Machine Learning: Lecture I

Michael Kagan

SLAC

University of Geneva

May 24, 2018

Lecture Structure

- Lecture I
  - What is Machine Learning
  - Linear Regression and Classification
  - Fitting a model: Cost Functions, Regularization, Gradient Descent
- Lecture II
  - Intro to Neural Networks, Deep Learning
  - Decision Trees and ensemble methods
  - Dimensionality reduction
  - Clustering
- Many topics we won't be able to cover in such a short time
  - SVM
  - Gaussian Processes
  - Variational Inference
  - Hidden Markov Models
  - …

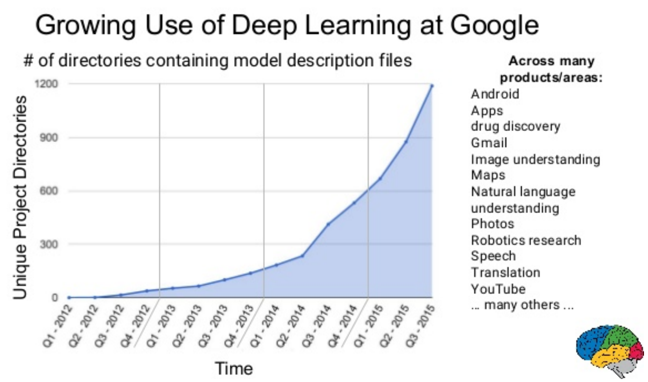
What is Machine Learning?

What is Machine Learning?

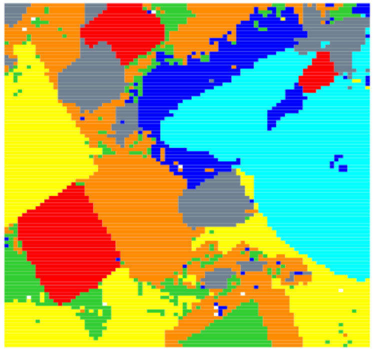
- Giving computers the ability to learn without explicitly programming them (Arthur Samuel, 1959)
- Statistics + Algorithms
- Computer Science + Probability + Optimization Techniques
- **Fitting data with complex functions**
- **Mathematical models** learnt from data that characterize the patterns, regularities,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

Where is ML Used, an Incomplete List

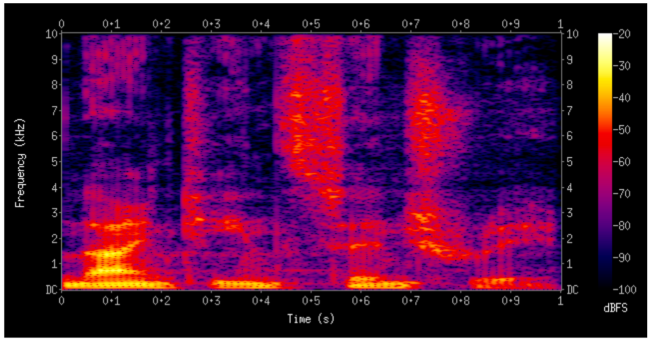
- Natural Language Processing
- Speech and handwriting recognition
- Object recognition and computer vision
- Fraud detection
- Financial market analysis
- Search engines
- Spam and virus detection
- Medical diagnosis
- Robotics control
- Automation: energy usage, systems control, video games, self-driving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

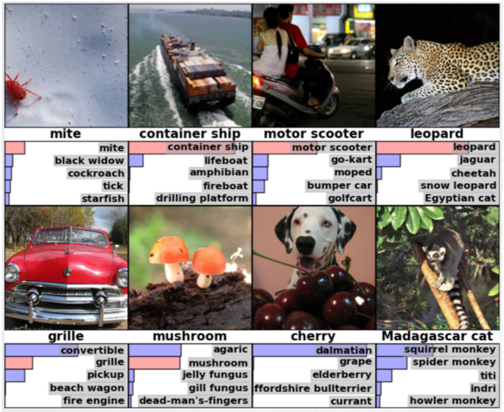


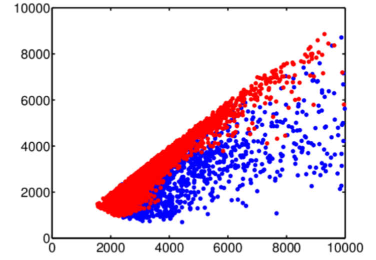
Predicted Land Usage



[ESL]



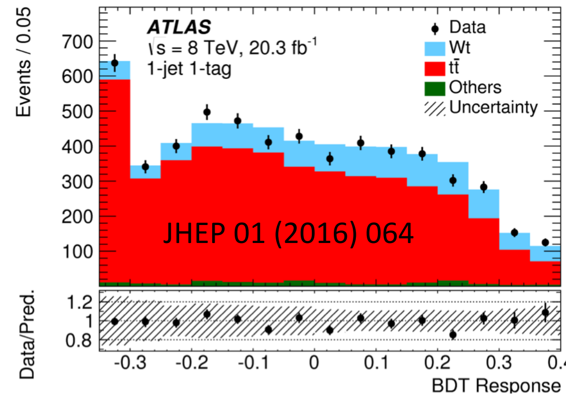


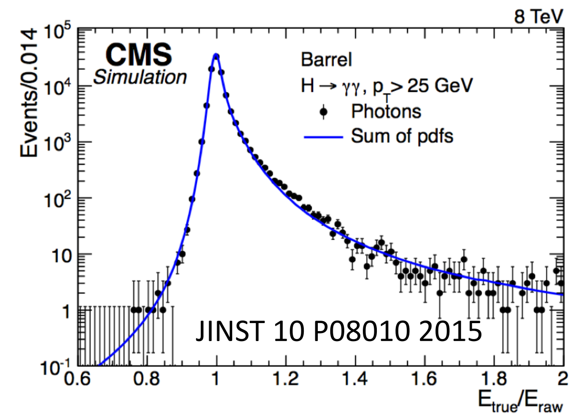


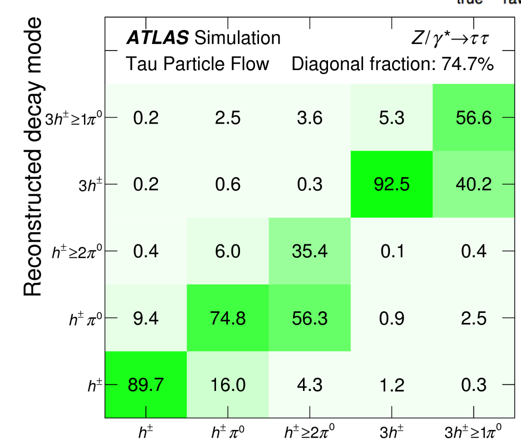
Minor elliptical axis (y) against Major elliptical axis (x) for stars (red) and galaxies (blue). (Amos Storkey)
http://www-wfau.roe.ac.uk/sss/

Machine Learning Applied Widely in HEP

- **In analysis**:
  - Classifying signal from background, especially in complex final states
  - Reconstructing heavy particles and improving the energy / mass resolution
  - …
- **In reconstruction**:
  - Improving detector level inputs to reconstruction
  - Particle identification tasks
  - Energy / direction calibration
  - …
- **In the trigger**:
  - Quickly identifying complex final states
  - …
- **In computing**:
  - Estimating dataset popularity, and determining how number and location of dataset replicas
  - …







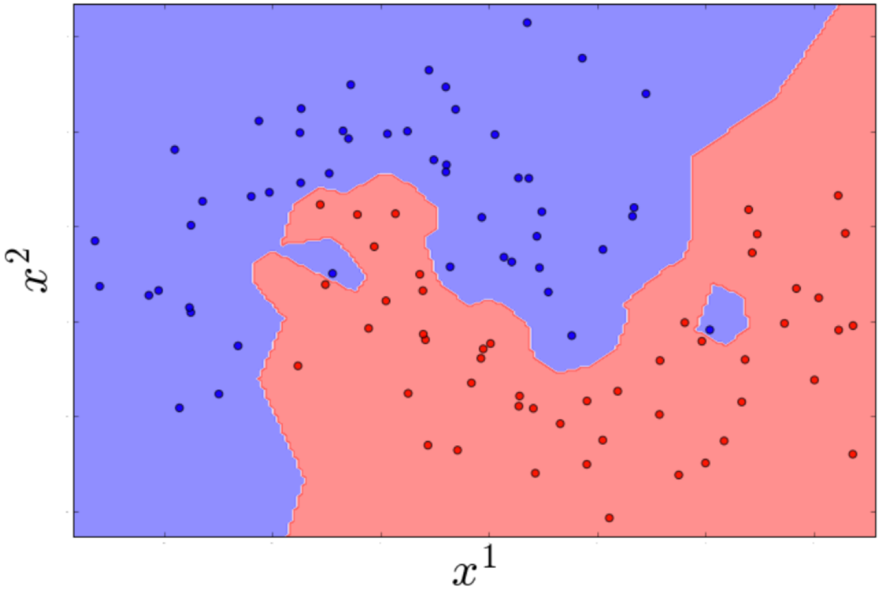
arXiv:1512.05955

Machine Learning: Models

- Key element in machine learning is a **mathematical model**
  - A mathematical characterization of system(s) of interest, typically via random variables
  - Chosen model depends on the task / available data

Machine Learning: Models

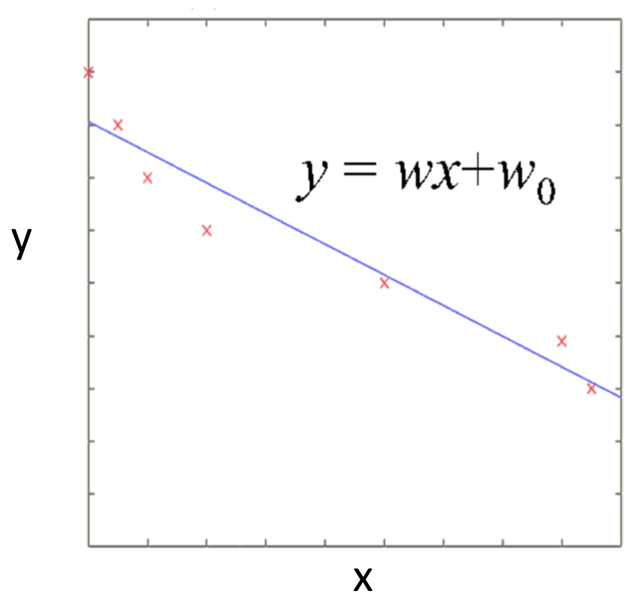
- Key element in machine learning is a **mathematical model**
  - A mathematical characterization of system(s) of interest, typically via random variables
  - Chosen model depends on the task / available data
  - **Classification:**



[Rogozhnikov]

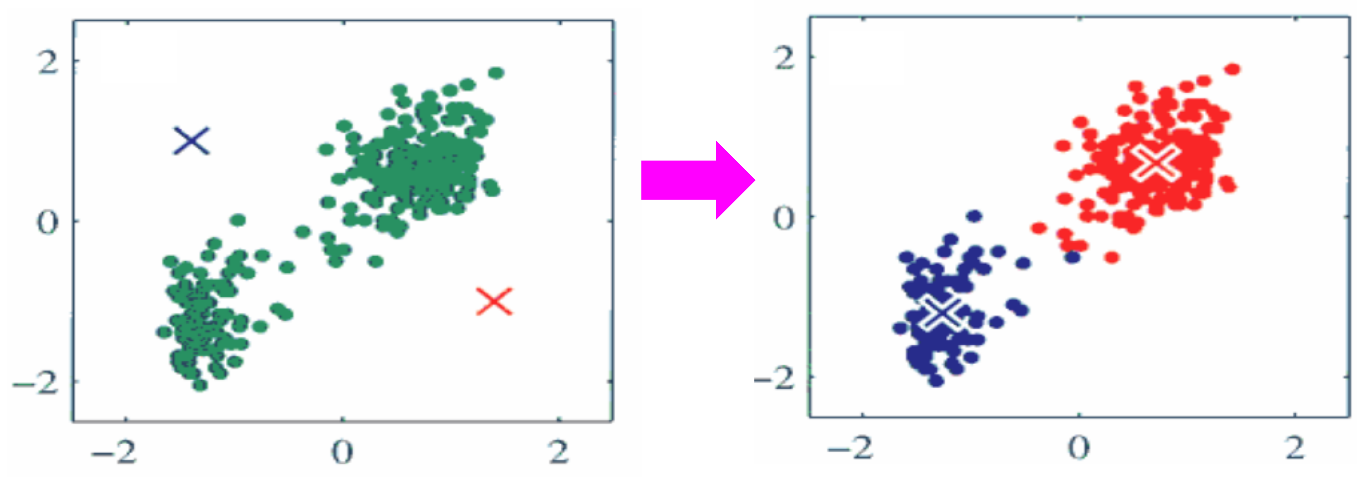
Machine Learning: Models

- Key element in machine learning 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

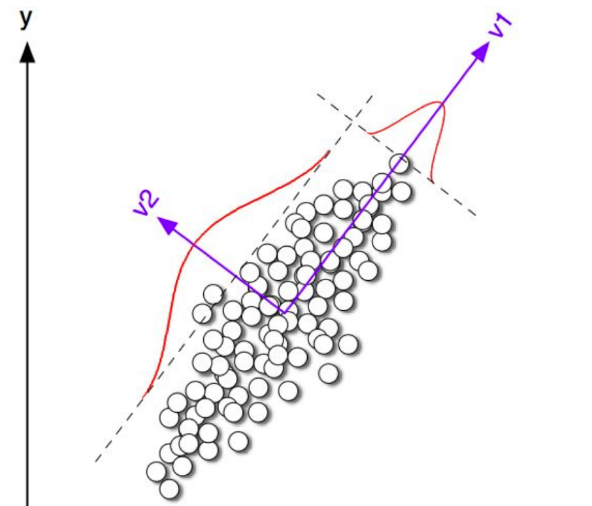
- Key element in machine learning is a **mathematical model**
  - A mathematical characterization of system(s) of interest, typically via random variables
  - Chosen model depends on the task / available data
  - **Clustering:**



[Bishop]

Machine Learning: Models

- Key element in machine learning 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:**



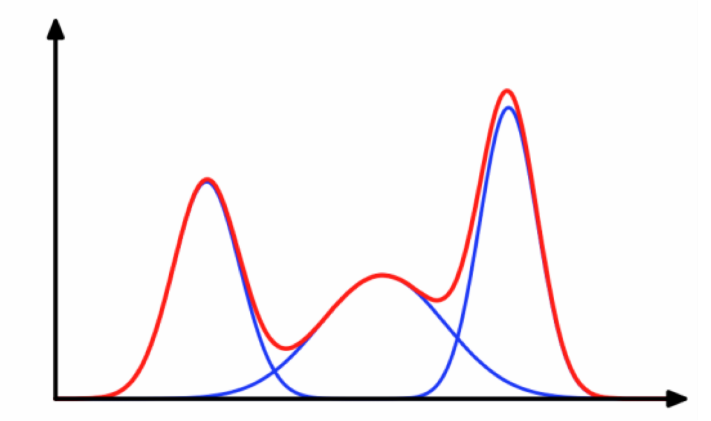
https://lazyprogrammer.me/tutorial-principal-components-analysis-pca/

Machine Learning: Models

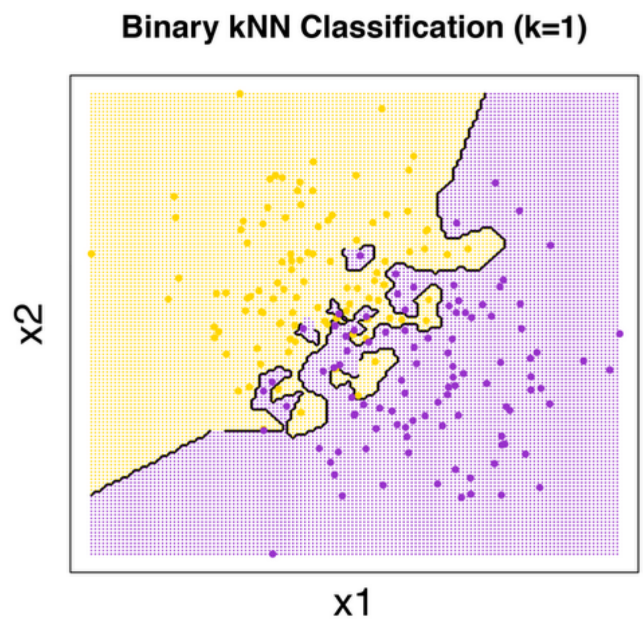
- Key element in machine learning is a **mathematical model**
  - A mathematical characterization of system(s) of interest, typically via random variables
  - Chosen model depends on the task / available data
- **Learning**: estimate statistical model from data
- **Prediction and Inference:** using statistical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

Parametric vs. Non-parametric Models

- **Parametric Models**: models that do not grow in complexity with dataset size. Fixed set of parameters to learn
  - Example: sum of Gaussians, each with mean, variance, and normalization

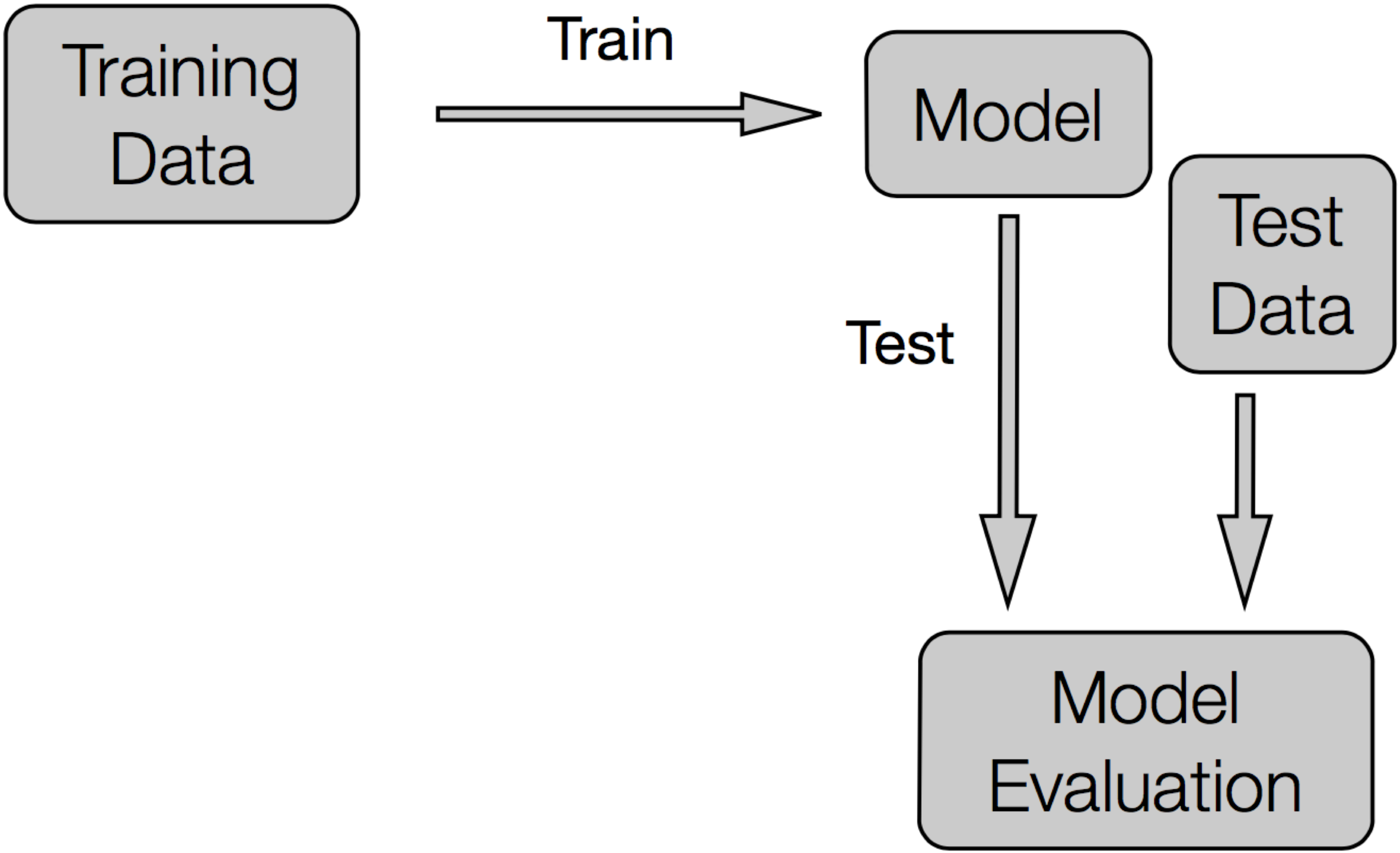


- **Non-Parametric Models**: models that do not have a fixed set of parameters, often grow in complexity with more data
  - Example: model predictions of a new data point using nearest known datapoint. The more known datapoints, the more complex is the model



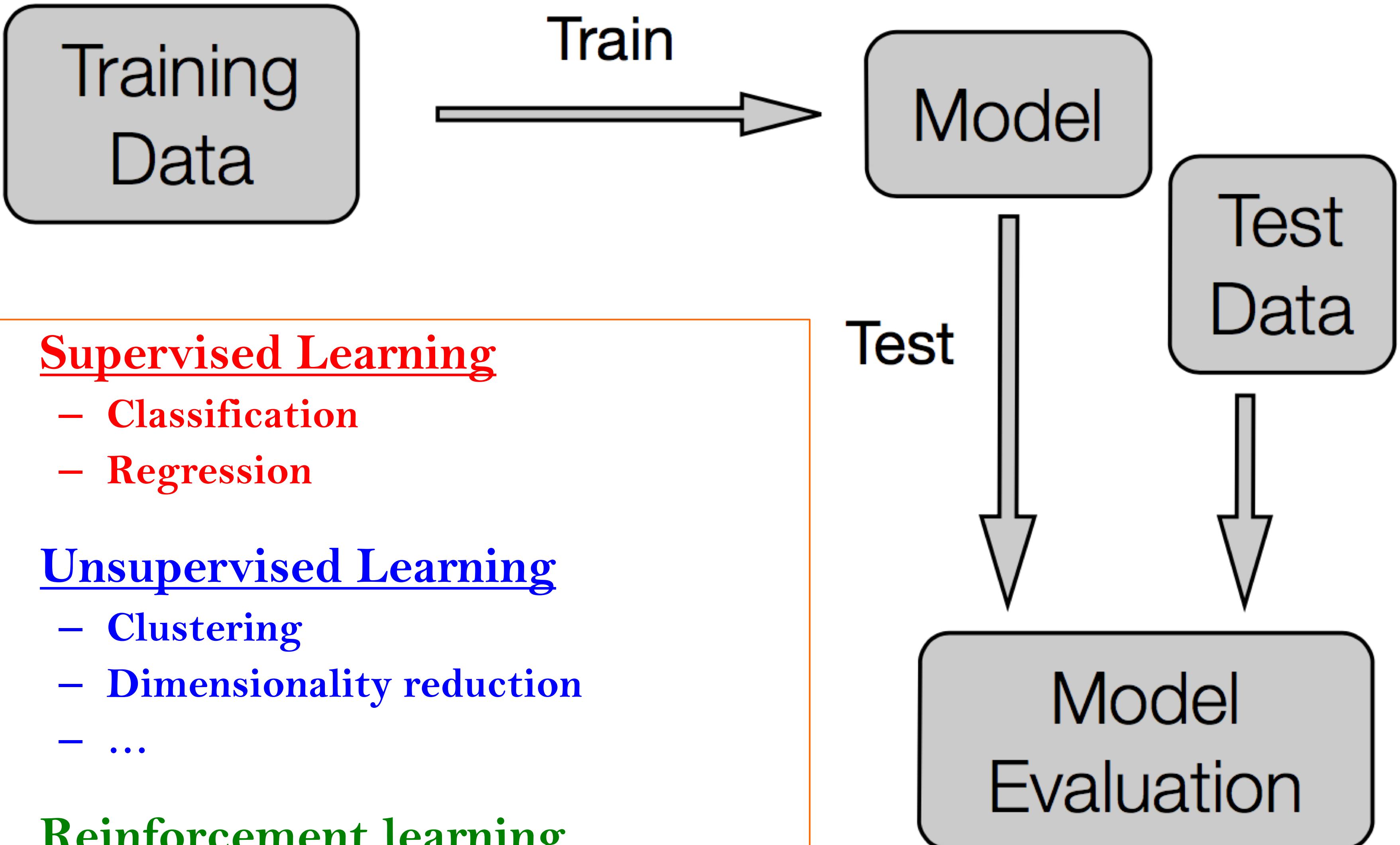
http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/

Learning



[Ravikumar]

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:

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

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

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

Probability Review

- Joint distribution of two variables: p(x,y)
- Marginal distribution: $p(x) = \int p(x,y)dy$
- Conditional distribution: $p(y|x) = \frac{p(x,y)}{p(x)}$
- Bayes theorem: $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$
- Expected value: $\mathbf{E}[f(x)] = \int f(x)p(x)\mathrm{d}x$
- Normal distribution:
  - $x \sim N(\mu, \sigma)$ → $p(x) = \frac{1}{\sqrt{2\pi\sigma_2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$

Supervised Learning

- Given N examples with features $\{x_i \in \mathcal{X}\}$ and **targets** $\{y_i \in \mathcal{Y}\}$, learn function mapping **h(x)=y**
  - **Classification**: $\mathcal{Y}$ is a finite set of **labels** (i.e. classes)

$\mathcal{Y} = \{0, 1\}$ for **binary classification**,
encoding classes, e.g. Higgs vs Background

$\mathcal{Y} = \{c_1, c_2, \ldots c_n\}$ for **multi-class classification**

represent with "**one-hot-vector**"

$\rightarrow y_i = (0, 0, \ldots, 1, \ldots 0)$

were k^{th} element is 1 and all others zero for class c_k

Supervised Learning

- Given N examples with features $\{x_i \in \mathcal{X}\}$ and **targets** $\{y_i \in \mathcal{Y}\}$, learn function mapping **h(x)=y**
  - **Classification**: $\mathcal{Y}$ is a finite set of **labels** (i.e. classes)
  - **Regression**: $\mathcal{Y}$ = Real Numbers

Supervised Learning

- Given N examples with features $\{\mathbf{x}_i \in \mathcal{X}\}$ and **targets** $\{y_i \in \mathcal{Y}\}$, learn function mapping **h(x)=y**
  - **Classification:** $\mathcal{Y}$ is a finite set of **labels** (i.e. classes)
  - **Regression:** $\mathcal{Y}$ = Real Numbers
- Often these are **discriminative models**, in which case we model:
  $$h(x) = p(y \mid x)$$
- Sometimes use **generative models**, estimate joint distribution p(y, x)
  - Could estimate class conditional density $p(x \mid y)$ and prior $p(y)$
  - Use Bayes theorem to then compute:

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

Unsupervised Learning

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

  - **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 $Z=h(X)$

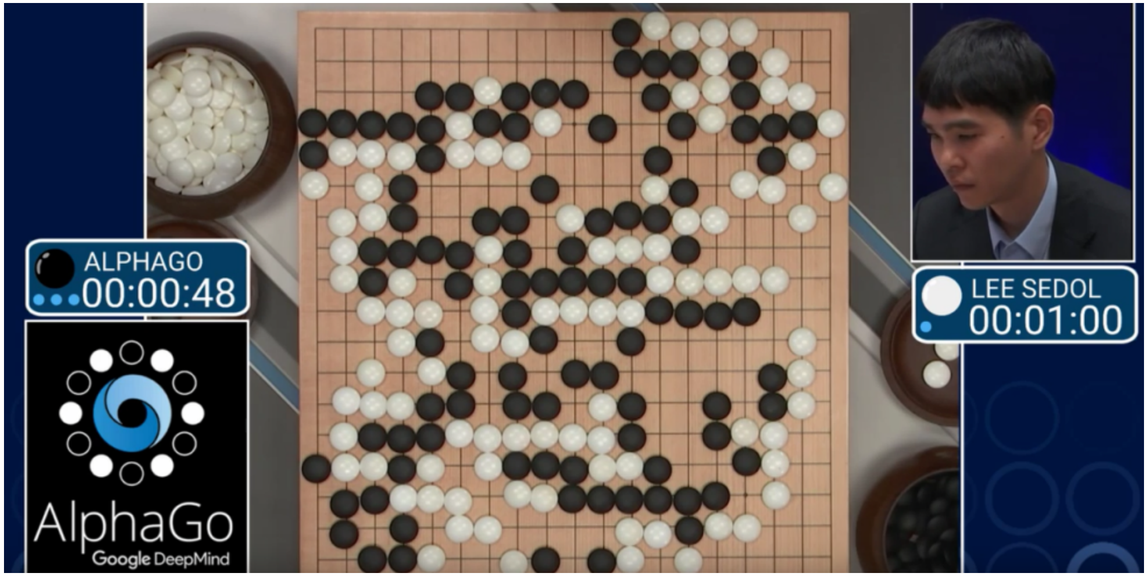
Reinforcement Learning

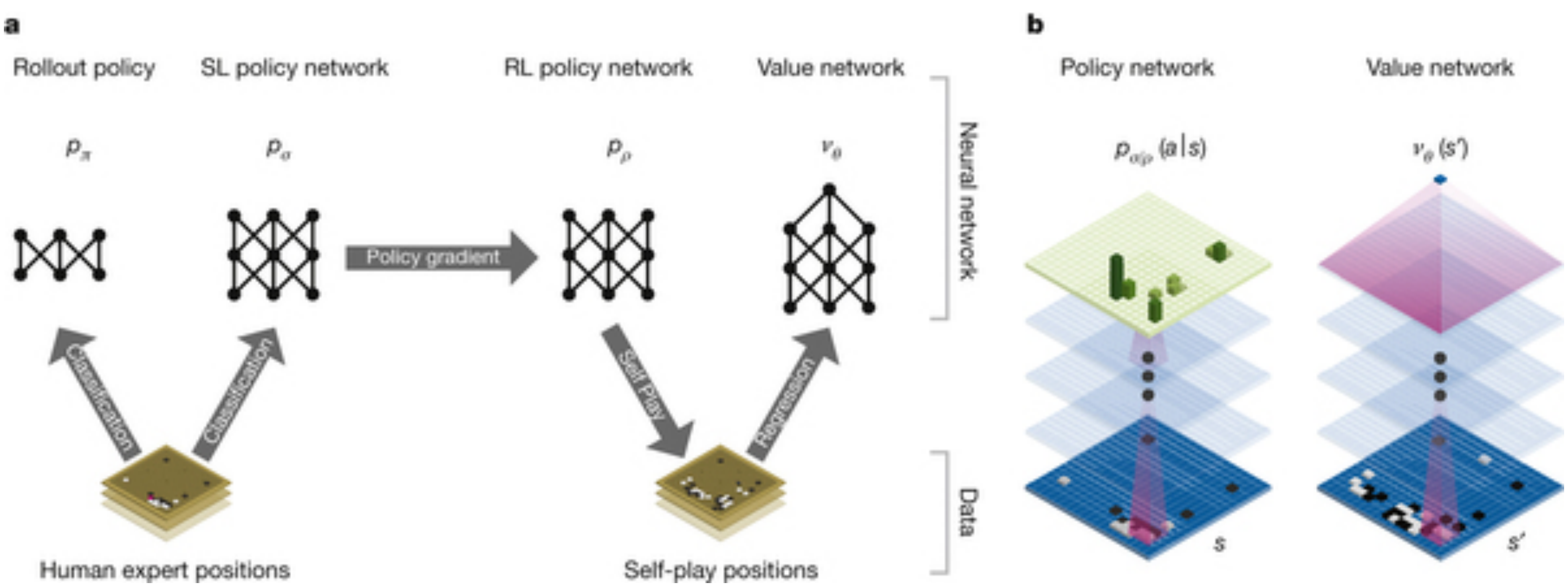
[Ravikumar]

- 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

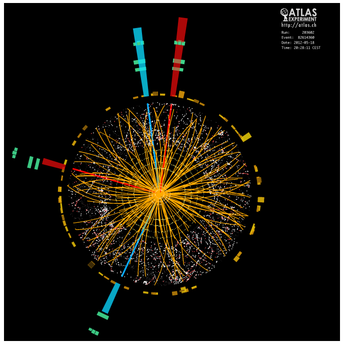




Nature 529, 484–489 (28 January 2016)

Supervised Learning: How does it work?

Supervised Learning: How does it work?



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

Loss Function

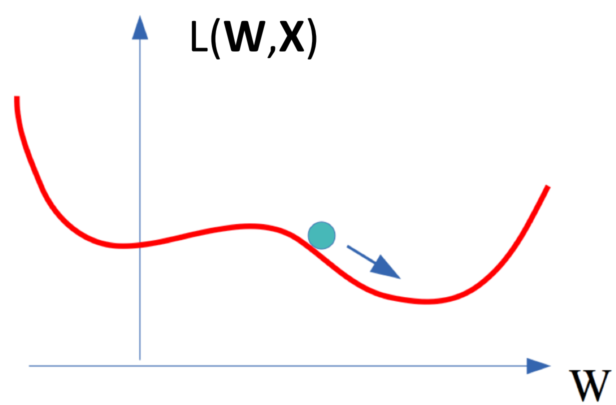
Compare prediction with true label

Loss

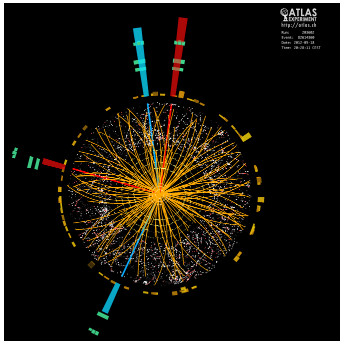
<u>True labels:</u>
Higgs = 1
Bkg = 0

- 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

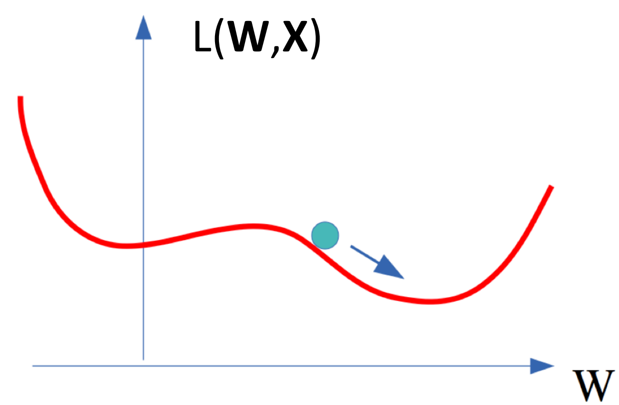
Compare prediction with true label

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
- Estimate final performance on *test-set*

Y. Le Cun



Empirical Risk Minimization

$$\arg\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i; \mathbf{w}), y_i) + \lambda \Omega(\mathbf{w})$$

Average expected loss

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 we penalize, and is a hyperparameter that we have to tune
    - We will come back to this later
- 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

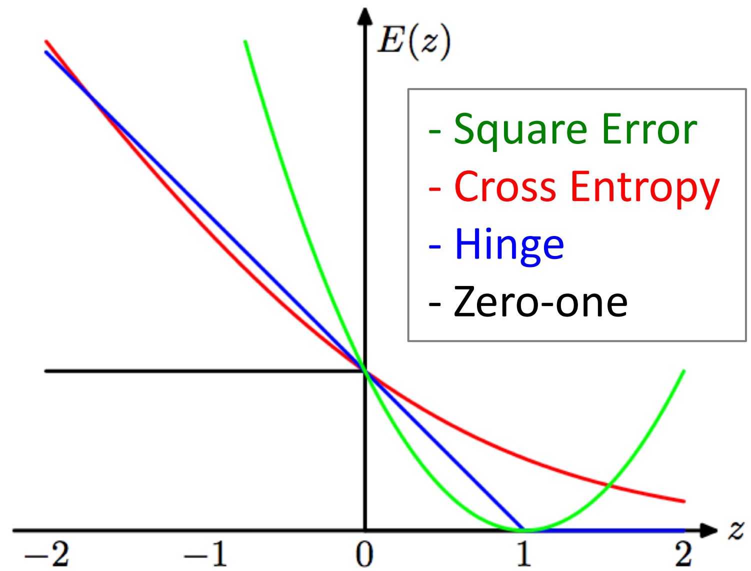
$$L(h(\mathbf{x};\mathbf{w}),y) = -\,y\log h(\mathbf{x};\mathbf{w}) - (1-y)\log(1-h(\mathbf{x};\mathbf{w}))$$

- 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]

Maximum Likelihood

- Describe a process behind the data
- Write down the likelihood of the observed data

$$\mathcal{L}(\mathbf{w}) = p(\mathbf{y}|\mathbf{X};\mathbf{w}) = \prod_i p(y_i|\mathbf{x}_i;\mathbf{w})$$

  - Where second equality holds if data is independent and identically distributed
  - Often minimize negative-log-likelihood for numerical stability
    - Same as maximizing likelihood since log is monotonic and differentiable away from zero

Maximum Likelihood

- Describe a process behind the data
- Write down the likelihood of the observed data

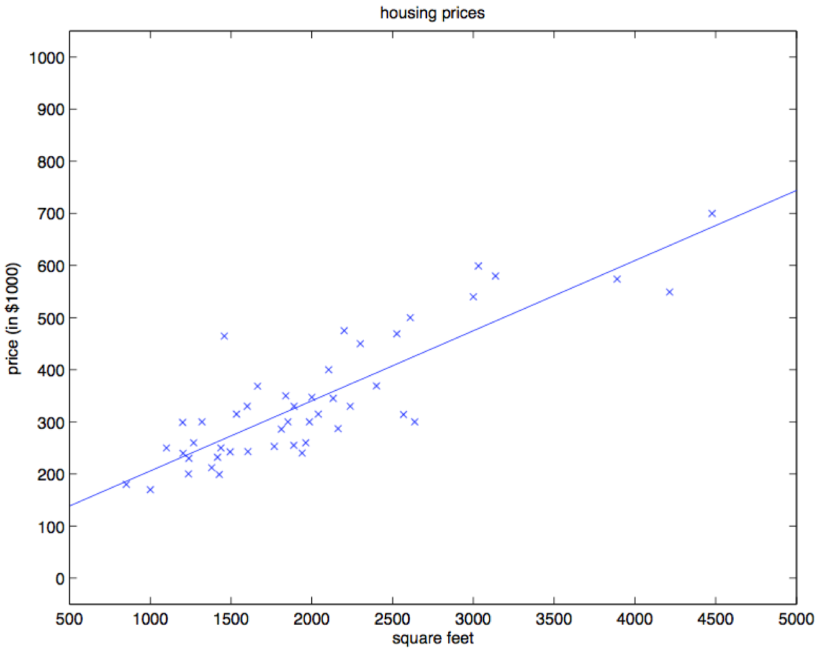
$$\mathcal{L}(\mathbf{w}) = p(\mathbf{y}|\mathbf{X};\mathbf{w}) = \prod_i p(y_i|\mathbf{x}_i;\mathbf{w})$$

- Select parameters that make data most likely
  - General strategy for parameter estimation

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} -\ln \mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} -\sum_i \ln p(y_i|\mathbf{x}_i;\mathbf{w})$$

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

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 $\mathbf{X} \in \mathbb{R}^{nxm}$
  - Target vector $\mathbf{y} \in \mathbb{R}^n$

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

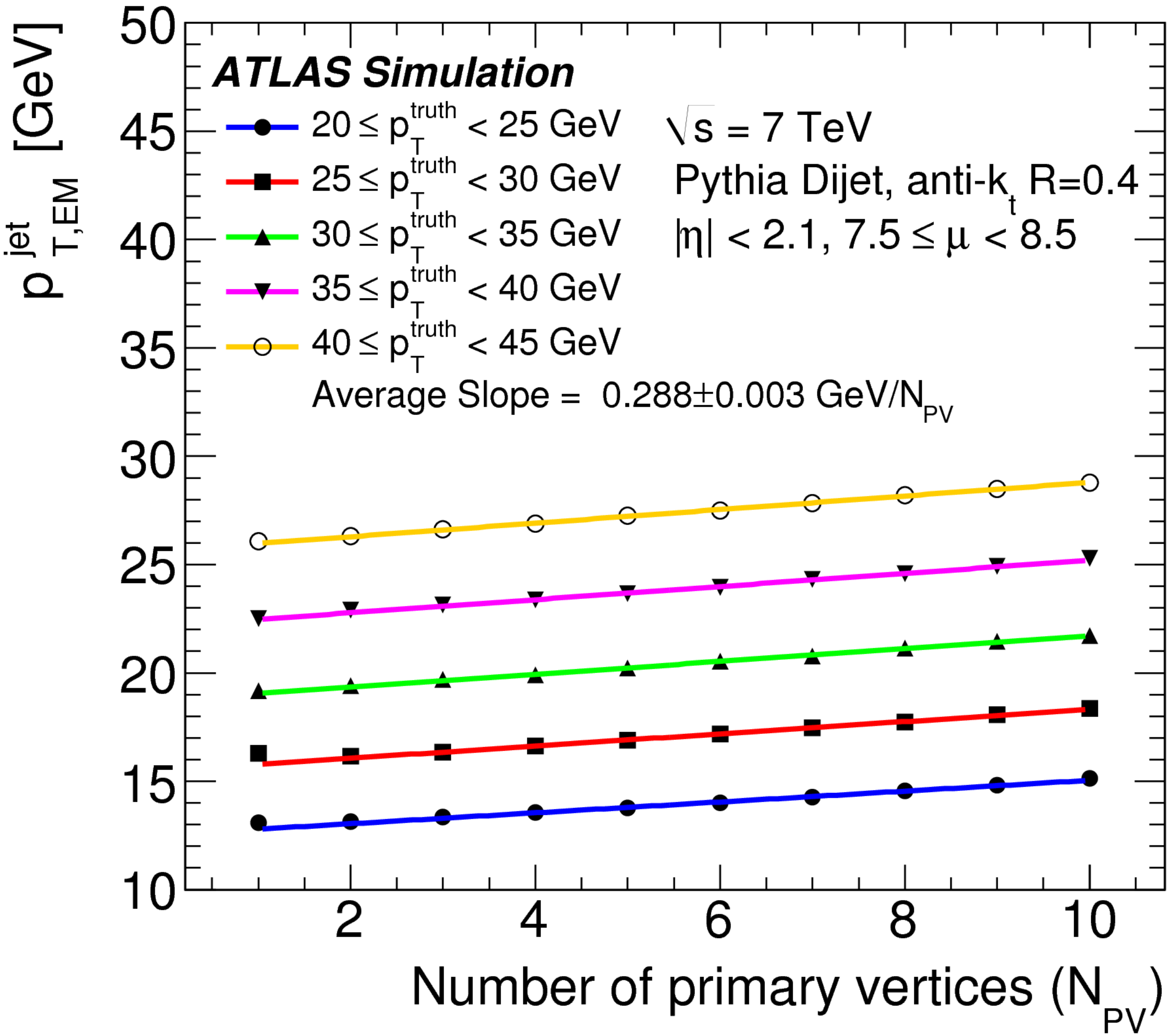
- Minimize w.r.t. **w**: $$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \arg\min_{\mathbf{w}} L(\mathbf{w})$$

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**: $$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \arg\min_{\mathbf{w}} L(\mathbf{w})$$
- What if we have correlated variables? *Multi-collinearity*
  - **X** is close to singular
  - Inverse is highly sensitive to random errors
- Hint: Regularization can help!

Linear Regression Example

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



- Reconstructed Jet energy vs. Number of primary vertices

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:

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

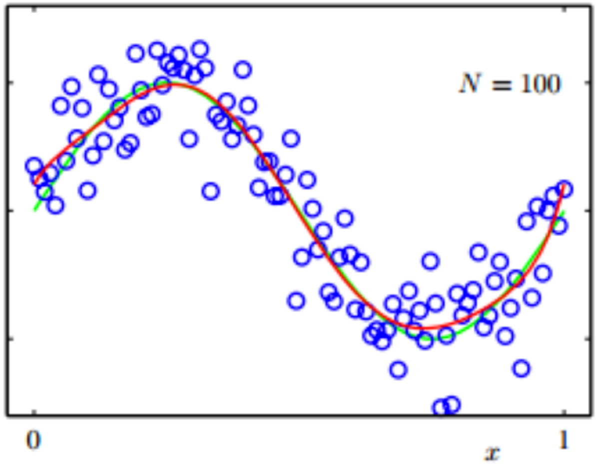
$$\rightarrow -\log L(m) \sim \sum_i (y_i - mx_i)^2$$

Squared loss function!

Why Take a Probabilistic Approach?

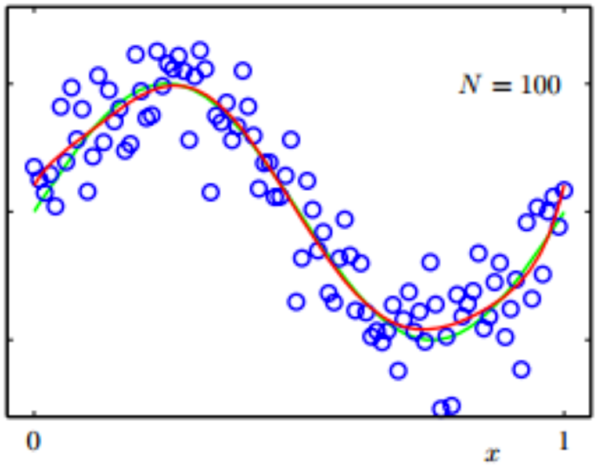
- Allows us to get calibrated estimates of $p(y|\mathbf{x})$
- Separates predictions from modeling
- A general framework for parameter estimation.
  - Can use to fit other parameters of the model.

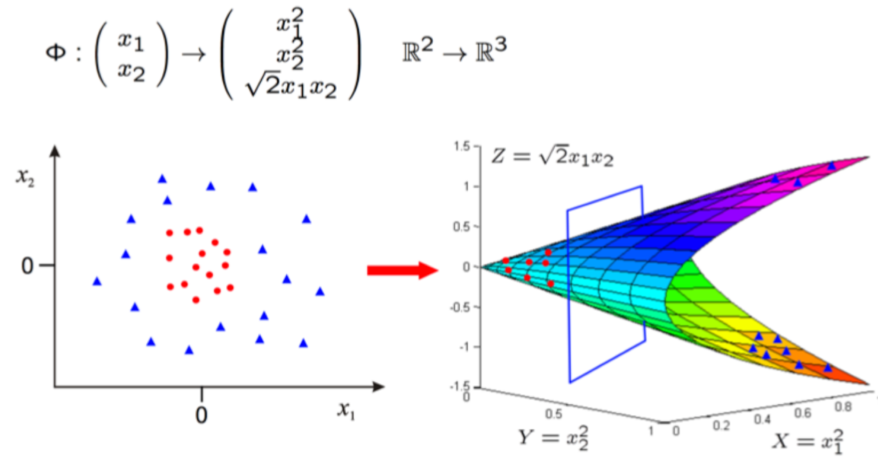
Basis Functions



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

Basis Functions





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

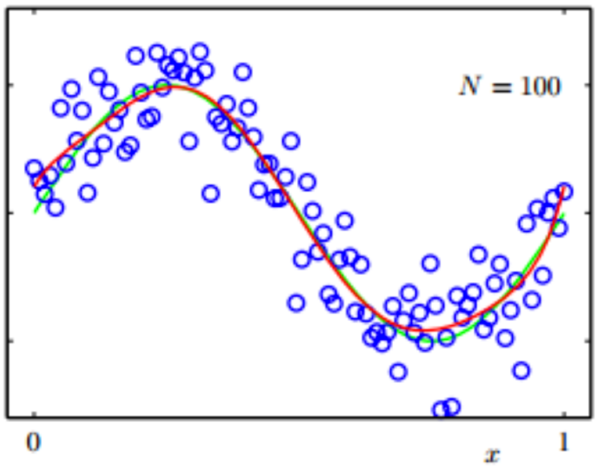
- Can choose basis functions $\phi(x)$ to form new features

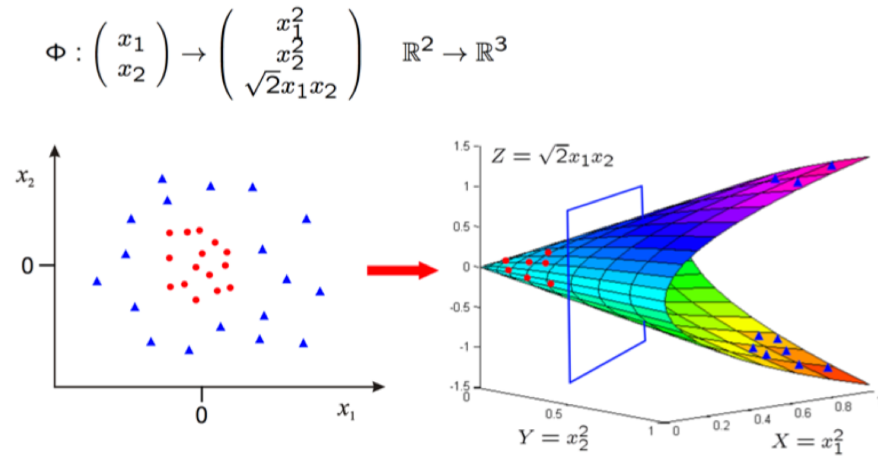
$$\mathbf{y}_i = \mathbf{w}^T \phi(x_i)$$

  - Polynomial basis $\phi(x) \sim \{1, x, x^2, x^3, \ldots\}$, Gaussian basis, …

  - **Linear regression on new features $\phi(x)$**

Basis Functions





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

- Can choose basis functions $\phi(x)$ to form new features

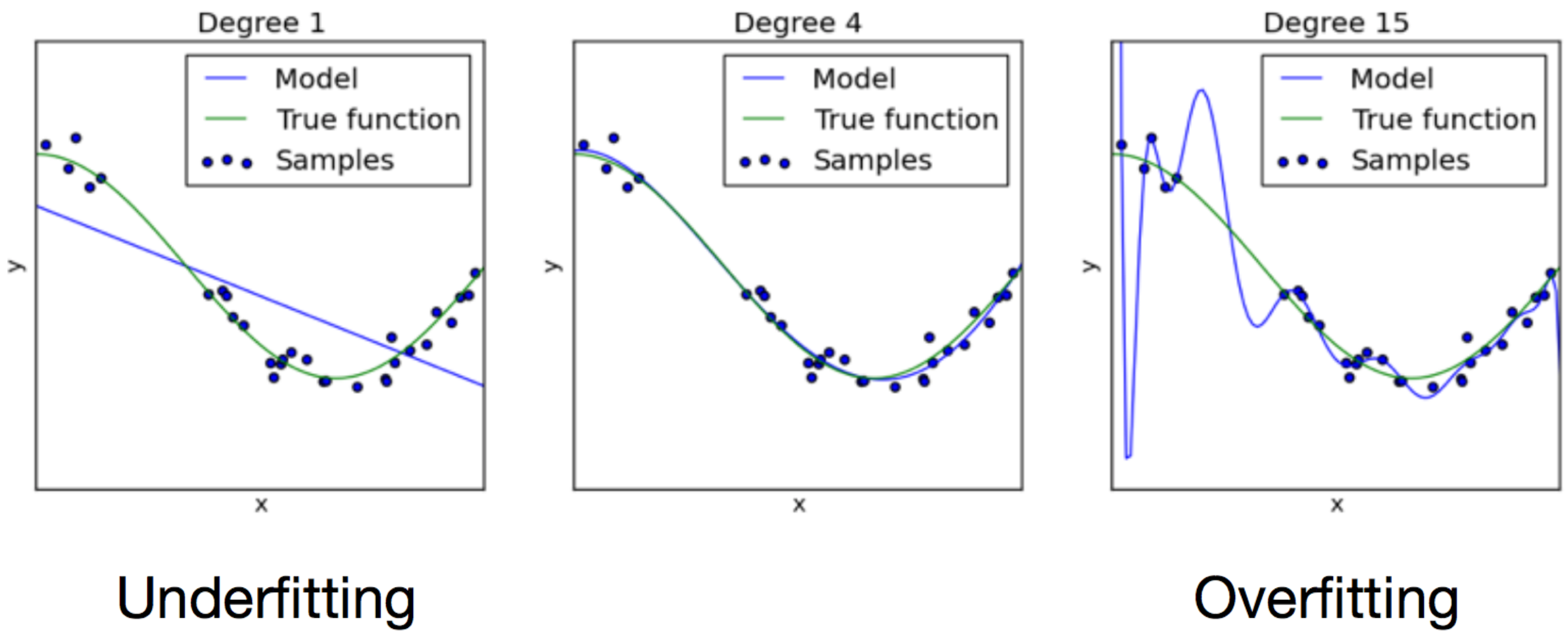
$$\mathbf{y}_i = \mathbf{w}^T \phi(x_i)$$

  - Polynomial basis $\phi(x) \sim \{1, x, x^2, x^3, \ldots\}$, Gaussian basis, …

  - Linear regression on new features $\phi(x)$

- 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

- 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

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

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

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

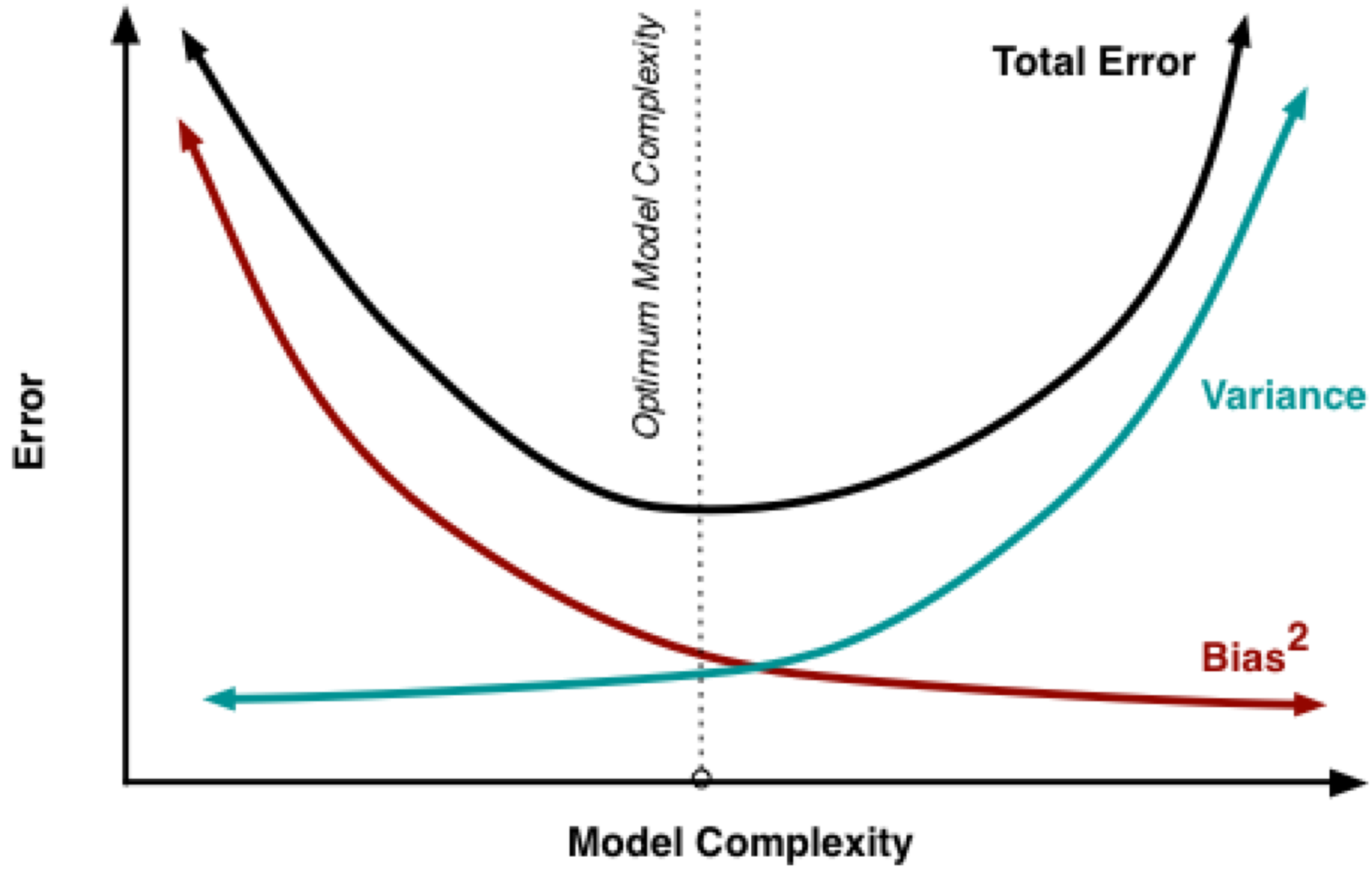
- Examining generalization error at x, w.r.t. possible training datasets

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

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

- **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**

Bias Variance Tradeoff



Regularization

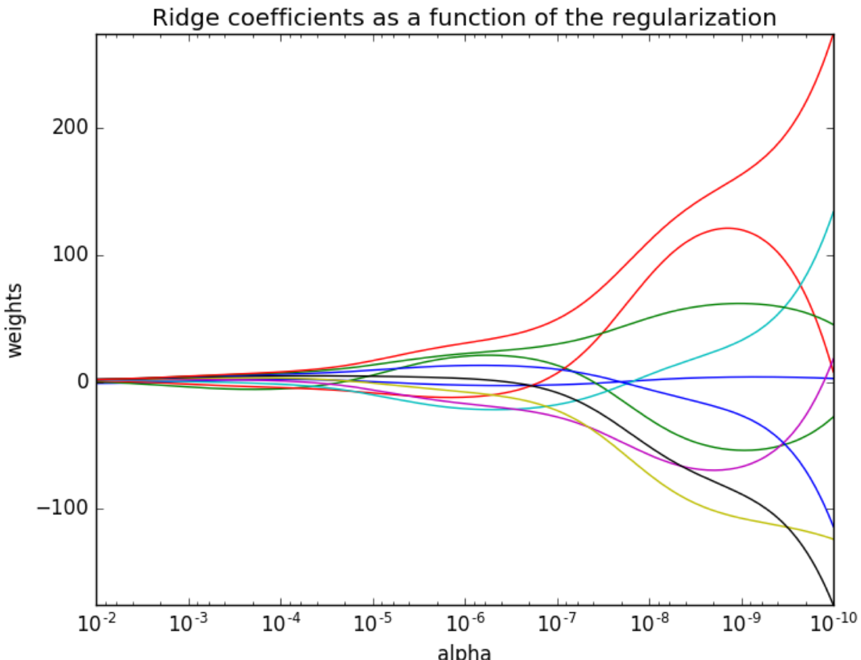
- Can control the complexity of a model by placing **constraints on the model parameters**
  - Trading some bias to reduce model variance
- **L2 norm**: $\Omega(\mathbf{w}) = ||\mathbf{w}||^2 = \sum_i w_i^2$
  - "Ridge regression", enforcing weights not too large
  - Equivalent to Gaussian prior over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

Regularized Linear Regression

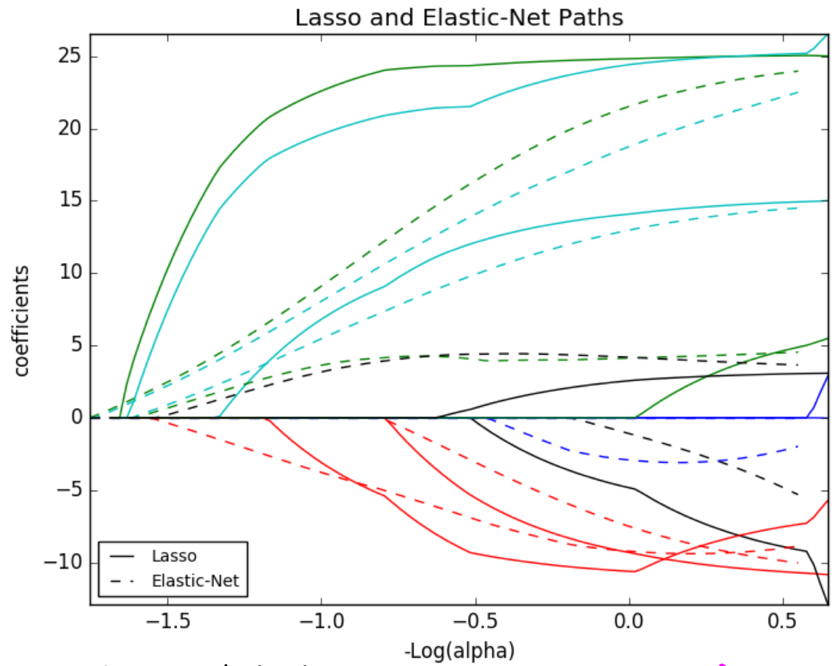
$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^2 + \alpha\Omega(\mathbf{w})$$

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

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



Less regularization



Less regularization

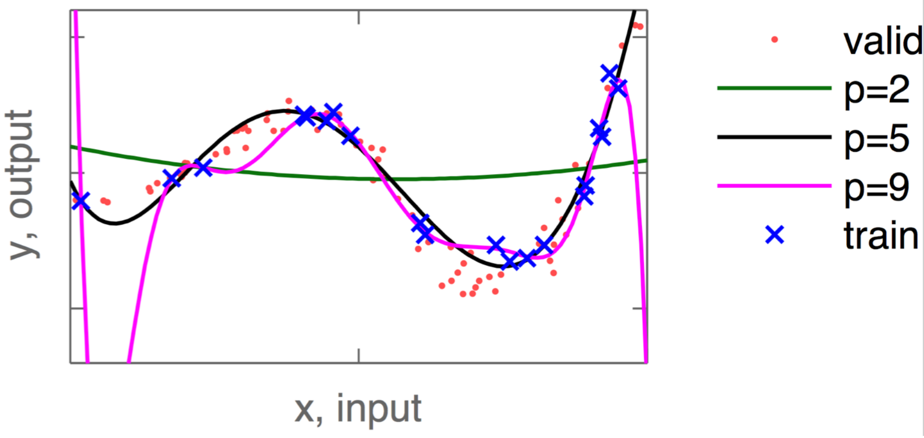
- L2 keeps weights small, L1 keeps weights sparse!
- But how to choose hyperparameter α?

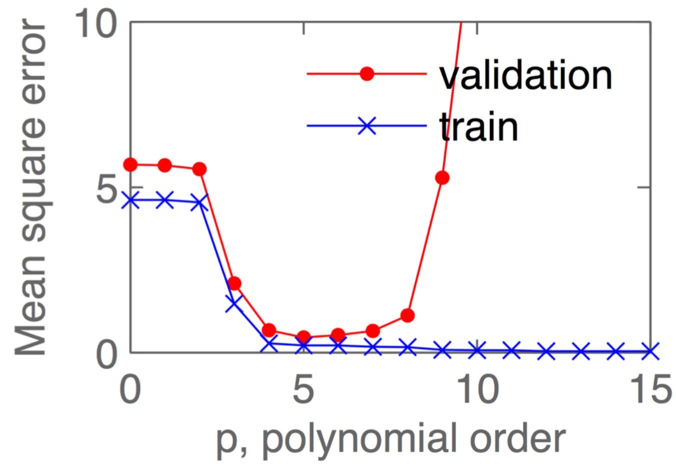
http://scikit-learn.org/

How to Measure Generalization Error?

Training set | Validation set | Test set

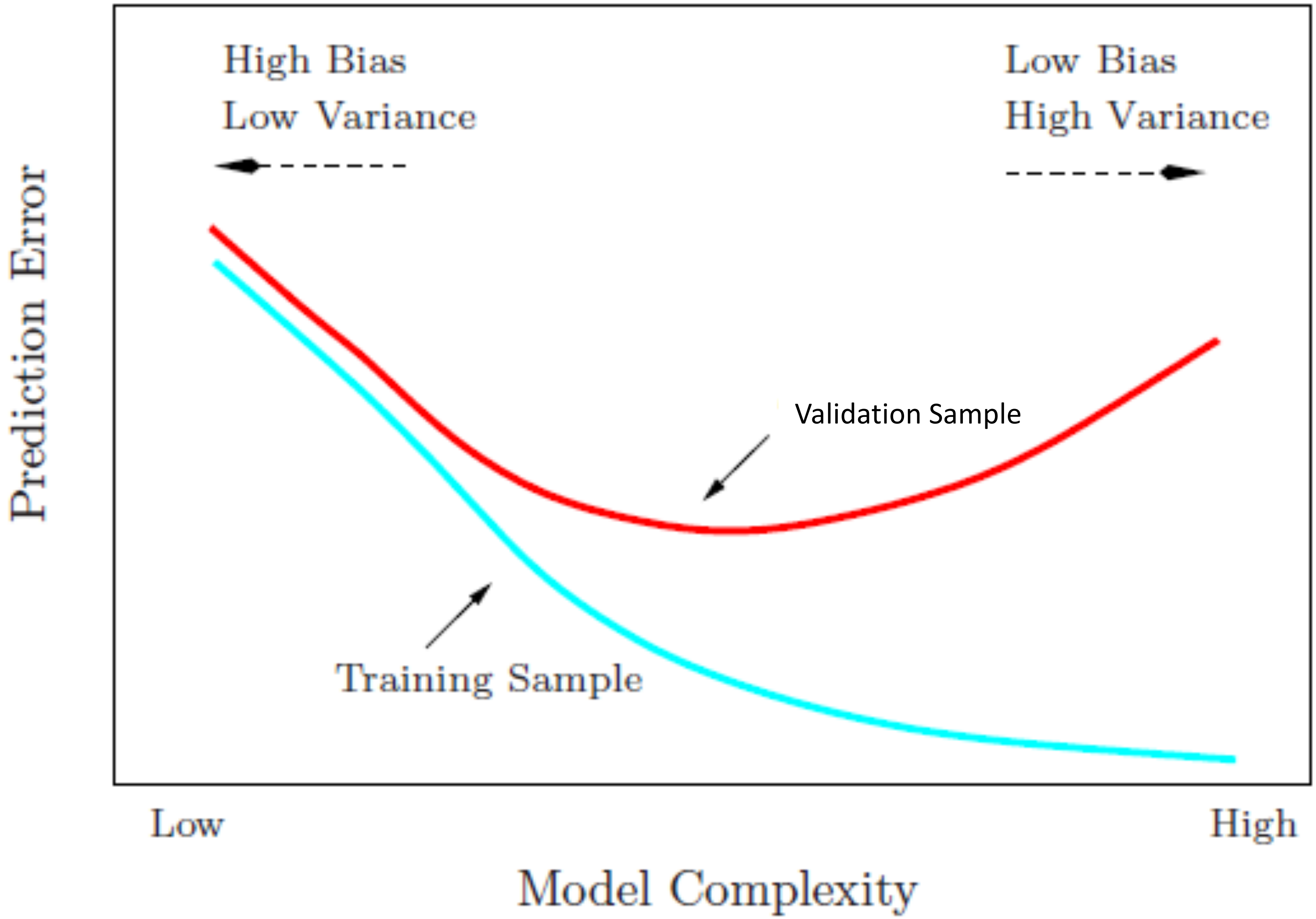
- Split dataset into multiple parts
- **Training set**
  - Used to fit model parameters
- **Validation set**
  - Used to check performance on independent data and tune hyper parameters
- **Test set**
  - final evaluation of performance after all hyper-parameters fixed
  - Needed since we tune, or "peek",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



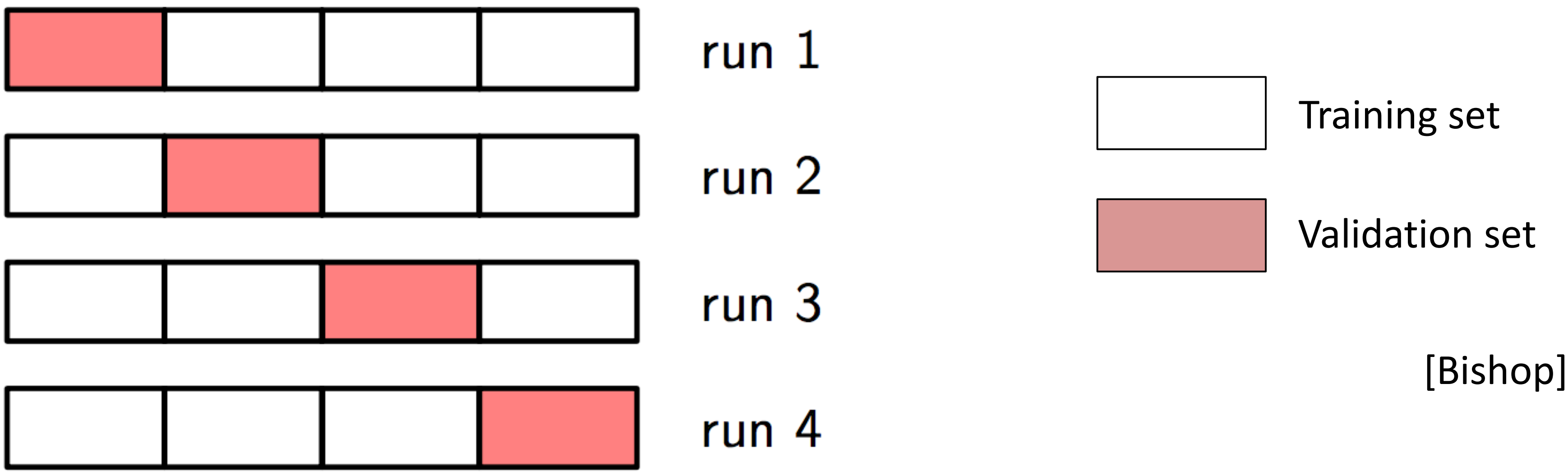


[Murray]

How to Measure Generalization Error?



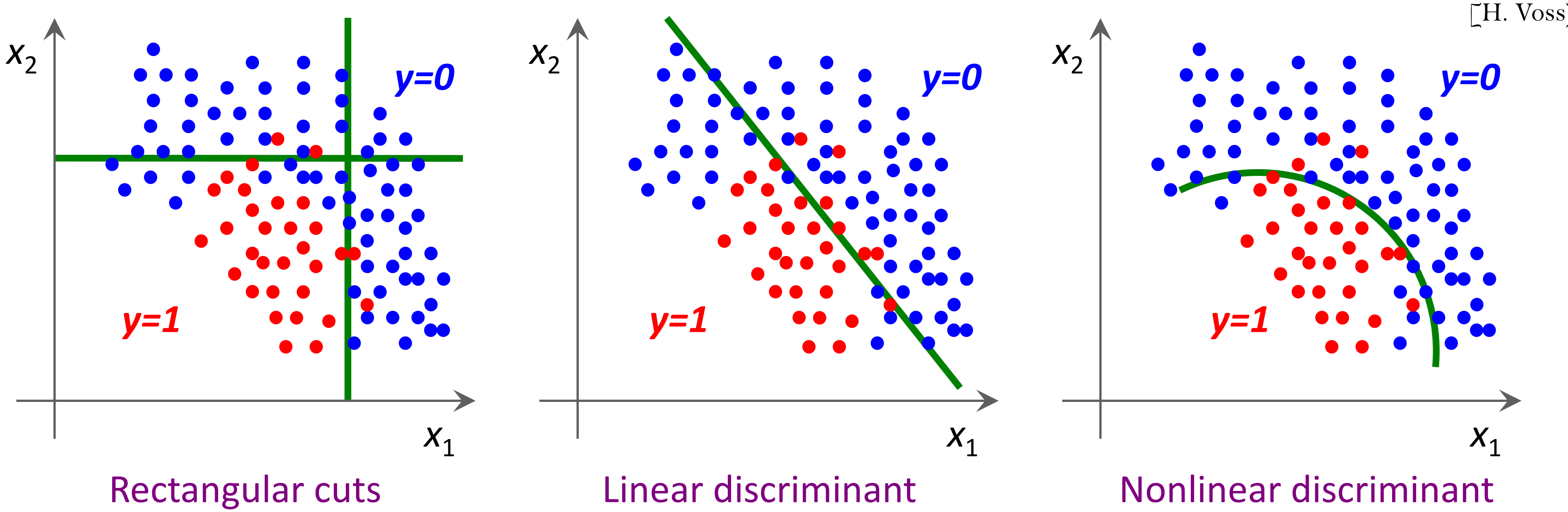
Cross Validation



- 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

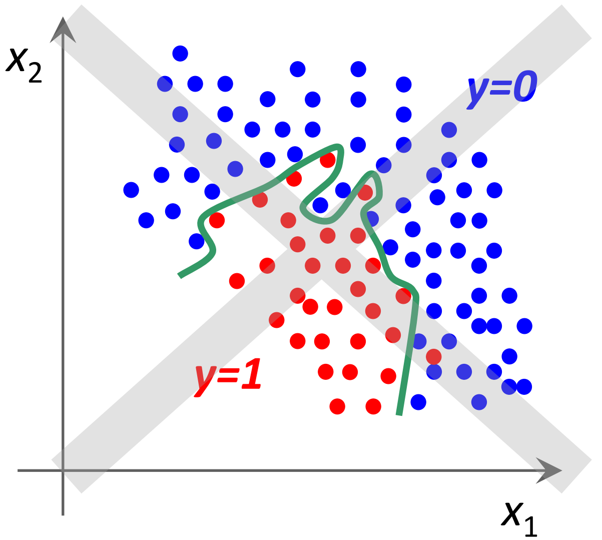
Classification

[H. Voss]



- Learn a function to separate different classes of data

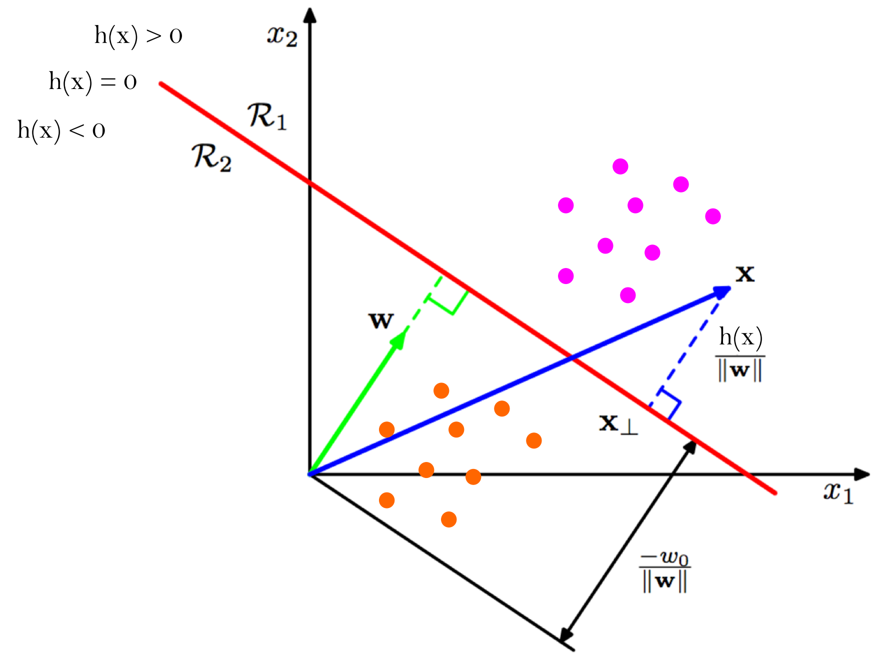
- Avoid over-fitting:
  - Learning too fined details about your training sample that will not generalize to unseen data



Linear Decision Boundaries

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

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

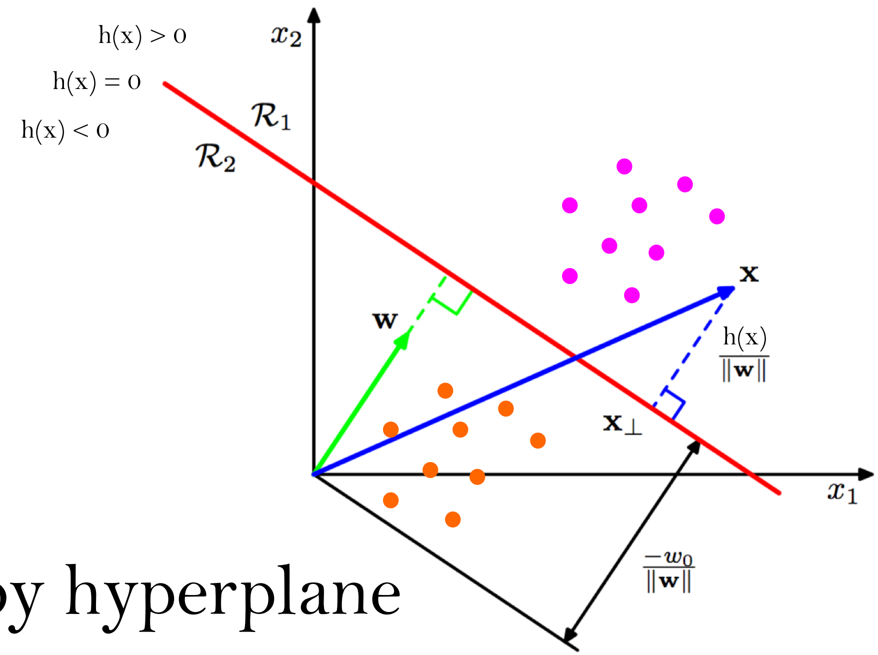


[Bishop]

Linear Decision Boundaries

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

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



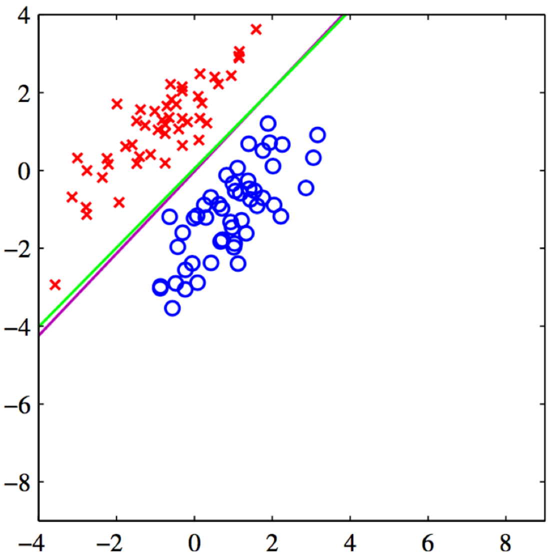
[Bishop]

- **Decision boundary** defined by hyperplane

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

  - Boundary is perpendicular to weight vector **w**
- Classifier Score($\mathbf{x}_i$) = $h(\mathbf{x}_i; \mathbf{w})$
- Class predictions: Predict class -1 if $h(\mathbf{x}_i ; \mathbf{w}) < 0$, else class 1

Linear Classifier with Least Squares?

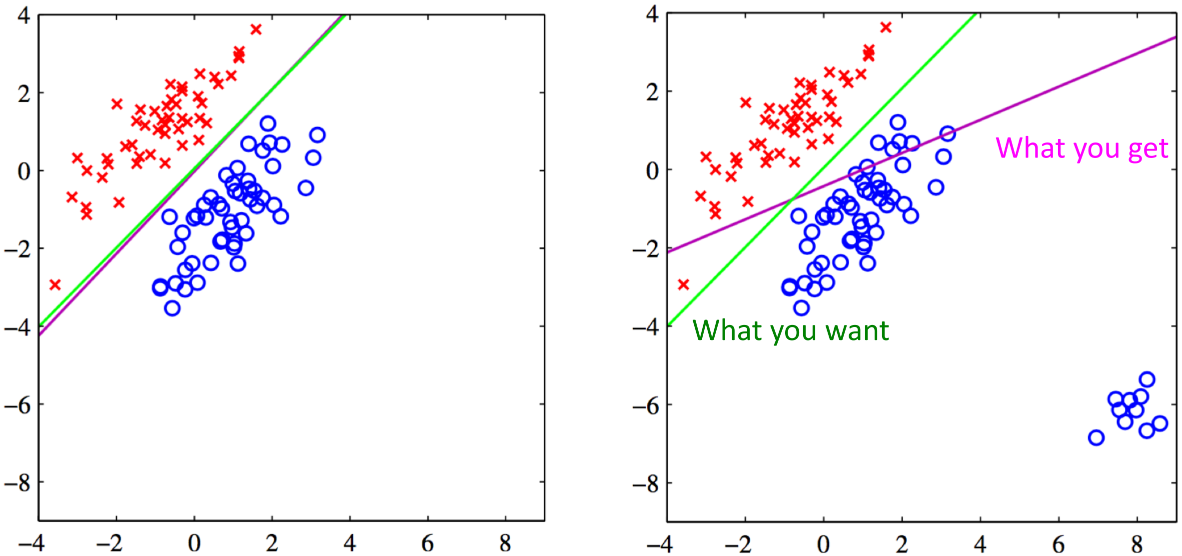


$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

[Bishop]

- Why not use least squares loss with binary targets?

Linear Classifier with Least Squares?

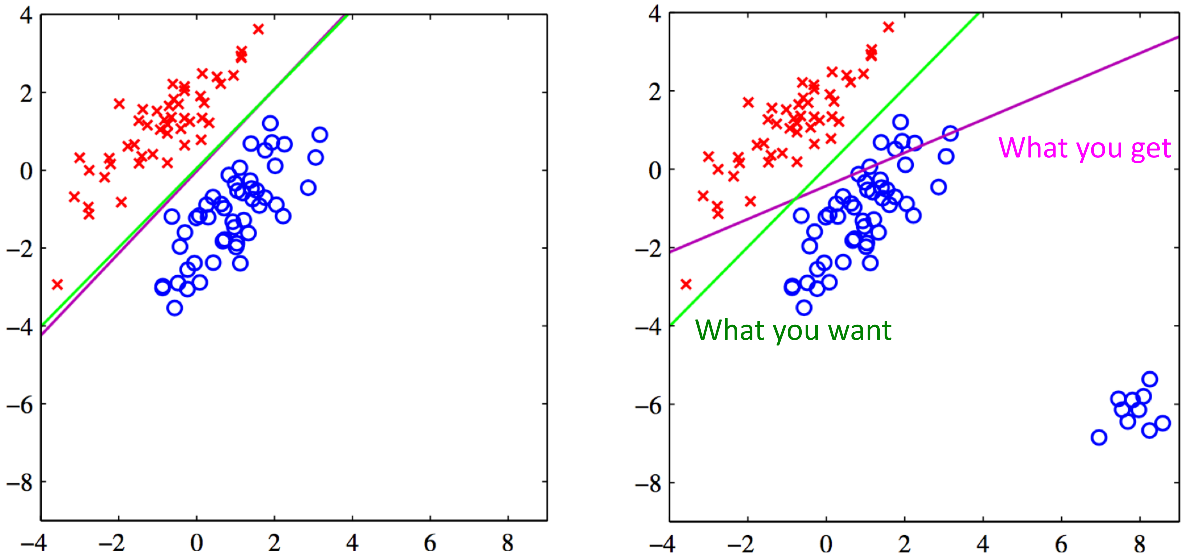


$$L(\mathbf{w}) = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2$$

[Bishop]

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

Linear 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
- Use only class labels?
  - Perceptron algorithm (not covered here)
- A probabilistic approach?

Logistic Regression for Classification

- Set of input / output pairs $D = \{\mathbf{x}_i , y_i\}_{i=1...n}$
  - $\mathbf{x}_i \in \mathbb{R}^m$
  - $y_i \in \{0, 1\}$

- Linear discriminant: $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}$

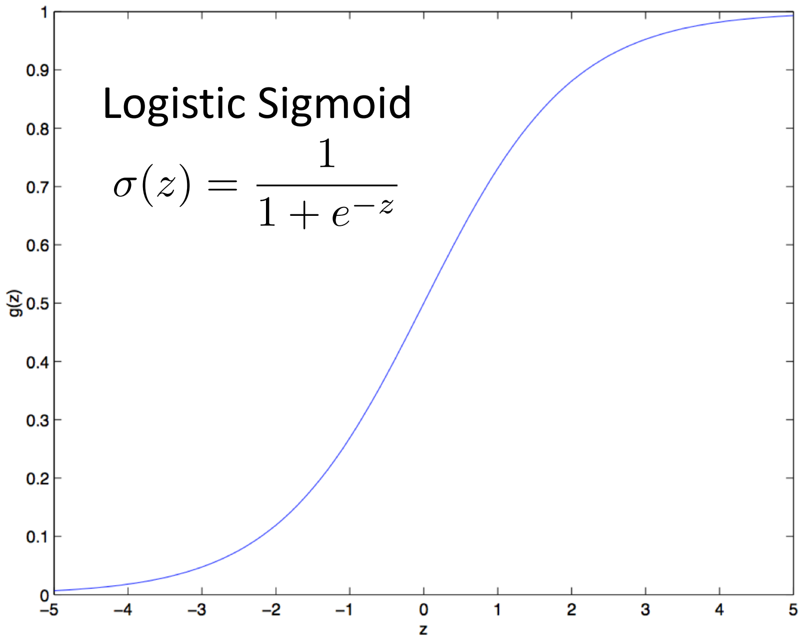
Logistic Regression for Classification

- Set of input / output pairs $D = \{\mathbf{x}_i, y_i\}_{i=1\ldots n}$
  - $\mathbf{x}_i \in \mathbb{R}^m$
  - $y_i \in \{0, 1\}$

- Linear discriminant: $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}$

- Model per example probability: $p(y = 1|\mathbf{x}) \equiv p_i = \dfrac{1}{1 + e^{-h(\mathbf{x};\mathbf{w})}}$

$$= \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$$



NOTE:
Not a random choice,
Natural choice for large class of models

See backups for more info

Logistic Regression for Classification

- Set of input / output pairs $D = \{\mathbf{x}_i, y_i\}_{i=1\ldots n}$
  - $\mathbf{x}_i \in \mathbb{R}^m$
  - $y_i \in \{0, 1\}$
- Linear discriminant: $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}$
- Model per example probability: $p(y = 1|\mathbf{x}) \equiv p_i = \dfrac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$
  - The farther from boundary $\mathbf{w}^T\mathbf{x}=0$, the more certain about class
  - Class decision rule: choose class 0 if $p_i<0.5$, else choose class 1

Logistic Regression for Classification

- Set of input / output pairs $D = \{\mathbf{x}_i, y_i\}_{i=1\ldots n}$
  - $\mathbf{x}_i \in \mathbb{R}^m$
  - $y_i \in \{0, 1\}$

- Linear discriminant: $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x}$

- Model per example probability: $p(y = 1|\mathbf{x}) \equiv p_i = \dfrac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}}$
  - The farther from boundary $\mathbf{w}^T\mathbf{x}=0$, the more certain about class
  - Class decision rule: choose class 0 if $p_i<0.5$, else choose class 1

- Concisely write $p(y|\mathbf{x})$ as Bernoulli random variable:

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

Logistic Regression

- Negative log-likelihood

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

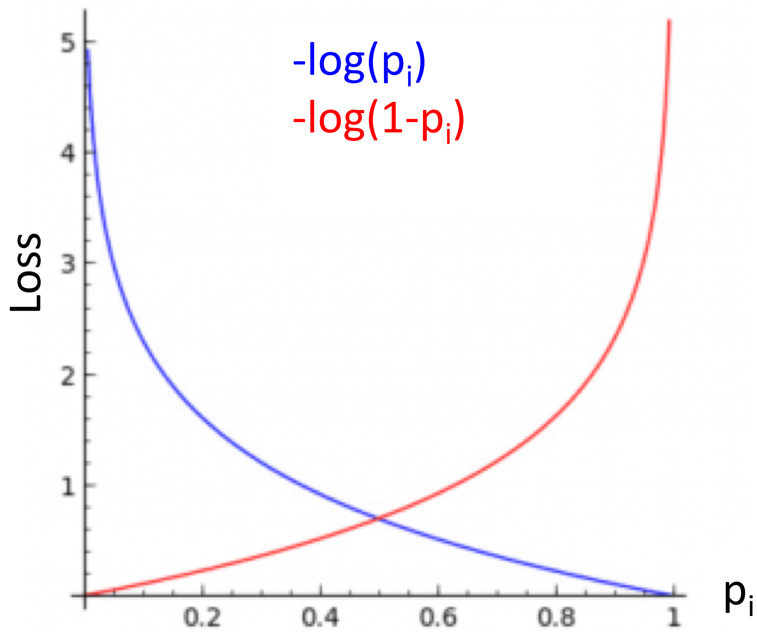
Logistic Regression

- 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

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

binary cross entropy loss function!

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

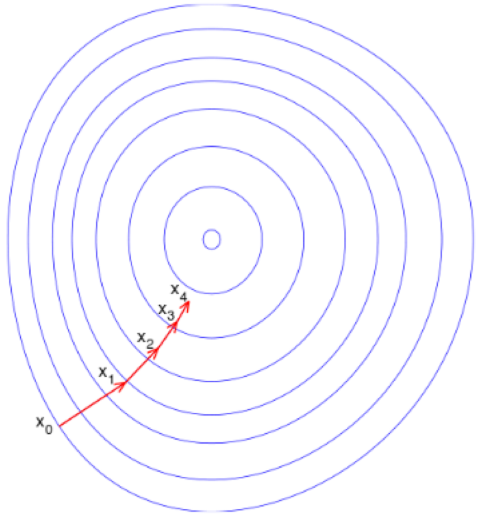
$$= \sum_i y_i \ln(1+e^{-\mathbf{w}^T\mathbf{x}}) + (1-y_i)\ln(1+e^{\mathbf{w}^T\mathbf{x}})$$

- No closed form solution to $\mathbf{w}^* = \arg\min_{\mathbf{w}}$ -ln L

- How to solve for **w**?

Gradient Descent

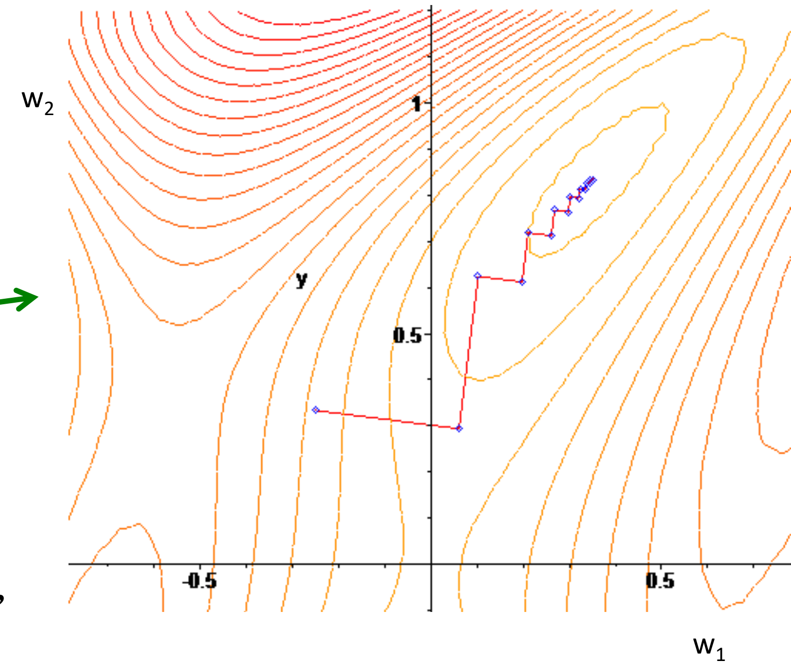
- Many methods to solve, lets use **Gradient Descent**
- Minimize loss by repeated gradient steps (when no closed form)
  - Compute gradient w.r.t. parameters: $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$
  - Update parameters $\mathbf{w}' \leftarrow \mathbf{w} - \eta \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}}$
  - η is called the learning rate, controls how big of a gradient step to take

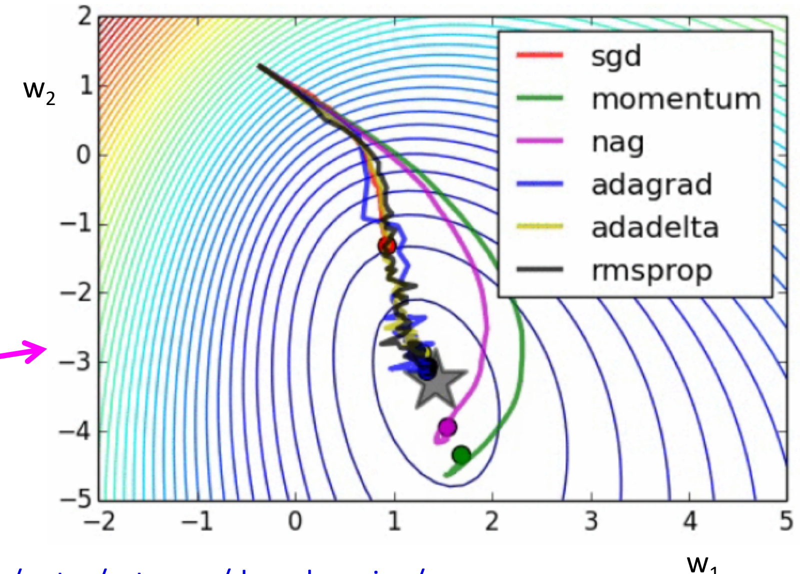


Stochastic Gradient Descent and Variants

- Gradient descent is computationally costly (since we compute gradient over full training set)

- Stochastic gradient descent
  - Compute gradient on one event at a time (in practice a small batch)
  - Noisy estimates average out
  - Stochastic behavior can allow "jumping" out of bad critical points
  - Scales well with dataset and model size
  - But can have some convergence difficulties
  - Improvements include: Momentum, RMSprop, AdaGrad, …





http://danielnouri.org/notes/category/deep-learning/

Gradient Descent for Logistic Regression

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

- Derivative of sigmoid: $\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1-\sigma(z))$

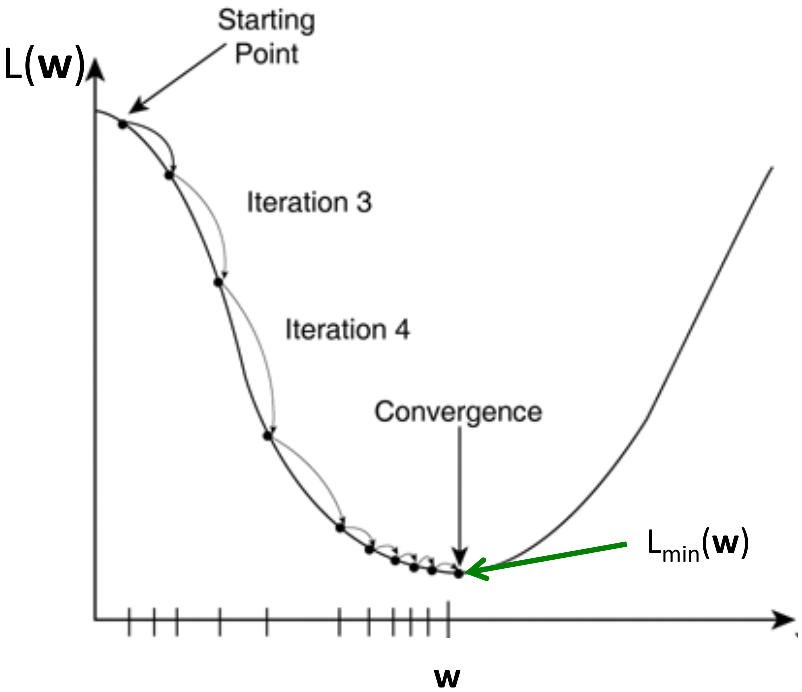
- Derivative of Loss: $\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_i (\sigma(\mathbf{w}^T\mathbf{x}) - y_i)\mathbf{x}_i$

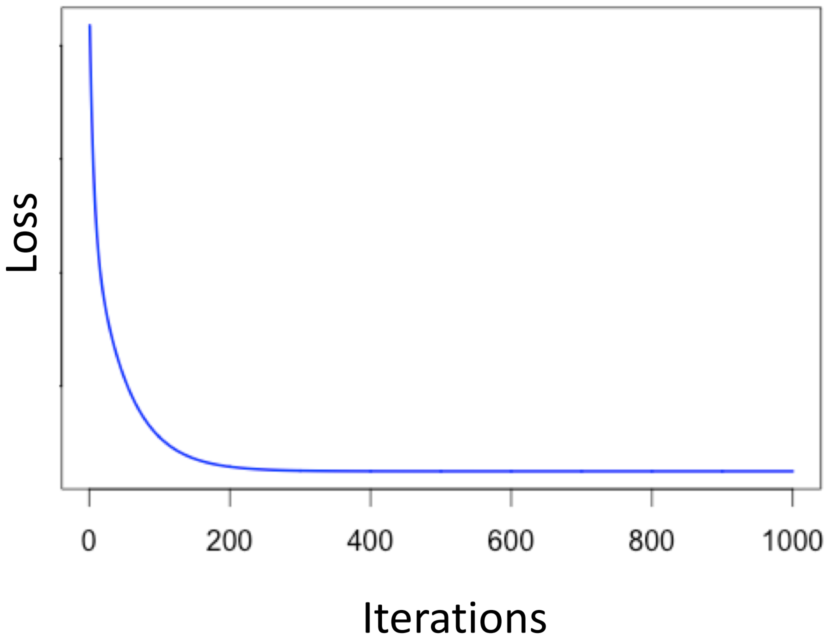
- Update rule:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w} - \eta \sum_i (\sigma(\mathbf{w}^T\mathbf{x}) - y_i)\mathbf{x}_i$$

- Repeat until parameters stable

Gradient Descent

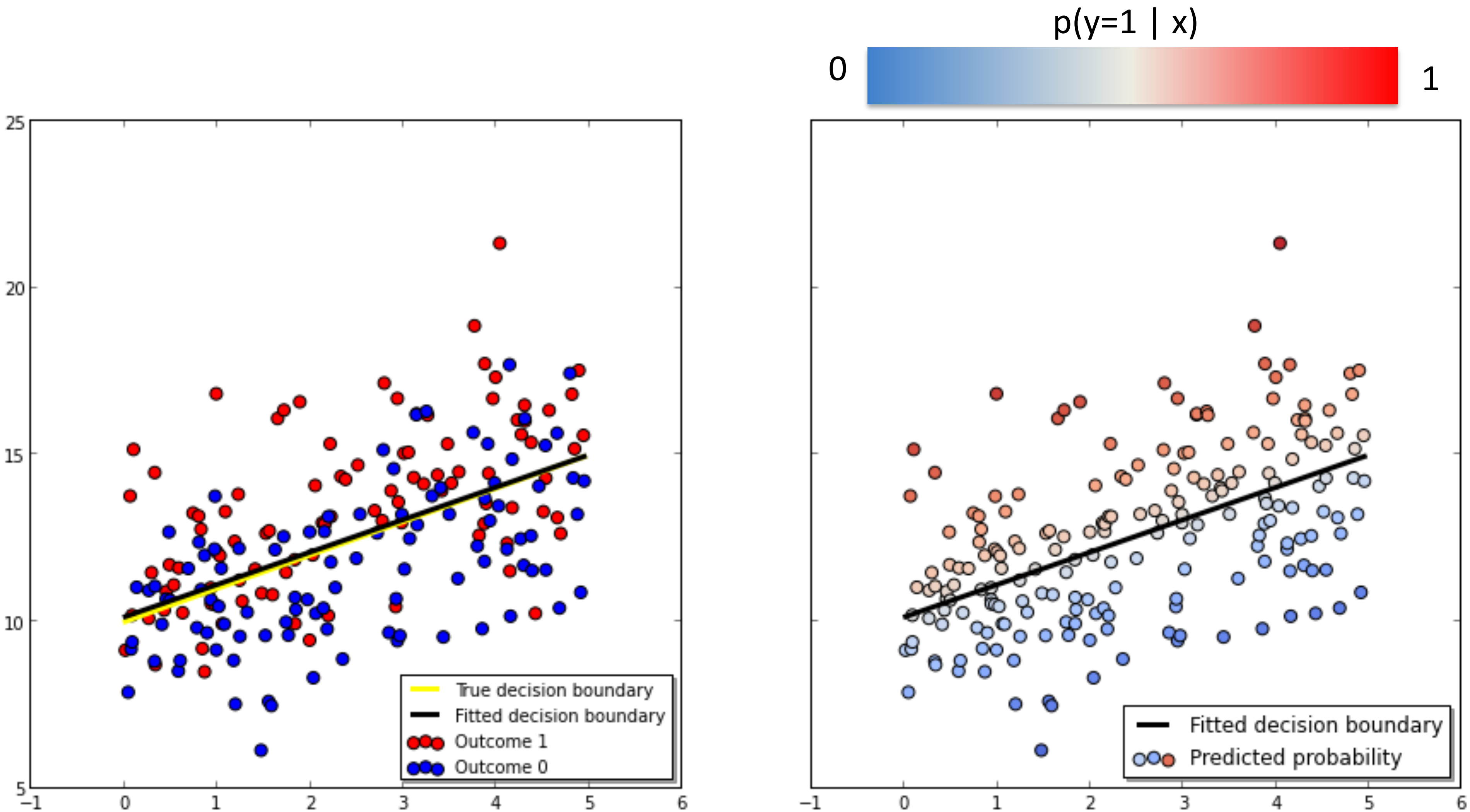




- Loss is convex
  - Single global minimum

- Iterations lower loss and move toward minimum

Logistic Regression Example

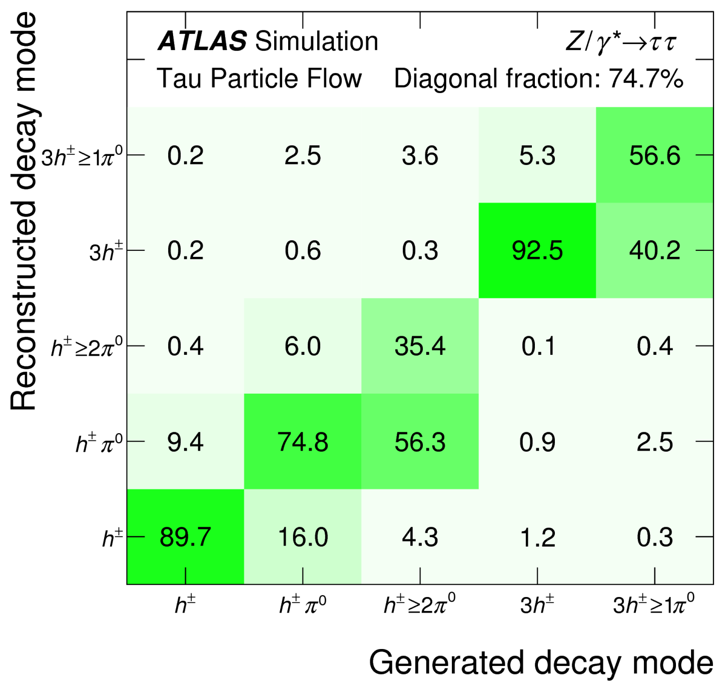


https://triangleinequality.wordpress.com/2013/12/02/logistic-regression/

Estimating a Classifier Performance

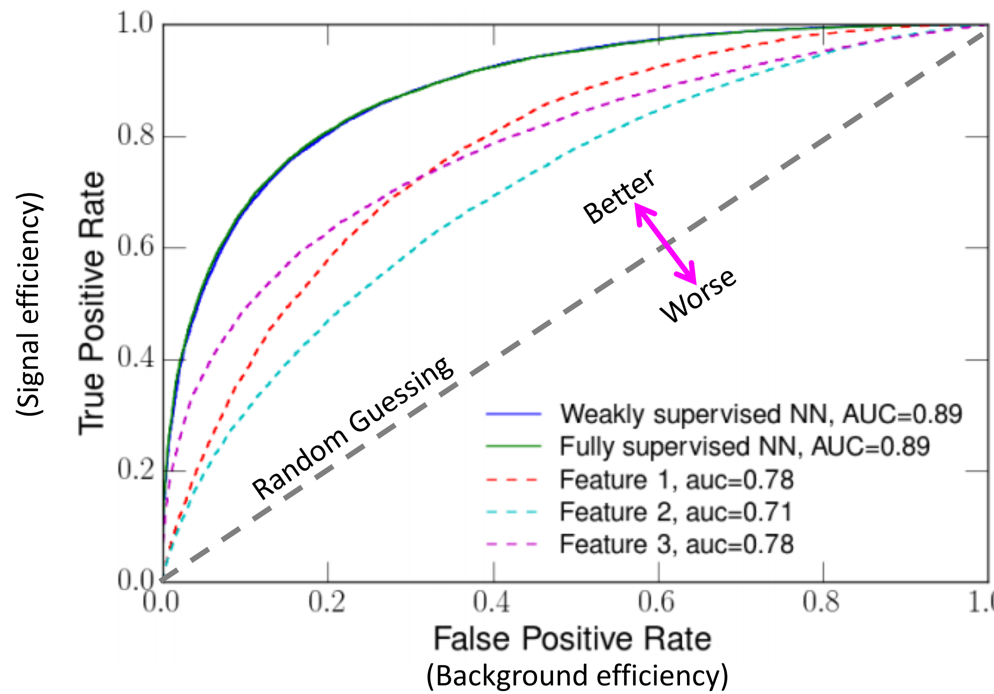
		Predicted Positive	Predicted Negative
True	Positive	True Positives (TP)	False Negatives (FN)
True	Negative	False Positives (FP)	True Negatives (TN)

Confusion Matrix
Classifying tau decays



arXiv:1512.05955

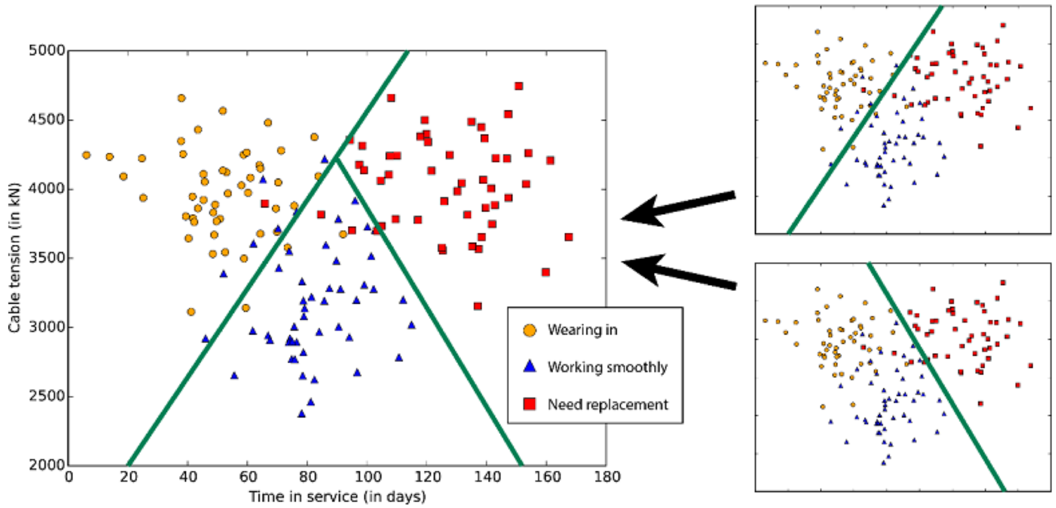
Receiver Operating Characteristic (ROC) Curve
classifying quarks vs. gluons



arXiv:1702.00414

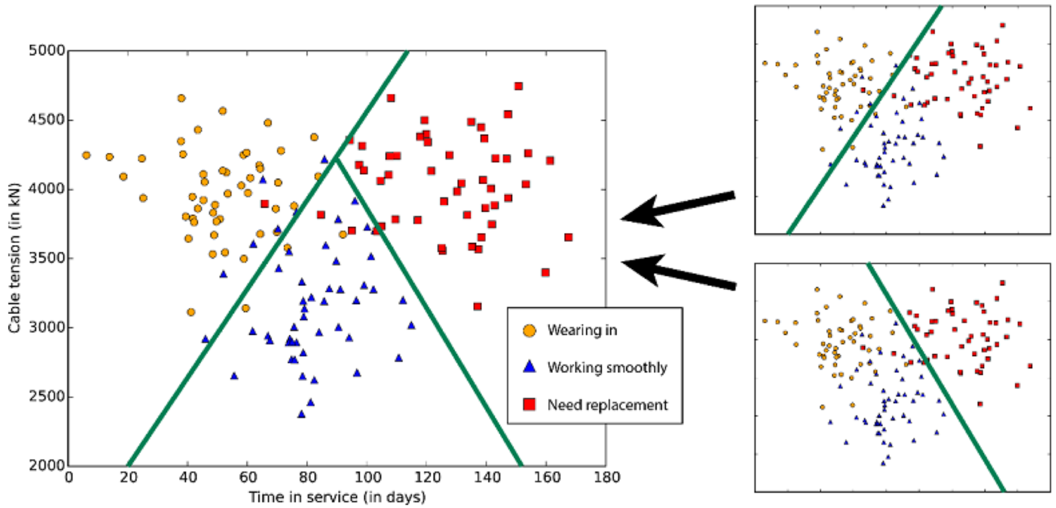
Multiclass Classification?

- 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 $\mathbf{w}_k$

Summary of Today

- 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*)

- Discussed linear models today
  - Many forms of linear models, we only touched the surface!

- Next time, some nonlinear models and unsupervised learning
  - Decision trees and ensemble methods
  - Neural network (intro)
  - Clustering
  - Dimensionality reduction

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
  - …

- **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/

Bayesian vs. Frequentist Models

- Mathematical models in ML typically described via random variables — in which case they are also called statistical models

- Statistical models typically specified by **unknown** parameters (to be learnt from data)

- **Frequentist:** there exist a "ground-truth" set of unknown parameters that are constant (i.e. not random)

- **Bayesian:** model parameters are themselves random, and typically specified by their own distribution/statistical model, with their own unknown "hyperparameters"

Probabilistic Motivation

- Posterior probability: $$p(y=1|\mathbf{x}) = \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=1)p(y=1)+p(\mathbf{x}|y=0)p(y=0)}$$ $$= \frac{1}{1+e^{-a(\mathbf{x})}} = \sigma(a(\mathbf{x}))$$ ← Logistic sigmoid

- Log-probability ratio: $$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=0)p(y=0)}$$

- In a large class of models a($\mathbf{x}$) is linear

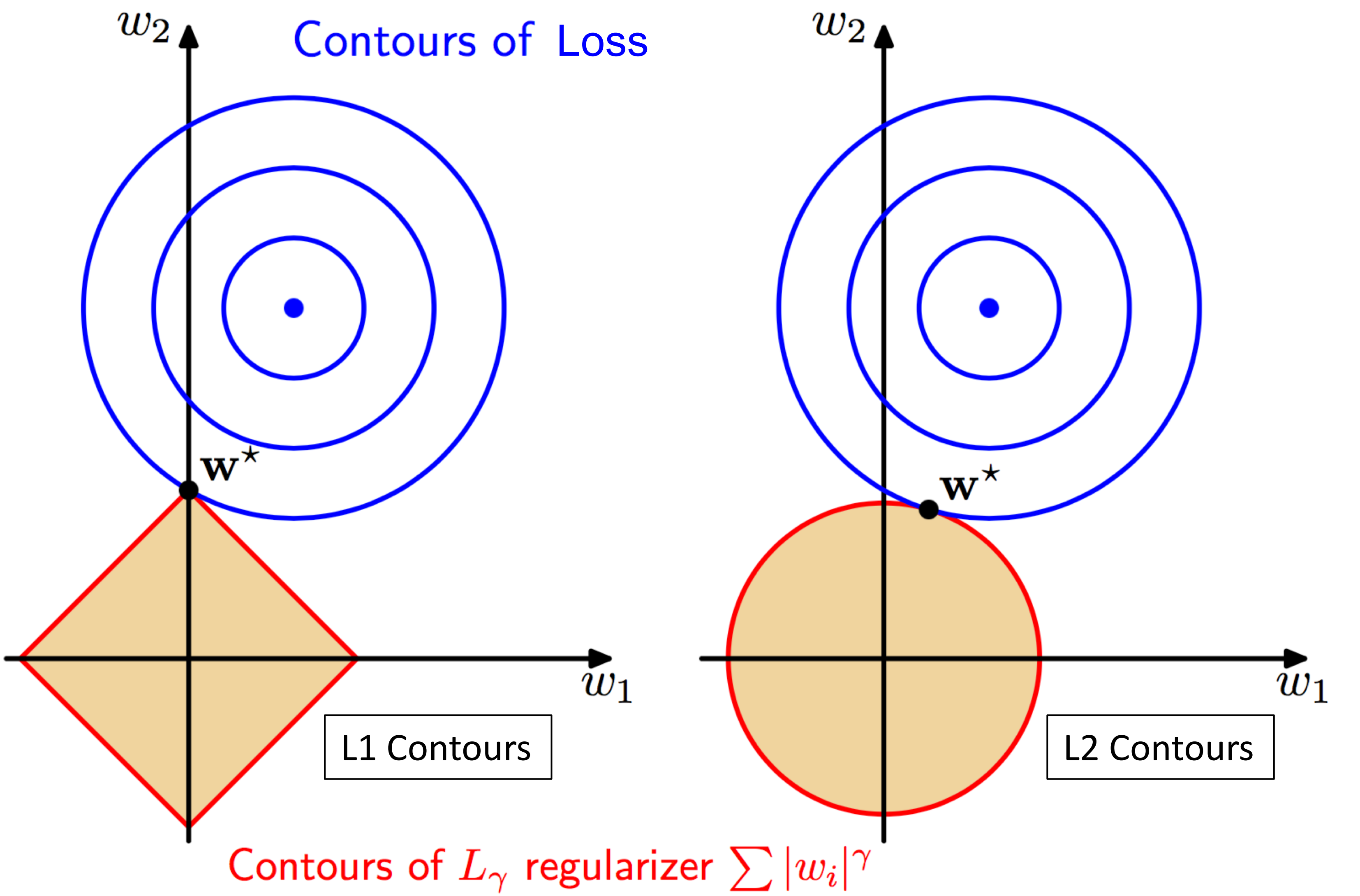
$$a(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$$

  - When class-conditional density p($\mathbf{x}$ | y) is in the exponential family of Generalized Linear Models,
    - Includes Gaussian, Exponential, Poisson, Beta, …

- Have linear discriminant and estimate of per-class probability

- Even if p($\mathbf{x}$ | y) unknown, motivation to model p(y | $\mathbf{x}$) with logistic sigmoid

Regularization

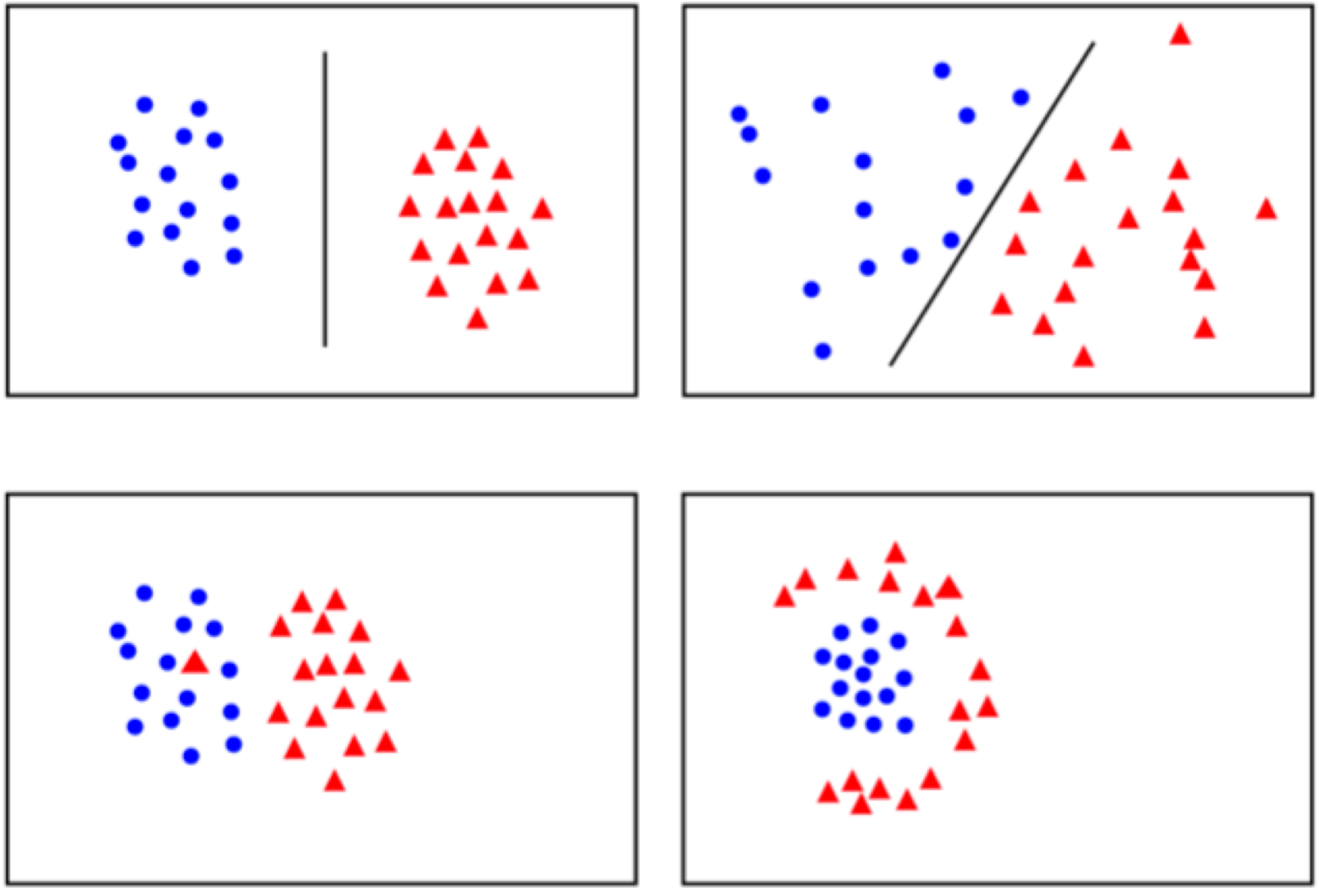


Pattern Recognition and Machine Learning C. M. Bishop (2006)

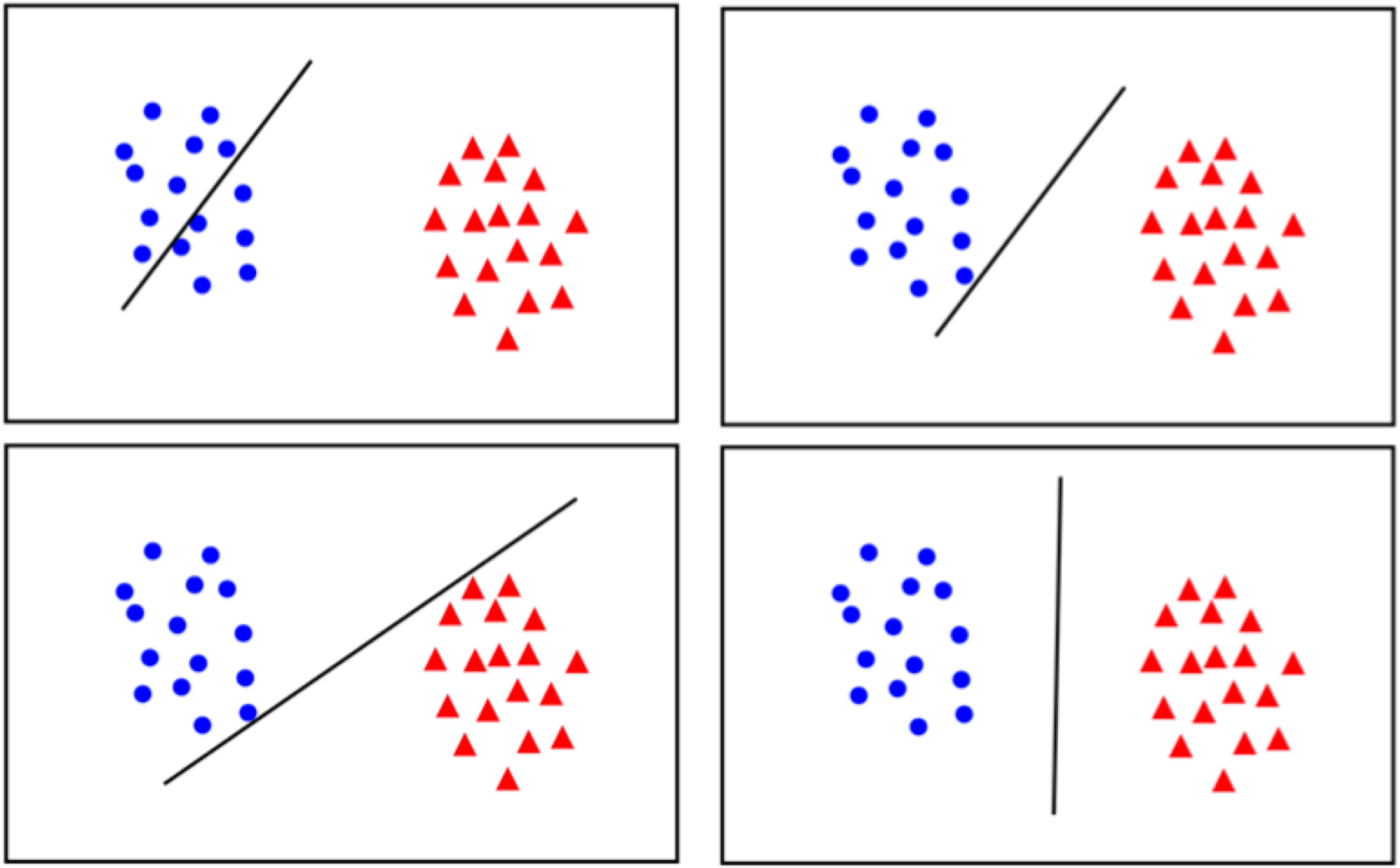
Linear Separability

linearly separable

not linearly separable



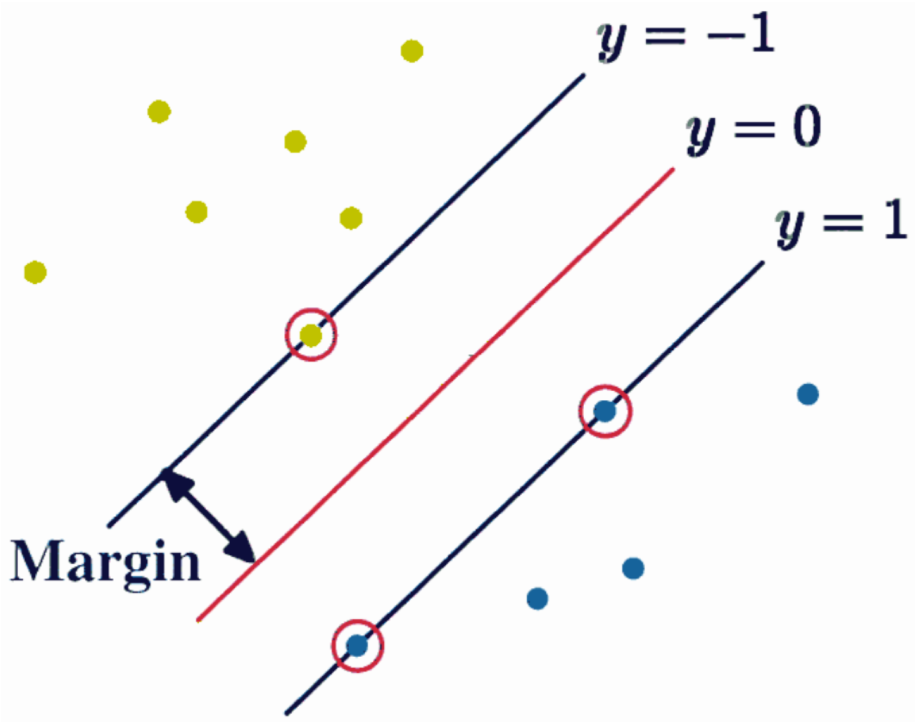
Decision Boundaries – Which is Best?



Maximum Margin Classifiers

- Many possible solutions to separating classes
  - Depends on the loss function chosen

- Assuming classes are linearly separable, what if we wanted to solution with the maximum distance between the decision boundary and the nearest data point?



Maximum Margin Classifier

- Assume we have:
  - x in R^d
  - y in {-1, 1}

- Linear classifier: $h(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T\mathbf{x} + w_0$

- Distance of data point, $\mathbf{x}_i$, to decision boundary $\dfrac{y_i(\mathbf{w}^T\mathbf{x}_i + w_0)}{\sqrt{\mathbf{w}^T\mathbf{w}}}$

- Optimization problem:

$$\arg\max_{\mathbf{w},\ w_0} \left\{ \frac{1}{\sqrt{\mathbf{w}^T\mathbf{w}}} \ \min_i y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \right\} \longrightarrow \arg\min_{\mathbf{w}, w_0} \ \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

$$s.\ t.\ \ y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 \text{ for all } i$$

  - Can solve with gradient descent methods!

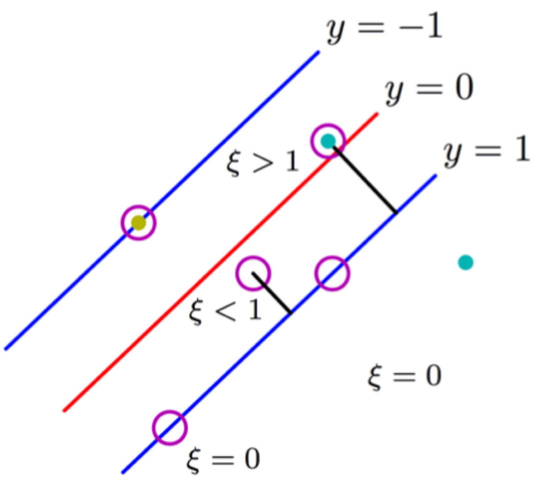
What if points not linearly separable?

$$\arg\min_{\mathbf{w},w_0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$

$$s.\ t.\quad y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i \text{ for all } i$$

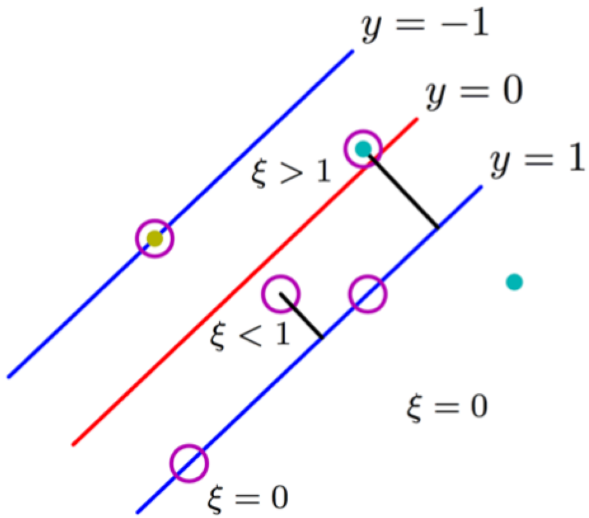
$$\text{and}\quad \xi_i \geq 0$$

- Add a smearing to the margin, $\xi \geq 0$
  - If $\xi = 0$, example correctly classifier
  - If $0 < \xi < 1$, example correctly classified, but in margin
  - If $\xi > 1$, example incorrectly classified



- Add regularizer to problem to constrain ξ_i not too large
  - C is the regularization hyperparameter that controls how much "softening" of the boundary is allowed, thus how big is margin

What if points not linearly separable?

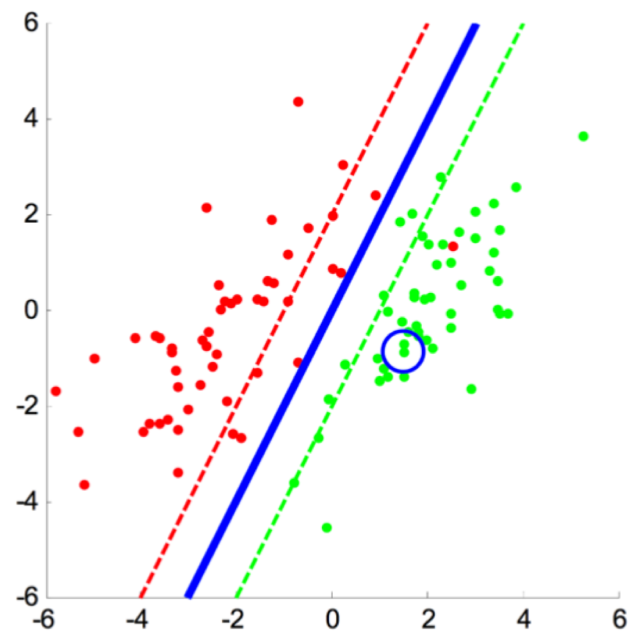


$$\arg\min_{\mathbf{w}, w_0} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} \ + \ C\sum_i \xi_i$$

$$s.\ t.\ \ y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq \ 1 - \xi_i \text{ for all } i$$

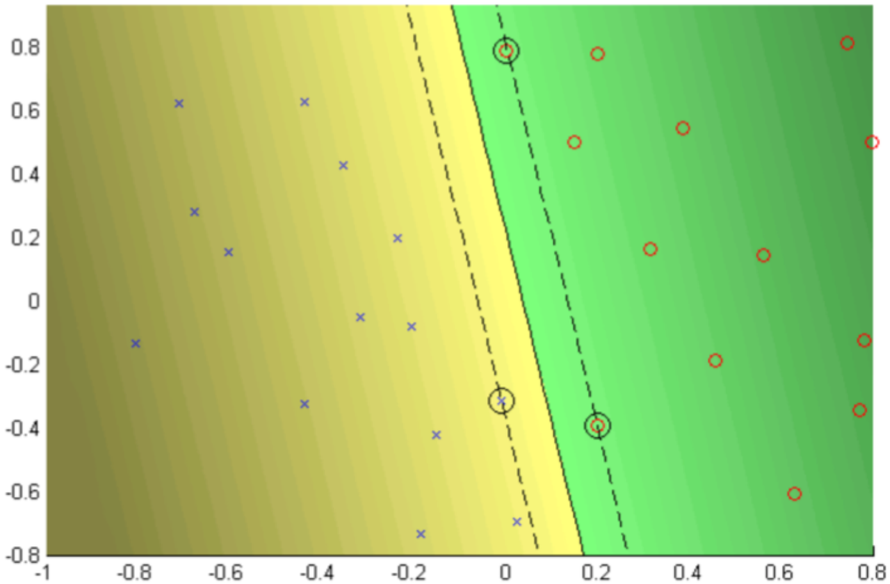
$$\text{and} \ \ \xi_i \geq 0$$

- Add a smearing to the margin, $\xi \geq 0$
- Add regularizer to problem to constrain ξ_i not too large
- C is the regularization hyperparameter
  - Controls how much "softening" of the boundary is allowed, thus how big is margin

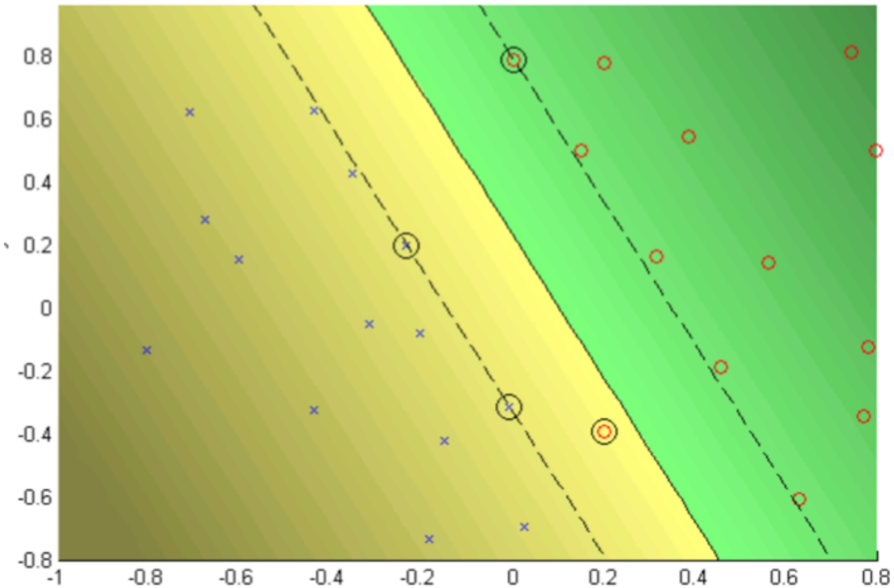


Soft Margin Formulation

C=infinity, hard margin



C=10, soft margin



Dual Formulation

- Use Lagrange multipliers (remember those!) to write a loss function for hard margin:

$$L(\mathbf{w}, w_0, \mathbf{a}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i a_i\{y_i(\mathbf{w}^T\mathbf{x}_i + w_0) - 1\}$$

$$s.\ t.\ \ \{a_i \geq 0\}$$

  - Where **a** are Lagrange multipliers
  - Minimize L w.r.t. **w** and w_0:

$$\rightarrow \mathbf{w} = \sum_i a_i y_i x_i$$

$$\rightarrow \sum_i a_i y_i = 0$$

- Dual form of optimization
  - Solve for **a** and w_0 using gradient methods, or SMO algorithm

$$\max_{\mathbf{a}} \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j$$

$$s.\ t.\ \sum_i a_i y_i = 0$$

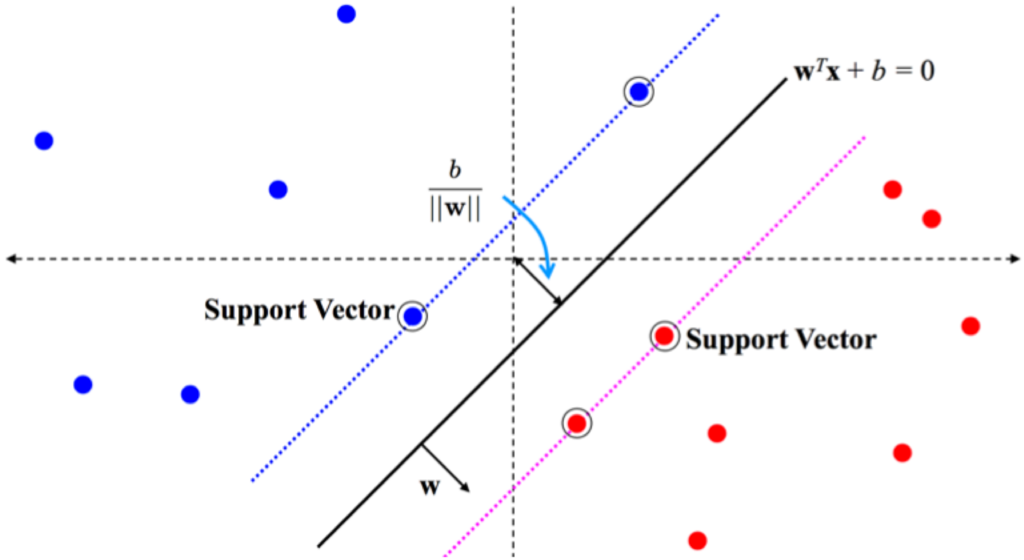
$$a_i \geq 0 \text{ for all } i$$

Discriminant Function

$$h(\mathbf{x}; \mathbf{a}, w_0) = \sum_i a_i y_i \mathbf{x}_i^t \mathbf{x} + w_0$$

Support Vector Machines

$$h(\mathbf{x}; \mathbf{a}, w_0) = \sum_i a_i y_i \mathbf{x}_i^t \mathbf{x} + w_0$$



- Only examples on margin will have a_i>0!
  - Follows from KKT conditions of constrained optimization
- Sum is only over a small number of examples on margin, the **support vectors**
  - Note: also only depends on inner produce! More later
- Margin on data = 1/||**w**||
  - At least one constraint will hold

Support Vector Machines: Recap

- Maximum Margin Optimization: $\max_{\mathbf{a}} \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j y_i y_j \mathbf{x}_i^t \mathbf{x}_j$
  - Dual formulation

$$s.\ t.\ \sum_i a_i y_i = 0$$

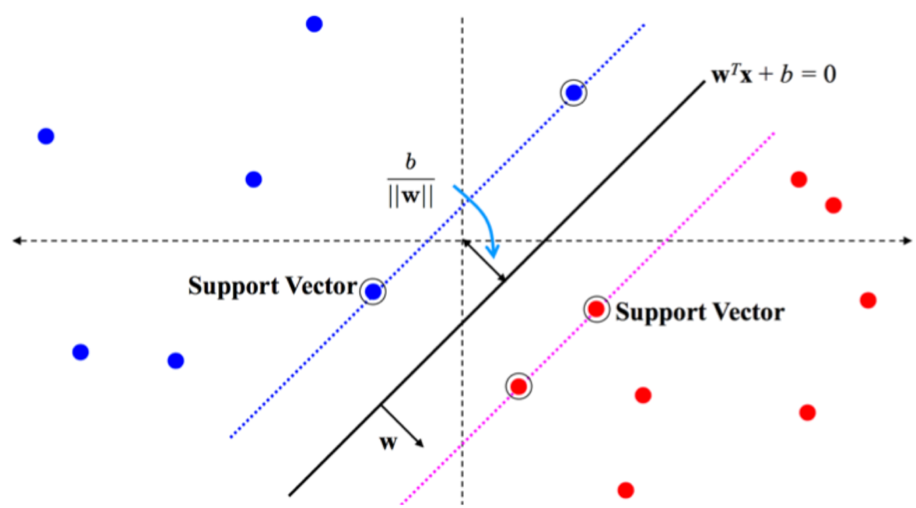
$$a_i \geq 0 \text{ for all } i$$

Data always in inner product

- Discriminant function:

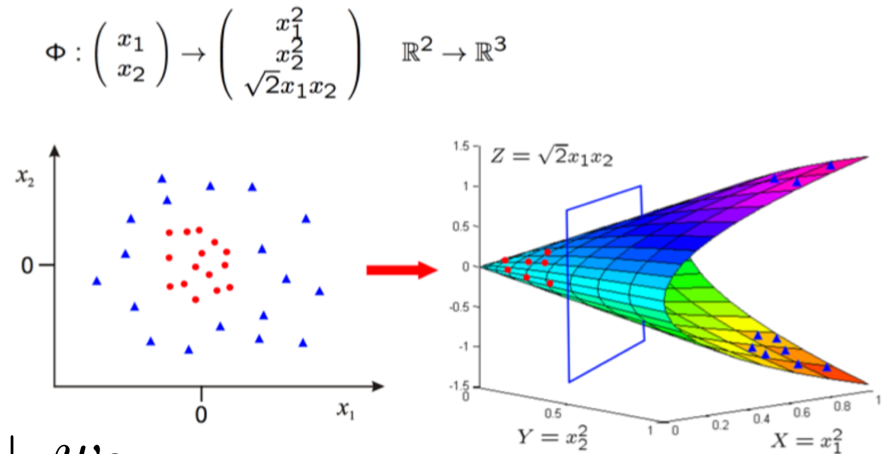
$$h(\mathbf{x}; \mathbf{a}, w_0) = \sum_i a_i y_i \mathbf{x}_i^t \mathbf{x} + w_0$$

  - Sum is only over a small number of examples on margin called the **support vectors**



Basis Functions Revisited

- When data is not linearly separable, can use basis functions



$$h(\mathbf{x};\mathbf{a},w_0)=\sum_i a_i y_i \phi(\mathbf{x}_i)^T\phi(\mathbf{x})+w_0$$

- Where ϕ is a map from $R^m \rightarrow R^k$

- But if k>>m (or if k infinite), inner product can be expensive to compute

- But we don't need the mapping ϕ, only inner products…

Kernels and the Kernel Trick

- A kernel function K(x,x')=ϕ(x)ϕ(x') is an inner product where ϕ is a mapping $R^m \rightarrow R^k$

- Kernelized discriminant and optimization problem

$$h(\mathbf{x}; \mathbf{a}, w_0) = \sum_i a_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0$$

$$\max_{\mathbf{a}} \sum_i a_i - \frac{1}{2} \sum_i \sum_j a_i a_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$s.\ t.\ \ \sum_i a_i y_i = 0$$

$$a_i \geq 0$$

- **Kernel Trick**: compute the Kernel K(x, x') without computing ϕ(x)!
  - So we just need to engineer the Kernel, not the exact features or exact mapping

Kernels

- Linear Kernel: $K(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$

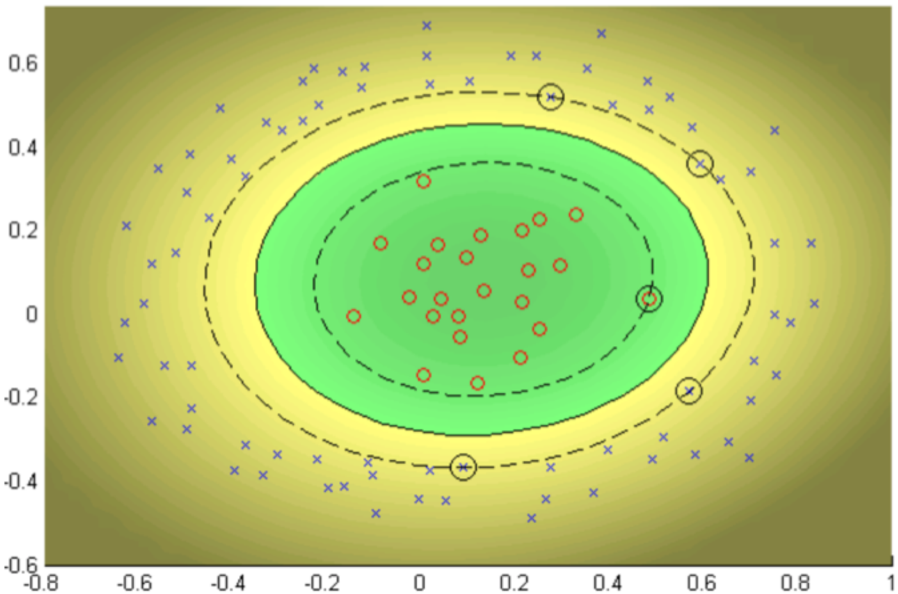
- Polynomial Kernel: $K(\mathbf{x},\mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^q$

- Gaussian Kernal: $K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2}\frac{(\mathbf{x}-\mathbf{x}')^2}{\sigma^2}\right)$

- As long as the Kernel matrix $K_{ij} = \phi(\mathbf{x}_i)\,\phi(\mathbf{x}_j)$ is a positive semi-definite matrix, it is a valid Kernel

SVM

Gaussian Kernel with σ=1



Gaussian Kernel with σ=0.25