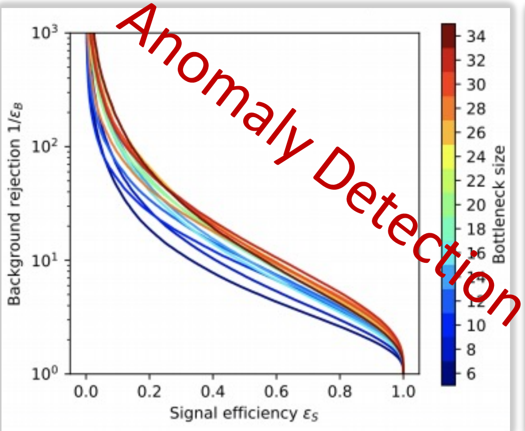
Rosenblatt 1958, 1960

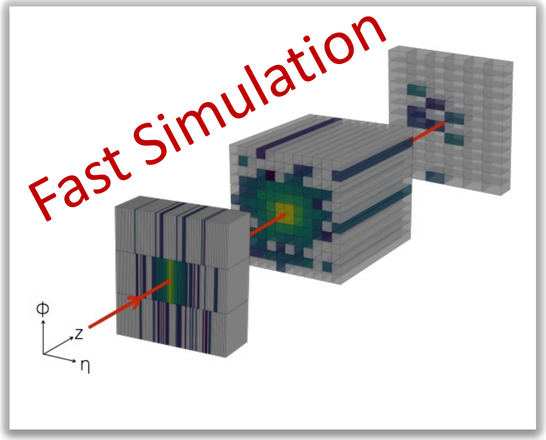
Vinyals et. al. 2019

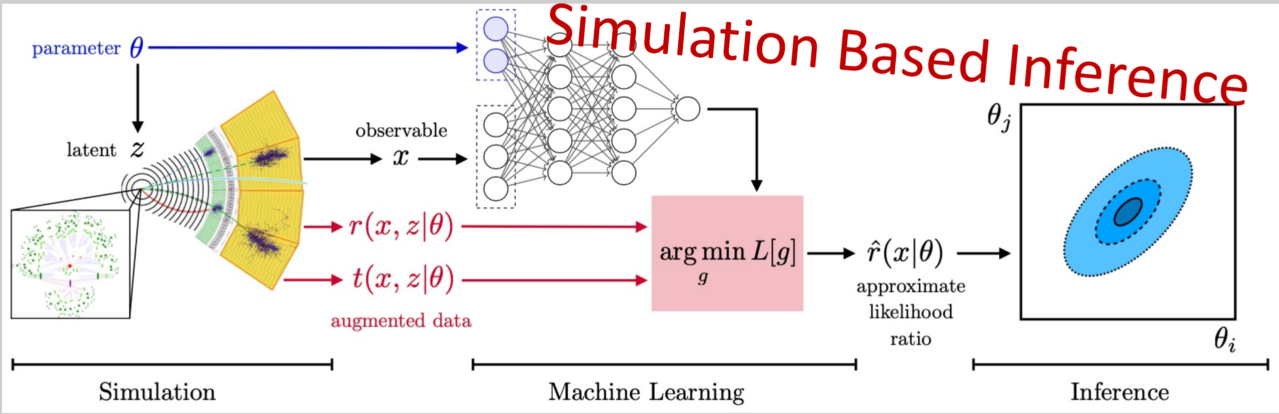
Machine Learning in HEP

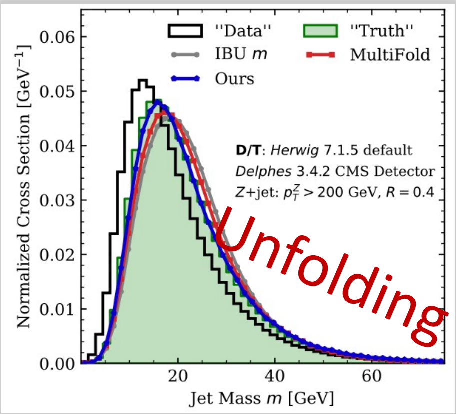

Particle Tagging

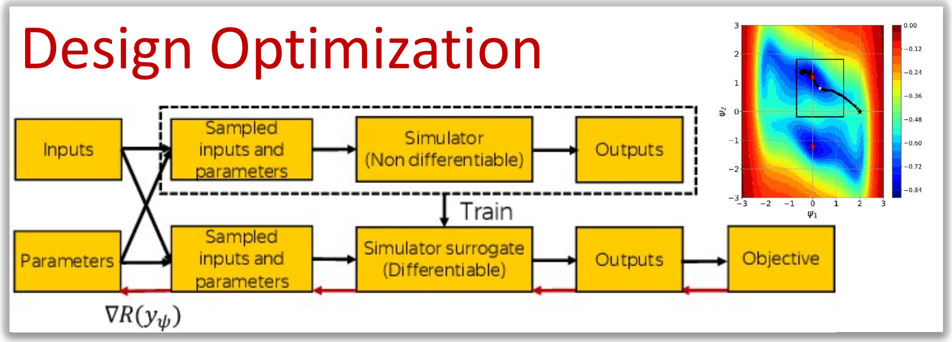

Signal Classification

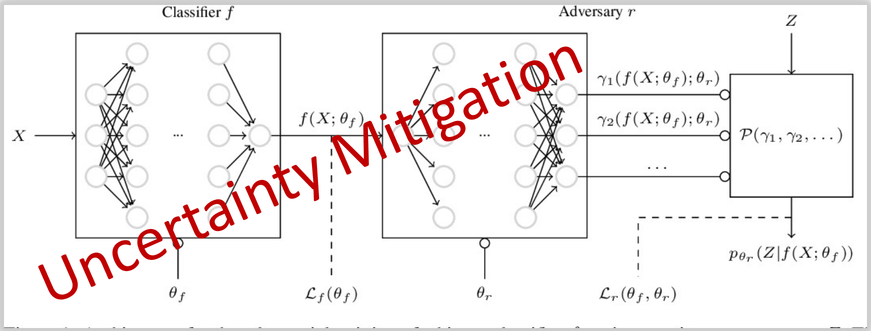

Anomaly Detection


Fast Simulation


Simulation Based Inference


Unfolding


Design Optimization


Uncertainty Mitigation

Classifier f
Adversary r
X
Z
$f(X;\theta_f)$
$\gamma_1(f(X;\theta_f);\theta_r)$
$\gamma_2(f(X;\theta_f);\theta_r)$
...
$\mathcal{P}(\gamma_1,\gamma_2,\dots)$
$p_{\theta_r}(Z|f(X;\theta_f))$
θ_f
$\mathcal{L}_f(\theta_f)$
θ_r
$\mathcal{L}_r(\theta_f,\theta_r)$

+ 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

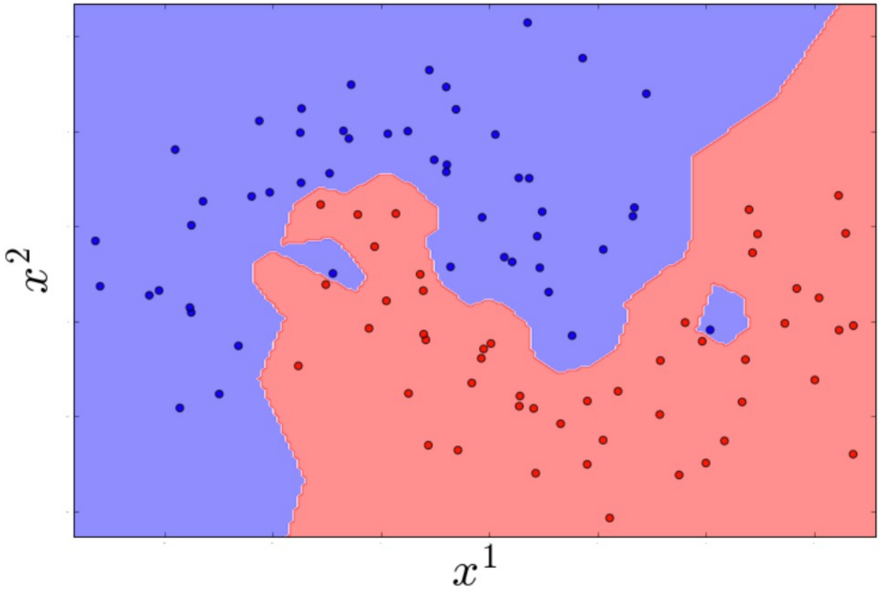
- **Fitting data with complex functions**

- **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

Classification

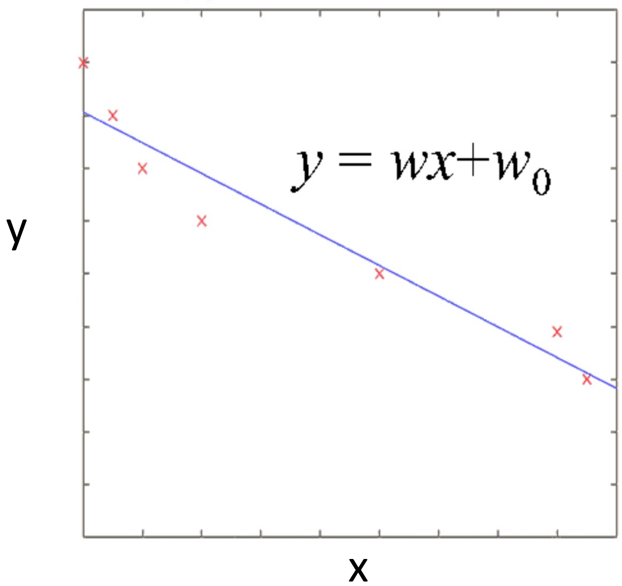


[Rogozhnikov]

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

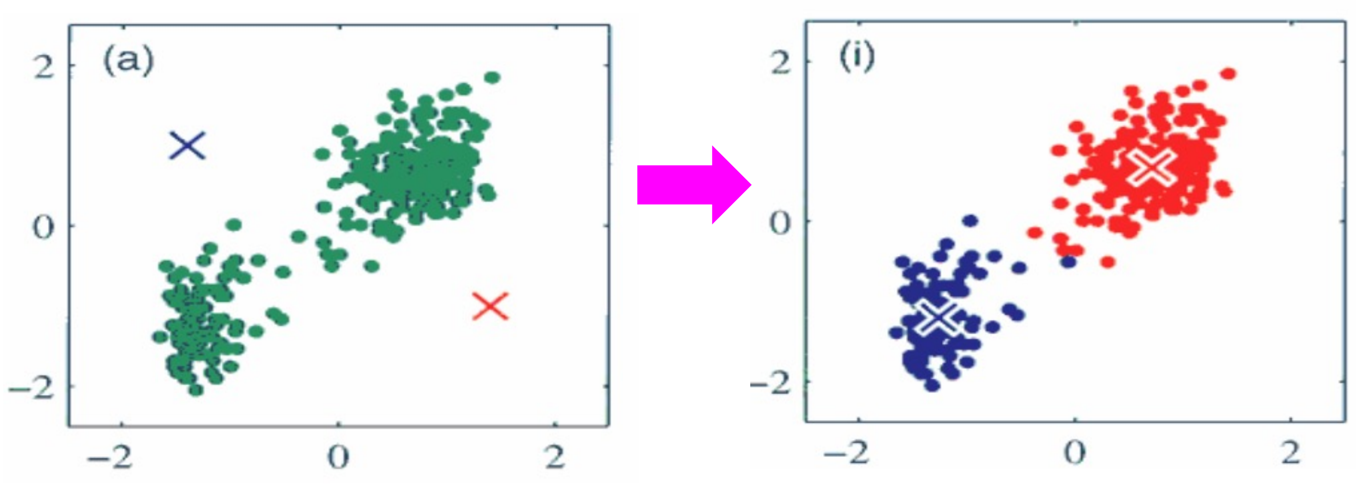
Regression



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

Clustering

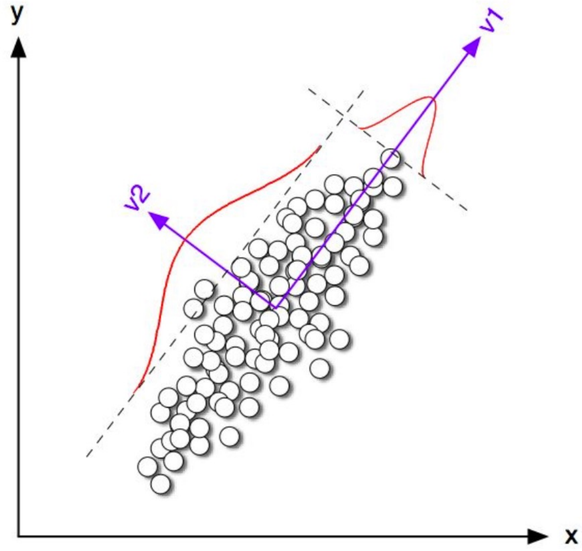


[Bishop]

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

Dimensionality Reduction

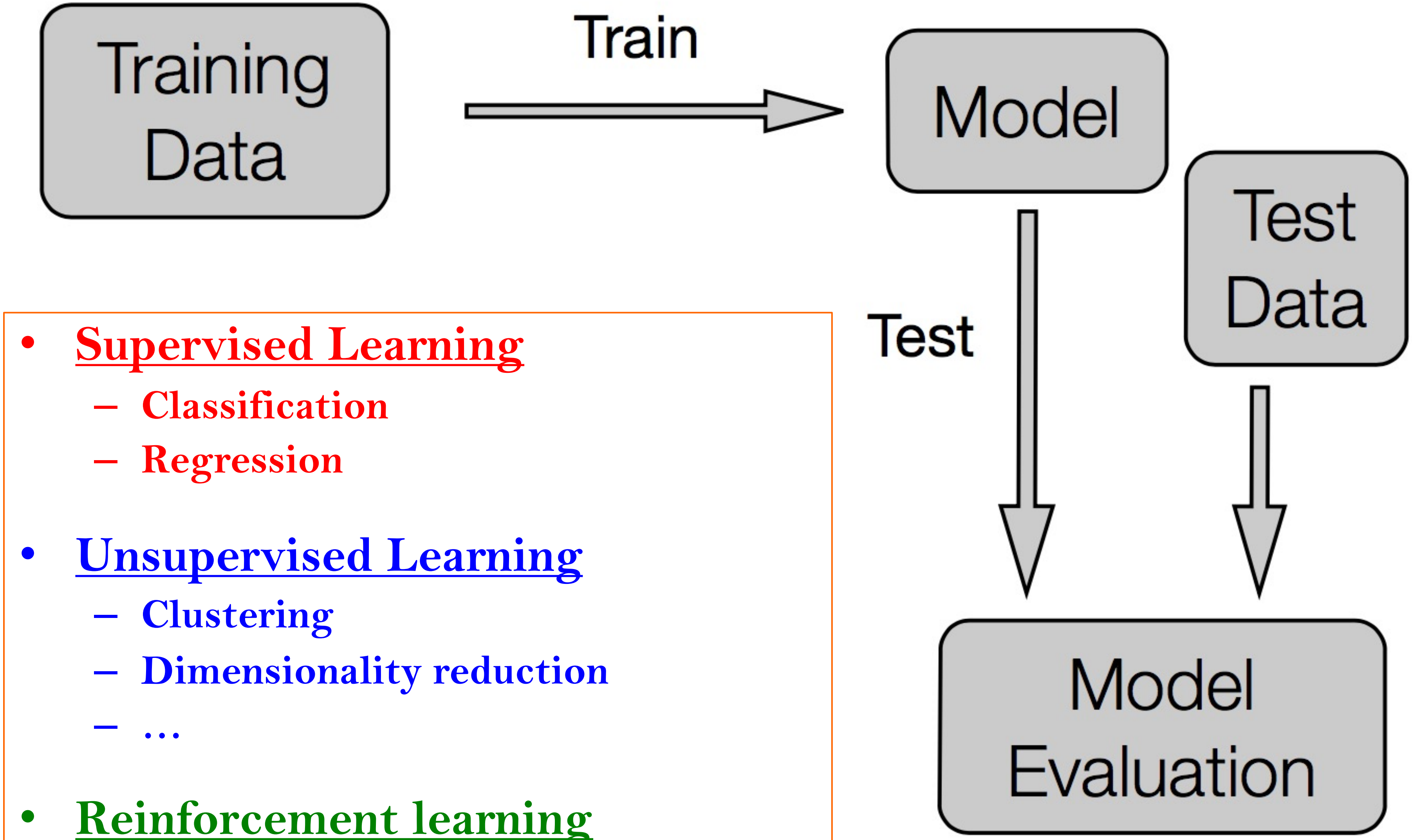


https://lazyprogrammer.me/tutorial-principal-components-analysis-pca/

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)

Learning



- **Supervised Learning**
  - **Classification**
  - **Regression**
- **Unsupervised Learning**
  - **Clustering**
  - **Dimensionality reduction**
  - **…**
- **Reinforcement learning**

[Ravikumar]

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:

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

$$\rightarrow \quad \mathbf{w} = \{w_0, w_1, \ldots, w_n\}$$

$$\rightarrow \quad \mathbf{x} = \{1, \quad x_1, \ldots, \quad x_n\}$$

Probability Review

- Joint distribution of two variables: p(x,y)

- Marginal distribution: $p(x) = \int p(x,y)dy$

- Conditional distribution: $p(y|x) = \dfrac{p(x,y)}{p(x)}$

- Bayes theorem: $p(y|x) = \dfrac{p(x|y)p(y)}{p(x)}$

- Expected value: $\mathbf{E}[f(x)] = \displaystyle\int f(x)p(x)\mathrm{d}x$

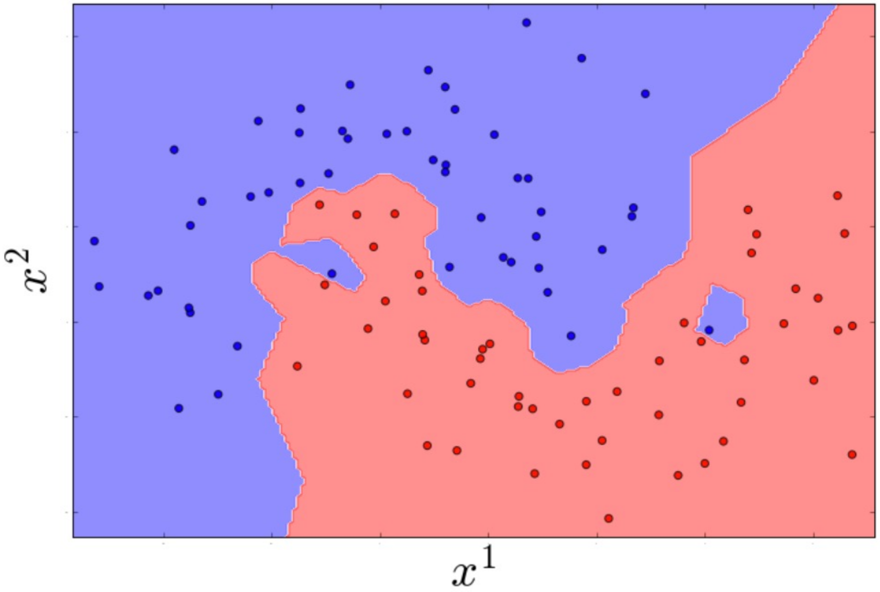
- Normal distribution:
  - x~N(μ, σ) → $p(x) = \dfrac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\dfrac{1}{2}\dfrac{(x-\mu)^2}{\sigma^2}\right)$

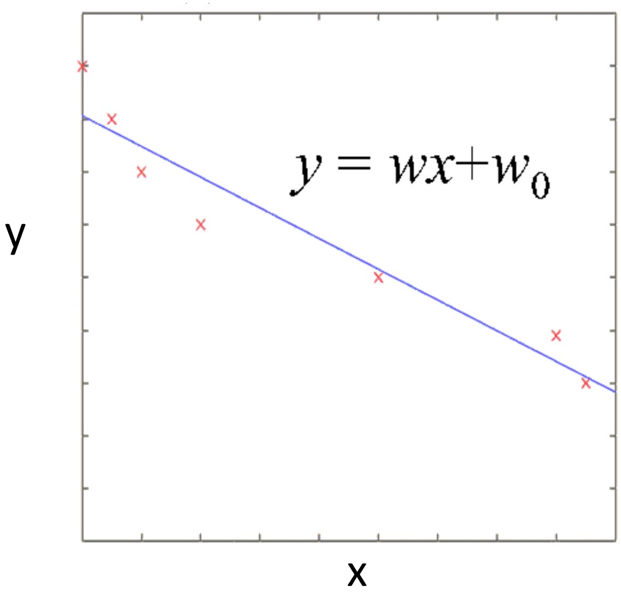
Supervised Learning

- 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

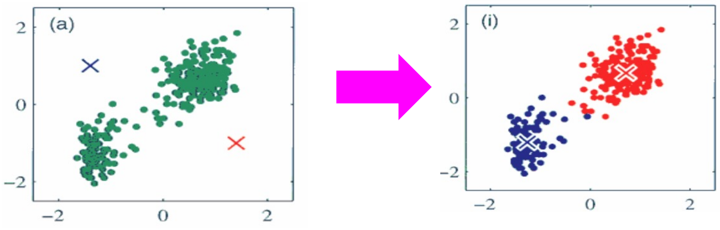




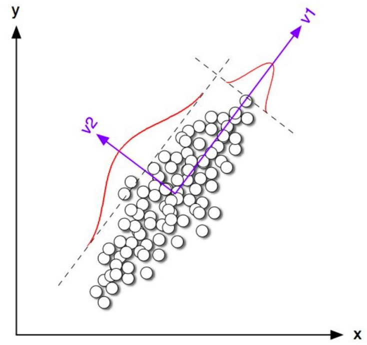
Unsupervised Learning

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

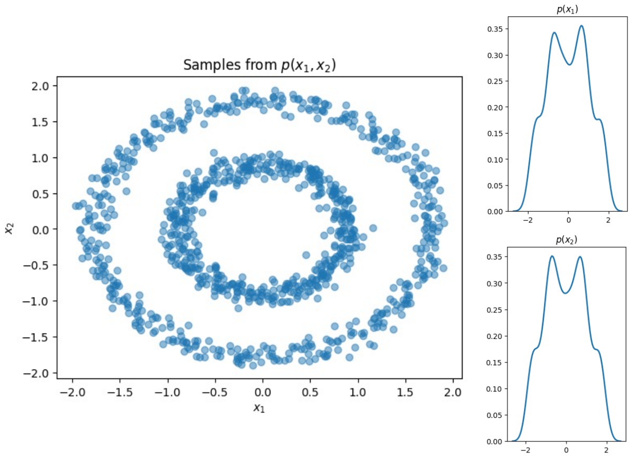
[Bishop]



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



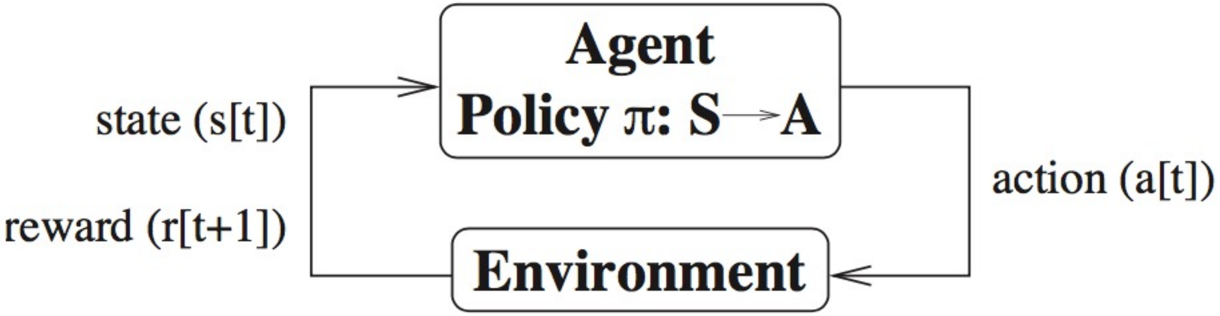
Dimensionality reduction: find a low dimensional (less complex) representation of the data with a mapping $Z=h(X)$



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

Image Credit - Link

Reinforcement Learning

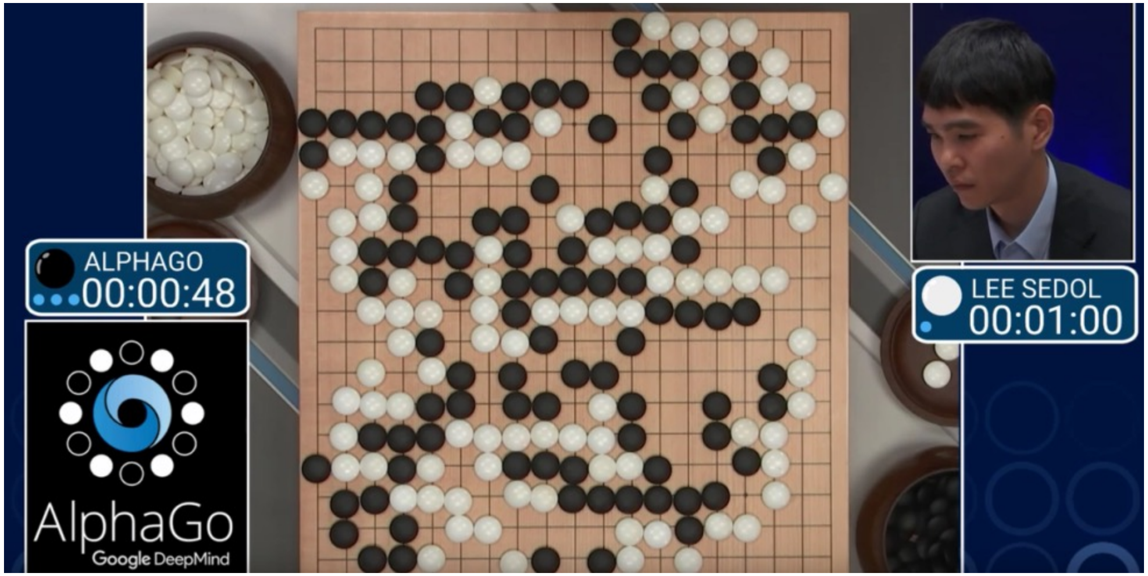


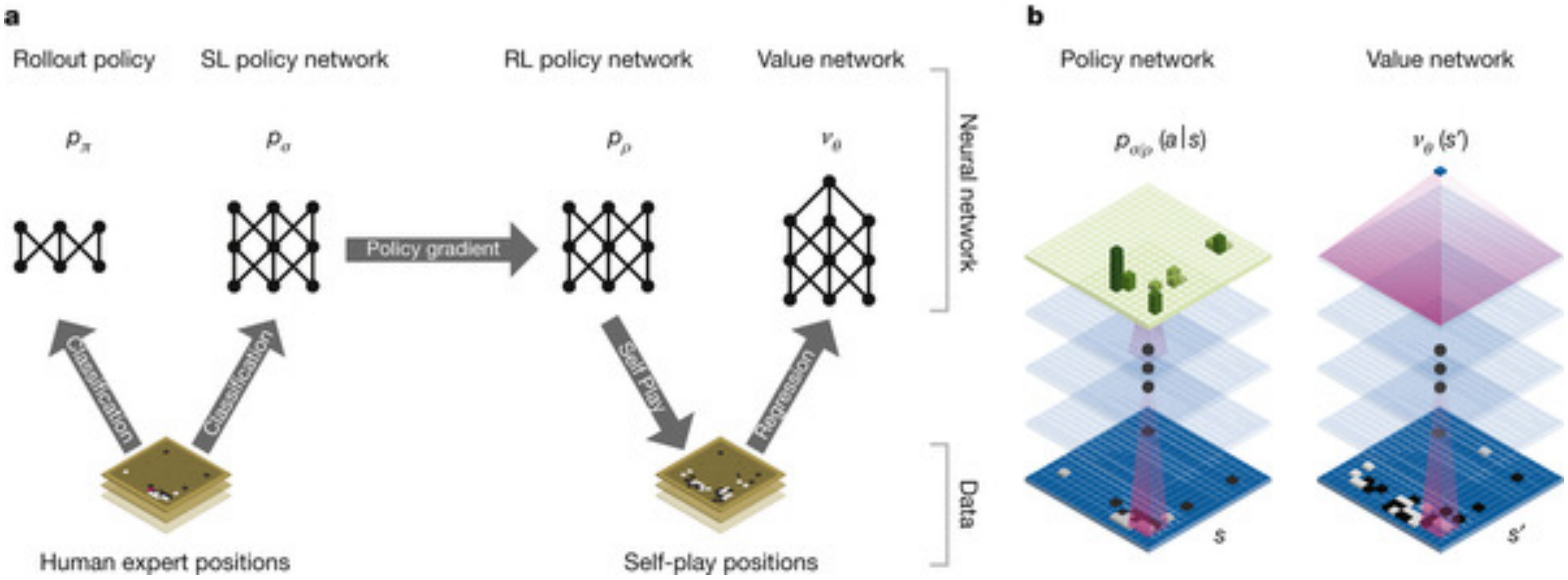
[Ravikumar]

- 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

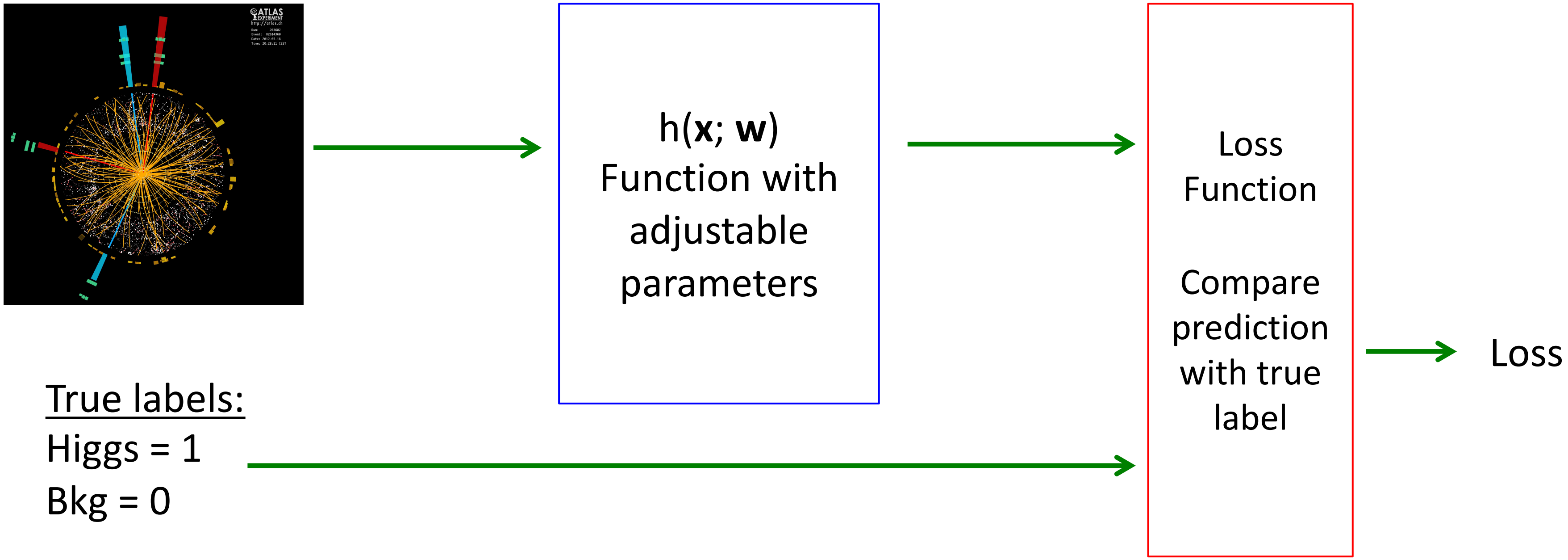




Nature 529, 484–489 (28 January 2016)

Supervised Learning: How does it work?

Supervised Learning: How does it work?



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

Loss Function

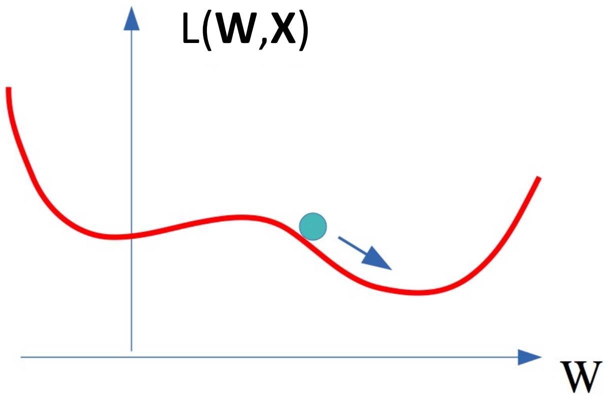
Compare prediction with true label

Loss

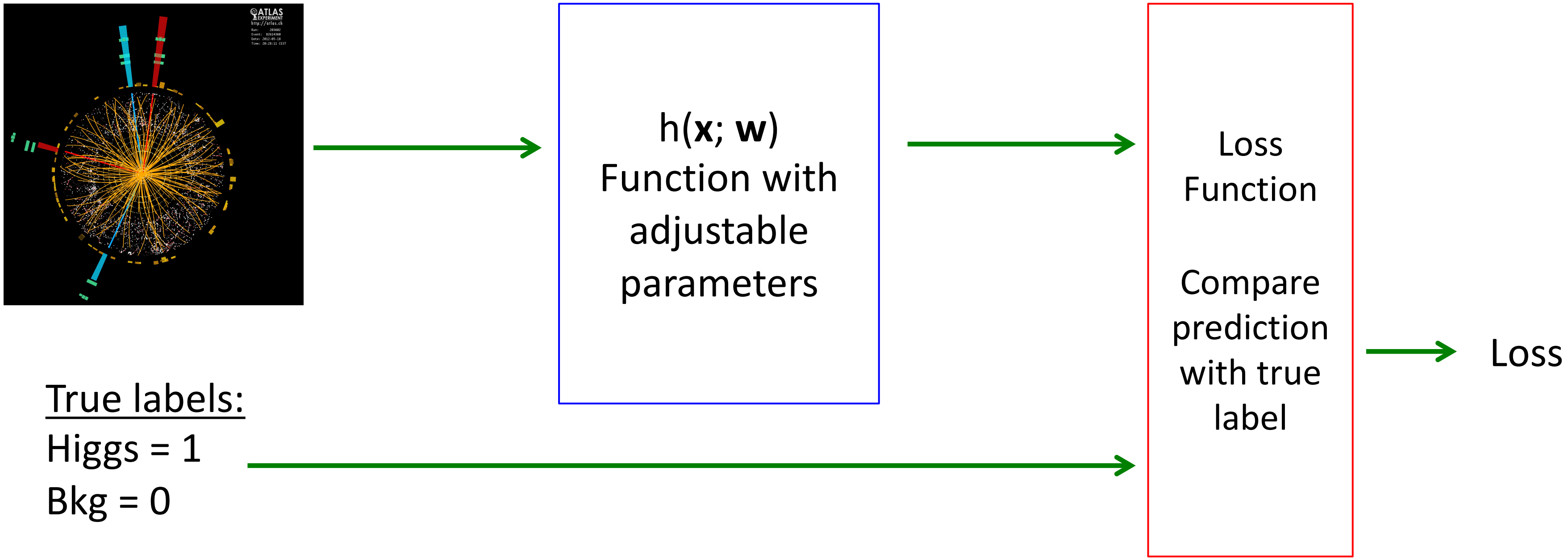
True labels:
Higgs = 1
Bkg = 0

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

Loss
Function

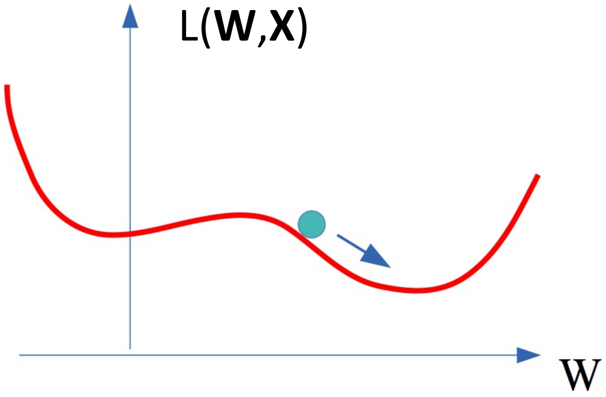
Compare
prediction
with true
label

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



Reminder: Empirical Risk Minimization

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

- Framework to design learning algorithms
  - L(·) is a loss function comparing prediction h(·) with target y
  - $\Omega(\mathbf{w})$ is a regularizer, penalizing certain values of $\mathbf{w}$
    - λ controls how much penalty… a hyperparameter we have to tune
- Learning is cast as an optimization problem

Example Loss Functions

- Square Error Loss:
  - Often used in regression

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

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

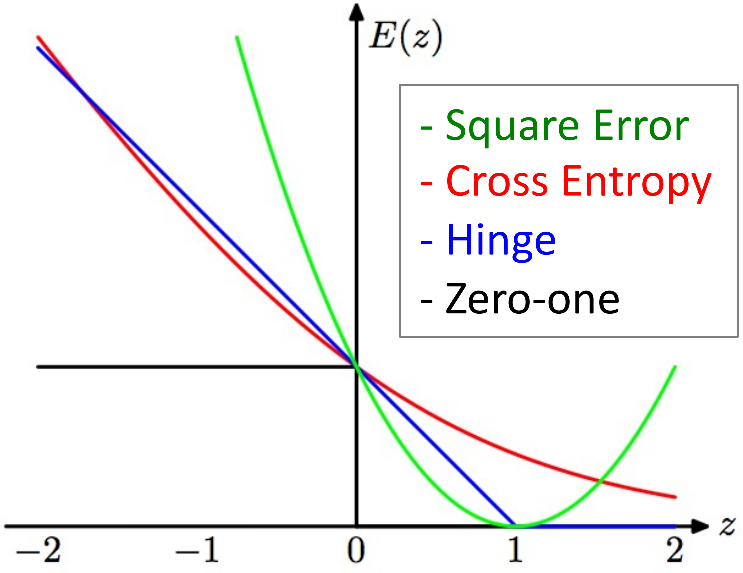
$$\begin{aligned} L(h(\mathbf{x};\mathbf{w}),y) = &-y\log h(\mathbf{x};\mathbf{w}) \\ &-(1-y)\log(1-h(\mathbf{x};\mathbf{w})) \end{aligned}$$

- 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

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

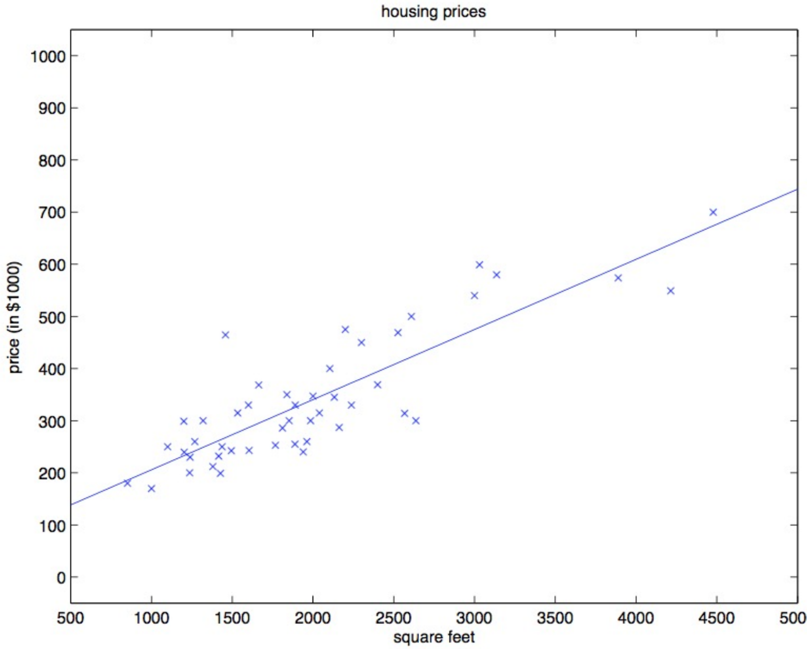


[Bishop]

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

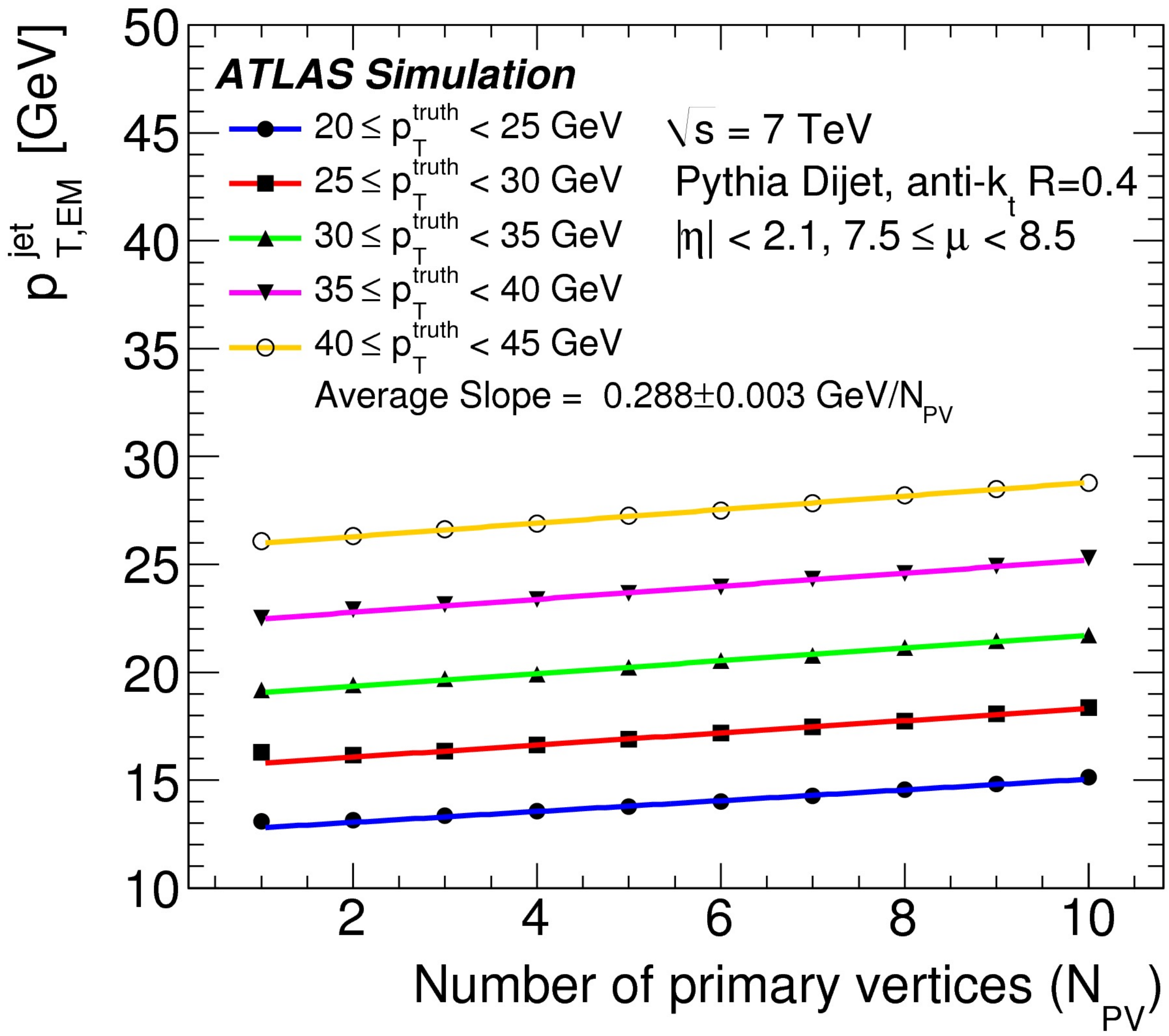
- Likelihood function:

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

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

Squared loss function!

Linear Regression Example



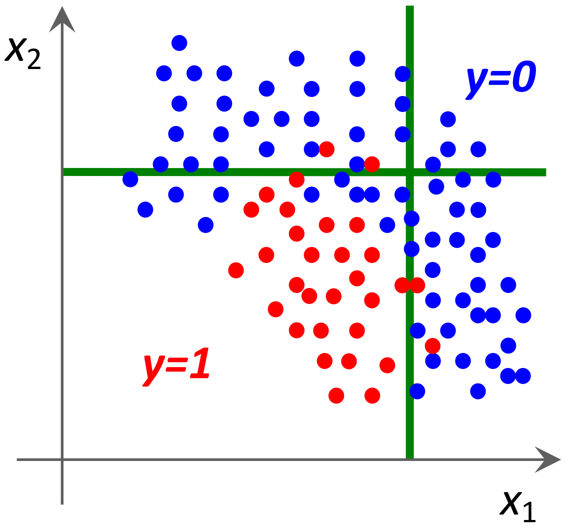
Eur. Phys. J. C (2015) 75:17

- Reconstructed Jet energy vs. Number of primary vertices

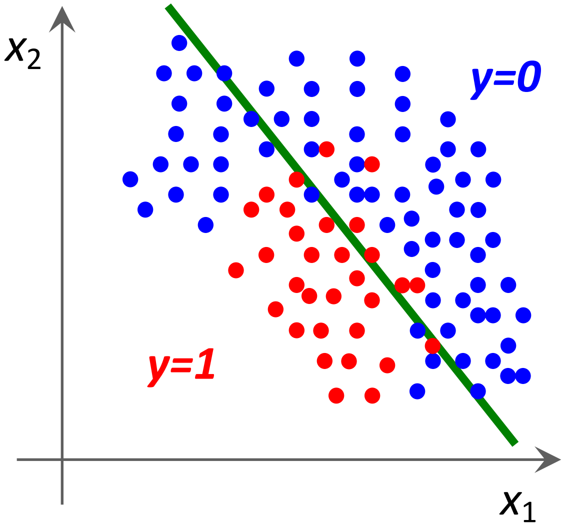
Linear Classification

Classification

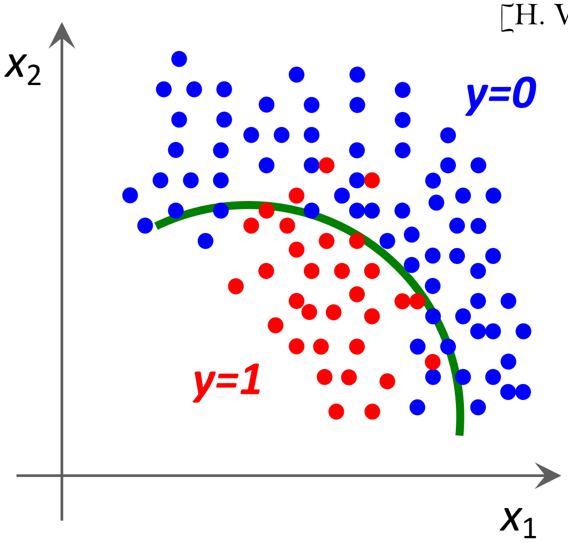
[H. Voss]



Rectangular cuts



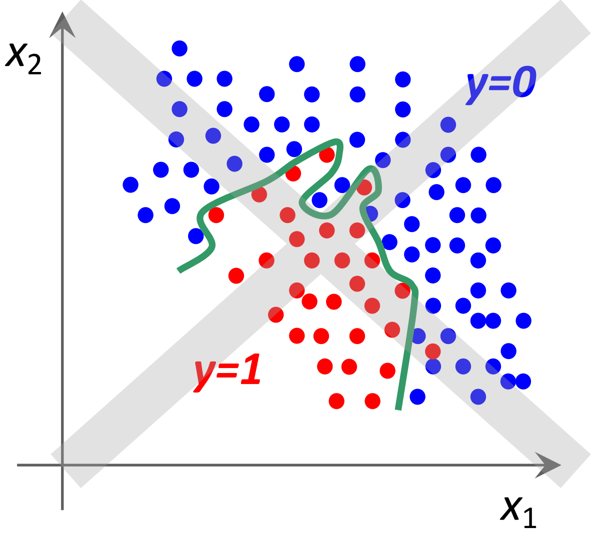
Linear discriminant



Nonlinear discriminant

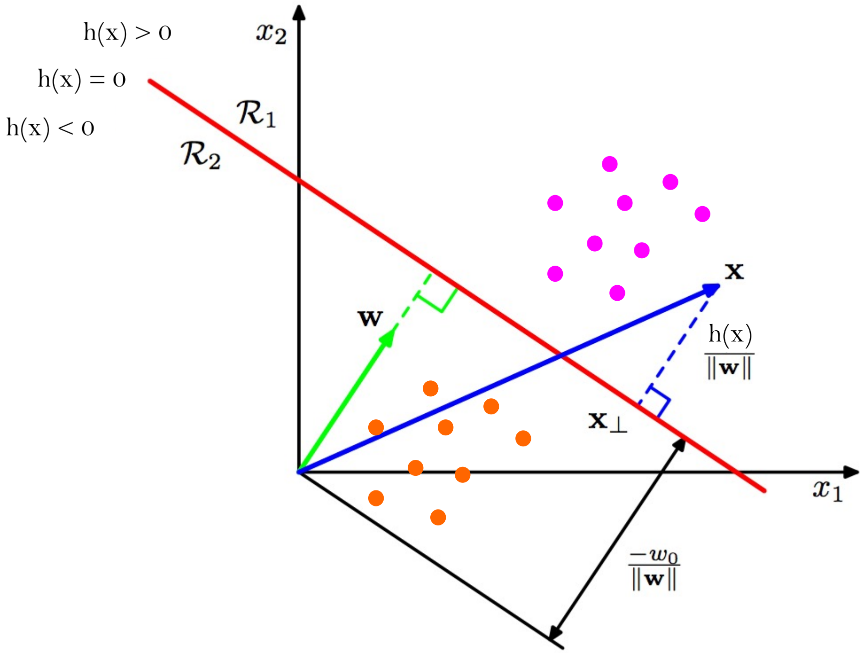
- Learn a function to separate different classes of data

- 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

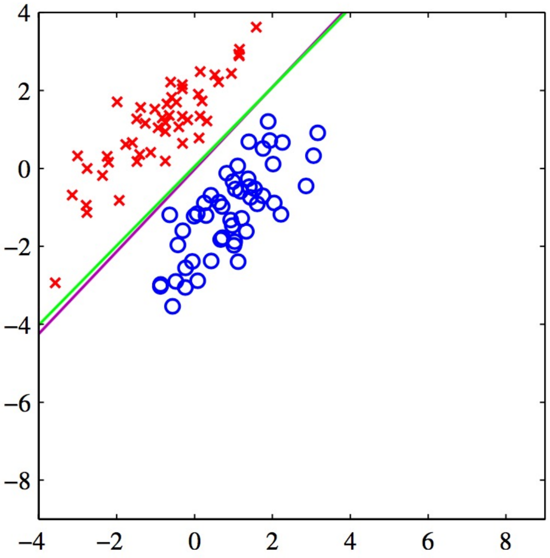
$$h(\mathbf{x};\, \mathbf{w}) = \mathbf{w}^T\mathbf{x}+b$$

- **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?

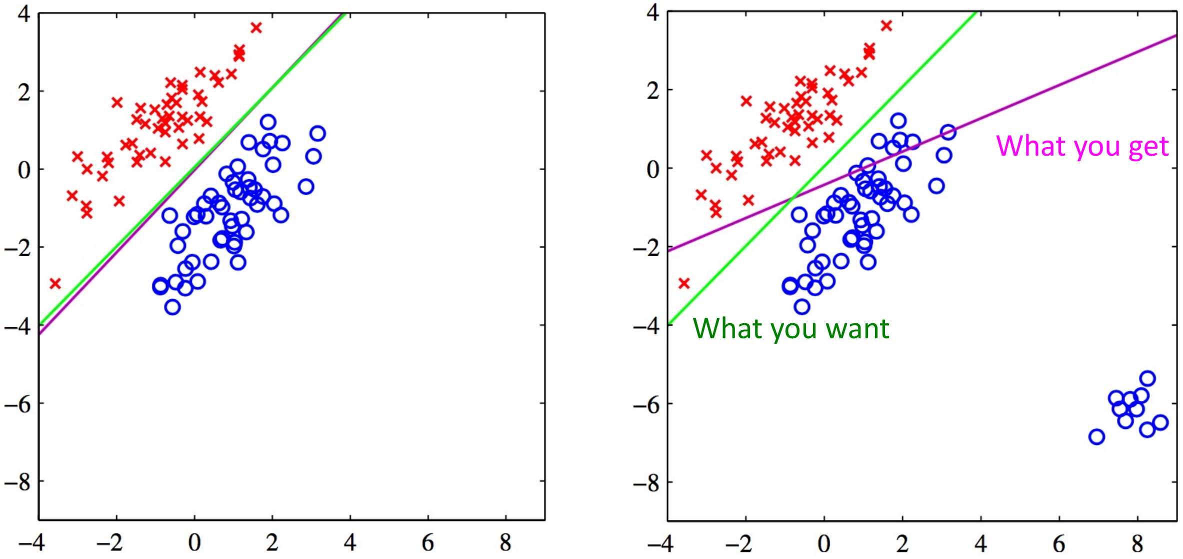


$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

[Bishop]

- Why not use least squares loss with binary targets?

Linear Classifier with Least Squares?



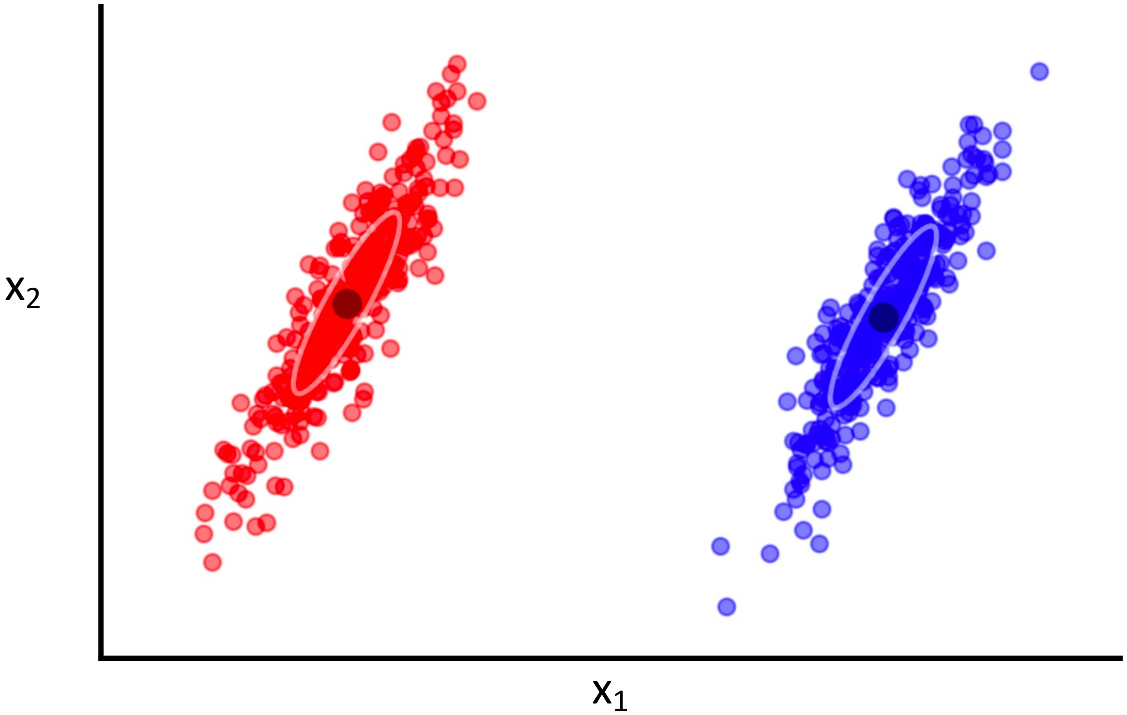
$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

[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

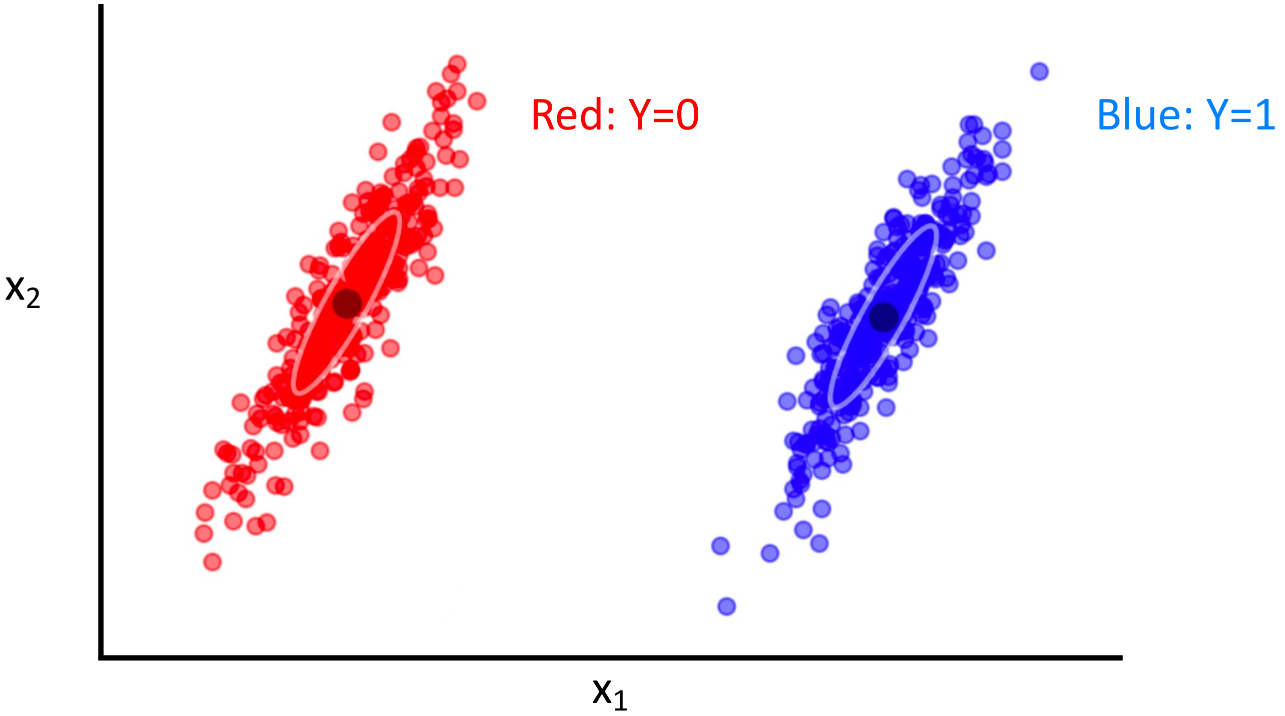
- Goal: Separate data from two classes / populations



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 (**x**, y) ~ p(**X**, Y)
  - Features: **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 $\mathbf{x}$

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

- Want to build a classifier to predict the label y given and input 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

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

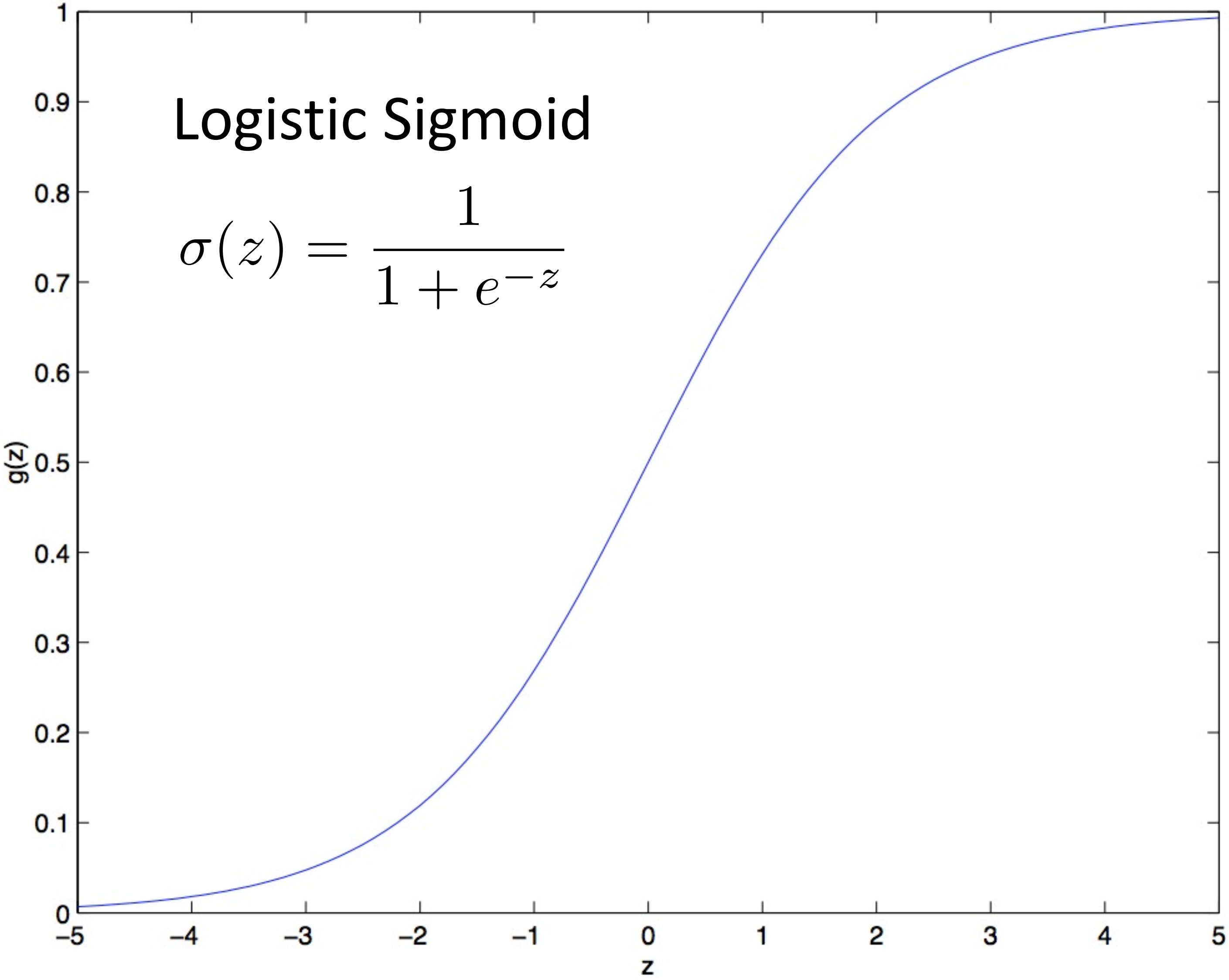
Marginal definition

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

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

Why?

Logistic Sigmoid Function



Logistic Sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Predicting Classes with Gaussian Likelihoods

$$p(y=1|\mathbf{x}) = \sigma\Big(\underbrace{\log \frac{p(\mathbf{x}|y=1)}{p(\mathbf{x}|y=0)}}_{\text{Log-likelihood ratio}} + \underbrace{\log \frac{p(y=1)}{p(y=0)}}_{\text{Constant w.r.t. } \mathbf{x}} \Big)$$

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:

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

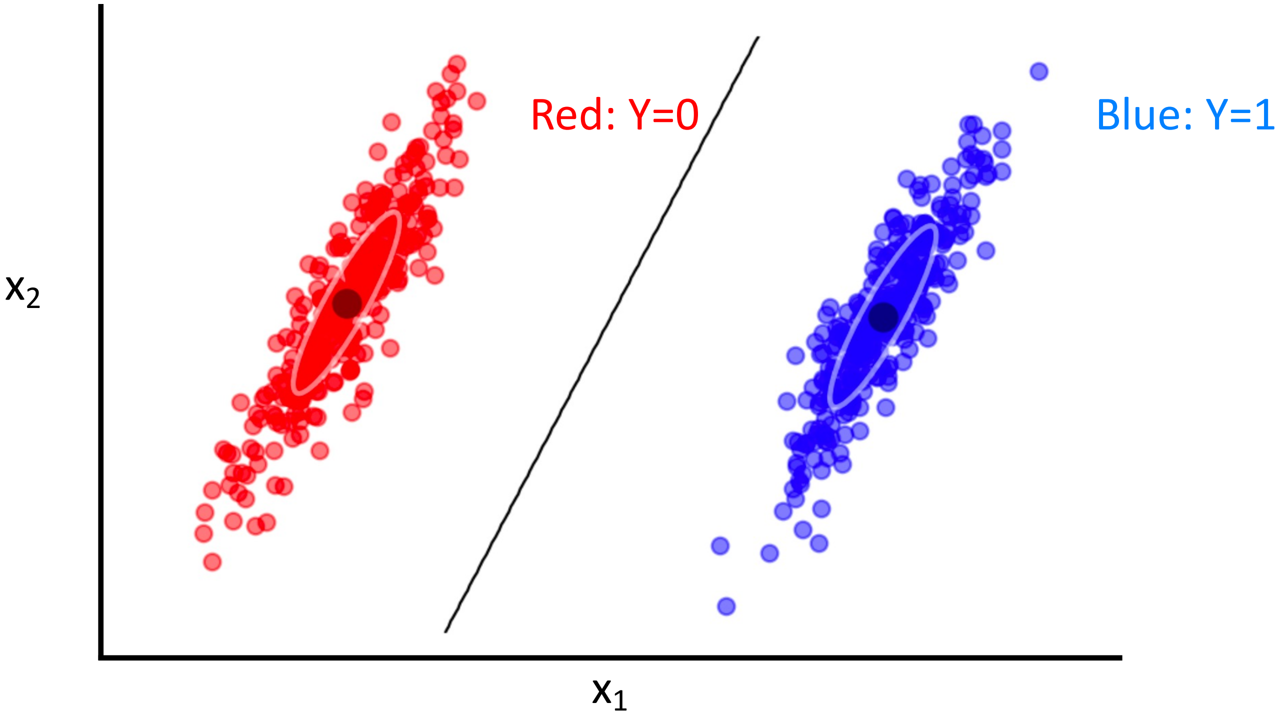
$$= \sigma\Big(-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1) + \frac{1}{2}(\mathbf{x}-\mu_0)^T\Sigma^{-1}(\mathbf{x}-\mu_0)$$
$$+ const.\Big)$$

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

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

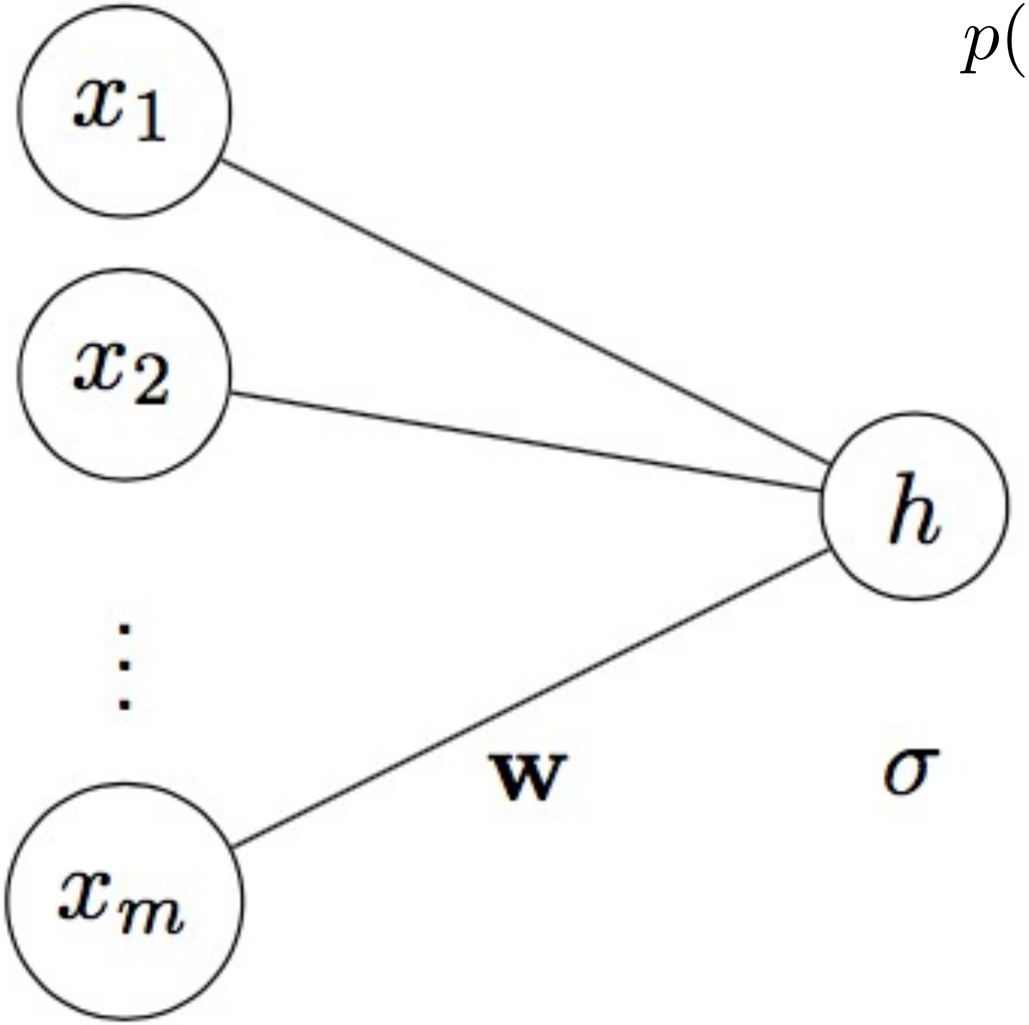
- What if we ignore Gaussian assumption on data?

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

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

- Sigmoid converts distance to class probability

Logistic Regression



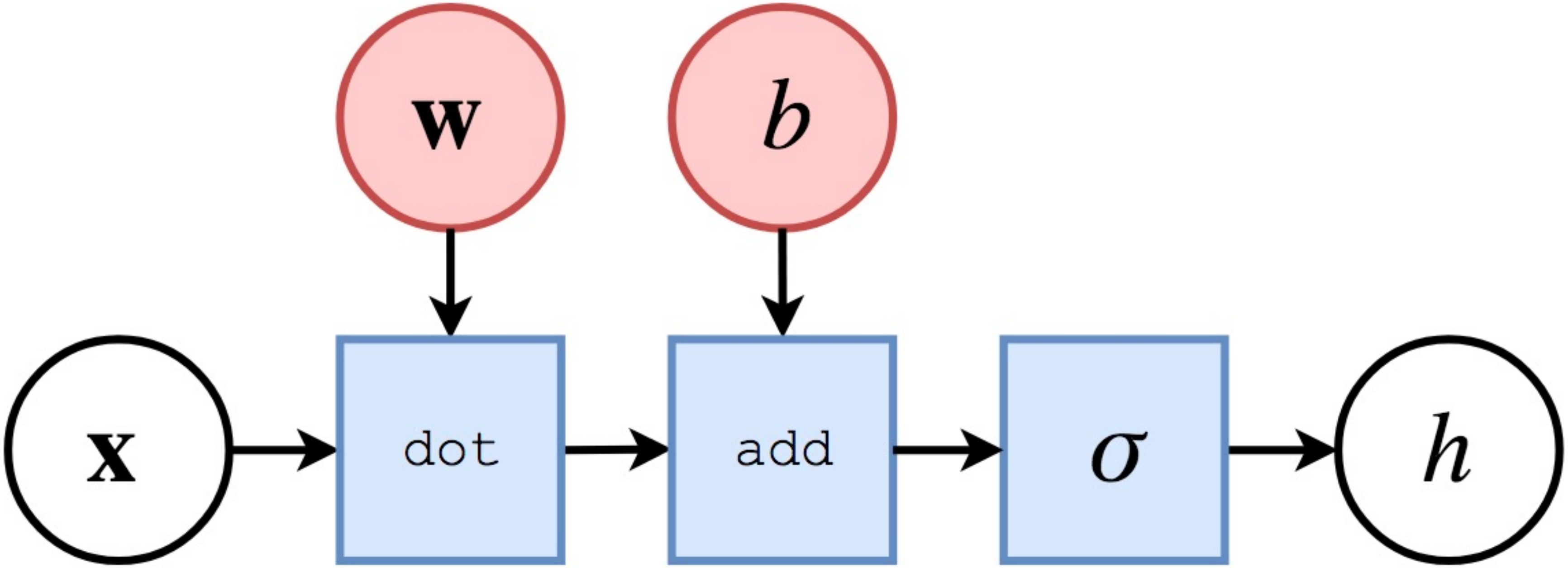
$$p(y=1|\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + b\right)$$
$$= \frac{1}{1+e^{-\mathbf{w}^T\mathbf{x}\text{-b}}}$$

This unit is the main building block of Neural Networks!

Logistic Regression

Slide credit: G. Louppe

- Computational Graph of function
  - White node = input
  - Red node = model parameter
  - Blue node = intermediate operations



This unit is the main building block of Neural Networks!

Logistic Regression

- What if we ignore Gaussian assumption on data?

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

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

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

- **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

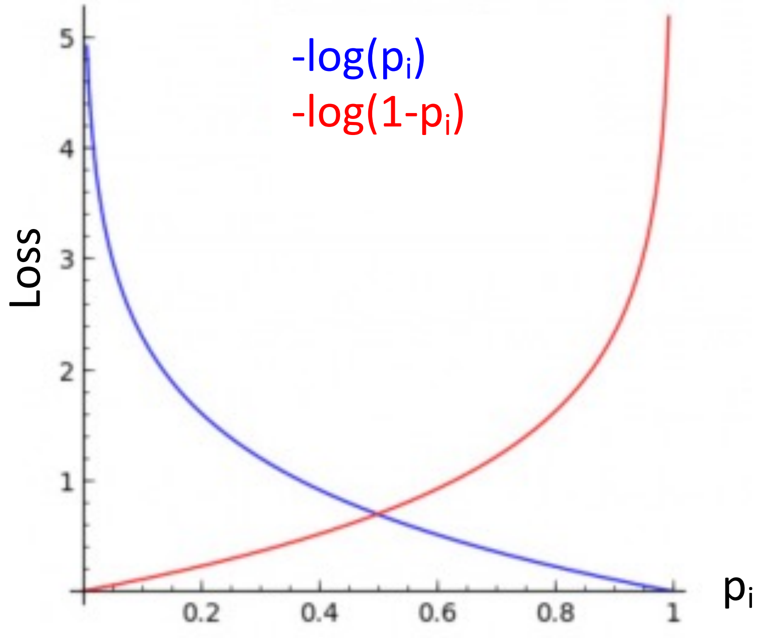
$$-\ln \mathcal{L} = -\ln \prod_i (p_i)^{y_i}(1-p_i)^{1-y_i}$$

Logistic Regression

- Negative log-likelihood

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

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 **w**?

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?

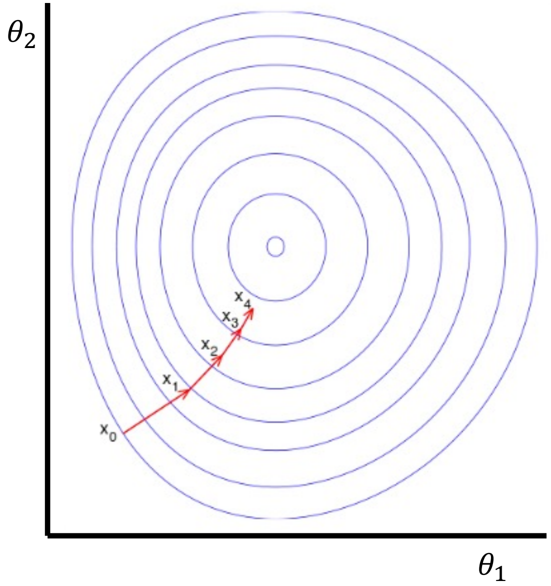
  Let λ be a perturbation along direction v

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

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

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



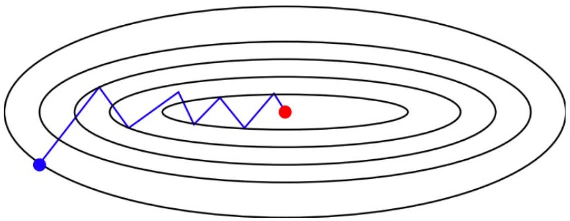
Stochastic Gradient Descent

- Loss is composed of a sum over samples:

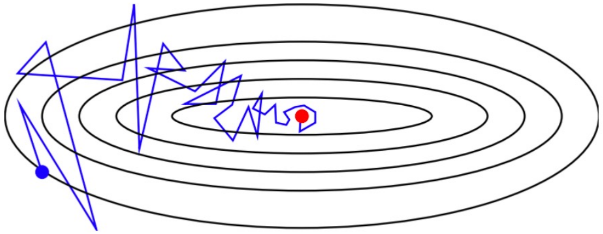
$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \mathcal{L}\big(y_i, h(x_i;\theta)\big)$$

  - 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

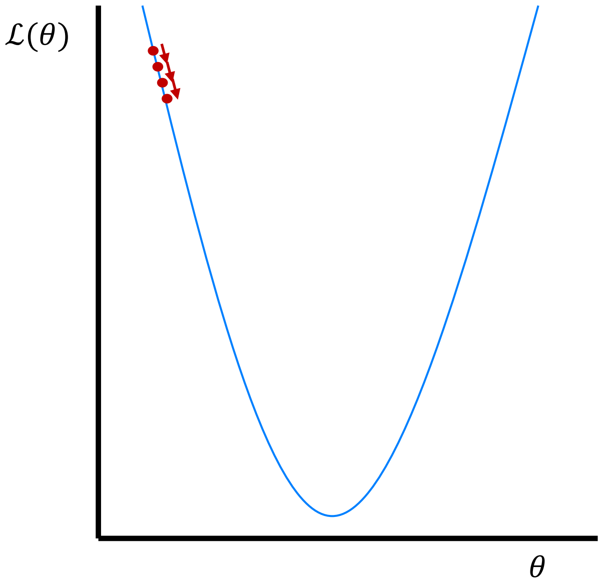
- Several updates to SGD, like momentum, ADAM, RMSprop to
  - Help to speed up optimization in flat regions of loss
  - Have adaptive learning rate
  - Learning rate adapted for each parameter
  - …

Step Sizes

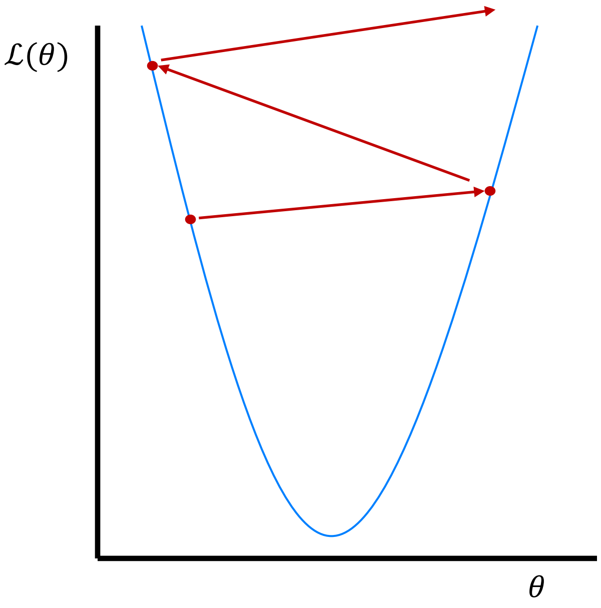
- Too small a learning rate, convergence very slow

- Too large a learning rate, algorithm diverges

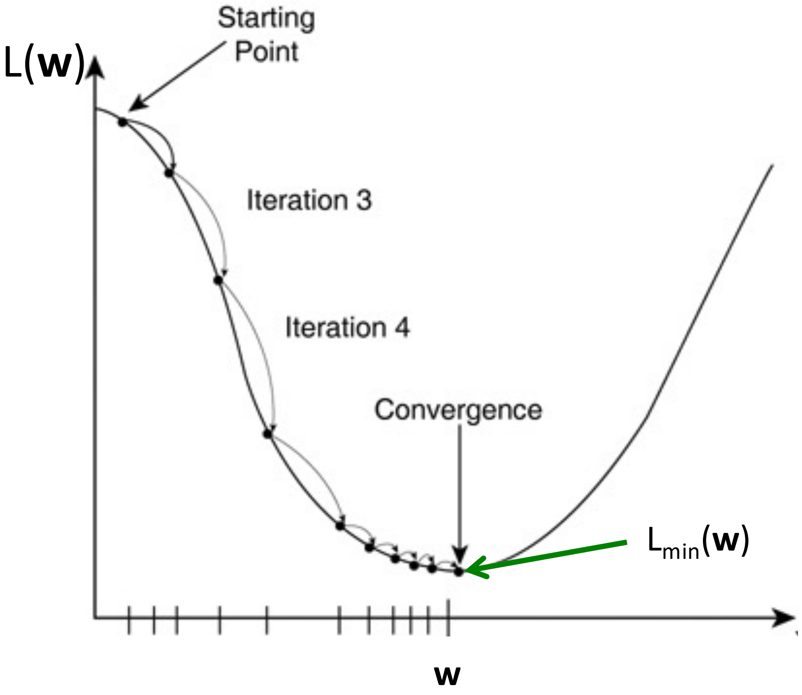
Small Learning rate

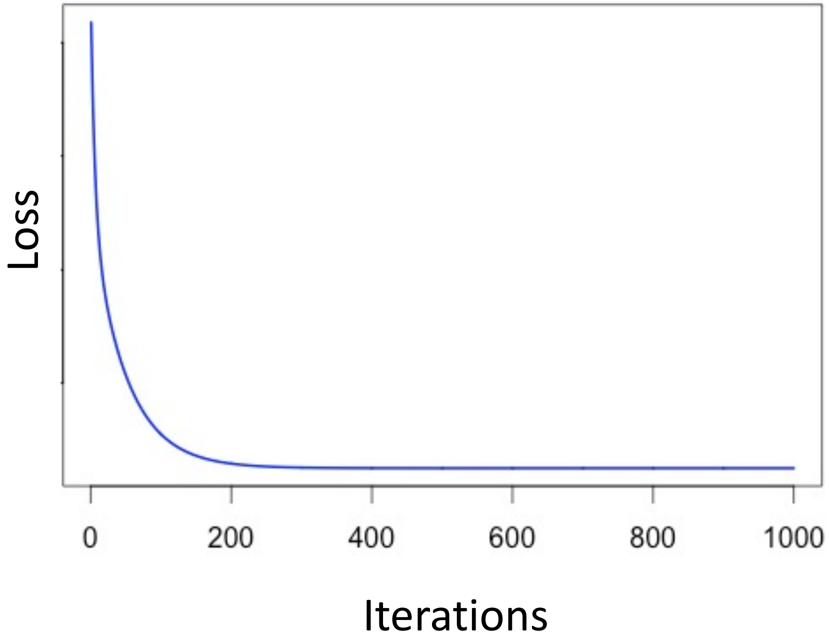


Large Learning rate



Gradient Descent





- Logistic Regression Loss is convex
  - Single global minimum

- Iterations lower loss and move toward minimum

Logistic Regression Example

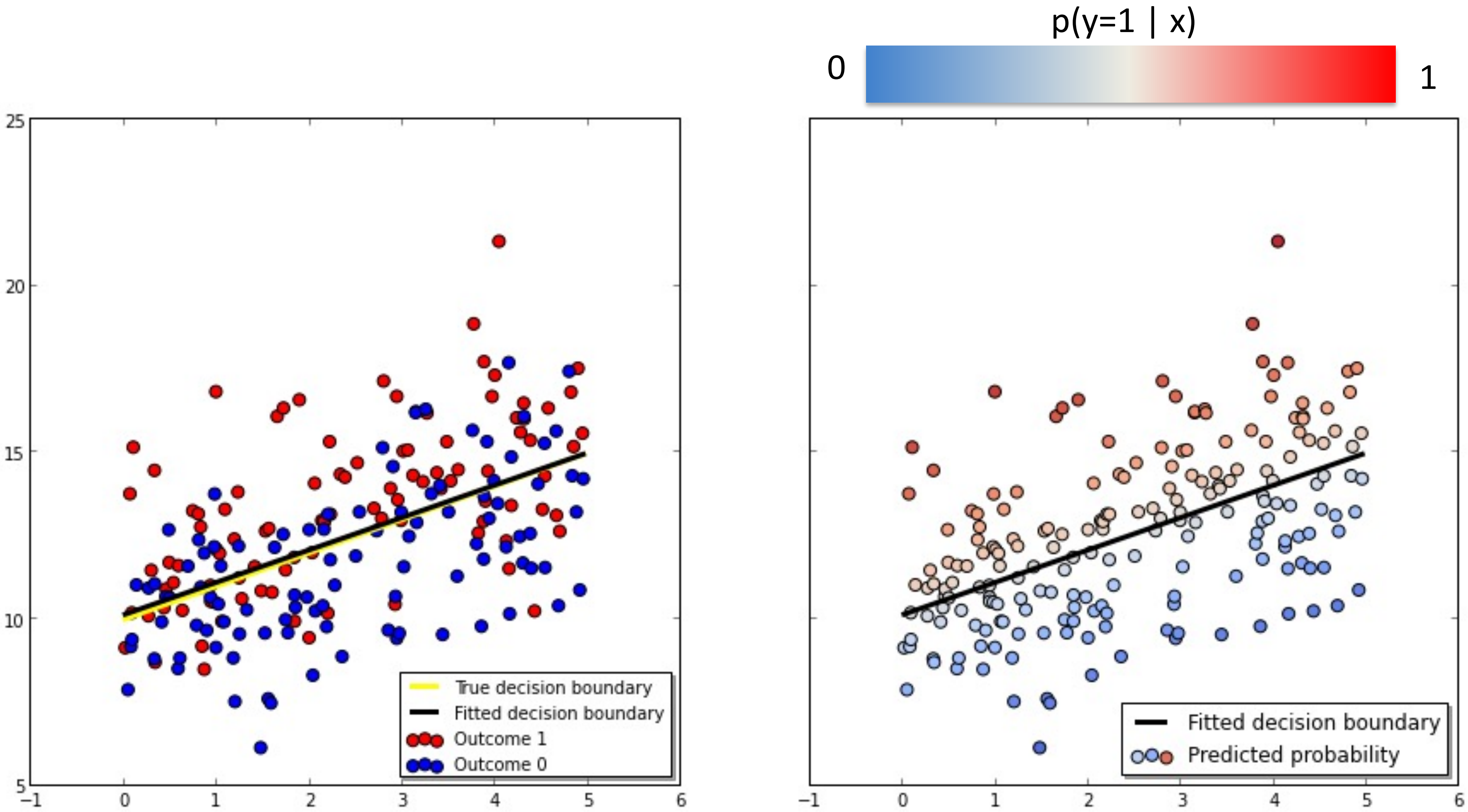
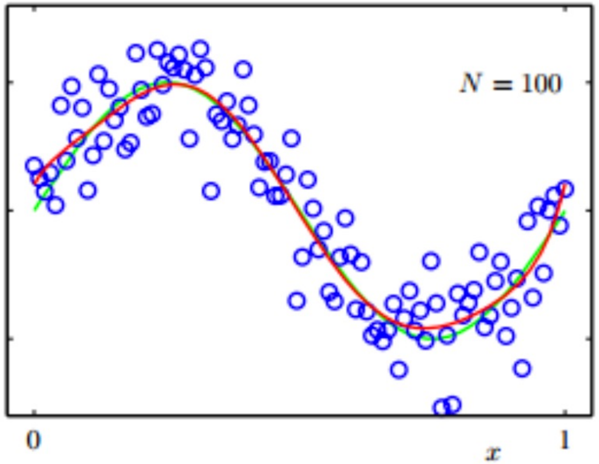


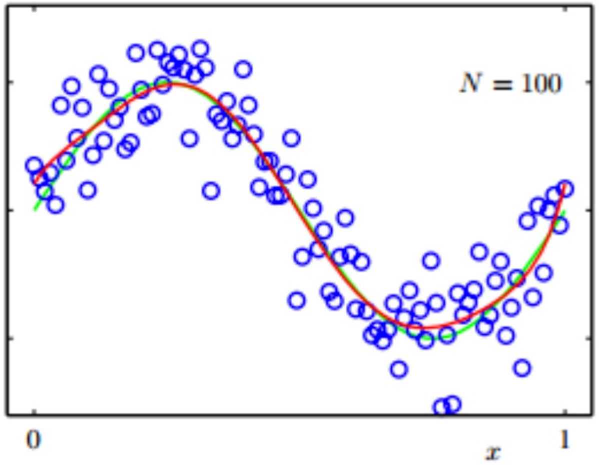
Image source

Basis Functions

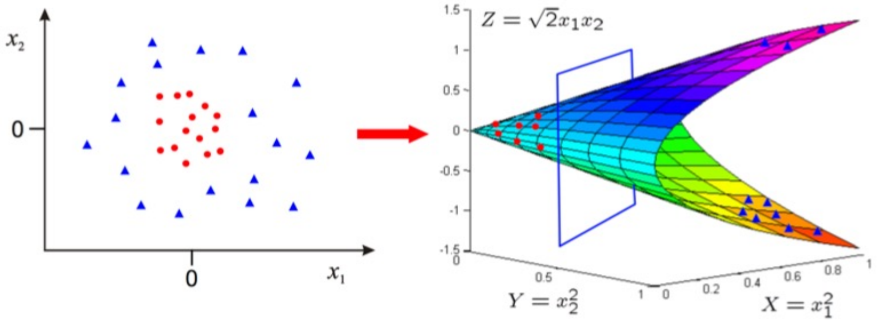


- What if non-linear relationship between **y** and **x**?

Basis Functions



$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$



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

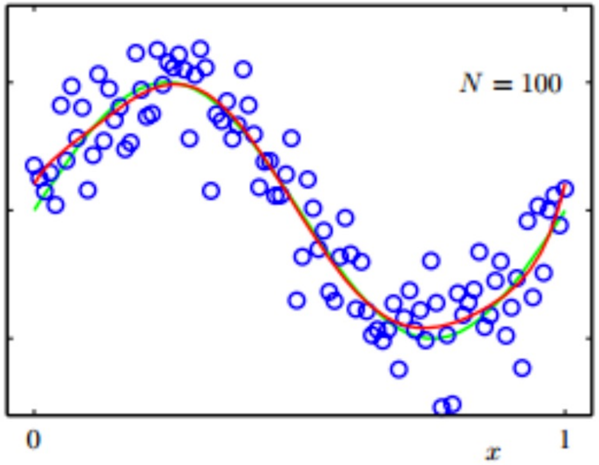
- Can choose basis functions ϕ(x) to form new features

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

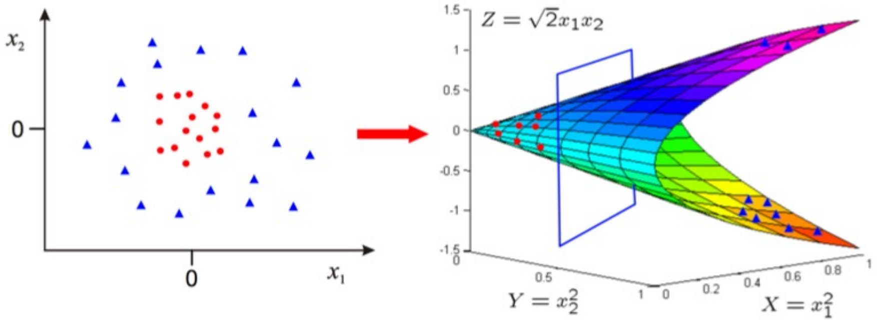
  - Polynomial basis ϕ(x) ~ {1, x, x^2, x^3, …}, Gaussian basis, …

  - **Logistic regression on new features ϕ(x)**

Basis Functions



$$\Phi : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \end{pmatrix} \qquad \mathbb{R}^2 \to \mathbb{R}^3$$



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

- Can choose basis functions ϕ(x) to form new features

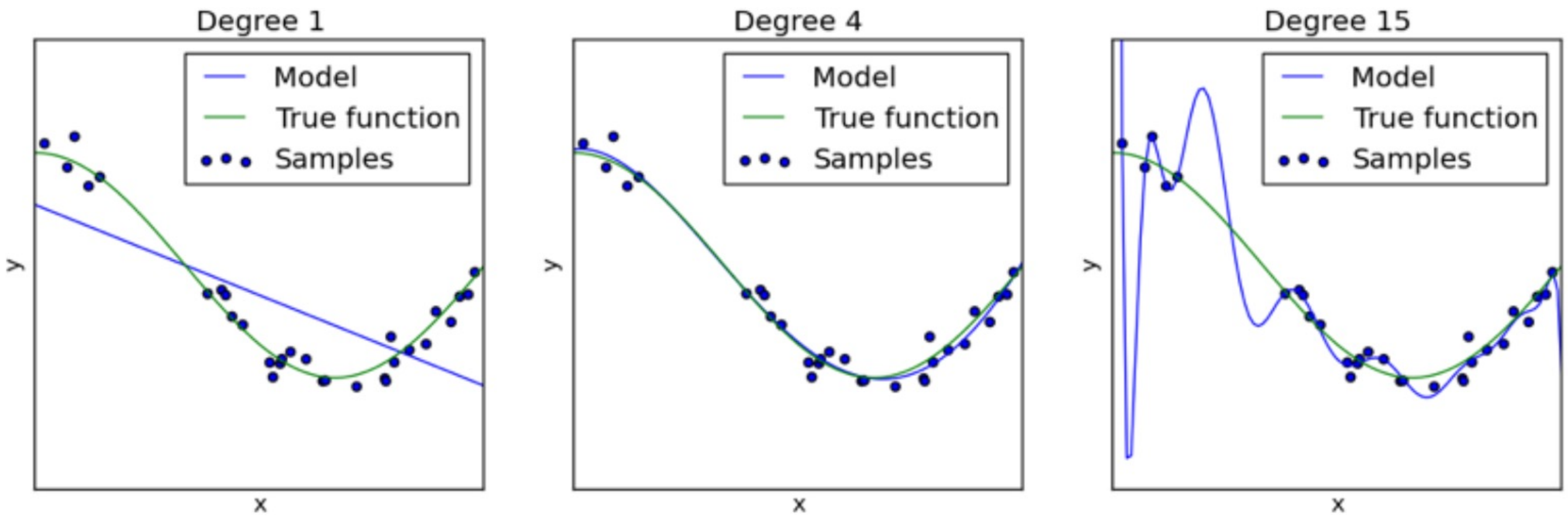
$$h(x; w) = \sigma\left(w^T \phi(x)\right)$$

  - Polynomial basis ϕ(x) ~ {1, x, x^2, x^3, …}, Gaussian basis, …

  - Logistic regression on new features ϕ(x)

- What basis functions to choose? *Overfit* with too much flexibility?

What is Overfitting



Underfitting

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).

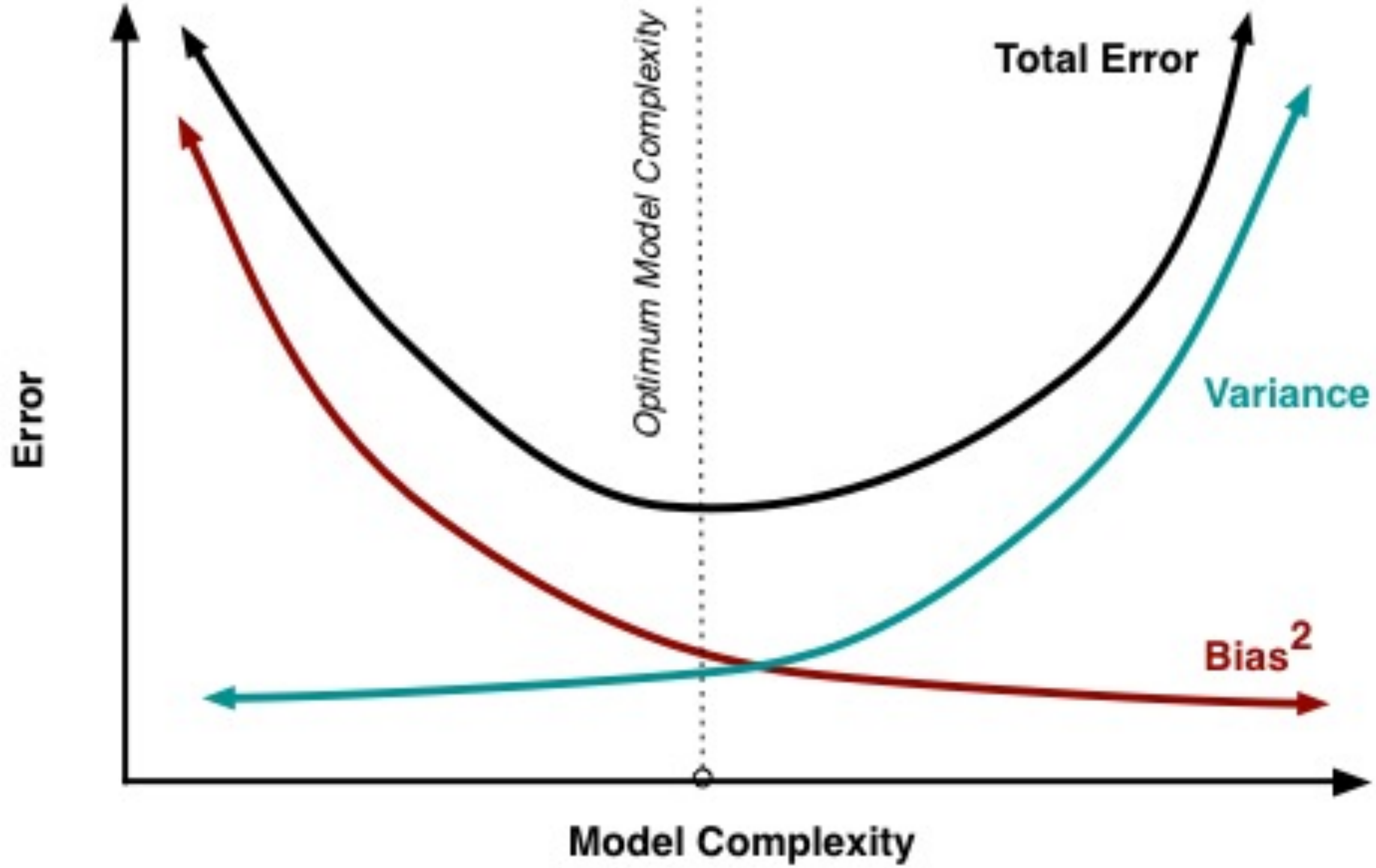
Bias Variance Tradeoff

- 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

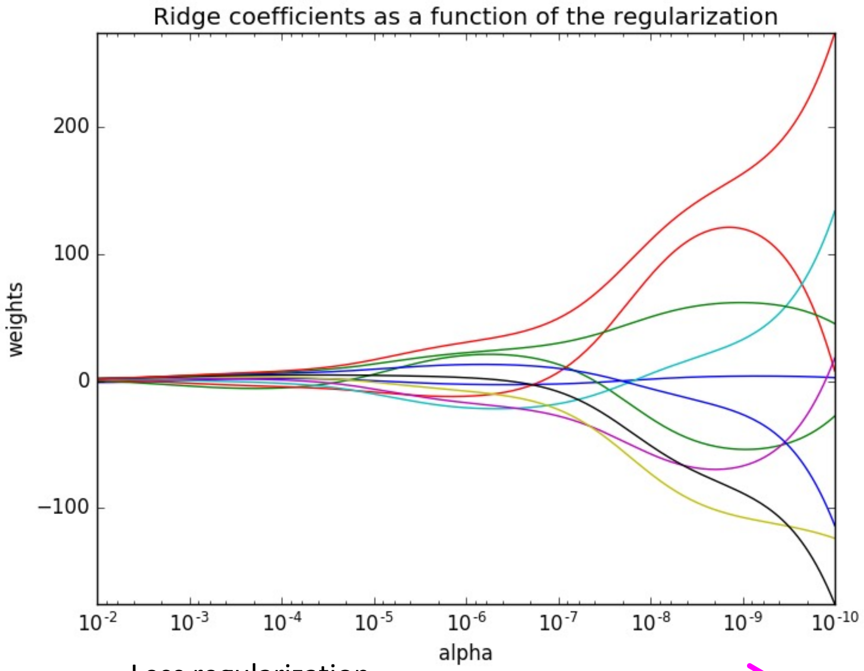


Regularization – Control Complexity

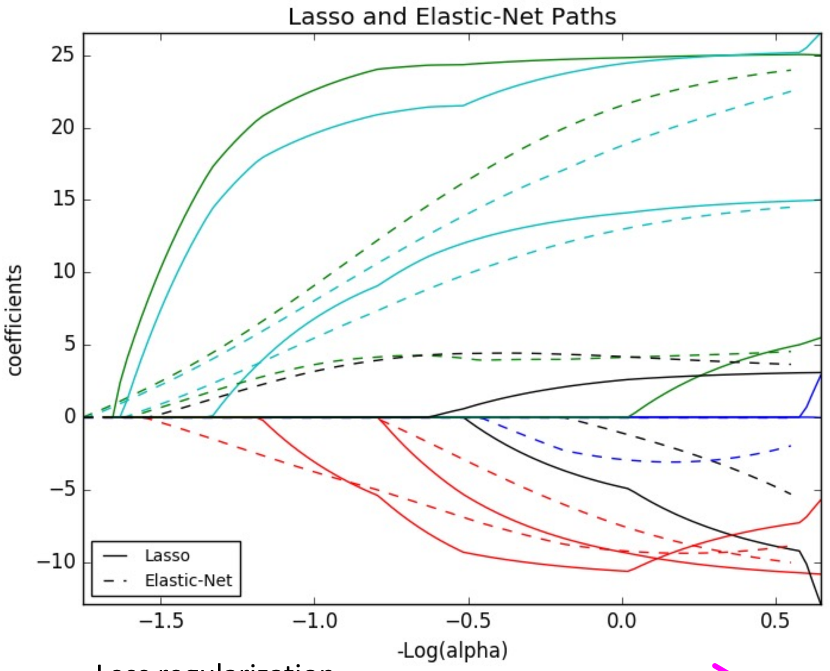
$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^2 + \alpha\Omega(\mathbf{w})$$

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

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



Less regularization →



Less regularization →

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

http://scikit-learn.org/

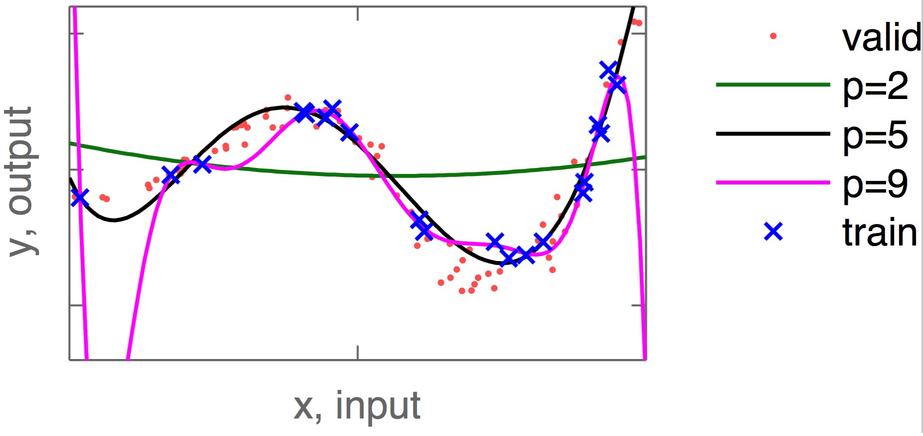
How to Measure Generalization Error?

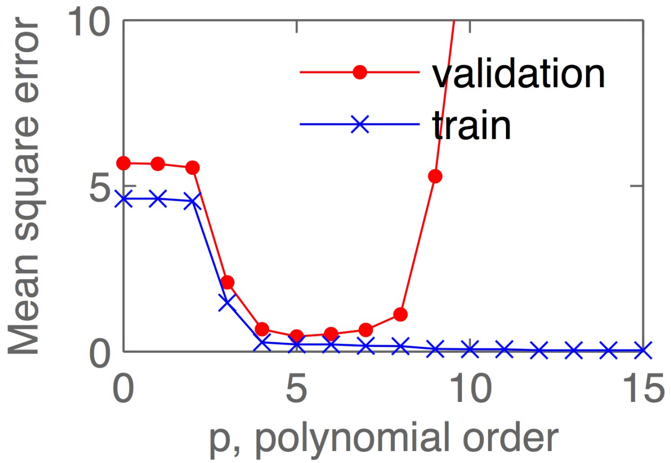
Training set

Validation set

Test set

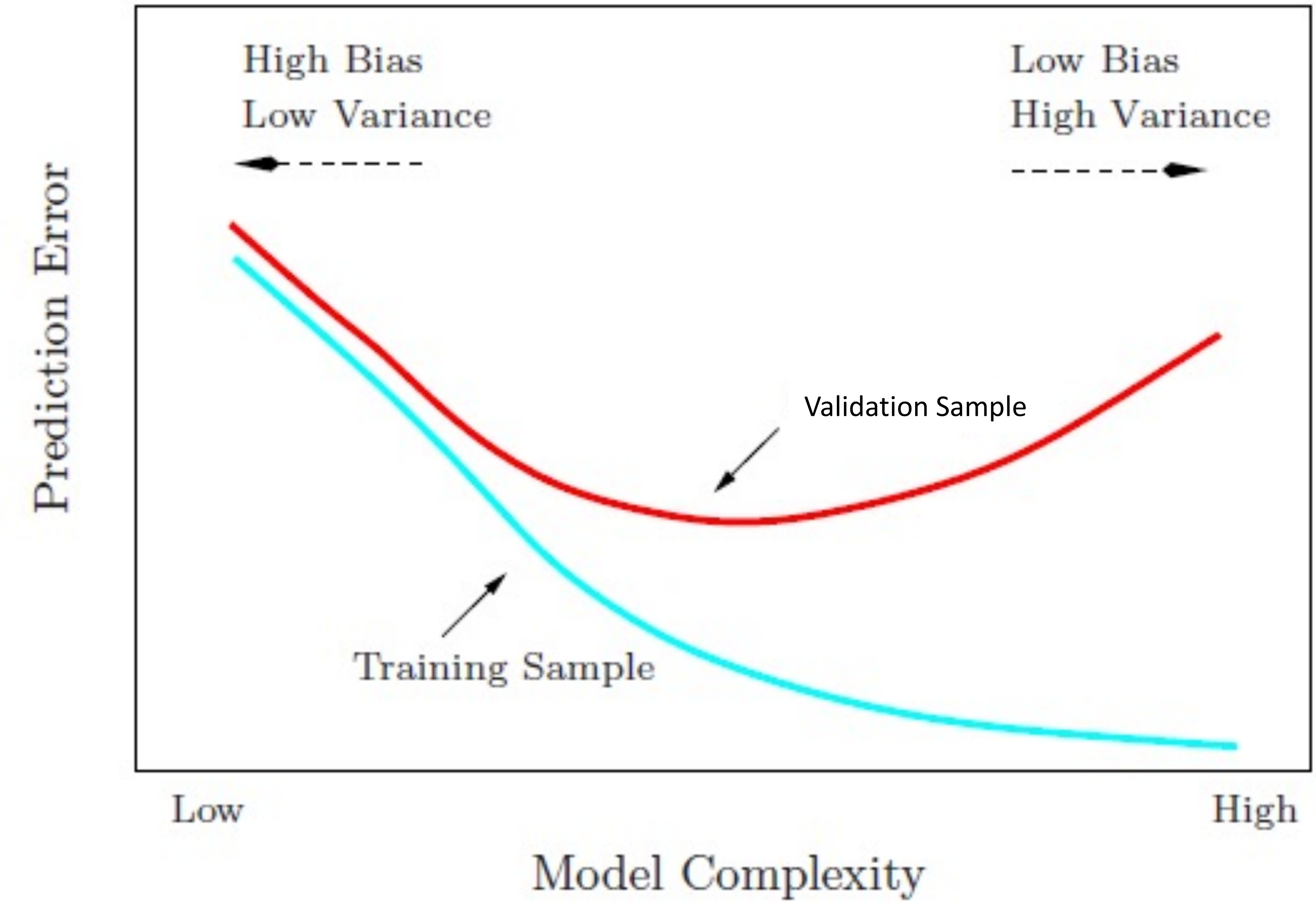
- Split dataset into multiple parts
- **Training set**
  - Used to fit model parameters
- **Validation set**
  - Used to check performance on independent data and tune hyper parameters
- **Test set**
  - final evaluation of performance after all hyper-parameters fixed
  - Needed since we tune, or “peek”,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

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/

Backup

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

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

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

$$E[(y-h(x))^2] = \boxed{\begin{array}{l} E[(y-\bar{y})^2] \\ \text{noise} \end{array}} \begin{array}{llll} + & (\bar{y}-\bar{h}(x))^2 & + & E[(h(x)-\bar{h}(x))^2] \\ + & (\text{bias})^2 & + & \text{variance} \end{array}$$

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

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

- 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

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

- 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

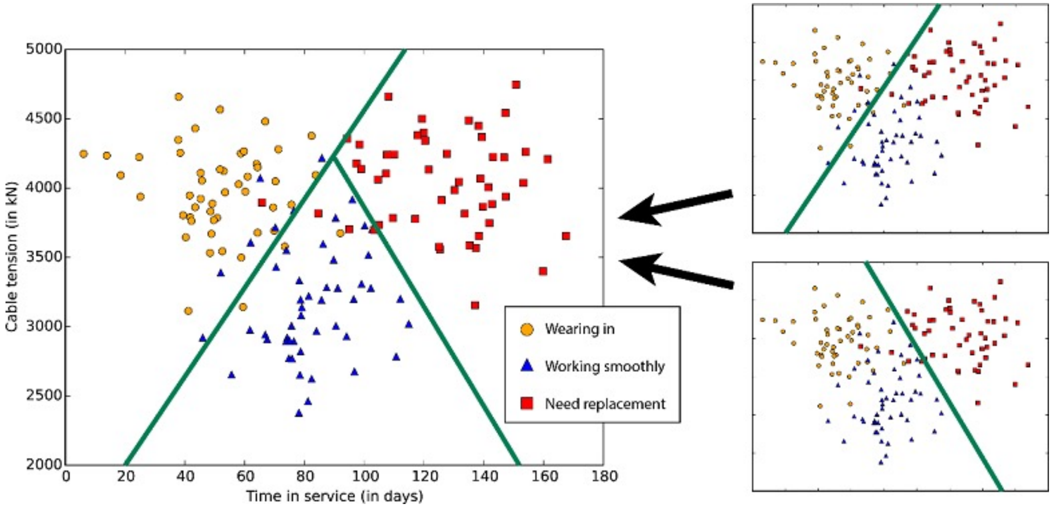
$$\begin{aligned} E[(y-h(x))^2] &= \boxed{E[(y-\bar{y})^2]} + \boxed{(\bar{y}-\bar{h}(x))^2} + \boxed{E[(h(x)-\bar{h}(x))^2]} \\ &= \boxed{\text{noise}} + \boxed{(\text{bias})^2} + \boxed{\text{variance}} \end{aligned}$$

- 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**

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 $\mathbf{y}_k = (0, \ldots, 1, \ldots 0)$

kth 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 $\mathbf{w}_k$