Model Selection: Beyond the Bayesian/Frequentist Divide

I. Guyon, et al.

Discussion of approaches to model selection

especially with reference to the problem of over-fitting

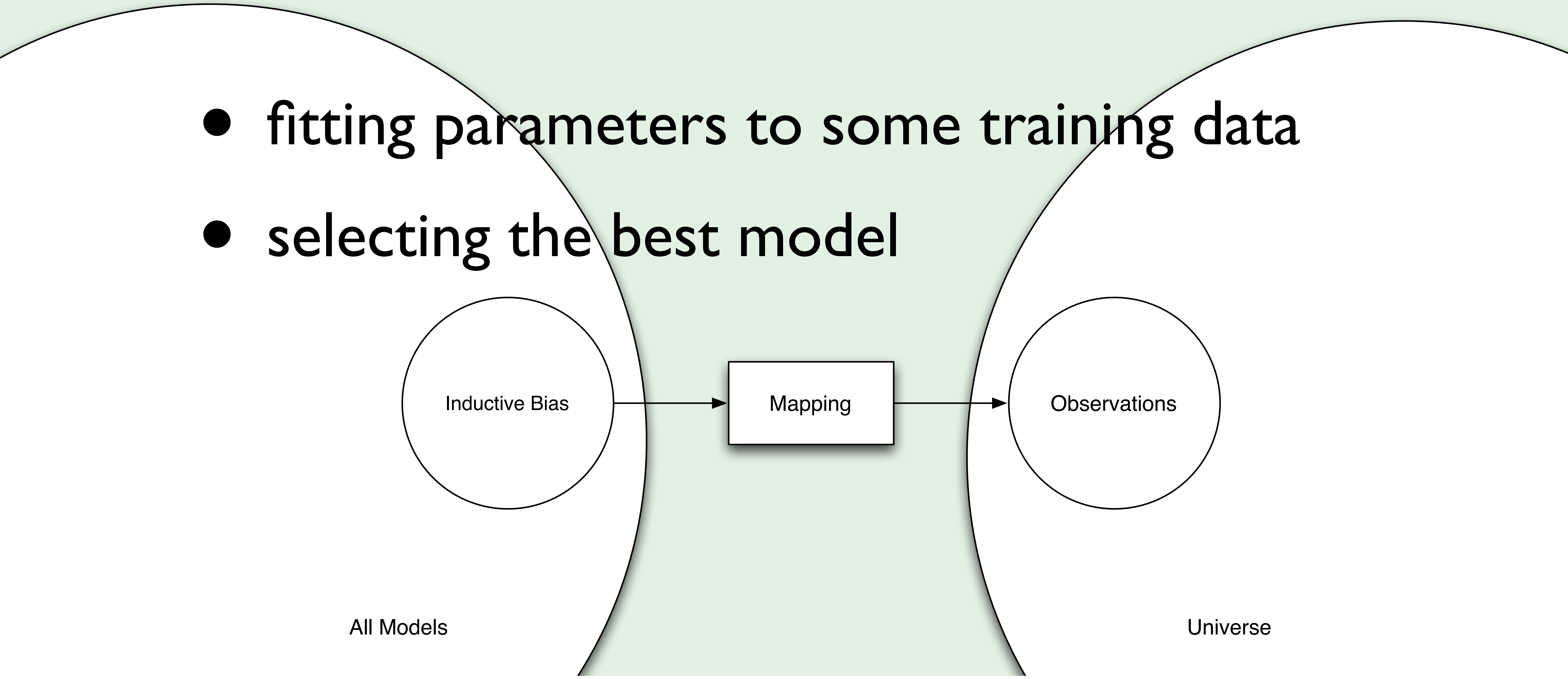
and the similarities between approaches

outline

- introduction to model selection
- Bayesians and Frequentists
- multi-level inference
- advances in model selection

learning as model selection

- fitting parameters to some training data
- selecting the best model



implicitly assumes, learning is the same as model selection, is it?

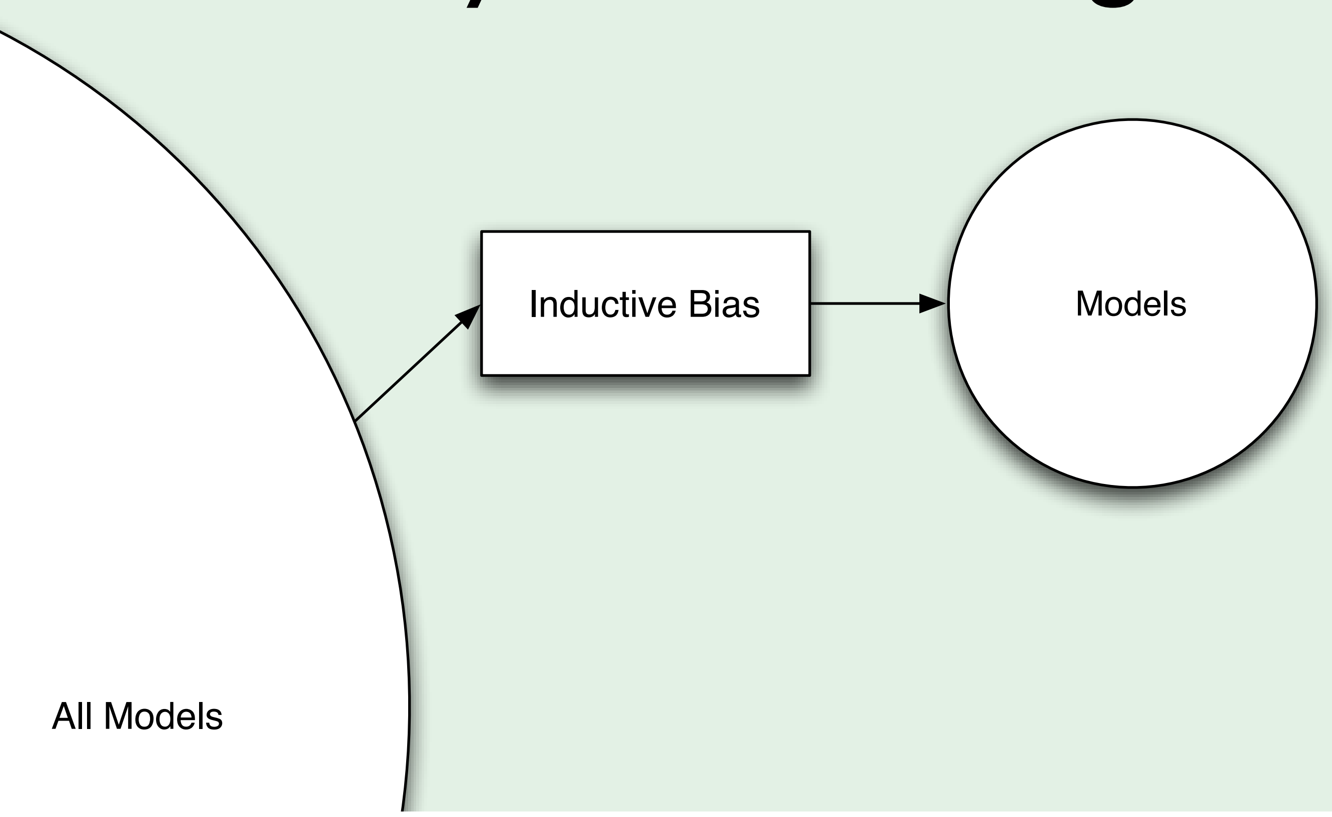
can you learn without making models? without choosing models?

if we choose a ‘better’ model have we done a ‘better’ job learning?

ex: cross-validation, optimizing cost/loss functions

can't we just average? or minimize risk?

- your still doing model selection

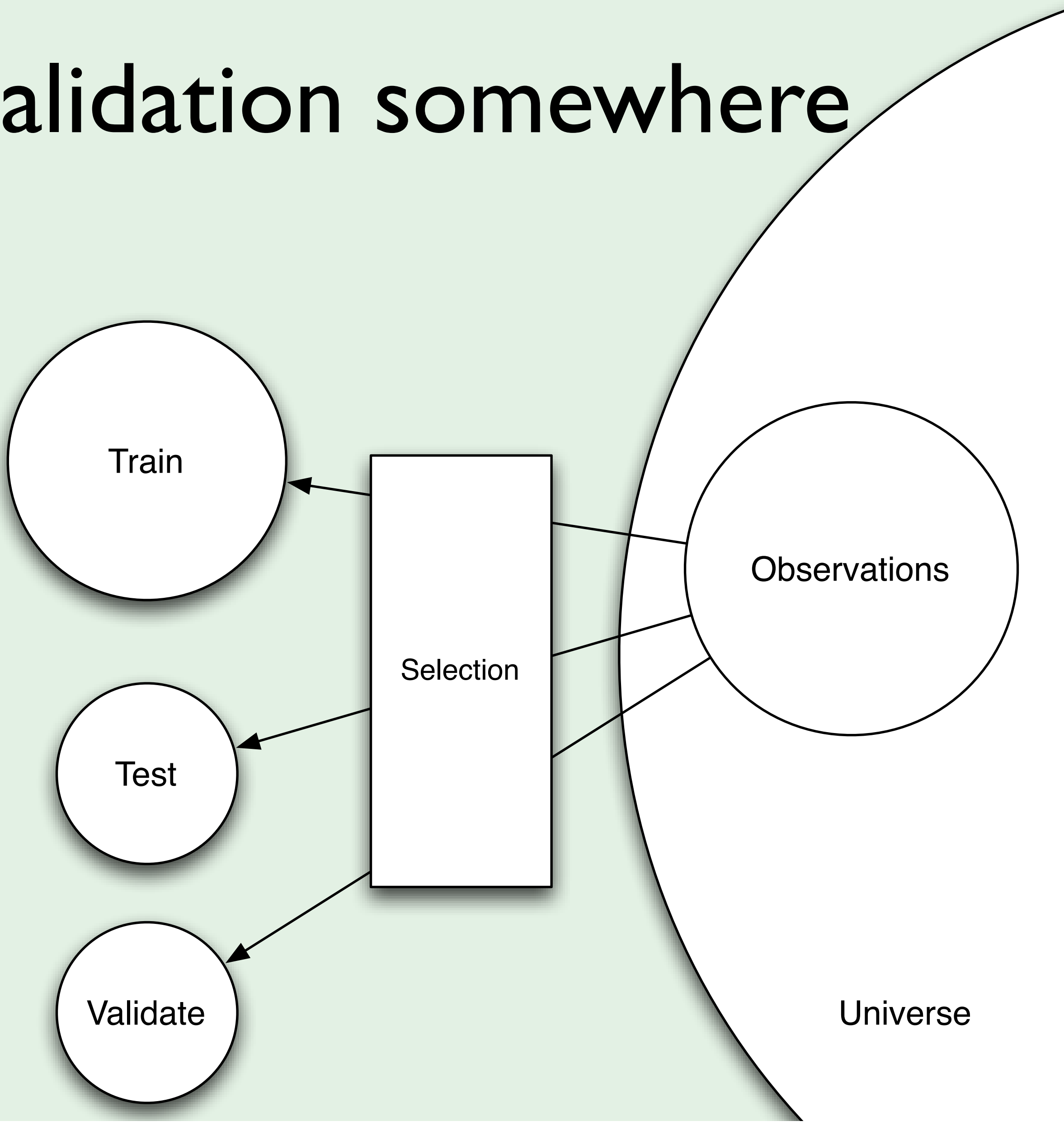


most model selection methods still have a hyper-parameter that's optimized through cross-validation

so, even in the principled Bayesian method of averaging over posteriors, or minimizing performance bounds, we use cross-validation

can't we just average? or minimize risk?

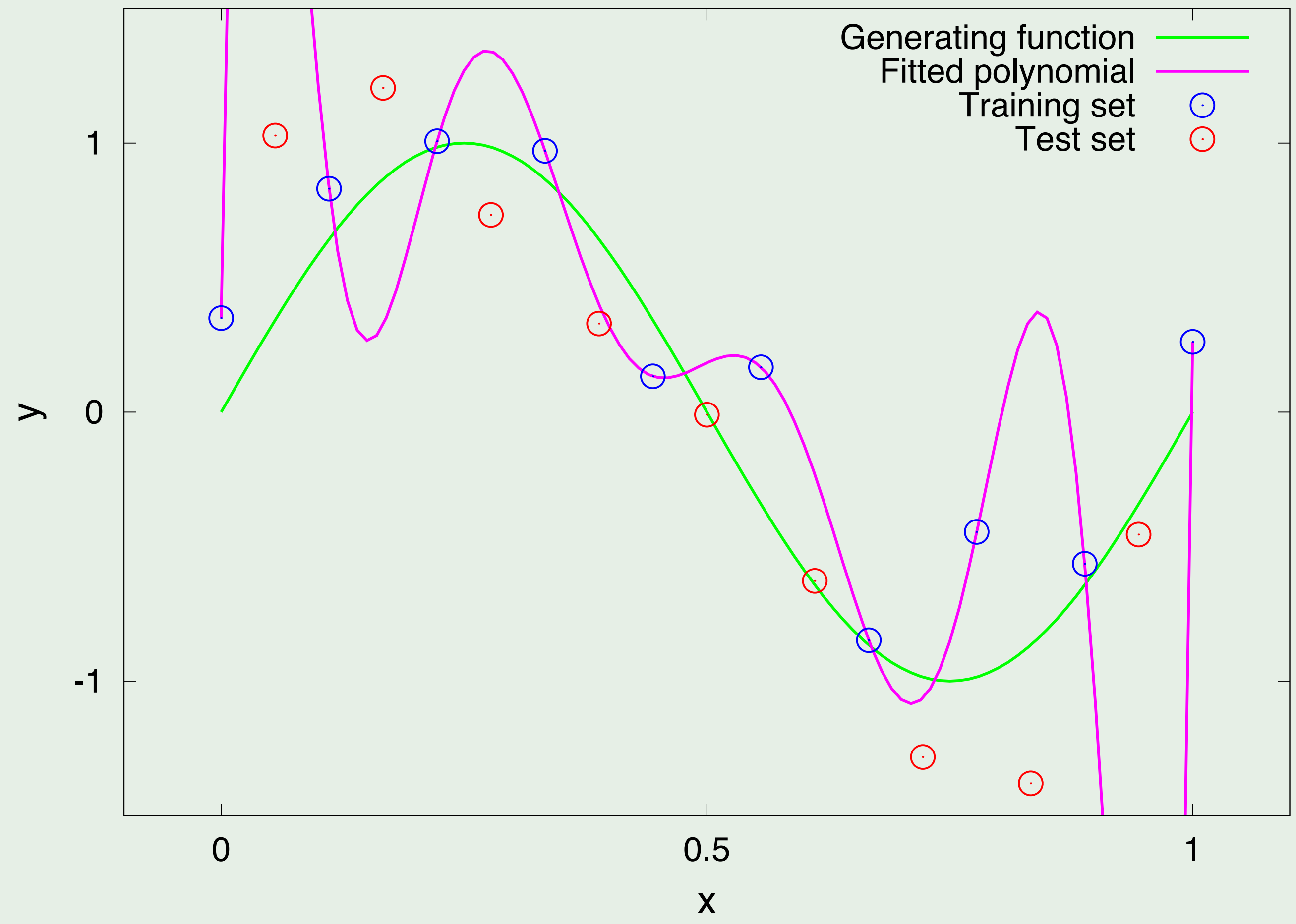
- probably rely on cross validation somewhere



there's no principled way to do cross-validation, e.g. choosing how to divide problem, how to allocate data to divisions

treat hyper-parameters as parameters?

- joint optimization is a non-convex problem
- joint optimization has infinite complexity



considering the class of kernel methods

non-convex lose unique solution guarantee

with hyper-parameters we can bound capacity, yet still search in a class of universal approximators

potentially alleviate over-fitting

we can structure parameter space

- hyper-parameters lets us monitor bias/variance tradeoff
- a regularizer enforces lower complexity

we can bound over-fitting with hyper-parameters

popular regularizers “weight decay” in NN, Gaussian processes, ridge regression, “hinge loss”

optimize hyper-parameter for regularizer at 2nd level of inference

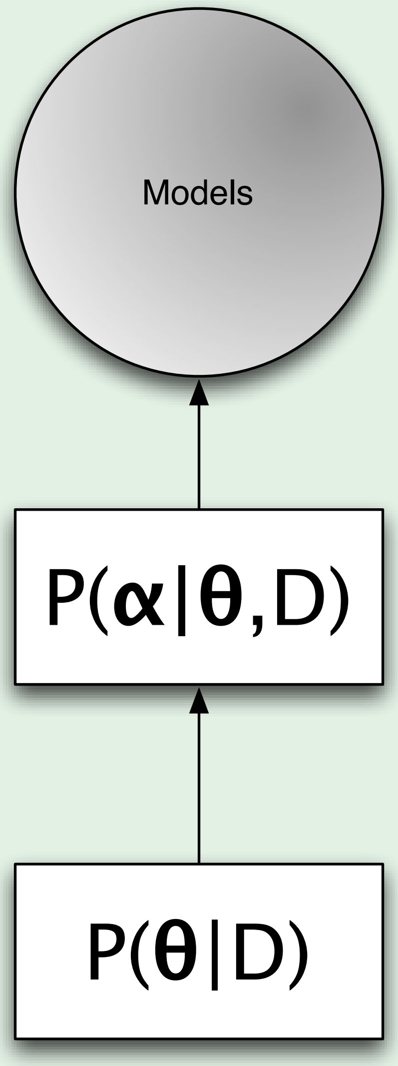
considering linear models: f(x)=\sum w_ix_i

“hinge loss” is: R_reg=R_tr+\gamma ||w||^2, \gamma>0

Bayesian Model Selection

decompose prior P($\boldsymbol{\alpha}$,$\boldsymbol{\theta}$) into

- parameter prior P($\boldsymbol{\alpha}$|$\boldsymbol{\theta}$)
- "hyper-prior" P($\boldsymbol{\theta}$)



given these parameters

make predictions according to an integral over the class of models

weighted by the likelihood of the parameters given the data

MAP Learning

- maximize evidence w.r.t the hyper-parameters

$$\boldsymbol{\theta}^* = \operatorname{argmax}_{\boldsymbol{\theta}} P(D|\boldsymbol{\theta}) = \operatorname{argmax}_{\boldsymbol{\theta}} \sum_{\boldsymbol{\alpha}} P(D|\boldsymbol{\alpha}, \boldsymbol{\theta}) P(\boldsymbol{\alpha}|\boldsymbol{\theta})$$

- maximize the posterior w.r.t the parameters

$$\boldsymbol{\alpha}^* = \operatorname{argmax}_{\boldsymbol{\alpha}} P(\boldsymbol{\alpha}|\boldsymbol{\theta}, D) = \operatorname{argmax}_{\boldsymbol{\alpha}} P(D|\boldsymbol{\alpha}, \boldsymbol{\theta}) P(\boldsymbol{\alpha}|\boldsymbol{\theta})$$

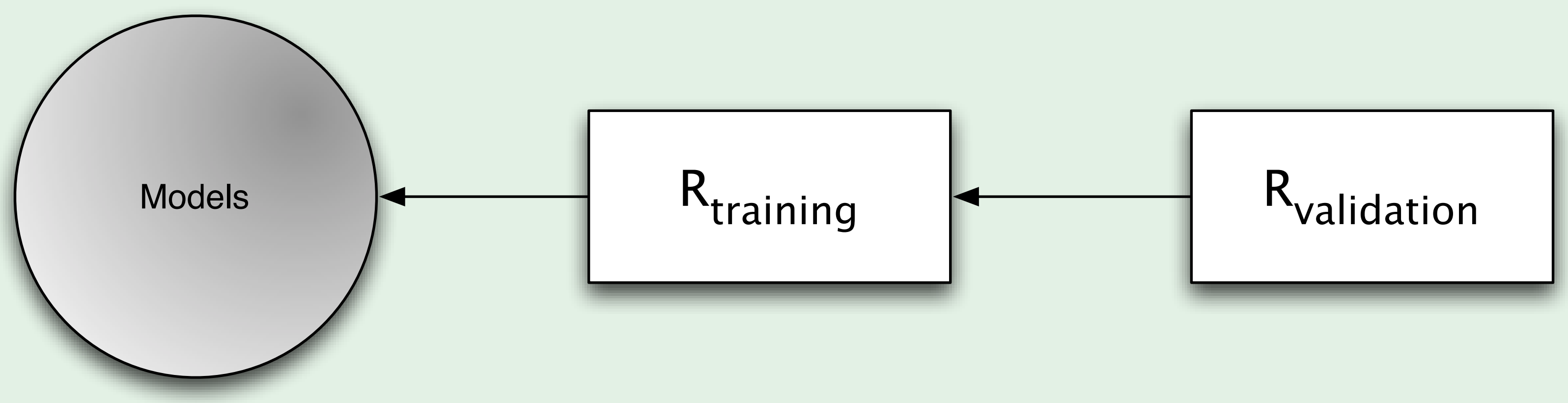
all familiar w/this

what’s important is that there are two levels of maximization

one w.r.t. \theta and then another, given \theta, w.r.t \alpha

Frequentist Model Selection

- adjust complexity to minimize risk of over-fitting or under-fitting
- ordering of models' expected error



performance prediction: estimate the generalization error R[f]

select models based on predicted performance, we want a monotonic function r, such that r [f_1]<r[f_2]=>R[f_1]<R[f_2]

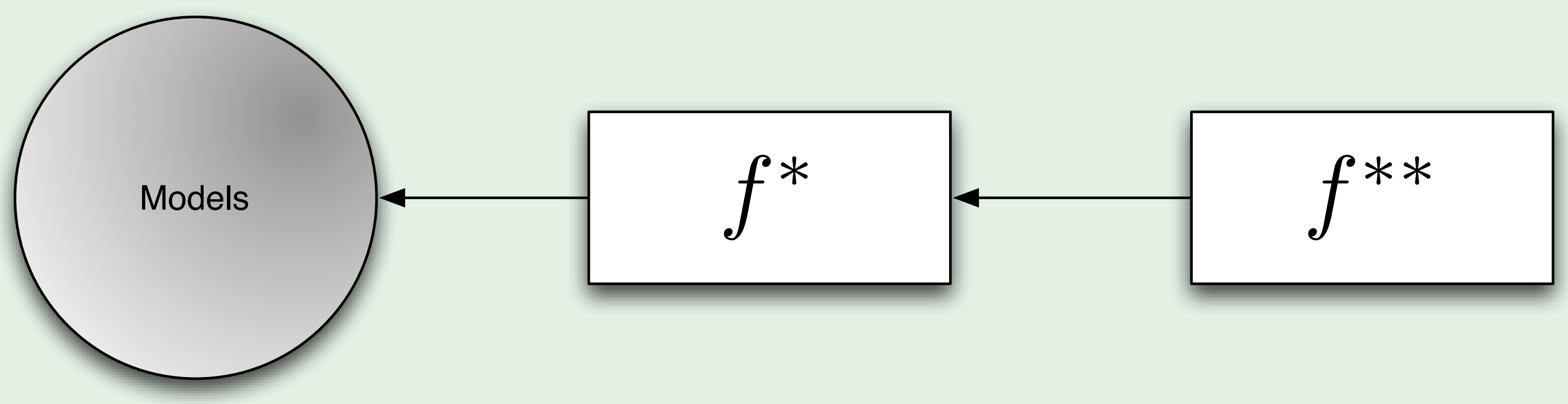
frequentists often train parameters on one part of data set, training examples

and train hyper-parameters on another part, validation examples

Multi-level Inference

- hierarchy of optimization problems
- each level infers a set of (hyper-)parameters

$$f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$$



consider a model class f, we want to optimize f according to f* and optimize f* according to f**

can view both frequentist and Bayesian learning as solving multi-level inference problems

Multi-level Inference Frequentist

- we determine our hyper-parameters:

$$f^{**} = \text{argmin}_{\boldsymbol{\theta}} R_2[f^*, D]$$

- then determine our parameters:

$$f^* = \text{argmin}_{\boldsymbol{\alpha}} R_1[f, D]$$

in frequentist models, given risk functionals R_1 and R_2, we solve the optimization problems f** and f*

Bayesian models are similarly expressed, but with integrals of the models and priors

Multi-level Inference

<u>Definition</u>: a *multi-level inference problem* is a learning problem organized into a hierarchy of learning problems

formalize by expressing the optimization problems, f*s, as the result of a training procedure

Multi-level Inference

- learning machines *A* with model space *B* of functions $f(\mathbf{x}; \boldsymbol{\theta})$ with parameters **θ**

$$f^{**} = \mathrm{train}(\mathcal{A}[\mathcal{B}, R_2], D)$$

- consider *B* as a learning machine in model space *F* of functions $f(\mathbf{x}; \boldsymbol{\alpha}, \boldsymbol{\theta})$ with variable **α** and fixed **θ**

$$f^{*} = \mathrm{train}(\mathcal{B}[\mathcal{F}, R_1], D)$$

think of train as a method, process data according to some training algorithm

R is an evaluation function

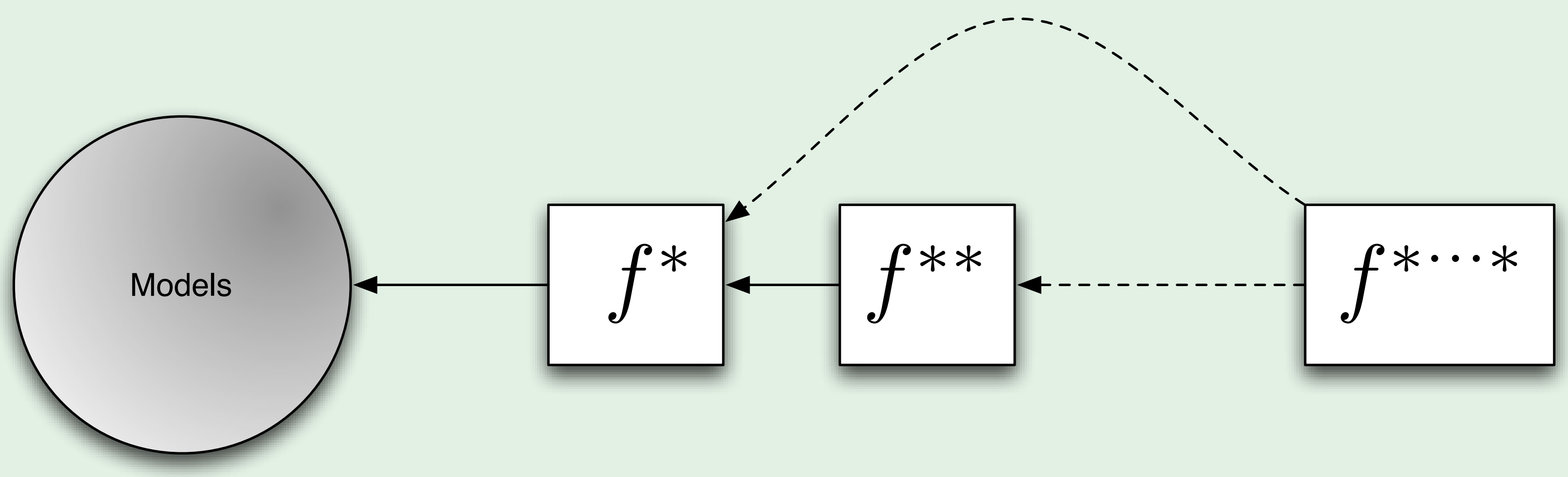
solution f** belongs to the convex closure of B

solution f* belongs to the convex closure of F

we may use different subsets of D at different levels of inference

Extensions

- more than two levels of inference
- ensemble methods



we can have an arbitrarily deep hierarchy. when would that be useful?

ensemble, have “train” return a linear combination of models

Inference Modules

Filter methods

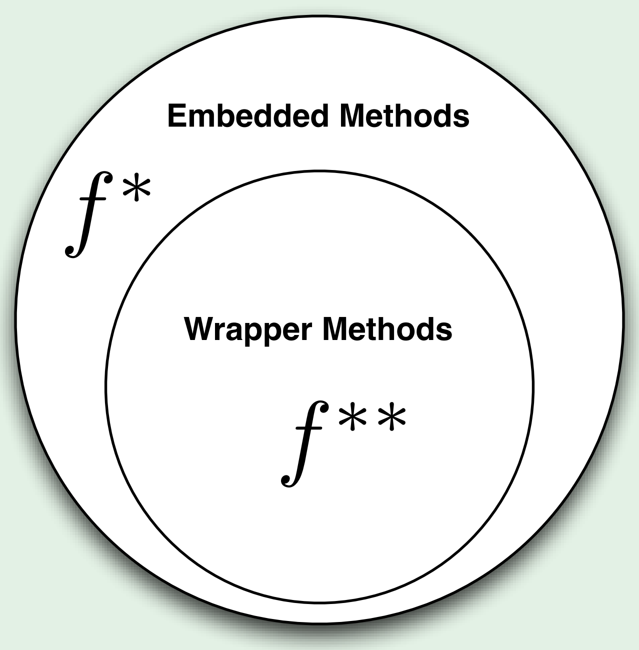
- narrow without training

Wrapper methods

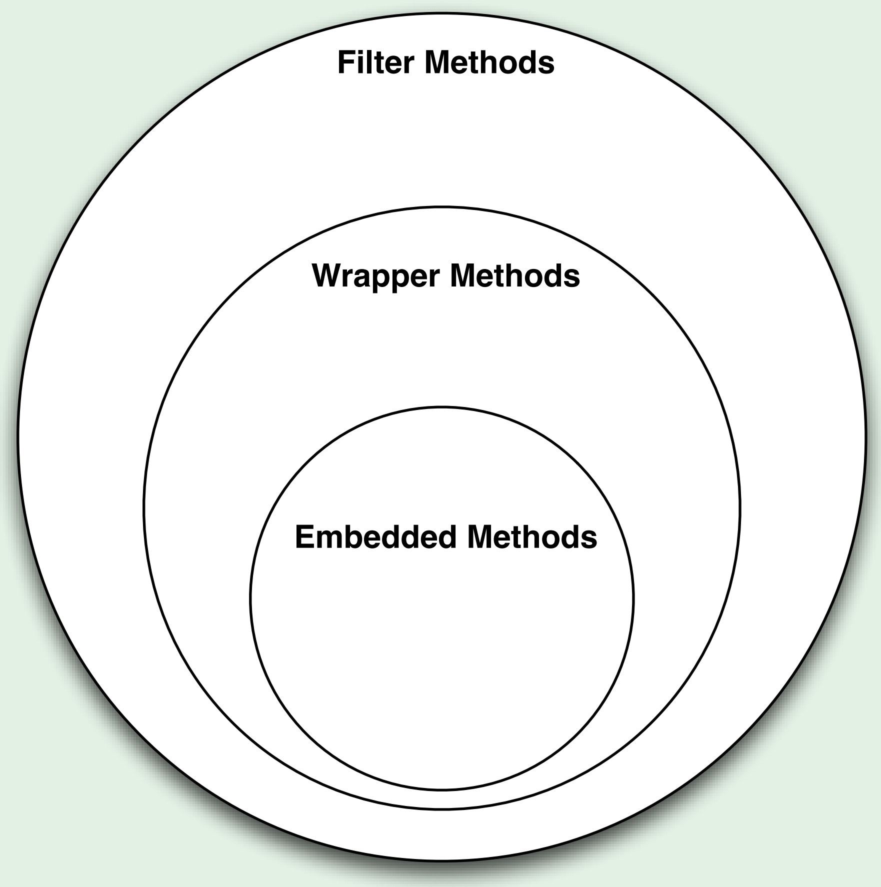
- invariant search

Embedded methods

- specific search



Parameter Space



Model Space

Filters at the highest level of inference, ex. preprocessing

Wrappers and Embedded optimize hyper-parameters

Wrappers treat learning machines as a black-box,assess performance with an evaluation function, ex. cross-validation

Embedded use knowledge of learning machine to search, jointly optimize parameters and hyper-parameters, ex. -log likelihood

Review some recently proposed methods implementing these modules

Filters

i) preprocessing and feature construction

- PCA, clustering

ii) designing regularizers or priors

- methods structuring parameter space

iii) noise modeling

- loss function embeds prior for noise

iv) feature selection

- reduce dimensions of feature space

goal of finding a good data representation, important and hard to automate: domain dependent

priors embed domain knowledge of model class, generally just enforce Occam's Razor

squared loss assumes Guassian noise, distorting training data adds noise

decrease computational costs, often pruning used

Wrappers

- no required knowledge of learning machines/algorithm
- search strategy to explore hyper-parameter space

select a classifier from a set of learning machines

search strategy decides which hyper-parameters considered in which order

regularization guards against over-fitting

Wrappers

- evaluation function to test performance
- select best machine or create ensemble

Bayesians usually use marginal likelihood “evidence”

Frequentists usually use cross-validation

Embedded Methods

- exploit specific features of learning machines/algorithm to search parameters
- Bayesians: compute posterior for parameters and hyper-parameters
- Frequentists: regularized functionals, include the empirical risk and a regularizer

like using gradient descent to find the optimum of a differentiable function

Bayes, hard in practice, often variational methods, which optimize parameters of simpler version of problem

Freq: or negative log likelihood and or a prior, often use wrapper for hyper-parameters

Advances

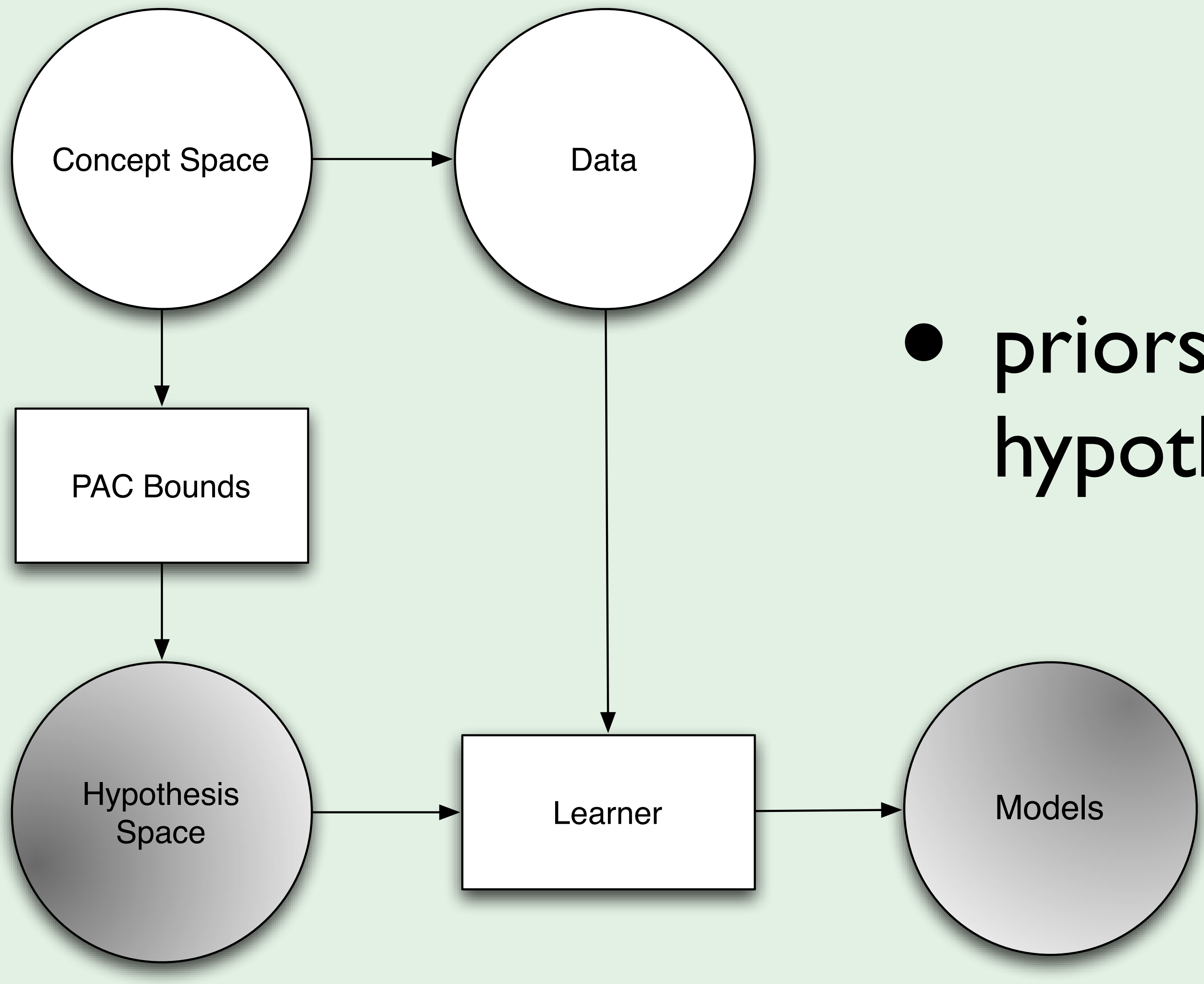
- Ensemble methods
  - Random Forests
  - Heterogenous learners

perform model selection by voting among models

RF subsamples both training examples and features to build learners

combining different types of learning machines successful in competitions

PAC-Bayes



- priors structure hypothesis space

no assumption model comes from concept space that generated the data

can use regularization at PAC–bounds step

Open Problems

- incorporating domain knowledge
- unsupervised learning

automatically incorporating domain knowledge hard.

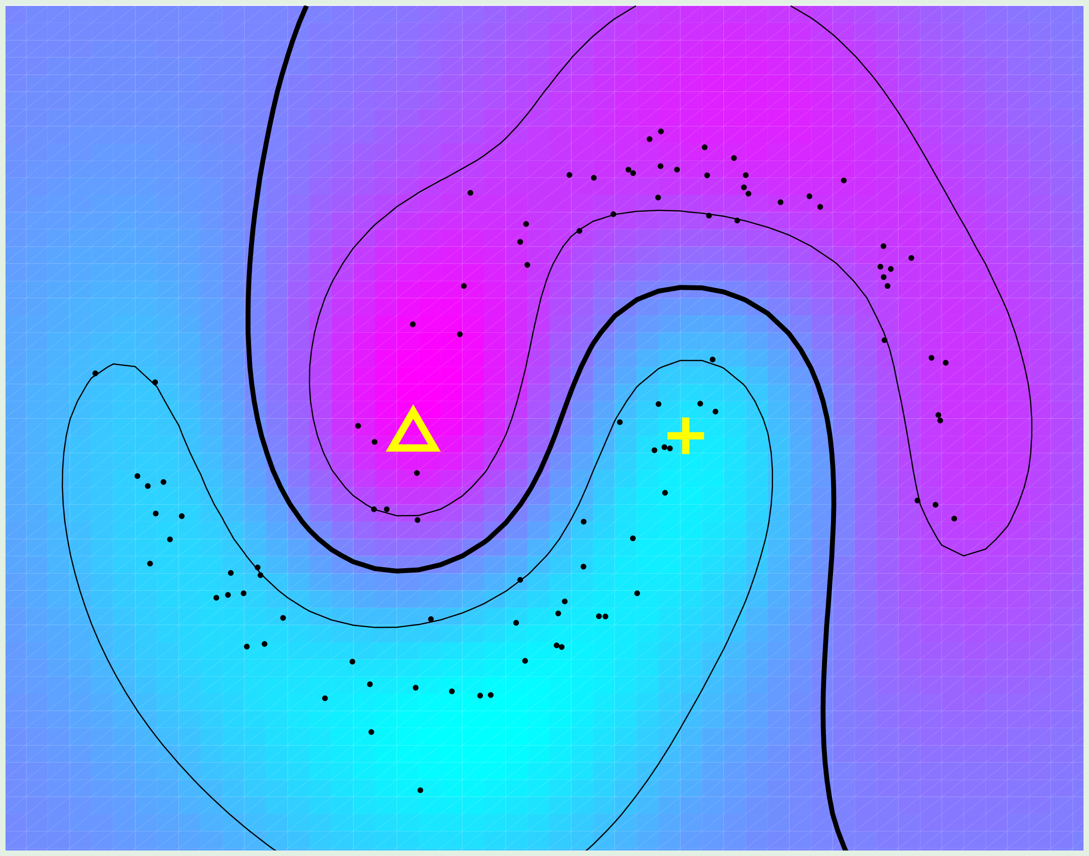
incorporating filter and wrapper methods into machine learning toolboxes can help

how do you validate model selection wrt unsupervised learning?

principled selection in unsupervised learning?

Open Problems

- semi-supervised learning



- what unlabeled data do we use?

Chapelle and others have success with semi-supervise support vector machines

choosing the data to use is a model selection problem

Open Problems

- non-i.i.d. data
- computational cost

when i.i.d. assumption fails significantly cross-validation may not work

better off selecting a model class instead of a single model

need systems that incorporate multiple objectives, accuracy and lower computation cost

applications to online learning

References

- Random Forests
  - http://www.stat.berkeley.edu/~breiman/RandomForests/
- S^3VM
  - http://olivier.chapelle.cc/research.html