

What You Should Know (so far)

Introduction to Statistical Machine Learning
ANU COMP 6467/4670, Sem 1, 2008

Scott Sanner
NICTA / RSISE
First.Last@nicta.com.au

Machine Learning

- You have a data set $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$
 - Data may be partially specified
- You want to learn $\mathbf{y} = \mathbf{f}(\mathbf{x})$ from \mathbf{D}
 - More precisely, you want to minimize some error $\mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{f}, \mathbf{w}, \mathbf{D})$



- Majority of problems are either
 - Classification: \mathbf{y} is discrete
 - Regression: \mathbf{y} is continuous

Inductive Bias

- **Avoid making assumptions about f**
 - Assume for simplicity that $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ is noise free
 - \mathbf{x}_i 's in \mathbf{D} only cover small subset of input space \mathbf{x}
- **What's the best we can do?**
 - If we've seen $\mathbf{x}=\mathbf{x}_i$ report $\mathbf{y}=\mathbf{y}_i$
 - If we have not seen $\mathbf{x}=\mathbf{x}_i$, can't say anything (no assumptions)
- **This is called rote learning... boring, eh?**
 - Key idea: *you can't generalize to unseen data w/o assumptions!*
- **Thus, key to ML is generalization**
 - Any ML algorithm *must* have some **inductive bias**
 - Bias usually in the form of a **restricted hypothesis space**
 - Important to understand restrictions (and whether appropriate)

Parametric vs. Non-parametric

- **Parametric:** has parameters
 - Most Probabilistic Approaches
 - Gaussians
 - Bernoulli / Binomial / Multinomial
 - Graphical Models
 - Linear Regression / Classifiers
 - SVM with linear kernel

What assumptions?

Data is generated by given distribution.

Linearly separable, regressable data, etc.

- **Non/semi-parametric:** data oriented
 - Neighbor-based approaches
 - (K-)nearest Neighbor
 - Parzen Windows
 - SVM with RBF kernel

What assumptions?

Encoded in distance function & K / width.

Smoothness... no abrupt changes!

Linear vs. Non-linear

- $y = f(\mathbf{x}, \mathbf{w})$
 - \mathbf{x} is your input vector
 - \mathbf{w} is your parameter vector (weights)

- **Which f is linear in \mathbf{w} ?**

i.e., $f(\mathbf{x}, \mathbf{w}) = \langle \mathbf{w}, \mathbf{x} \rangle$ (assume $\mathbf{x}_0 = 1$)

– $f_1(\mathbf{x}) = w_1 x_1 + w_2 x_2$ ✓

– $f_2(\mathbf{x}) = w_1 x_1^2 + w_2 x_1 x_2 + w_3 x_2^2$ ✓

– $f_3(\mathbf{x}) = w_1 x_1 + w_2 w_3 x_2 + w_3^2 x_3$ ✗

Any transformation of input \mathbf{x} maintains linear function in \mathbf{w} !

Your Linear Fun. Approx. Toolbox

- **Classification**

- Naïve Bayes (simple)
- Logistic Regression (better than NB for dependent features)
- Perceptron (didactic, rarely used in practice)
- SVM and Kernel Methods (very powerful)

- **Regression**

- Linear Regression (closed-form solution)
- SVR and Kernel Methods (very powerful)

- **Key Advantage**

- All of above lead to convex optimization problems
→ global optima will be found.

How Powerful are Linear Models?

- **Short answer:**

- $\mathbf{y} = \langle \mathbf{w}, \mathbf{x} \rangle$ is surprisingly powerful

- Especially if \mathbf{x} transformed: $\mathbf{x} \rightarrow \Phi(\mathbf{x})$

- $\mathbf{y} = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$


input space *feature space*

- **Can use expressive feature spaces $\Phi(\mathbf{x})$**

- Use cross-validation (CV) to prune features

- If data \ll features / \mathbf{w} , may overfit even with CV

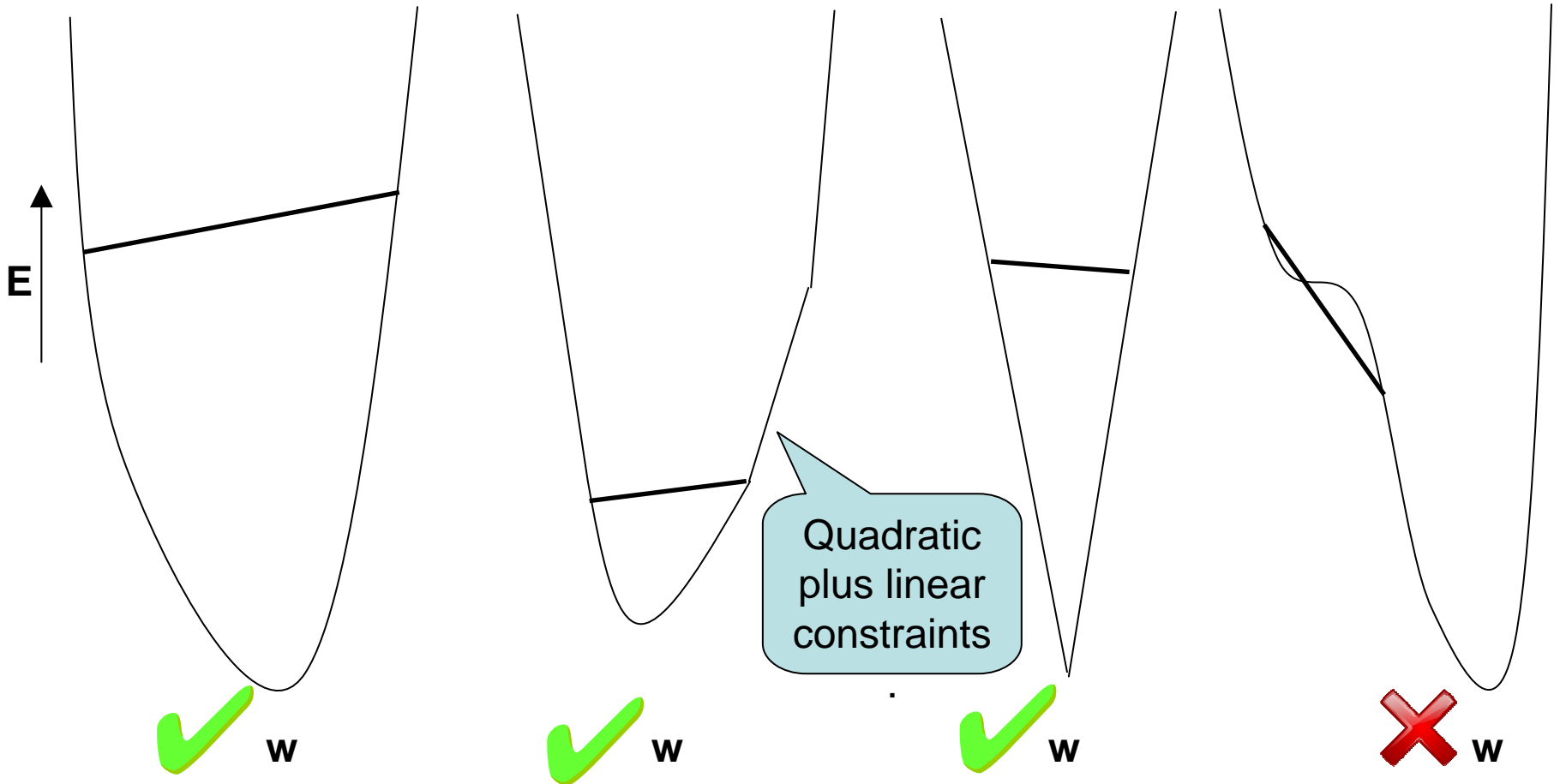
This is Worth Investigating...

- **Seems counterintuitive...**
 - Fit crazy non-linear functions & find global optimum?
 - WTF?  Why's that fine?
- **Think about abstracted problem...**
 - $E(\mathbf{w}) = E(\mathbf{x}, \mathbf{y}, f, \mathbf{w}, \mathbf{D})$ (b/c $\mathbf{x}, \mathbf{y}, f, \mathbf{D}$ fixed, given problem)
 - We change the weights \mathbf{w} , we get different $E(\mathbf{w})$.
 - Which setting of weights optimizes $E(\mathbf{w})$?
- **Question: how does $E(\mathbf{w})$ look w.r.t. weights?**
 - Linear regression: linear f & SSE, $E(\mathbf{w})$ looks quadratic
 - All 2nd derivatives > 0
 - Sufficient to show Hessian (2nd derivative matrix) is positive semidefinite → convex...




Convexity

For convex problems, (sub)gradient descent gets us to global minima. Q.E.D.

- Graphically... which of these is convex?



Empirical Risk Minimization

- A **general framework** for function approximation
- Minimize $E'(\mathbf{w}) = \text{Loss}(\mathbf{w}) + \text{Regularizer}(\mathbf{w})$
- Loss functions penalize errors in different ways, e.g.,
 - Sum of squared error (SSE), a.k.a. LMS 
 - Hinge loss 
 - ϵ -insensitive loss 
- Regularizer expresses preference on \mathbf{w} , e.g.,
 - $\|\mathbf{w}\|_2$: assumes Gaussian prior
 - $\|\mathbf{w}\|_1$: can encourage sparsity
 - $\mathbf{w} \cdot \log \mathbf{w}$: maximizes entropy for prob. interpretation of \mathbf{w}
- **Many $E'(\mathbf{w})$ possibilities are convex for linear f !**

The Joy of Convex(ity)

All of these losses are convex for linear f :

Table 1: Scalar loss functions and their derivatives, depending on $f := \langle w, x \rangle$, and y .

	Loss $l(f, y)$	Derivative $l'(f, y)$
Hinge [20]	$\max(0, -yf)$	0 if $yf \geq 0$ and $-y$ otherwise
Squared Hinge [26]	$\frac{1}{2} \max(0, -yf)^2$	0 if $yf \geq 0$ and f otherwise
Soft Margin [4]	$\max(0, 1 - yf)$	0 if $yf \geq 1$ and $-y$ otherwise
Squared Soft Margin [10]	$\frac{1}{2} \max(0, 1 - yf)^2$	0 if $yf \geq 1$ and $f - y$ otherwise
Exponential [14]	$\exp(-yf)$	$-y \exp(-yf)$
Logistic [13]	$\log(1 + \exp(-yf))$	$-y / (1 + \exp(yf))$
Novelty [32]	$\max(0, 1 - f)$	0 if $f \geq 0$ and -1 otherwise
Least mean squares [43]	$\frac{1}{2}(f - y)^2$ ← linear regression	$f - y$
Least absolute deviation	$ f - y $	$\text{sgn}(f - y)$
Quantile regression [27]	$\max(\tau(f - y), (1 - \tau)(y - f))$	τ if $f > y$ and $\tau - 1$ otherwise
ϵ -insensitive [41]	$\max(0, f - y - \epsilon)$	0 if $ f - y \leq \epsilon$ and $\text{sgn}(f - y)$ otherwise
Huber's robust loss [31]	$\frac{1}{2}(f - y)^2$ if $ f - y < 1$, else $ f - y - \frac{1}{2}$	$f - y$ if $ f - y \leq 1$, else $\text{sgn}(f - y)$
Poisson regression [16]	$\exp(f) - yf$	$\exp(f) - y$

Table 2: Vectorial loss functions and their derivatives, depending on the vector $f := Wx$ and on y .

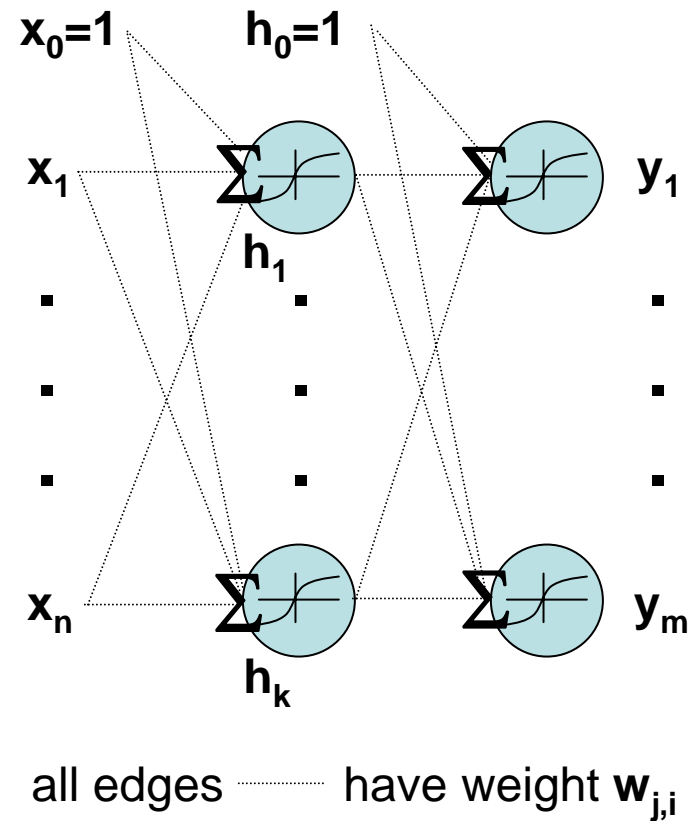
	Loss	Derivative
Soft Margin [38]	$\max_{y'}(f_{y'} - f_y + \Delta(y, y'))$	$e_{y^*} - e_y$, where y^* is the argmax of the loss
Scaled Soft Margin [40]	$\max_{y'} \Delta^\beta(y, y')(f_{y'} - f_y + \Delta(y, y'))$	$\Delta^\beta(y, y')(e_{y^*} - e_y)$, where y^* is the argmax of the loss
Softmax [14]	$\log \sum_{y'} \exp(f_{y'}) - f_y$	$\frac{\sum_{y'} e_{y'} \exp(f'_{y'})}{\sum_{y'} \exp(f'_{y'})} - e_y$
Multivariate Regression	$\frac{1}{2}(f - y)^T M(f - y)$ where $M \succeq 0$	$M(f - y)$

Your Convex Optimization Toolbox

- **If lucky, convex problem can be solved...**
 - in closed-form (linear regression)
 - or via gradient descent (logistic regression)
- **But convex functions may be piecewise**
 - Often due to linear constraints
- **Then may need to use one of**
 - Linear program (LP) optimization:
 - Minimize $w \cdot d$ s.t. $Cw < 0$ (C is a linear constraint matrix)
 - Quadratic (objective) program (QP) optimization
 - Semidefinite (constraints) program (SDP) optimization
- **Don't fear! Good off-the-shelf optimizers**
 - SVMs require solving QP, but very robust & scalable

Artificial Neural Networks

- **Neural Net:**
non-linear weighted combination of *shared* sub-functions
- **Backpropagation:** to minimize SSE, train weights using *gradient descent* and *chain rule*
- **Backprop. ideas useful beyond ANNs!**



Probability / Density Estimation

- **Maximum Likelihood (ML)**

$$\vec{\theta}^* = \arg \max_{\vec{\theta}} P(D|\vec{\theta})$$

Derivations
for Gaussian,
Binomial, etc.

- **Bayesian**

$$P(\vec{\theta}|D^n) \propto P(x^n|\vec{\theta})P(\vec{\theta}|D^{n-1})$$

Conjugate Priors

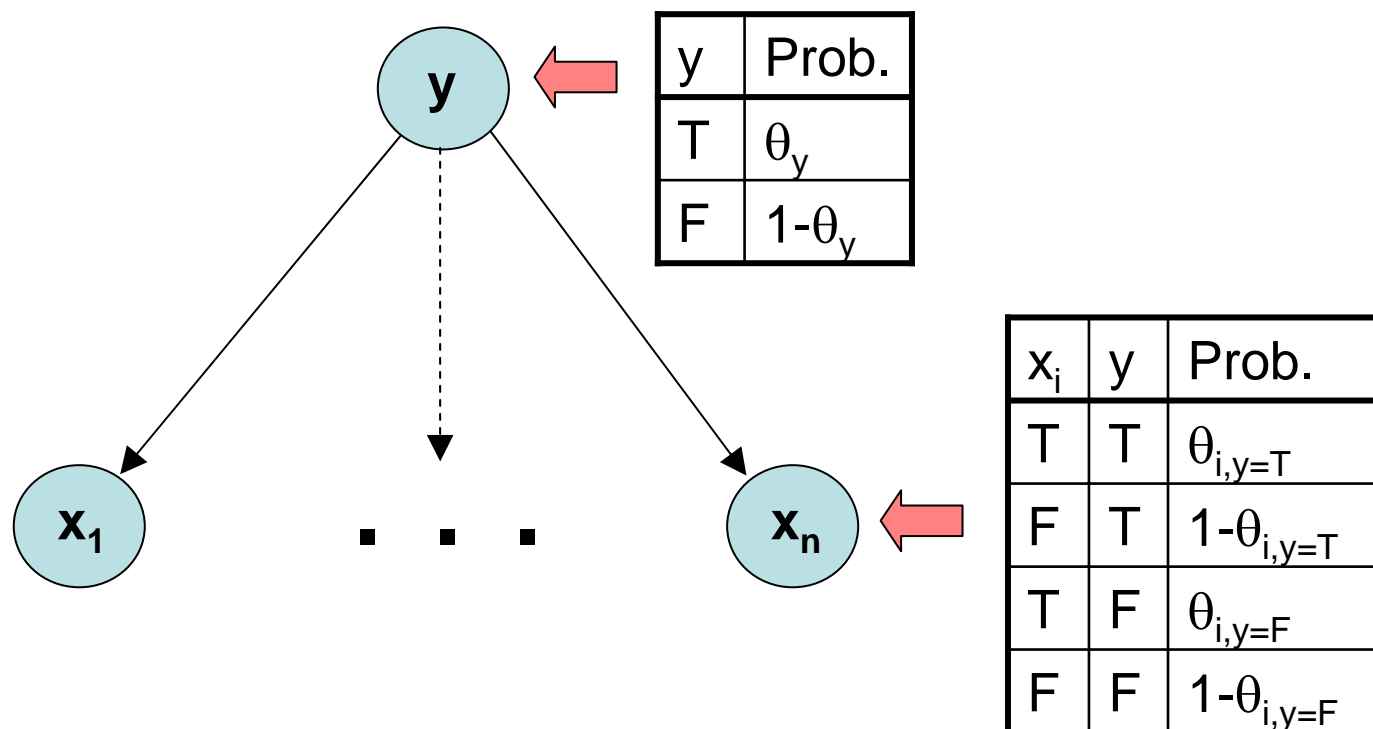
- **Maximum a Posteriori (MAP)**

$$\vec{\theta}^* = \arg \max_{\vec{\theta}} P(\vec{\theta}|D)$$

$$\propto \arg \max_{\vec{\theta}} P(D|\vec{\theta})P(\vec{\theta})$$

Naïve Bayes (Classifier)

- Make naïve independence assumption
 - \mathbf{x}_i conditionally independent of \mathbf{x}_j ($i \neq j$) given \mathbf{y}_i
 - For simplicity, assume all variables are binary



Max Likelihood for Naïve Bayes

$$\begin{aligned}
 \vec{\theta}^* &= \arg \max_{\vec{\theta}} \overbrace{P(D|\vec{\theta})}^{L(\theta)} \\
 &= \arg \max_{\vec{\theta}} \prod_{d \in D} P(y^d, x_1^d, \dots, x_n^d | \vec{\theta}) \\
 &= \arg \max_{\vec{\theta}} \prod_{d \in D} P(x_1^d, \dots, x_n^d | y^d, \vec{\theta}) P(y^d | \vec{\theta}) \\
 &= \arg \max_{\vec{\theta}} \prod_{d \in D} P(y^d | \vec{\theta}) \prod_{i=1}^n P(x_i^d | y^d, \vec{\theta}) \\
 &= \arg \max_{\vec{\theta}} \prod_{d \in D} \theta_y^{\mathbb{1}_{y=T}[d]} (1 - \theta_y)^{\mathbb{1}_{y=F}[d]} \prod_{i=1}^n \prod_{v \in \{F, T\}} \theta_{y=v, i}^{\mathbb{1}_{y=v, x_i=T}[d]} (1 - \theta_{y=v, i})^{\mathbb{1}_{y=v, x_i=F}[d]} \\
 &= \arg \max_{\vec{\theta}} \theta_y^{\#D_{y=T}} (1 - \theta_y)^{\#D_{y=F}} \prod_{i=1}^n \prod_{v \in \{F, T\}} \theta_{y=v, i}^{\#D_{y=v, x_i=T}} (1 - \theta_{y=v, i})^{\#D_{y=v, x_i=F}} \\
 &= \arg \max_{\vec{\theta}} \underbrace{\#D_{y=T} \log \theta_y + \#D_{y=F} \log(1 - \theta_y)}_{\substack{\sum_{i=1}^n \sum_{v \in \{F, T\}} \left[\#D_{y=v, x_i=T} \log \theta_{y=v, i} + \#D_{y=v, x_i=F} \log(1 - \theta_{y=v, i}) \right] \\ l(\vec{\theta})}}
 \end{aligned}$$

Max Likelihood for NB (Cont.)

- Unique maxima for log-linear models at slope = 0

$$\frac{\partial l(\vec{\theta})}{\partial \theta_y} = \frac{\#D_{y=T}}{\theta_y} + \frac{\#D_{y=F}}{(1 - \theta_y)} = 0$$

$$\frac{\theta_y}{(1 - \theta_y)} = \frac{\#D_{y=T}}{\#D_{y=F}}$$

$$\theta_y = \frac{\#D_{y=T}}{\#D_{y=T} + \#D_{y=F}}$$

$$\frac{\partial l(\vec{\theta})}{\partial \theta_{y=v,i}} = \frac{\#D_{y=v,x_i=T}}{\theta_{y=v,i}} + \frac{\#D_{y=v,x_i=F}}{(1 - \theta_{y=v,i})} = 0$$

$$\frac{\theta_{y=v,i}}{(1 - \theta_{y=v,i})} = \frac{\#D_{y=v,x_i=T}}{\#D_{y=v,x_i=F}}$$

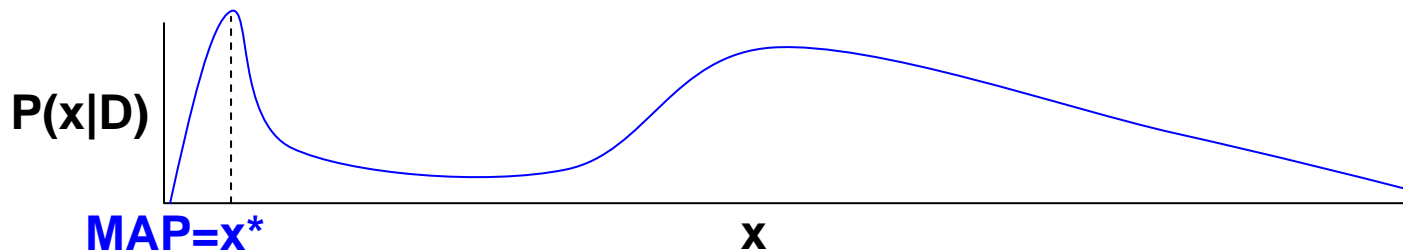
$$\theta_{y=v,i} = \frac{\#D_{y=v,x_i=T}}{\#D_{y=v,x_i=T} + \#D_{y=v,x_i=F}}$$

- ML parameters are just the empirical probabilities!

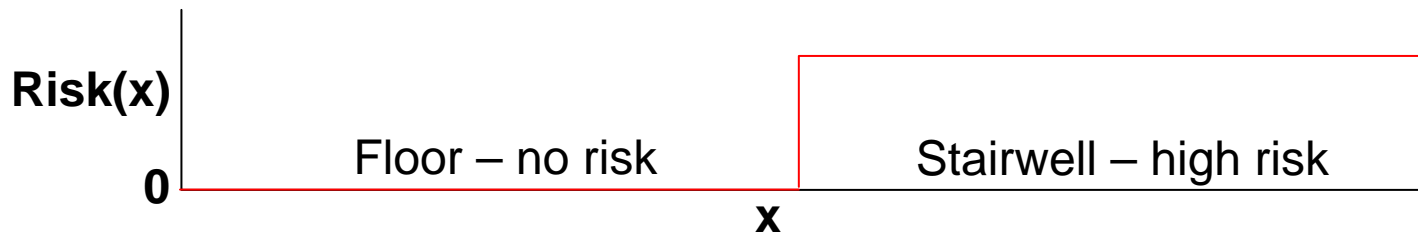
Also see [discussion of NB vs. Logistic Regression](#) (w\ LogRegr derivation).

Bayesian Decision Theory

- Robot has belief $P(x|D)$ over position
 - D consists of noisy range finder readings



- Associate Risk(x) w/ position x (e.g., stairs!)



- MAP Risk = $Risk(x^*) = 0$
- Full Bayesian Risk = $\int_x Risk(x)p(x|D) > 0$
- Which risk estimate would you use?

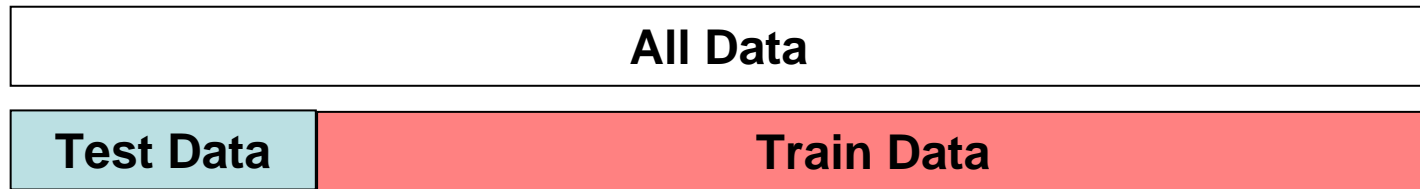
Unsupervised Learning

- **$D = \{(x_i, y_i)\}$, y unknown (latent/hidden variable)**
- **Fill in labellings of latent variable**
 - K-Means, Expectation Maximization (EM)
 - Have to make assumptions on what variable filling in
- **Extract latent variables – find feature structure in data**
 - PCA, ICA
 - Different approaches prefer different features
- **Applications**
 - Filling in partially labelled or hidden variables
 - E.g., 2d & 3d modeling
 - Noise Filtering
 - Compression with minimal loss
 - Project a high dimensional space to lower dimensions (PCA)
 - Source separation (ICA)
 - Collaborative filtering (semi-supervised)

(Cross) Validation

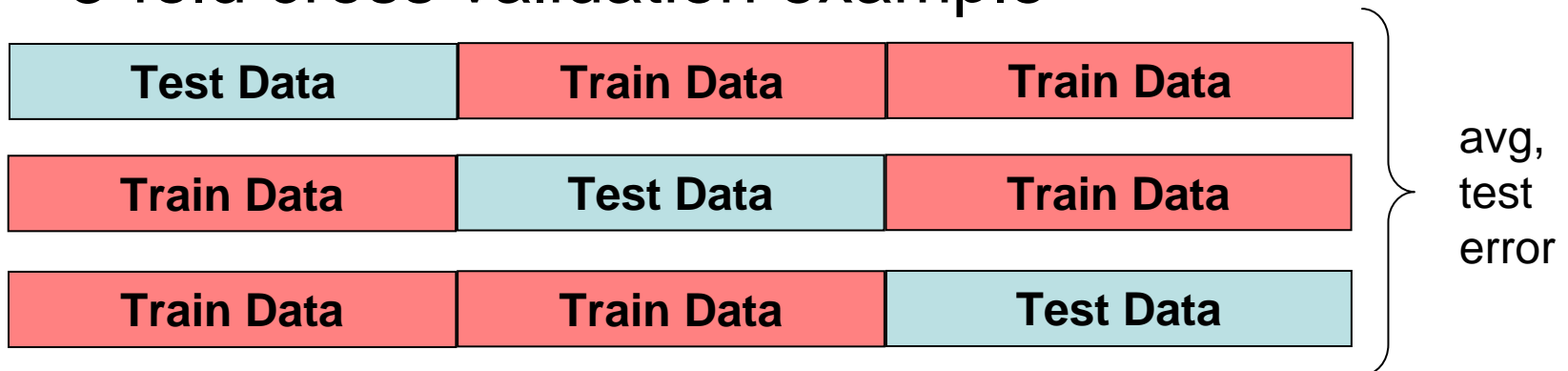
- **Validation:**

- Never calculate test error on training data



- **Cross Validation:**

- Reduce variance of test error estimate
- 3-fold cross validation example



Model and Feature Selection

- Data set $\mathbf{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$
- Learn $\mathbf{y} = \mathbf{f}(\mathbf{x})$ by minimizing $\mathbf{E}(\mathbf{x}, \mathbf{y}, \mathbf{f}, \mathbf{w}, \mathbf{D})$
- Model Selection:
 - Which \mathbf{f} to use
 - Linear / Non-linear
 - Parametric / Non- / Semi-parametric
 - Parameters within each model
- Feature Selection
 - Which \mathbf{x} to use
 - Original / Transformed (e.g., polynomials,
- How to choose between different models or feature sets
 - Can choose model / feature set with lowest CV error

Introductory Books to Consider

