University of Potsdam
Dept. of Computer Science

Transfer Learning
Adversarial Learning

Tobias Scheffer

Shopping for PhD Students, Postdocs

- Have you got a Master‘s Degree in CS or Mathematics?
- Or do you have a PhD in a field related to machine learning with a number of first-tier publications?
- Do you feel connected to statistics, programming, and data?
- Do you feel attracted to research in a highly relevant area and a growing research community?
- Do you feel like machine learning is a part of you that has been missing all along?
- Act on these feelings, give me your cv!

University of Potsdam
Dept. of Computer Science

Transfer Learning

Tobias Scheffer

Prerequisites

- Statistics
  - Random variables, distributions
  - Bayes‘ equation
- Linear algebra
  - Vectors and matrices
  - Transposed, inverted matrices
  - Eigenvalues and eigenvectors
- Calculus
  - Derivative, partial derivative
  - Gradient

Respawning Points

- In places, the material may get technically somewhat involved.
- This icon markes points at which you can catch up, if you dropped out earlier.

Overview

- Recap: Supervised learning
  - Empirical inference, graphical models
  - Logistic regression, linear models
- Learning multiple distinct, related problems (transfer learning, domain adaptation)
  - Hierarchical Bayesian Inference
- Learning under covariate shift (differing marginal input distribution)
  - Importance sampling
  - Direct estimation of important weights
- Importance sampling for domain adaptation

Classification

- Input: instance $\mathbf{x} \in X$
  - X may be vector space.
  - Instance is a set of values for these attributes

- Output: class $y \in Y$; finite set Y.
  - Class y also called class label.

Classification: Example

- Input: instance $\mathbf{x} \in X$

  X = Set of all combinations of a base set of substances

Attribute — Instance $\mathbf{x}$

Contains substance 1?

$$\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Combination of drugs

Contains substance 6?

- Output:

$$y \in Y = \{toxic, ok\}$$

Classifier

Tobias Scheffer

Classification: Learning Problem

- Input to learning problem: Data T_n.

$$\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

$$y = (y_1 \quad \ldots \quad y_n)^{\mathrm{T}}$$

- Training data:

$$T_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

Classification: Learning Problem

- Input to learning problem: Data T_n.
  - $\mathbf{X} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$
  - $y = (y_1 \quad \ldots \quad y_n)^{\mathrm{T}}$
- Output: Model
  - $\mapsto$
  - $y_\theta : X \to Y$
  - For instance,

$$y_\theta(\mathbf{x}) = \begin{cases} , & \text{if } \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} \geq 0 \\ , & \text{otherwise} \end{cases}$$

Linear classifier with parameters $\boldsymbol{\theta}$.

- Training data:

$$T_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

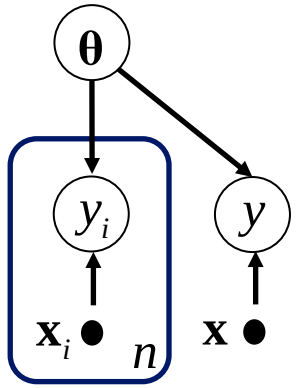
Tobias Scheffer

Empirical Inference

- What is the modt likely class y given instance $\mathbf{x}$ and given the training data?
  - $y^* = \arg\max_y P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{y})$
- Need assumptions about data generation process to solve.

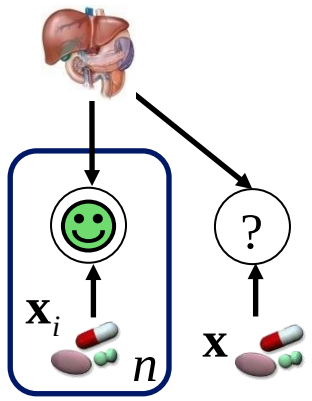
Classification: Graphical Model

- Graphical model defines stochastic process
- Modeling assumptions on data generation process
- First, parameter vector θ is drawn
- This θ parameterizes training data $P(y_i | \mathbf{x}_i, \theta)$
- Marginal distribution $p(\mathbf{x}_i)$ is not part of the model; instances are treated as if constant: discriminative model!



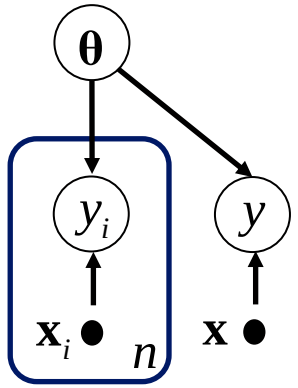
Example

- Evolution guides physiological parameters of humans.
- Given these parameters and a combination of substances, nature rolles dice to determine whether an individual survives ingestion.
- Survival is governed by $p(y_i \mid x_i, \boldsymbol{\theta})$



Empirical Inference

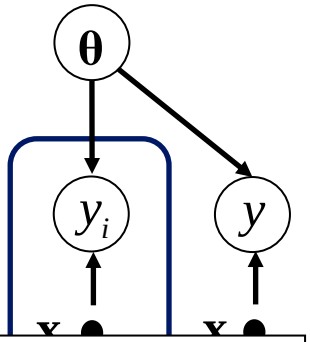
- Inference of the probability of y given $\mathbf{x}$ and training data:
  - $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{y}) = \int p(y, \boldsymbol{\theta} \mid \mathbf{x}, \mathbf{X}, \mathbf{y}) \mathrm{d}\boldsymbol{\theta}$



Empirical Inference

- Inference of the probability of y given $\mathbf{x}$ and training data:
  - $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{y}) = \int p(y, \boldsymbol{\theta} \mid \mathbf{x}, \mathbf{X}, \mathbf{y}) \mathrm{d}\boldsymbol{\theta}$

$$= \int P(y \mid \mathbf{x}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})\, \mathrm{d}\boldsymbol{\theta}$$



Integration over space of all model parameters: Bayesian model averaging

Independence assumptions from graphical model

- No closed-form solution for classification
- Numerical integragtion over space of all model parameters generally infeasible.

Tobias Scheffer

Empirical Inference

- Inference of the probability of y given $\mathbf{x}$ and training data:
  - $$P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{y}) = \int p(y, \boldsymbol{\theta} \mid \mathbf{x}, \mathbf{X}, \mathbf{y}) \mathrm{d}\boldsymbol{\theta}$$
    $$= \int P(y \mid \mathbf{x}, \boldsymbol{\theta})\, p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})\, \mathrm{d}\boldsymbol{\theta}$$
    $$\approx P(y \mid \mathbf{x}, \boldsymbol{\theta}_{\mathrm{MAP}}) \text{ with } \boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$$
- Approximation using only the single most likelyparameter vector: MAP model.

Empirical Inference

- Inference of the probability of y given $\mathbf{x}$ and training data:
  - $P(\cdot \mid \cdot, \mathbf{X}, \mathbf{y}) = \int p(\cdot, \cdot \mid \cdot, \mathbf{X}, \mathbf{y}) \mathrm{d}\cdot$

    $= \int P(\cdot \mid \cdot, \cdot)\, p(\cdot \mid \mathbf{X}, \mathbf{y})\, \mathrm{d}\cdot$

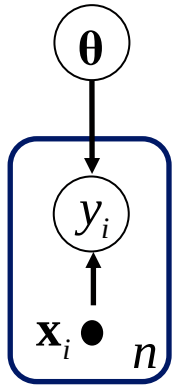
    $\approx P(\cdot \mid \cdot, \cdot_{MAP})$ with $\cdot_{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$

    Training data

    Model
- Approximation using only the single most likelyparameter vector: MAP model.

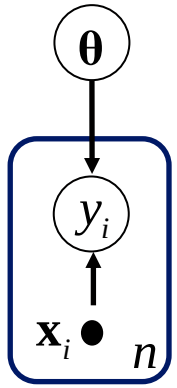
Empirical Inference

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$$
    $$= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})}$$
    $$= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$



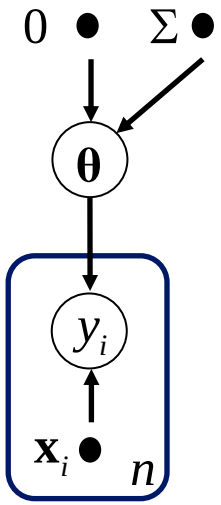
Empirical Inference

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $$\begin{aligned}\boldsymbol{\theta}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y}) \\ &= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})} \\ &= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta})\end{aligned}$$



Empirical Inference: Logistic Regression

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$$
    $$= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})}$$
    $$= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$
    $$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) N[0, \Sigma](\boldsymbol{\theta})$$



Assuming that $p(\mathbf{x}_i \mid y_i, \boldsymbol{\theta})$ is an exponential family leads to logistic regression

Empirical Inference: Logistic Regression

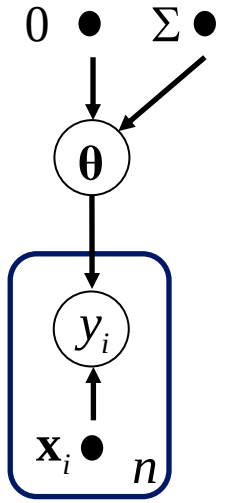
- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$

$$= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})}$$

$$= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$

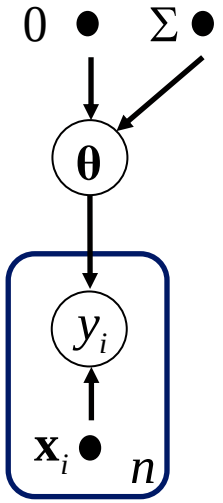
$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) N[0, \Sigma](\boldsymbol{\theta})$$

$$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left(\log \sum_{y'} e^{\phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y'}} - \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y_i} \right) + \frac{1}{2} \boldsymbol{\theta}^{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\theta}$$



Empirical Inference: Logistic Regression

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$$
    $$= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})}$$
    $$= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$$
    $$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) N[0, \Sigma](\boldsymbol{\theta})$$
    $$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left(\log \sum_{y'} e^{\phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y'}} - \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y_i} \right) + \frac{1}{2} \boldsymbol{\theta}^{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\theta}$$



- Inference of most likely class:
  - $$\arg\max P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \arg\max_{y} \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y_i} + b_y$$

Linear model in features ϕ.

Empirical Inference: Logistic Regression

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$

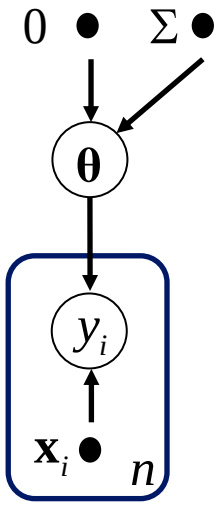
$$= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X})}$$

$$= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1} [\quad, \sigma^2](\boldsymbol{\theta})$$

$$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left(\log \sum_{y'} e^{\phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y'}} - \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y_i} \right) + \frac{1}{2} \boldsymbol{\theta}^{T} \boldsymbol{\Sigma}^{-1} \boldsymbol{\theta}$$

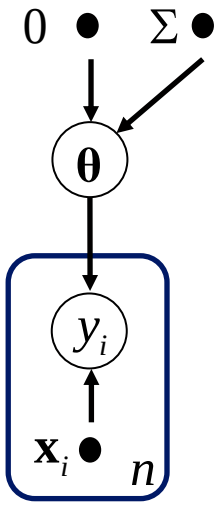
Training data



- Inference of most likely class:
  - $\arg\max P(\text{☺}_i \mid \text{💊}_i, \boldsymbol{\theta}) = \arg\max_y \phi(\text{💊}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y_i} + b_y$

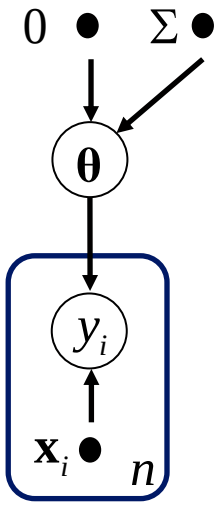
Empirical Inference

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $$\begin{aligned}\boldsymbol{\theta}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y}) \\ &= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})} \\ &= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta})\end{aligned}$$



Empirical Inference: reg. ERM

- Inference of $\boldsymbol{\theta}_{\text{MAP}}$:
  - $$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y})$$
    $$= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})}$$
    $$= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$
    $$= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta})$$
    $$= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$$

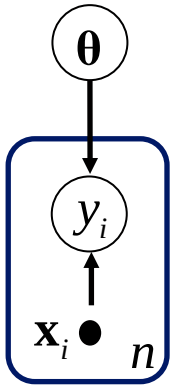


Substituting log-likelihood for general loss function and…

… substituting log-prior for general regularizer leads to regularized empirical risk minimization

Empirical Inference: reg. ERM

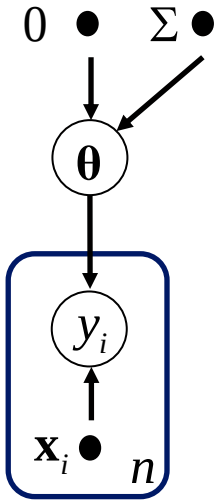
- Inference of $\boldsymbol{\theta}^*$:
  - $$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y}) \\
&= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})} \\
&= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})
\end{aligned}$$



Regularized empirical risk minimization

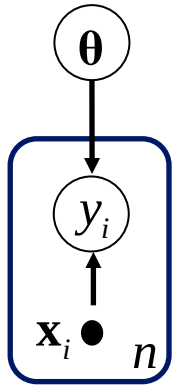
Empirical Inference: reg. ERM

- Inference of $\boldsymbol{\theta}^*$:
  - $$\begin{aligned}
\boldsymbol{\theta}^* &= \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}, \mathbf{y}) \\
&= \arg\max_{\boldsymbol{\theta}} \frac{P(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})}{p(\mathbf{X}, \mathbf{y})} \\
&= \arg\max_{\boldsymbol{\theta}} P(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \boldsymbol{\theta}^{\mathrm{T}} \Sigma^{-1} \boldsymbol{\theta}
\end{aligned}$$



Empirical Inference: reg. ERM

- Alternative rationalization: minimize risk = expected loss
  - $\boldsymbol{\theta}^{\otimes} = \arg\min_{\boldsymbol{\theta}} \int \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) p(\mathbf{x}_i, y_i)$
- Distribution $p(\mathbf{x}_i, y_i)$ not known
  $\rightarrow$ approximateby sum over sample
  - $\approx \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i))$
- Minimization problem is ill-posed; small change in data can lead to large change in parameters. Smooth by adding Tikhonov regularizer
  - $\approx \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})$



Regularized empirical risk minimization

Solving the Optimization Problem

- Goal: Minimize function $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i \right) + \Omega(\boldsymbol{\theta})$ for given loss function and regularizer

- Numeric solutions:
  - Gradient descent
  - Cutting plane method
  - Interior point method
  - …

Solving the Optimization Problem

- Goal: Minimize function $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i \right) + \Omega(\boldsymbol{\theta})$ for given loss function and regularizer

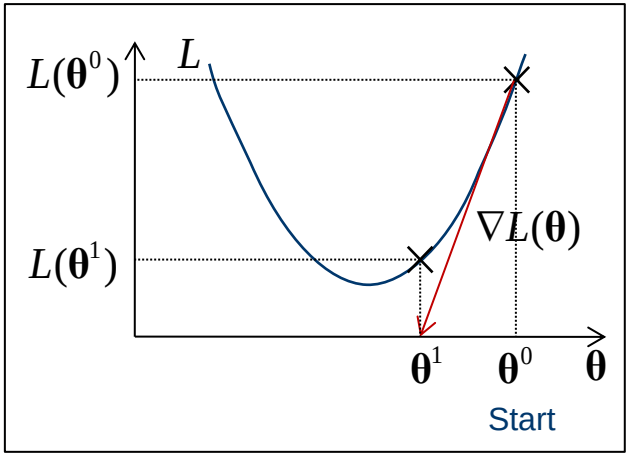
- Gradient: vector of partial derivatives
- Ascent direction for function $L(\boldsymbol{\theta})$.

$$\nabla L(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial}{\partial \theta_1^{\phi,1}} L(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_m^{\phi,1}} L(\boldsymbol{\theta}) \\ \vdots \\ \frac{\partial}{\partial \theta_m^{\phi,k}} L(\boldsymbol{\theta}) \end{pmatrix}$$

Solving the Optimization Problem

- Goal: Minimize function $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i \right) + \Omega(\boldsymbol{\theta})$ for given loss function and regularizer

- Gradient descent:
  - Iterative procedure.
  - In each step, move into direction of steepest descent
  - Descent direction given by negative gradient.



Solving the Optimization Problem

- Goal: Minimize function $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i \right) + \Omega(\boldsymbol{\theta})$ for given loss function and regularizer

- Gradient descent:

RegERM(*Data* $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$)

Let $\boldsymbol{\theta}^0 = \mathbf{0}$ and $k = 0$

DO

determine gradient $\nabla L(\boldsymbol{\theta}^k)$

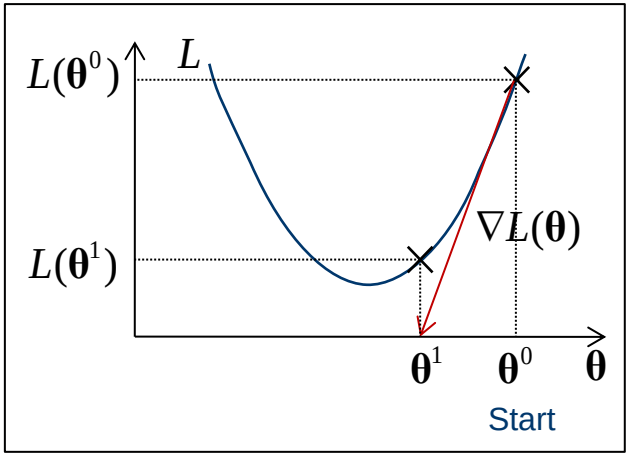
determine step size α^k

Let $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \alpha^k \nabla L(\boldsymbol{\theta}^k)$

Let $k = k + 1$

WHILE $\left\| \boldsymbol{\theta}^k - \boldsymbol{\theta}^{k-1} \right\| > \varepsilon$

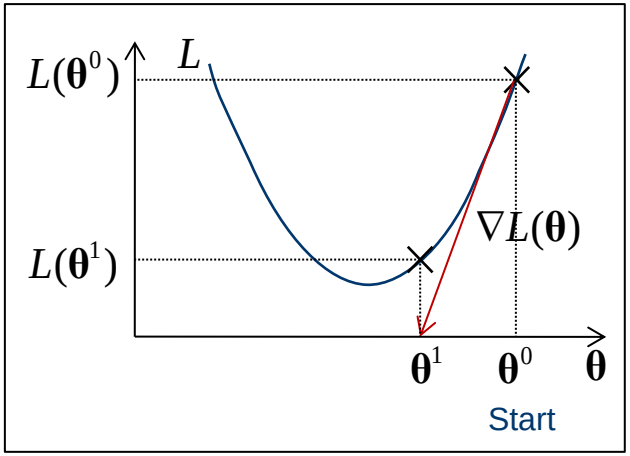
RETURN $\boldsymbol{\theta}^k$



Solving the Optimization Problem

- Goal: Minimize function $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \ell\left(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i \right) + \Omega(\boldsymbol{\theta})$ for given loss function and regularizer

- Gradient descent:
  - $L(\boldsymbol{\theta})$ decreases in each step.
  - If $L(\boldsymbol{\theta})$ is convex, converges to global minimum
  - If loss function and regularizer are convex, $L(\boldsymbol{\theta})$ is convex.



Setting the Regularization Parameter

- Optimization criterion has a parameter:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \boldsymbol{\theta}^{\mathrm{T}} \Sigma^{-1} \boldsymbol{\theta}$$

$$\approx \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \frac{1}{\lambda} \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\theta}$$

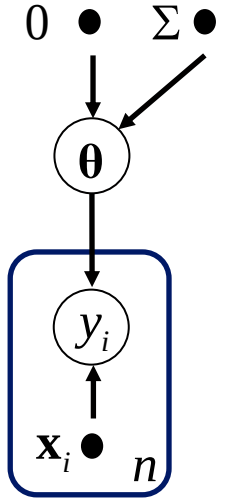
One parameter per element of θ, even if Σ is diagonal

One parameter λ in total

- To set parameter, use grid search and n-fold cross validation (when training sample is large, one training-and-test split)

Empirical Inference

- Graphical model formulates assumptions that lead to
  - $\boldsymbol{\theta}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$
- $\boldsymbol{\theta}_{\text{MAP}}$ is the most likely model given prior and given training data.
- Substituting log-likelihood for general loss function and log-prior for general regularizer leads to regularized empirical risk minimization
  - $\boldsymbol{\theta}^{*} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \ell(y_i, y_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})$
- Minimum can be found by gradient descent.

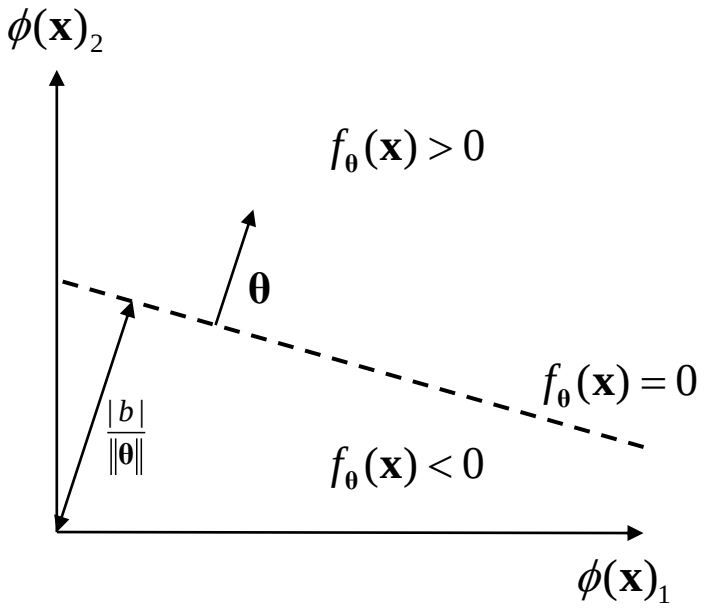


Linear Models

- Hyperplane defined by normal vector and offset:

$$H_{\boldsymbol{\theta},b} = \{\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + b = 0\}$$

- Class probability (two classes, logistic regression):
  - $P(y_i = +1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = \dfrac{1}{1 + e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b}}$
- Decision function:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b$
- Classifier:
  - $y_{\boldsymbol{\theta}}(\mathbf{x}_i) = \mathrm{sign}(\, f_{\boldsymbol{\theta}}(\mathbf{x}_i))$



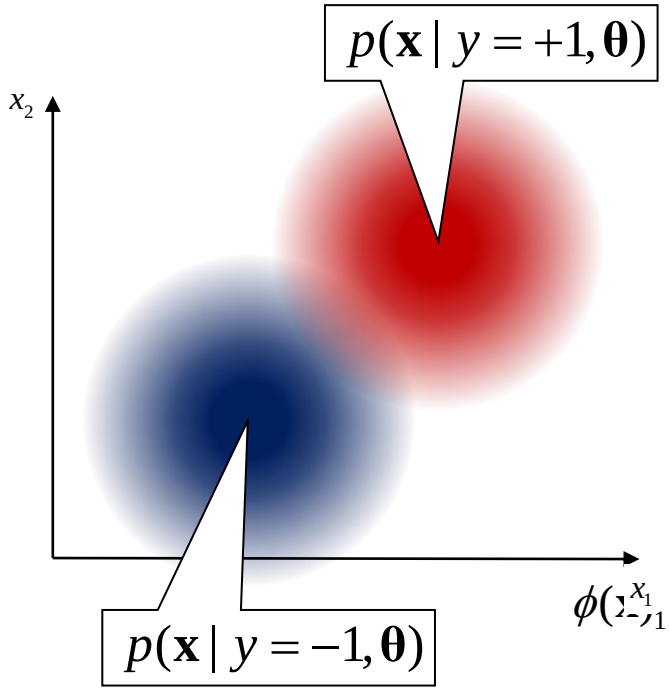
Linear Models

- Hyperplane defined by normal vector and offset:

$$H_{\boldsymbol{\theta},b} = \{\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + b = 0\}$$

$$\phi(\mathbf{x}_i) = \mathbf{x}_i$$

- Class probability (two classes, logistic regression):
  - $P(y_i = +1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = \dfrac{1}{1 + e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b}}$
- Decision function:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b$
- Classifier:
  - $y_{\boldsymbol{\theta}}(\mathbf{x}_i) = \mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))$



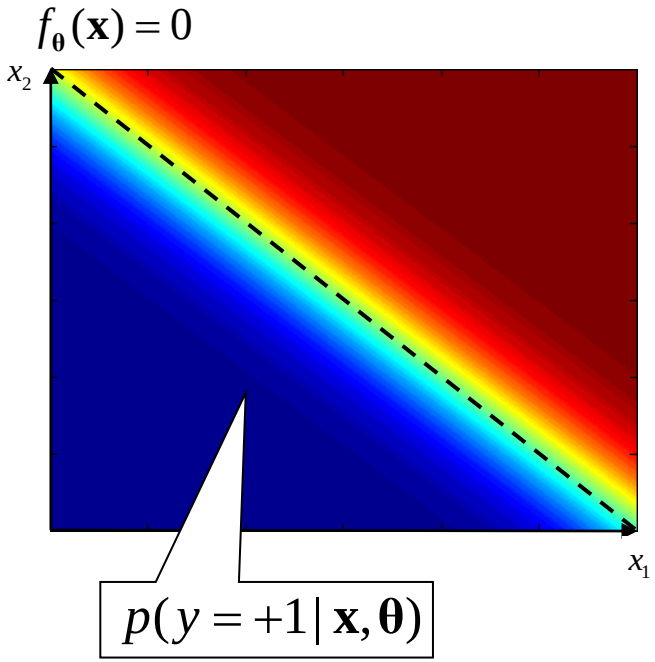
Linear Models

- Hyperplane defined by normal vector and offset:

$$H_{\boldsymbol{\theta},b} = \{\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + b = 0\}$$

$$\phi(\mathbf{x}_i) = \mathbf{x}_i$$

- Class probability (two classes, logistic regression):
  - $P(y_i = +1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = \dfrac{1}{1 + e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b}}$
- Decision function:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b$
- Classifier:
  - $y_{\boldsymbol{\theta}}(\mathbf{x}_i) = \mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))$



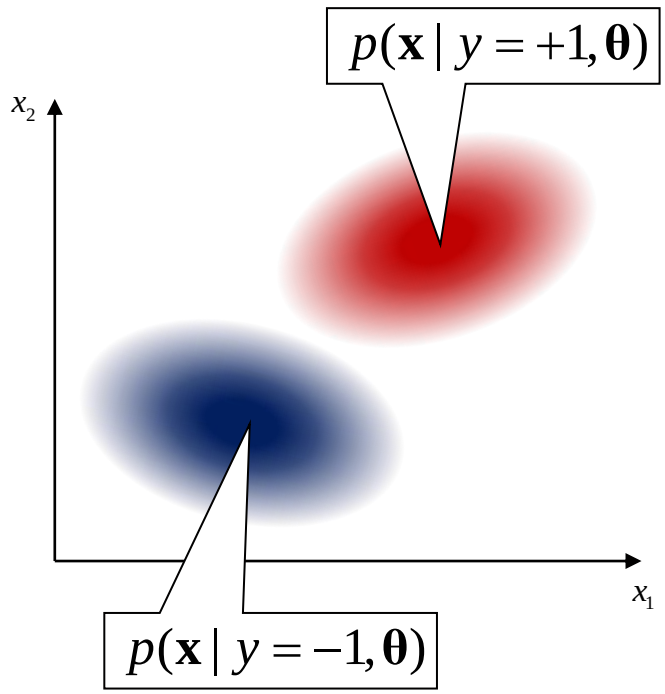
Linear Models

- Hyperplane defined by normal vector and offset:

$$H_{\boldsymbol{\theta},b} = \{\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + b = 0\}$$

$$\phi(\mathbf{x}_i) = \begin{pmatrix} 1 \\ \mathbf{x}_i \\ \mathbf{x}_i \otimes \mathbf{x}_i \end{pmatrix}$$

- Class probability (two classes, logistic regression):
  - $P(y_i = +1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = \dfrac{1}{1 + e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b}}$



- Decision function:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b$
- Classifier:
  - $y_{\boldsymbol{\theta}}(\mathbf{x}_i) = \mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))$

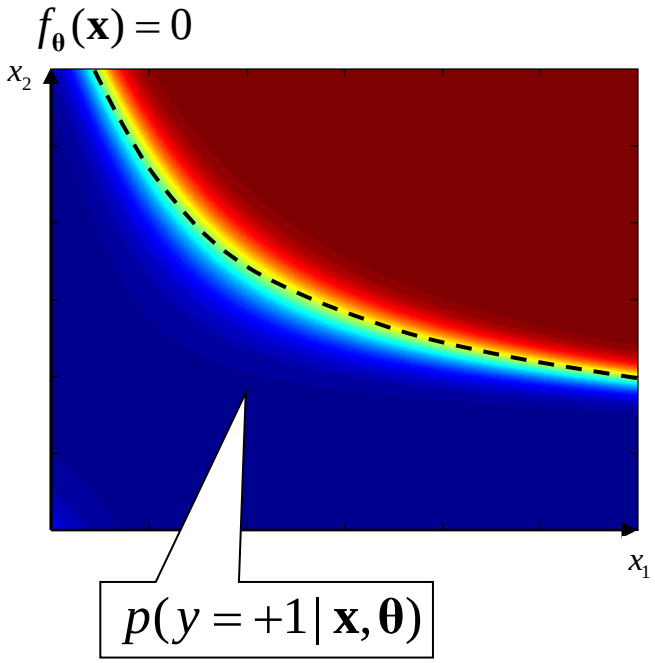
Linear Models

- Hyperplane defined by normal vector and offset:

$$H_{\boldsymbol{\theta},b} = \{\mathbf{x} \mid f_{\boldsymbol{\theta}}(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\boldsymbol{\theta} + b = 0\}$$

$$\phi(\mathbf{x}_i) = \begin{pmatrix} 1 \\ \mathbf{x}_i \\ \mathbf{x}_i \otimes \mathbf{x}_i \end{pmatrix}$$

- Class probability (two classes, logistic regression):
  - $P(y_i = +1 \mid \mathbf{x}_i, \boldsymbol{\theta}) = \dfrac{1}{1 + e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b}}$



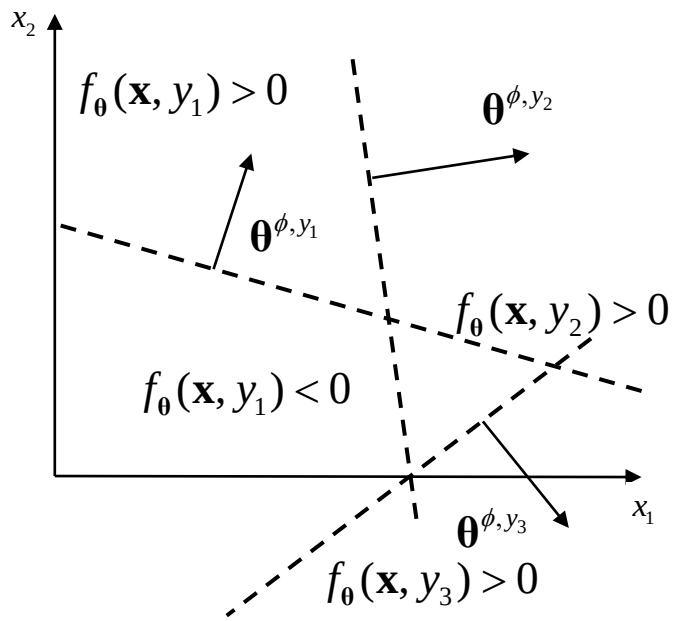
- Decision function:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\theta}^{\phi} + b$
- Classifier:
  - $y_{\boldsymbol{\theta}}(\mathbf{x}_i) = \mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))$

Linear Models – Multi-Class Case

- Hyperplanes defined by normal vector and offset:

$$H_{\boldsymbol{\theta}^{\phi,y_i},b_{y_i}} = \{\mathbf{x} \mid f_\theta(\mathbf{x}, y_i) = \phi(\mathbf{x})^{\mathrm{T}} \boldsymbol{\theta}^{\phi,y_i} + b_{y_i} = 0\}$$

- Class probability (logistic regression):
  - $$P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \frac{e^{\phi(\mathbf{x})^{\mathrm{T}} \boldsymbol{\theta}^{\phi,y_i} + b_{y_i}}}{\sum_{y'} e^{\phi(\mathbf{x})^{\mathrm{T}} \boldsymbol{\theta}^{\phi,y'} + b_{y'}}}$$
- Decision function:
  - $$f_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i) = \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi,y_i} + b_{y_i}$$
- Classifier:
  - $$y_{\boldsymbol{\theta}}(\mathbf{x}_i) = \arg\max_y \; f_{\boldsymbol{\theta}}(\mathbf{x}_i, y)$$

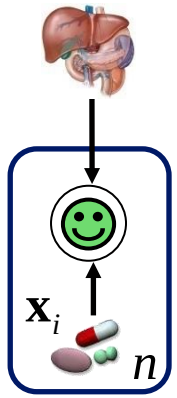


Linear Models

- Decision function of generalized linear classifiers:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi} + b$
- Multi-class case:
  - $f_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i) = \phi(\mathbf{x}_i)^{\mathrm{T}} \boldsymbol{\theta}^{\phi, y_i} + b_{y_i}$
- Linear case: $\phi(\mathbf{x}) = \mathbf{x}$
- General feature mapping leads to kernel machines.

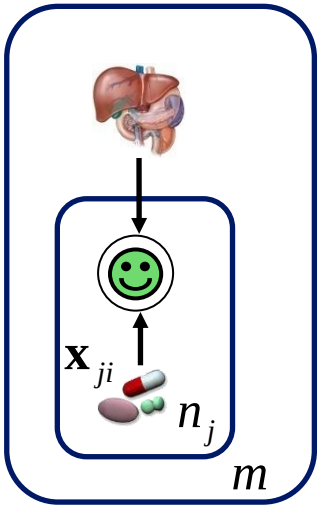
Multiple Kids Experimenting with Drugs

- So far: only one distribution $p(y_i \mid x_i, \boldsymbol{\theta})$ for everyone.
- What about different metabolic rates? Differing genetic factors? Levels of tolerance?



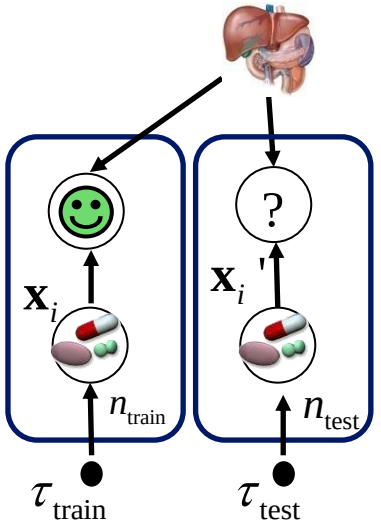
Multiple Kids Experimenting with Drugs

- We could learn distinct models for each user j:
  - $\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathbf{X}_j, \mathbf{y}_j)$
  - $\boldsymbol{\theta}_j^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n_j} \ell(y_{ji}, y_{\boldsymbol{\theta}}(\mathbf{x}_{ji})) + \Omega(\boldsymbol{\theta})$
- But this would treat individuals as independent and disregard much data (if many previous users died, maybe this does tell you something)

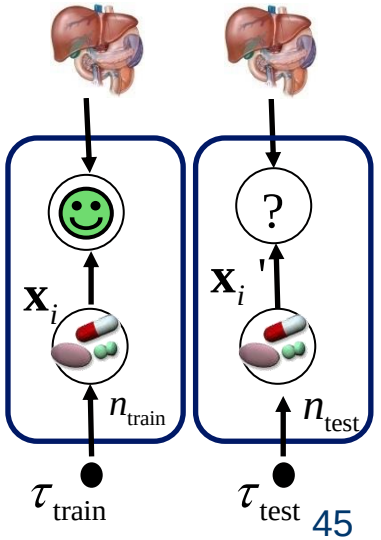


Forms of Transfer Learning

- Learning under covariate shift
  - Only marginal distributions $p(\mathbf{x}_i \mid \tau_{\text{train}})$ and $p(\mathbf{x}_i \mid \tau_{\text{test}})$ differ; conditional $P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta})$ is constant.
  - Goal: minimize risk over test distribution.

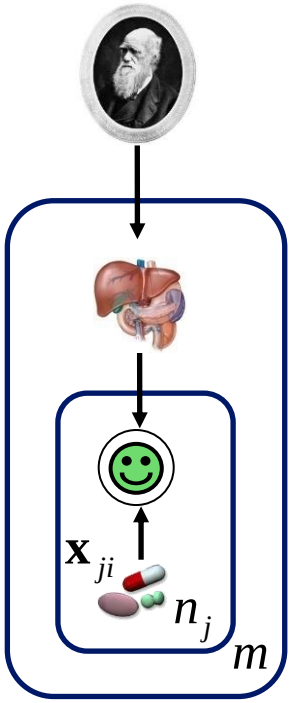


- Multi-task learning, domain adaptation
  - Both, marginal distributions and conditional distributions $P(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}_{train/test})$ may differ



Hierarchical Bayesian Inference

- Instead, model a common prior over individuals (tasks) $p(y_{ji} \mid x_{ji}, \boldsymbol{\theta}_j)$
- Physiology of everyone has been produced by the same process of evolution.

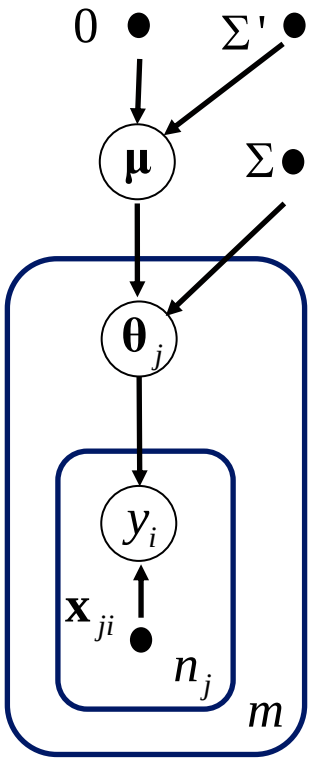


Hierarchical Bayesian Inference

- Instead, model a common prior over individuals (tasks) $p(y_{ji} \mid x_{ji}, \boldsymbol{\theta}_j)$
- Physiology of everyone has been produced by the same process of evolution.
- $p(\boldsymbol{\mu} \mid \boldsymbol{\Sigma}') = N[0, \boldsymbol{\Sigma}'](\boldsymbol{\mu})$
- $p(\boldsymbol{\theta}_j \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = N[\boldsymbol{\mu}, \boldsymbol{\Sigma}](\boldsymbol{\theta}_j)$

- Substitution: $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{v}$

  $\Rightarrow p(\mathbf{v}_j \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = N[0, \boldsymbol{\Sigma}](\boldsymbol{\theta}_j)$

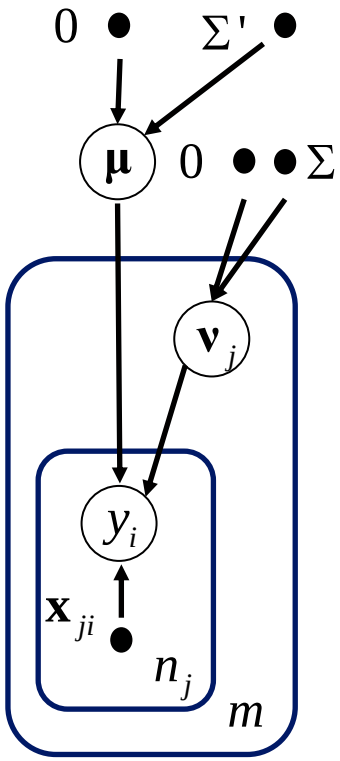


Hierarchical Bayesian Inference

- Instead, model a common prior over individuals (tasks) $p(y_{ji} \mid x_{ji}, \boldsymbol{\theta}_j)$
- Physiology of everyone has been produced by the same process of evolution.
- $p(\boldsymbol{\mu} \mid \boldsymbol{\Sigma}') = N[0, \boldsymbol{\Sigma}'](\boldsymbol{\mu})$
- $p(\boldsymbol{\theta}_j \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = N[\boldsymbol{\mu}, \boldsymbol{\Sigma}](\boldsymbol{\theta}_j)$

- Substitution: $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{v}$

  $\Rightarrow p(\mathbf{v}_j \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = N[0, \boldsymbol{\Sigma}](\mathbf{v}_j)$



Hierarchical Bayesian Inference

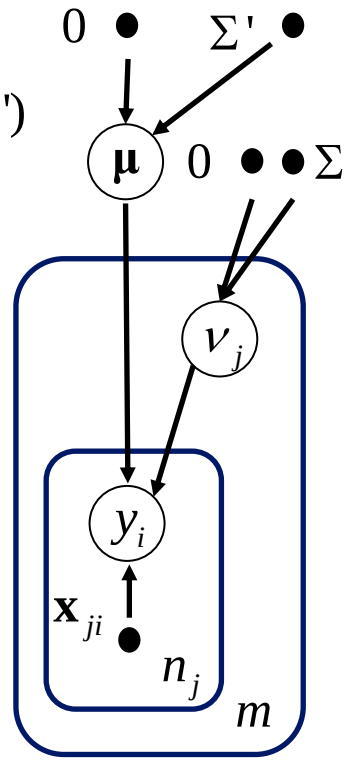
- Inference of $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\mu}+\boldsymbol{\nu}} p(\boldsymbol{\mu}+\boldsymbol{\nu} \mid \mathbf{X}, \mathbf{y}, \boldsymbol{\Sigma}, \boldsymbol{\Sigma}')$$

$$= \arg\max_{\boldsymbol{\mu}+\boldsymbol{\nu}} \prod_j P(\mathbf{y}_j \mid \mathbf{X}_j, \boldsymbol{\mu}+\boldsymbol{\nu}) \prod_j p(\boldsymbol{\nu}_j \mid \boldsymbol{\Sigma}) p(\boldsymbol{\mu} \mid \boldsymbol{\Sigma}')$$

$$= \arg\min_{\boldsymbol{\mu}+\boldsymbol{\nu}} \left(\begin{array}{l} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \log P(y_{ji} \mid \mathbf{x}_{ji}, \boldsymbol{\mu}+\boldsymbol{\nu}) \\ +\sum_j \log p(\boldsymbol{\nu}_j \mid \boldsymbol{\Sigma}) + \log p(\boldsymbol{\mu} \mid \boldsymbol{\Sigma}') \end{array} \right)$$

$$= \arg\min_{\boldsymbol{\mu}+\boldsymbol{\nu}} \left(\begin{array}{l} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \ell(y_{ji}, y_{\boldsymbol{\mu}+\boldsymbol{\nu}}(\mathbf{x}_{ji})) \\ +\sum_j \boldsymbol{\nu}_j^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\nu}_j + \boldsymbol{\mu}_j^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_j \end{array} \right)$$

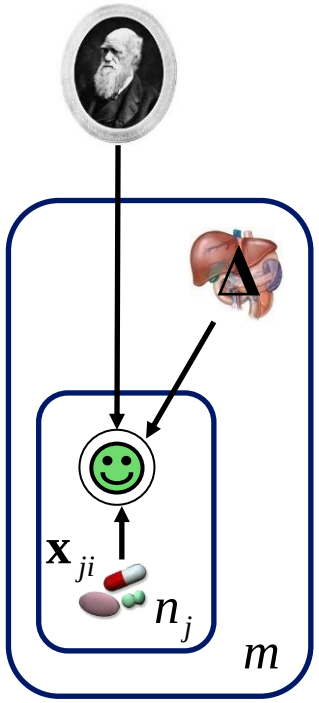


- For general loss and regularizers:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\mu}+\boldsymbol{\nu}} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \ell(y_{ji}, y_{\boldsymbol{\mu}+\boldsymbol{\nu}}(\mathbf{x}_{ji})) + \sum_j \Omega(\boldsymbol{\nu}_j) + \Omega(\boldsymbol{\mu})$$

Hierarchical Bayesian Inference

- Parameters = sum of population-specific and individual parameters.
- Decision function for individual j:
  - $f_{\boldsymbol{\theta},j}(\mathbf{x}) = (\boldsymbol{\mu} + \boldsymbol{\nu}_j)^{\mathrm{T}} \phi(\mathbf{x})$



- Optimization criterion (learning):

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\mu}+\boldsymbol{\nu}} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \ell(\mathbf{x}_{ji}, y_{\boldsymbol{\mu}+\boldsymbol{\nu}}(\mathbf{x}_{ji})) + \sum_j \Omega(\boldsymbol{\mu}+\boldsymbol{\nu}_j) + \Omega(\boldsymbol{\mu})$$

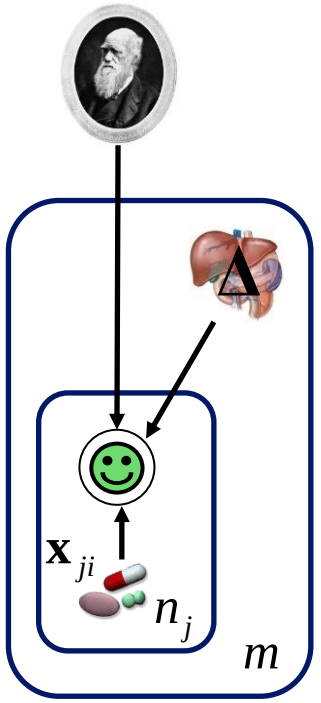
Individual parameters

Individual parameters + population average

population average

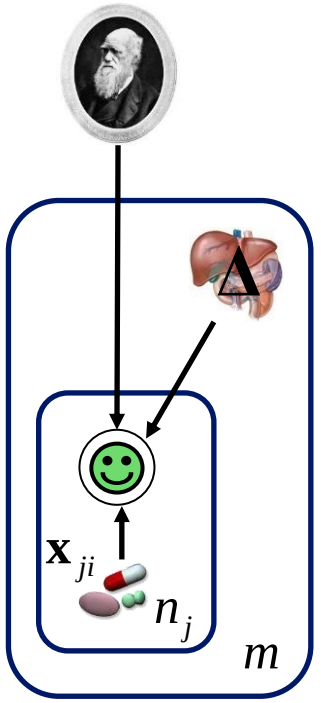
Hierarchical Bayesian Inference

- A simple trick for implementing hierarchical Bayes
- Decision function for individual j:
  - $$f_{\boldsymbol{\theta},j}(\mathbf{x}) = (\boldsymbol{\mu} + \boldsymbol{\nu}_j)^{\mathrm{T}} \phi(\mathbf{x})$$
    $$= \boldsymbol{\mu}^{\mathrm{T}} \phi(\mathbf{x}) + \boldsymbol{\nu}_1^{\mathrm{T}} \phi(\mathbf{x}) \delta(j=1)$$
    $$+ \ldots + \boldsymbol{\nu}_m^{\mathrm{T}} \phi(\mathbf{x}) \delta(j=m)$$



Hierarchical Bayesian Inference

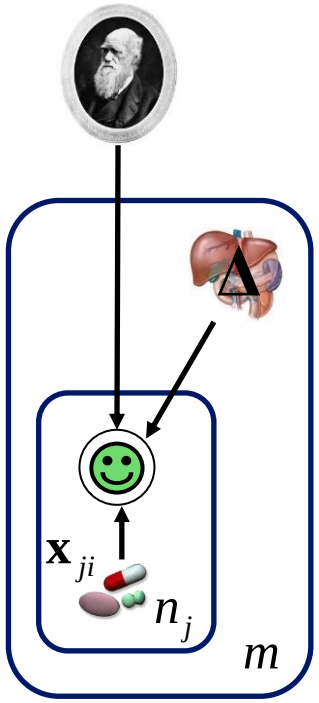
- A simple trick for implementing hierarchical Bayes
- Decision function for individual j:
  - $$f_{\boldsymbol{\theta},j}(\mathbf{x}) = (\boldsymbol{\mu} + \boldsymbol{\nu}_j)^{\mathrm{T}} \phi(\mathbf{x})$$
    $$= \boldsymbol{\mu}^{\mathrm{T}} \phi(\mathbf{x}) + \boldsymbol{\nu}_1^{\mathrm{T}} \phi(\mathbf{x}) \delta(j=1) + \ldots + \boldsymbol{\nu}_m^{\mathrm{T}} \phi(\mathbf{x}) \delta(j=m)$$
    $$= \begin{pmatrix} \boldsymbol{\nu}_1 \\ \ldots \\ \boldsymbol{\nu}_m \\ \boldsymbol{\mu} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \phi(\mathbf{x})\delta(j=1) \\ \ldots \\ \phi(\mathbf{x})\delta(j=m) \\ \phi(\mathbf{x}) \end{pmatrix}$$



[Daume III et al., Frustratingly easy domain adaptation, 2010]

Hierarchical Bayesian Inference

- A simple trick for implementing hierarchical Bayes
- Decision function for individual j:
  - $$f_{\boldsymbol{\theta},j}(\mathbf{x}) = (\boldsymbol{\mu} + \boldsymbol{\nu}_j)^{\mathrm{T}} \phi(\mathbf{x})$$
    $$= \boldsymbol{\mu}^{\mathrm{T}} \phi(\mathbf{x}) + \boldsymbol{\nu}_1^{\mathrm{T}} \phi(\mathbf{x}) \delta(j=1) + \ldots + \boldsymbol{\nu}_m^{\mathrm{T}} \phi(\mathbf{x}) \delta(j=m)$$
    $$= \begin{pmatrix} \boldsymbol{\nu}_1 \\ \ldots \\ \boldsymbol{\nu}_m \\ \boldsymbol{\mu} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} \phi(\mathbf{x})\delta(j=1) \\ \ldots \\ \phi(\mathbf{x})\delta(j=m) \\ \phi(\mathbf{x}) \end{pmatrix}$$

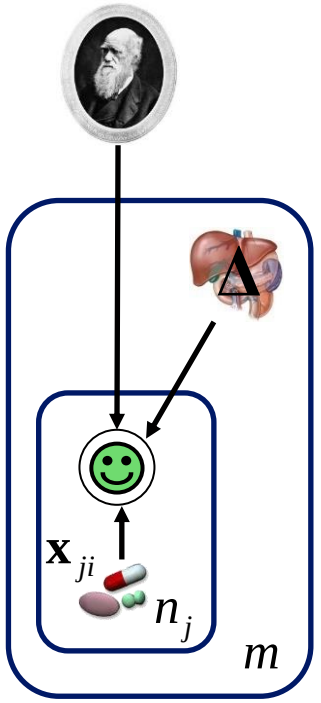


One shared copy of the attributes for entire population

One copy of the attributes for each individual (task)

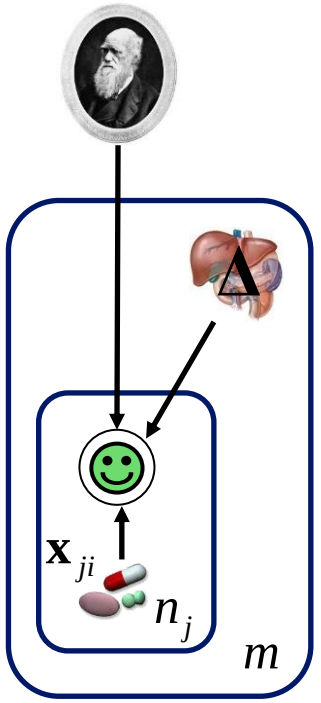
Hierarchical Bayesian Inference

- A simple trick for implementing hierarchical Bayes
- Decision function for individual j:
  - $$f_{\boldsymbol{\theta},j}(\mathbf{x}) = \underbrace{\begin{pmatrix} \mathbf{v}_1 \\ \dots \\ \mathbf{v}_m \\ \boldsymbol{\mu} \end{pmatrix}^{\mathrm{T}}}_{\boldsymbol{\theta}} \underbrace{\begin{pmatrix} \phi(\mathbf{x})\delta(j=1) \\ \dots \\ \phi(\mathbf{x})\delta(j=m) \\ \phi(\mathbf{x}) \end{pmatrix}}_{\boldsymbol{\Phi}_j(\mathbf{x})}$$
- Learning problem:
  - $$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \ell(y_{ji}, \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\Phi}_j(\mathbf{x}_{ji})) + \Omega(\boldsymbol{\theta})$$



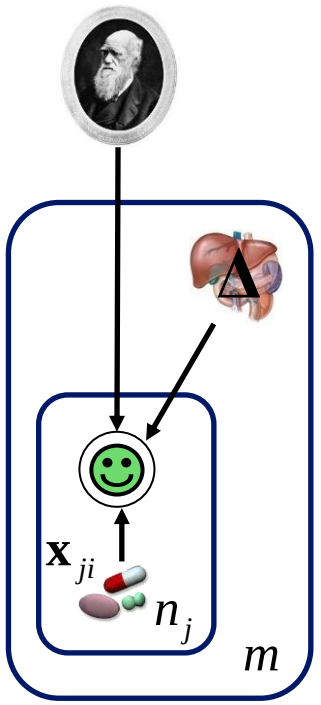
Hierarchical Bayesian Inference

- A simple trick for implementing hierarchical Bayes
- Decision function for individual j:
  - $$f_{\boldsymbol{\theta},j}(\mathbf{x}) = \underbrace{\begin{pmatrix} \mathbf{v}_1 \\ \dots \\ \mathbf{v}_m \\ \boldsymbol{\mu} \end{pmatrix}^{\mathrm{T}}}_{\boldsymbol{\theta}} \underbrace{\begin{pmatrix} \phi(\mathbf{x})\delta(j=1) \\ \dots \\ \phi(\mathbf{x})\delta(j=m) \\ \phi(\mathbf{x}) \end{pmatrix}}_{\boldsymbol{\Phi}_j(\mathbf{x})}$$
- Learning problem:
  - $$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \ell(y_{ji}, \boldsymbol{\theta}^{\mathrm{T}} \boldsymbol{\Phi}_j(\mathbf{x}_{ji})) + \Omega(\boldsymbol{\theta})$$
- Implement using any learning algorithm by constructing appropriate $\boldsymbol{\Phi}$



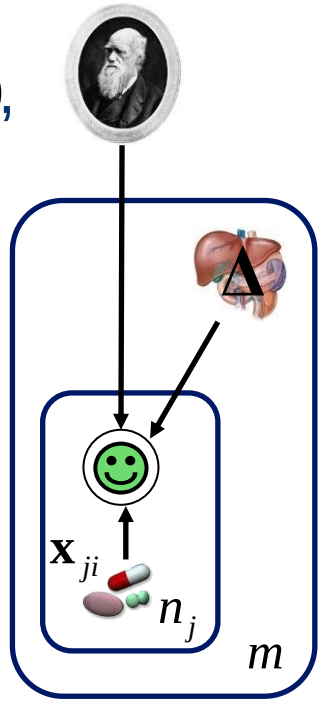
Hierarchical Bayesian Inference: Exercise

- Three kids, training set
  - $\{\{(\mathbf{x}_{11}, y_{11}), (\mathbf{x}_{12}, y_{12}), (\mathbf{x}_{13}, y_{13})\},$
    $\{(\mathbf{x}_{21}, y_{21}), (\mathbf{x}_{22}, y_{22})\},$
    $\{(\mathbf{x}_{31}, y_{31})\}\}$
- Construct the training set that maps this multi-task problem to a regular SVM
- Kid 3 takes substances $\mathbf{x}$. Will he live? How do you use the resulting SVM for inference?
- Kid 4 takes substances $\mathbf{x}$. Can we say anything about whether she will live?



Hierarchical Bayesian Inference

- Graphical model: population-specific and individual factors determine model parameters.
- When $p(\boldsymbol{\mu} \mid \boldsymbol{\Sigma}') = N[0, \boldsymbol{\Sigma}'](\boldsymbol{\mu})$, $p(\mathbf{v}_j \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = N[0, \boldsymbol{\Sigma}](\mathbf{v}_j)$, and $\boldsymbol{\theta} = \boldsymbol{\mu} + \mathbf{v}$, then
  - $\boldsymbol{\theta}_{\text{MAP}} = \arg\min_{\boldsymbol{\mu}+\mathbf{v}} \left(\sum_{j=1}^{m} \sum_{i=1}^{n_j} \log P(y_{ji} \mid \mathbf{x}_{ji}, \boldsymbol{\mu} + \mathbf{v}) + \sum_j \log p(\mathbf{v}_j \mid \boldsymbol{\Sigma}) + \log p(\boldsymbol{\mu} \mid \boldsymbol{\Sigma}') \right)$
- For general loss and regularizers,
  - $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\mu}+\mathbf{v}} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \ell(y_{ji}, y_{\boldsymbol{\mu}+\mathbf{v}}(\mathbf{x}_{ji})) + \sum_j \Omega(\mathbf{v}_j) + \Omega(\boldsymbol{\mu})$
- Can be mapped to regular learning problem using mapping
  - $f_{\boldsymbol{\theta},j}(\mathbf{x}) = \underbrace{\begin{pmatrix} \mathbf{v}_1 \\ \dots \\ \mathbf{v}_m \\ \boldsymbol{\mu} \end{pmatrix}}_{\boldsymbol{\theta}}^{\mathrm{T}} \underbrace{\begin{pmatrix} \phi(\mathbf{x})\delta(j=1) \\ \dots \\ \phi(\mathbf{x})\delta(j=m) \\ \phi(\mathbf{x}) \end{pmatrix}}_{\boldsymbol{\Phi}_j(\mathbf{x})}$



Importance Sampling for Covariate Shift

- Goal: minimize risk on test distribution
  - $R_{\text{test}} = \int\int \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) P(y_i \mid \mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{test})\, d\mathbf{x}_i dy_i$

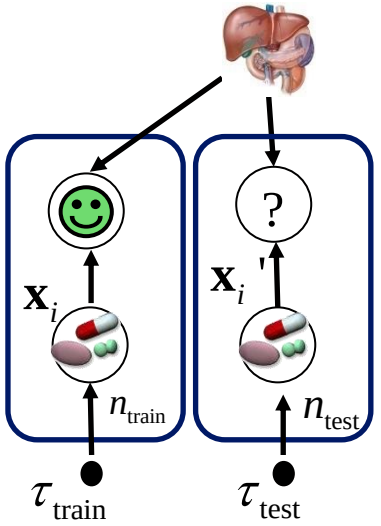
Integral over test distribution

- Idea: Write as expected value over training distribution by using appropriate weights.
  - $R_{\text{test}} = \int\int \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) P(y_i \mid \mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}})\, d\mathbf{x}_i dy_i$

Integral over training distribution

- Regularized empirical risk:
  - $\hat{R}_{\text{test}} = \sum_{i=1}^{n_{\text{train}}} \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})$

Sum over labeled training sample

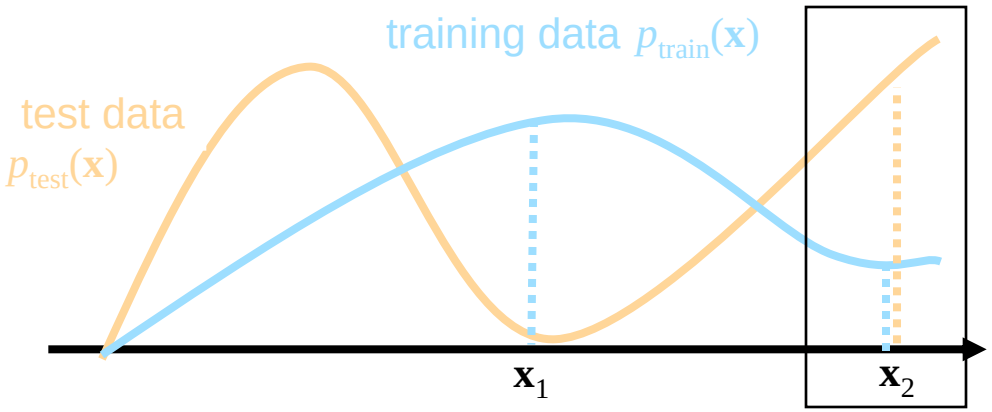


Importance Sampling for Covariate Shift

- Regularized empirical risk:
  - $$\hat{R}_{\text{test}} = \sum_{i=1}^{n_{\text{train}}} \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})$$
- Densities $p(\mathbf{x}_i \mid \tau_{\text{test}})$ and $p(\mathbf{x}_i \mid \tau_{\text{train}})$ unknown and high-dimensional (impractical to estimate).
- Density ratio is just a number for each training data point (should be easier to estimate).
- Direct estimation of optimal reweighting factors.

Direct Estimation of Importance Weights

- Regularized empirical risk:
  - $\hat{R}_{\text{test}} = \sum_{i=1}^{n_{\text{train}}} \frac{p(\mathbf{x}_i \mid p_{\text{test}})}{p(\mathbf{x}_i \mid p_{\text{train}})} \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})$



$$p_{\text{test}}(\mathbf{x}_2) = .6$$

$$p_{\text{train}}(\mathbf{x}) = .2$$

Direct Estimation of Importance Weights

- Density ratio can be rephrased:
  - $$\frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})} \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} \frac{p(\tau_{\text{test}})}{p(\tau_{\text{train}})}$$

$$= \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})} \left(1 + \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} \frac{p(\tau_{\text{test}})}{p(\tau_{\text{train}})} - 1 \right)$$

$$= \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})} \left(\frac{p(\tau_{\text{train}}) p(\mathbf{x} \mid \tau_{\text{train}}) + p(\tau_{\text{test}}) p(\mathbf{x} \mid \tau_{\text{test}})}{p(\tau_{\text{train}}) p(\mathbf{x} \mid \tau_{\text{train}})} - 1 \right)$$

$$= \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})} \left(\frac{p(\tau_{\text{train}} \mid \mathbf{x}) + p(\tau_{\text{test}} \mid \mathbf{x})}{p(\tau_{\text{train}} \mid \mathbf{x}_i)} - 1 \right)$$

$$= \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})} \left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i)} - 1 \right)$$

[Bickel, Brückner, Scheffer, Discriminative learning for differing training and test distributions, ICML 2007]

Direct Estimation of Importance Weights

- Density ratio can be rephrased:
  - $$\frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})}\left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i)} - 1\right)$$

Ratio of training-to-test set size

Given x, how likely is x to have come from training set?

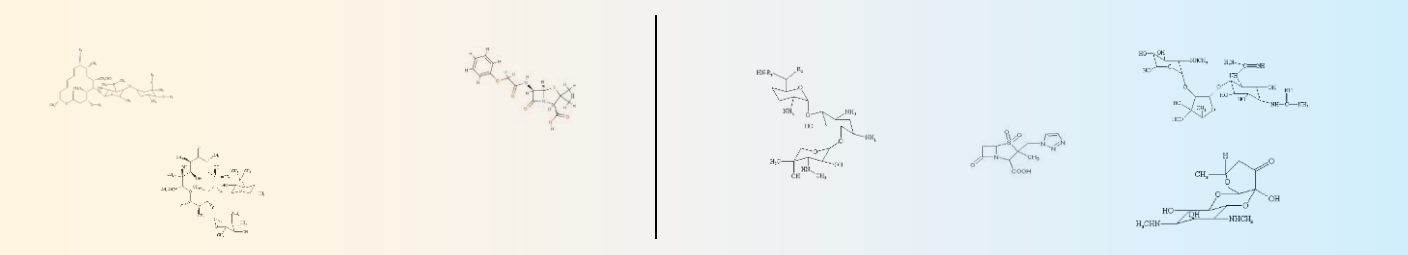
Tobias Scheffer

Direct Estimation of Importance Weights

- Density ratio can be estimated directly:
  - $$\frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})}\left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i)} - 1\right)$$
- Train logistic regression model that discriminates training from test data. $$p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta) = \frac{1}{1 + e^{\phi(\mathbf{x}_i)^{\mathrm{T}} \vartheta + b}}$$

Unlabeled test data

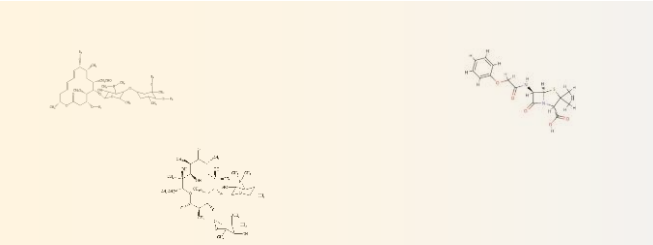
Training data (labels ignored)



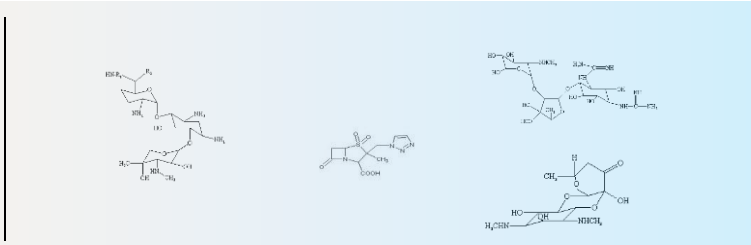
Direct Estimation of Importance Weights

1. Train model $p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta) = \frac{1}{1+e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\vartheta}}$ using training data as positive class and unlabeled test data as negative class.

Unlabeled test data



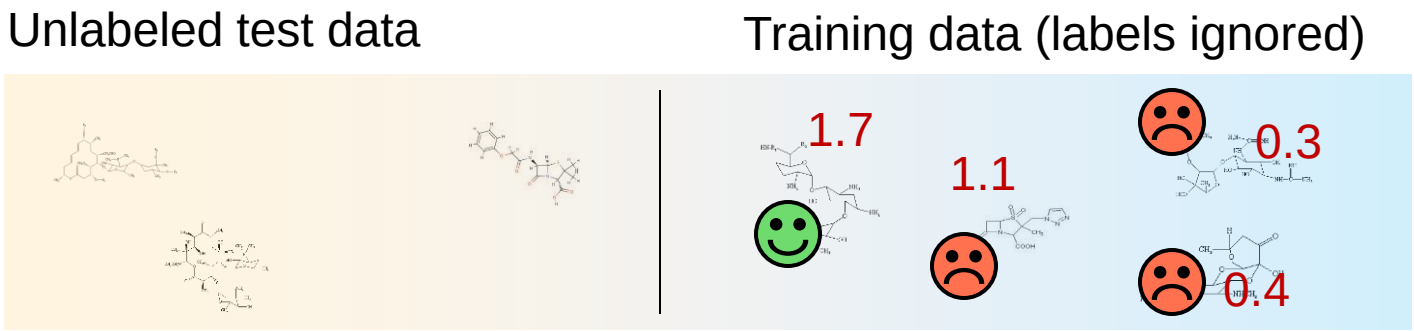
Training data (labels ignored)



Direct Estimation of Importance Weights

1. Train model $p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta) = \frac{1}{1+e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\vartheta}}$ using training data as positive class and unlabeled test data as negative class.

2. Infer Weights $\frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})}\left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta)} - 1\right)$

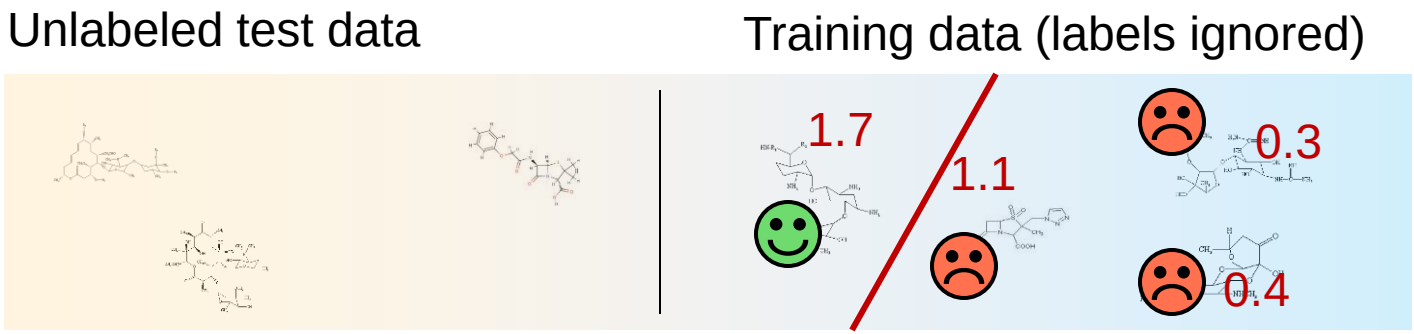


Direct Estimation of Importance Weights

1. Train model $p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta) = \frac{1}{1+e^{\phi(\mathbf{x}_i)^{\mathrm{T}}\vartheta}}$ using training data as positive class and unlabeled test data as negative class.

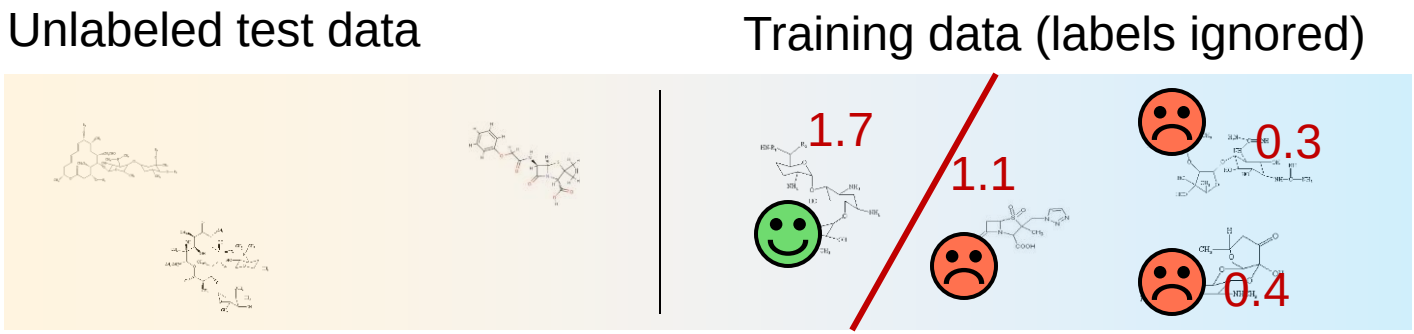
2. Infer Weights $\frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})}\left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta)} - 1\right)$

3. Train final classifier on weighted training data.



Direct Estimation of Importance Weights

- The logistic regression of the first step also has a regularization parameter.
- Set parameter using grid search and n-fold cross validation.
- (Label information is whether instance is training or test instance.)
- Hence, possible to tune regularization parameter.



[Bickel, Brückner, Scheffer, Discriminative learning for differing training and test distributions, ICML 2007]

Direct Estimation of Importance Weights

- Toxicity study conducted in country X has yielded labeled data.
- In country Y, physicians prefer different drugs, leading to a different distribution over drug prescriptions.
- How would you learn a toxicity model that works well for country Y?
- Does importance sampling make a difference at all? How does the decision whether or not a combination of substances is toxic depend on the distribution of substances administered?

Direct Estimation of Importance Weights

- Goal: Model that minimizes risk on $p(\mathbf{x}_i \mid \tau_{\text{test}})$
- Training data governed by $p(\mathbf{x}_i \mid \tau_{\text{train}})$
- Optimal resampling weights:
  - $$\frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})}\left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i)} - 1\right)$$
- Train logistic regression model for $p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta)$ using training data as positive data and unlabeled test data as negative class.
- Calculate weights (above formula) and train classifier on weighted training data.

Scheffer

KLIEP

- Alternative way of estimating density ratio.
- Define weights $w(\mathbf{x}_i)$ such that $w(\mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}}) = p(\mathbf{x}_i \mid \tau_{\text{test}})$
- Minimize KL-divergence between distributions of test data and weighted training data.
  - $KL[\, p(\mathbf{x}_i \mid \tau_{\text{test}}) \,\|\, w(\mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}})]$

$$= \int p(\mathbf{x}_i \mid \tau_{\text{test}}) \log \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{w(\mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}})} d\mathbf{x}_i$$

$$= \int p(\mathbf{x}_i \mid \tau_{\text{test}}) \log \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} d\mathbf{x}_i - \int p(\mathbf{x}_i \mid \tau_{\text{test}}) \log w(\mathbf{x}_i) d\mathbf{x}_i$$

$$= -\int p(\mathbf{x}_i \mid \tau_{\text{test}}) \log w(\mathbf{x}_i) d\mathbf{x}_i + const$$

$$\approx -\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \log w(\mathbf{x}_i) = -\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \log \phi(\mathbf{x}_i)^{\mathrm{T}} \vartheta$$

KLIEP

- Minimize KL-divergence between distributions of test data and weighted training data.
  - $KL[\, p(\mathbf{x}_i \mid \tau_{\text{test}}) \,||\, w(\mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}})] \approx -\frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \log \phi(\mathbf{x}_i)^{\mathrm{T}}\, \vartheta$
- Weighted training data $w(\mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}}) = p(\mathbf{x}_i \mid \tau_{\text{test}})$ have to be nortmalized, weights have to be positive.
- Optimization problem:
  - $KL[\, p(\mathbf{x}_i \mid \tau_{\text{test}}) \,||\, w(\mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{\text{train}})] \approx$

    $\vartheta^* = \arg\max_{\vartheta} \sum_{i=1}^{n_{\text{test}}} \log \phi(\mathbf{x}_i)^{\mathrm{T}}\, \vartheta$

    $\text{subject to} \sum_{i=1}^{n_{\text{train}}} \log \phi(\mathbf{x}_i)^{\mathrm{T}}\, \vartheta = n_{\text{train}}$

    $\text{for all } j : \vartheta_j \geq 0$

[Sugiyama, Suzuki, Nakajima, Kashima, H., von Bünau, Kawanabe. Direct importance estimation for covariate shift adaptation. Annals of the Institute of Statistical Mathematics, vol.60, no.4, 2008.]

Kernel Mean Matching

- Alternative way of estimating density ratio.
- Define weights $w(\mathbf{x}_i)$ such that $w(\mathbf{x}_i)\,p(\mathbf{x}_i \mid \tau_{\text{train}}) = p(\mathbf{x}_i \mid \tau_{\text{test}})$
- Expected mean feature mapping:
  - $\mu_\tau(\phi(\mathbf{x})) = \int \phi(\mathbf{x}) p(\mathbf{x} \mid \tau) d\mathbf{x}$
- When $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}} \phi(\mathbf{x}')$ is a universal kernel, then there is a one-to-one relationship between $\mu_\tau(\phi(\mathbf{x}))$ and $p(\mathbf{x} \mid \tau)$
- Idea: set weights such that
  - $\mu_{\tau_{\text{test}}}(\phi(\mathbf{x})) = \mu_{\tau_{\text{train}}}(w(\mathbf{x})\phi(\mathbf{x}))$

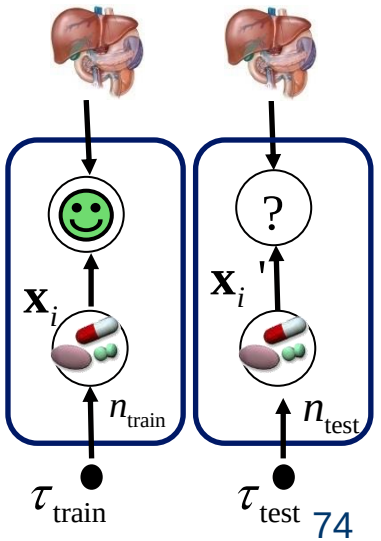
Kernel Mean Matching

- Idea: set weights such that
  - $\mu_{\tau_{\text{test}}}(\phi(\mathbf{x})) = \mu_{\tau_{\text{train}}}(w(\mathbf{x})\phi(\mathbf{x}))$
- Weights have to be positive and normalized.
- Optimization problem:
  - $w^* = \arg\min_w \| \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} w(\mathbf{x}_i)\phi(\mathbf{x}_i) - \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \phi(\mathbf{x}_i) \|$
- Optimization problem is convex.
- Regularization: impose upper bound on individual weights and sum over weights.
- Problem: need labeled test data to tune regularizer.

[Huang, Smola, Gretton, Borgwardt, Schölkopf, Correcting sample selection bias by unlabeled data. NIPS 2007.]

Importance Sampling for Domain Adaptation

- Minimize risk on test distribution:
  - $R_{\text{test}} = \int\int \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) P(y_i \mid \mathbf{x}_i)\, p(\mathbf{x}_i \mid \tau_{test})\, d\mathbf{x}_i dy_i$
- Rephrase as expected weighted training loss
  - $R_{\text{test}}(\boldsymbol{\theta}) = \int\int \frac{p(\mathbf{x}_i, y_i \mid \tau_{\text{test}}, \boldsymbol{\theta}_{\text{train}})}{p(\mathbf{x}_i, y_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})} \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i))\, p(\mathbf{x}_i, y_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})\, d\mathbf{x}_i dy_i$
- Estimate on training sample:
  - $\hat{R}_{\text{test}}(\boldsymbol{\theta}) = \sum_{i=1}^{n_{\text{train}}} \underbrace{\frac{p(\mathbf{x}_i, y_i \mid \tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})}{p(\mathbf{x}_i, y_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})}}_{w(\mathbf{x}_i, y_i)} \ell(y_i, f_{\boldsymbol{\theta}}(\mathbf{x}_i)) + \Omega(\boldsymbol{\theta})$



Importance Sampling for Domain Adaptation

- Density ratio can be rephrased:
  - $$\frac{p(\mathbf{x}_i \mid \tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})} = \frac{p(\tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})}{p(\tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})} \left(\frac{1}{p(\tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}} \mid \mathbf{x}_i, y_i)} - 1 \right)$$

Ratio of training-to-test set size

Given x,y, how likely is instance to have come from training set?

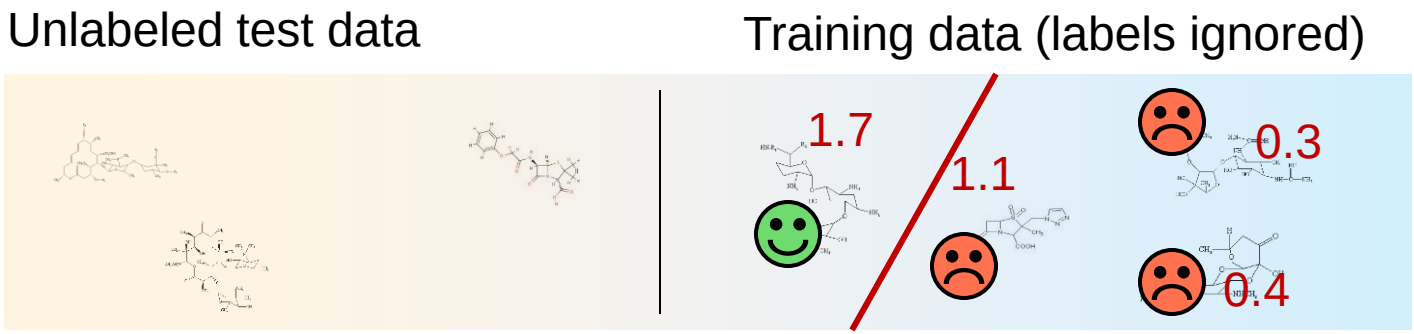
[Bickel, Sawade, Scheffer. Transfer Learning by Distribution Matching for Targeted Advertising. NIPS 2008.]

Direct Estimation of Importance Weights

1. Train model $p(\tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}} \mid \mathbf{x}_i, y_i, \vartheta) = \dfrac{1}{1+e^{\phi(\mathbf{x}_i, y_i)^{\mathrm{T}} \vartheta}}$ using training data as positive class and labeled test data as negative class.

2. Infer Weights $\dfrac{p(\mathbf{x}_i \mid \tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})} = \dfrac{p(\tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})}{p(\tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})} \left(\dfrac{1}{p(\tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}} \mid \mathbf{x}_i, y_i, \vartheta)} - 1 \right)$

3. Train final classifier on weighted training data.



Estimation of Importance Weights

- Importance weights match weighted training data to distribution of test data.
- Learn logistic regression model $p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta)$ and let
  - $$w(\mathbf{x}_i) = \frac{p(\mathbf{x}_i \mid \tau_{\text{test}})}{p(\mathbf{x}_i \mid \tau_{\text{train}})} = \frac{p(\tau_{\text{train}})}{p(\tau_{\text{test}})} \left(\frac{1}{p(\tau_{\text{train}} \mid \mathbf{x}_i, \vartheta)} - 1 \right)$$
- Alternatively, minimize KL divergence between weighted training and test data
- Alternatively, minimize distance between mean of weighted training and test data
- Applies to domain adaptation too (some labeled data from target distribution required).

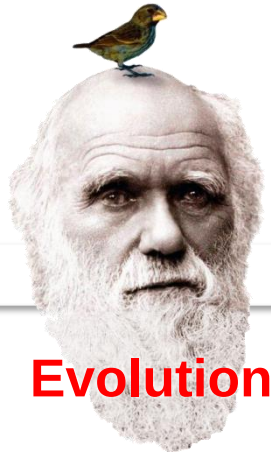
University of Potsdam
Dept. of Computer Science

Adversarial Learning

Tobias Scheffer

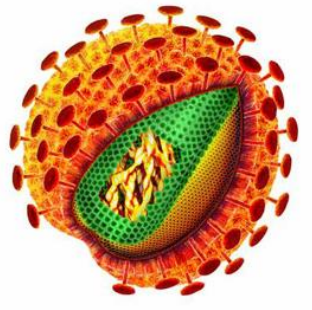
Adersarial Learning

Current virus



Evolution

Mutated virus

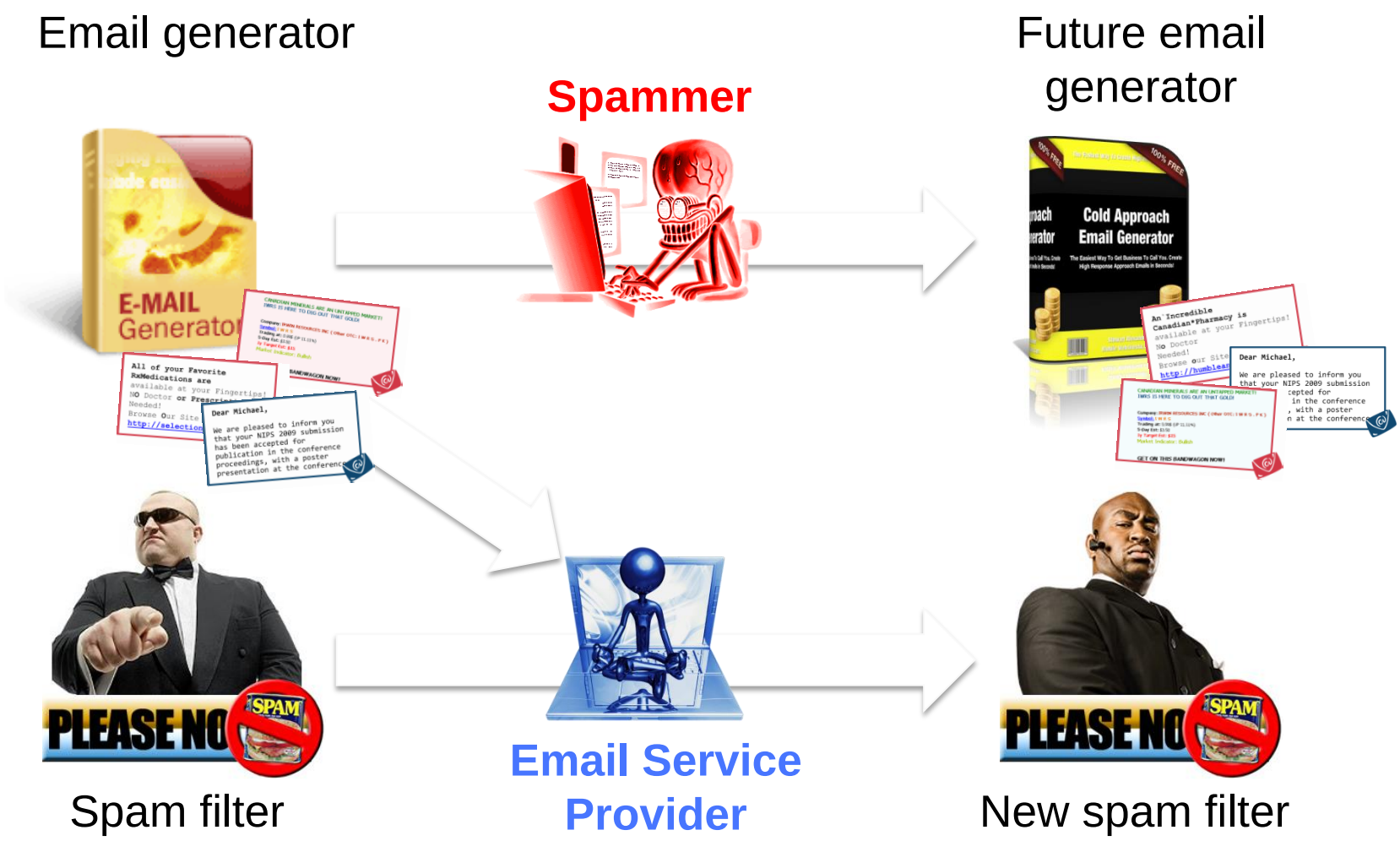


Current therapy

Physician

New therapy

Adersarial Learning



Overview

- Learning with Invariances
  - Make result of learning process invariant to specific types of transformations $\Phi(\mathbf{x}, y)$
- Minimax probability machine
- Game Theory Basics
- Game-Theoretic Learning Models

Robust Learning Models

- Standard learning assumes training data drawn iid from distribution at application time.
- Optimistic, if adversary tampers with distribution.
- Robust learners minimize *maximum risk for any distribution* over a certain class of distributions.
- Underlying assumption: adversary can choose distribution within certain class, and will try to inflict greatest possible damage for learner.

Learning with Invariances

- Training set is drawn according to $p(\mathbf{x}_i, y_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})$
- Adversary gets to exercise transformation on distribution: $\tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}} = \Phi(\tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})$
- Test set is drawn according to $p(\mathbf{x}_i, y_i \mid \tau_{\text{test}}, \boldsymbol{\theta}_{\text{test}})$
- Equivalent to drawing according to $p(\mathbf{x}_i, y_i \mid \tau_{\text{train}}, \boldsymbol{\theta}_{\text{train}})$ and transforming instances: $\mathbf{x}', y' = \Phi(\mathbf{x}, y)$
- Goal: Minimize regularized empirical risk:
  - $\hat{R} = \sum_{i=1}^{n_{\text{test}}} \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), y_i) + \Omega(\boldsymbol{\theta})$

Heuristic SVM Learning with Invariances

- Dual SVM decision function
  - $f_{\boldsymbol{\beta}}(\mathbf{x}) = \sum_{i=1}^{n_{\text{train}}} \beta_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$
- Is determined by support vectors
  - $S = \{\mathbf{x}_i : \beta_i > 0\}$

- Heuristic invariant learning algorithm:
  - Learn SVM on training data L
  - If set of transformations is finite: add all possible $\{\mathbf{x}', y' = \Phi(\mathbf{x}, y) : \mathbf{x}, y \in L \cap S, \Phi \in \boldsymbol{\Phi}\}$ to training set
  - Otherwise: draw some $\Phi \in \boldsymbol{\Phi}$ and $(\mathbf{x}, y) \in L \cap S$ and add $\mathbf{x}', y' = \Phi(\mathbf{x}, y)$ to L.
  - Learn from enhanced sample.

Reminder: SVM-Struct

- Hinge-loss for binary SVM:
  - $\ell(y, f_{\boldsymbol{\theta}}(\mathbf{x})) = \max(0, 1 - y f_{\boldsymbol{\theta}}(\mathbf{x}))$
- Loss for multi-class SVM and SVM-Struct
  - $\ell(y, f_{\boldsymbol{\theta}}(\mathbf{x}, \cdot)) = \max_{y' \neq y}(0, 1 - f_{\boldsymbol{\theta}}(\mathbf{x}, y) + f_{\boldsymbol{\theta}}(\mathbf{x}, y'))$
- Corresponding constraint optimization problem
  - $\min_{\boldsymbol{\theta}, \xi} \lambda \sum_{i=1}^{n} \xi_i + \frac{1}{2} \sum_{i=1}^{k} \boldsymbol{\theta}^{i\mathrm{T}} \boldsymbol{\theta}^i$

    subject to the constraints:

    $\forall y \neq y_i : f_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i) \geq f_{\boldsymbol{\theta}}(\mathbf{x}_i, y) + 1 - \xi_i$

    and $\xi_i \geq 0.$

SVM Learning with Invariances

- Define loss
  - $\ell(y, \mathbf{x}, f_{\boldsymbol{\theta}}) = \max_{\Phi \in \boldsymbol{\Phi}, y' \neq y} (0, 1 - f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}), y) + f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}), y'))$

Max over all transformations and incorrect labels y‘

Margin

[Teo, Globerson, Roweis, Smola, Convex learning with invariances, NIPS 2007]

SVM Learning with Invariances

- Define loss
  - $\ell(y, \mathbf{x}, f_{\boldsymbol{\theta}}) = \max_{\Phi \in \boldsymbol{\Phi}, y' \neq y} (0, 1 - f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}), y) + f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}), y'))$
- Loss is a convex upper bound on zero-one loss.
- Constrained optimization problem
  - $\min_{\boldsymbol{\theta}, \xi} \lambda \sum_{i=1}^{n} \xi_i + \frac{1}{2} \sum_{i=1}^{k} \boldsymbol{\theta}^{i\mathrm{T}} \boldsymbol{\theta}^i$

    subject to the constraints:

    $\forall y' \neq y_i \forall \Phi \in \boldsymbol{\Phi} : f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}_i), y_i) \geq f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}_i), y') + 1 - \xi_i$

    and $\xi_i \geq 0$.

SVM Learning with Invariances

- Define loss
  - $\ell(y, \mathbf{x}, f_{\boldsymbol{\theta}}) = \max_{\Phi \in \boldsymbol{\Phi}, y' \neq y} (0, 1 - f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}), y) + f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}), y'))$
- Constrained optimization problem
  - $\min_{\boldsymbol{\theta}, \xi} \lambda \sum_{i=1}^{n} \xi_i + \frac{1}{2} \sum_{i=1}^{k} \boldsymbol{\theta}^{i\mathrm{T}} \boldsymbol{\theta}^i$ subject to the constraints:

    $\forall i \forall y' \neq y_i \forall \Phi \in \boldsymbol{\Phi} : f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}_i), y_i) \geq f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}_i), y') + 1 - \xi_i$

    and $\xi_i \geq 0$.
- SVM-Struct working set algorithm:
  - Iterate over examples, find the one y‘ and transformation that violates margin the most:

    $y'^*, \Phi^* = \arg\max_{y', \Phi} 1 - f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}_i), y_i) + f_{\boldsymbol{\theta}}(\Phi(\mathbf{x}_i), y')$
  - Add to working set, solve OP, reiterate.

Minimax Probability Machine

- Let $\mathbf{P}_{\mu,\Sigma}$ be the set of *all* distributions with mean μ and covariance Σ.
- Distribution of training data:
  - Positive class has mean μ_{+1}, covariance matrix Σ_{+1}.
  - Negative class has mean μ_{-1}, covariance matrix Σ_{-1}.
- At application time, adversary can choose *any* input distribution $p(\mathbf{x},+1) \in \mathbf{P}_{\mu_{+1},\Sigma_{+1}}$, $p(\mathbf{x},-1) \in \mathbf{P}_{\mu_{-1},\Sigma_{-1}}$

- What is the right optimization criterion for learner?
  - Based on zero-one loss

Tobias Scheffer

Minimax Probability Machine

- Let $\mathbf{P}_{\mu,\Sigma}$ be the set of *all* distributions with mean μ and covariance Σ.
- Distribution of training data:
  - Positive class has mean μ_{+1}, covariance matrix Σ_{+1}.
  - Negative class has mean μ_{-1}, covariance matrix Σ_{-1}.
- At application time, adversary can choose *any* input distribution $p(\mathbf{x},+1) \in \mathbf{P}_{\mu_{+1},\Sigma_{+1}}$, $p(\mathbf{x},-1) \in \mathbf{P}_{\mu_{-1},\Sigma_{-1}}$

- Optimization criterion for learner:
  - $\max_{p(\mathbf{x},+1)\in\mathbf{P}_{\mu_{+1},\Sigma_{+1}},\, p(\mathbf{x},-1)\in\mathbf{P}_{\mu_{-1},\Sigma_{-1}}} \sum_y \int \ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}), y)\, p(\mathbf{x}, y) d\mathbf{x}$

[Lanckriet et al., Minimax Probability Machine, NIPS 2002]

Minimax Probability Machine

- Let $\mathbf{P}_{\mu,\Sigma}$ be the set of *all* distributions with mean μ and covariance Σ.
- Distribution of training data:
  - Positive class has mean μ_{+1}, covariance matrix Σ_{+1}.
  - Negative class has mean μ_{-1}, covariance matrix Σ_{-1}.
- At application time, adversary can choose *any* input distribution $p(\mathbf{x},+1) \in \mathbf{P}_{\mu_{+1},\Sigma_{+1}}$, $p(\mathbf{x},-1) \in \mathbf{P}_{\mu_{-1},\Sigma_{-1}}$
- Optimization criterion for learner:
  - $\max_{p(\mathbf{x},+1)\in\mathbf{P}_{\mu_{+1},\Sigma_{+1}},\, p(\mathbf{x},-1)\in\mathbf{P}_{\mu_{-1},\Sigma_{-1}}} \sum_y \int \ell_{0/1}(f_{\boldsymbol{\theta}}(\mathbf{x}), y)\, p(\mathbf{x}, y) d\mathbf{x}$
- Can be rewritten as convex optimization problem.

[Lanckriet et al., Minimax Probability Machine, NIPS 2002]

Invariant SVMs

- Family of methods for making SVM more robust against transformations of instances by adversary.
- Simplistic approach: add transformed support vectors to training set.
- More sophisticated: minimize, for all examples, maximal loss over transformation space.
- Minimax probability machine: minimize maximal loss over all input distributions with observed mean value and variance.

Game Theory Basics

- Game defined in terms of
  - Players $P_1, \ldots, P_n$
  - Action spaces $A_1, \ldots, A_n$
  - Cost functions $C_1, \ldots, C_n$ with $C_i : A_1 \times \cdots \times A_n \rightarrow \mathbb{R}$
  - Cost functions are interleaved optimization problems.
- Zero-sum game: two players, $C_1(A_1, A_2) + C_2(A_1, A_2) = 0$
  - One player‘s loss is other player‘s gain.
- Non-cooperative game:
  - Players cannot exchange messages

Game Theory Basics

- Static game:
  - Players act simultaneously, not knowing their opponents‘ move.
  - Learner and adversary act simultaneously.
- Dynamic, extensive-form game:
  - Some ordering defined over moves, information about previous actions.
  - Stackelberg competition: learner committs first to model, then adversary transforms distribution.

Game Theory Basics

- Complete information:
  - Each player know other players‘ cost functions.
- Incomplete information:
  - At least one player is uncertain about opponents‘ cost functions.
  - Bayesian players infer expected cost functions over their beliefs.

Game Theory: Zero Sum Games

- Security level for player P_1
  - $\bar{C}_1 = \min_{a_1 \in A_1} \max_{a_2 \in A_2} C_1(a_1, a_2)$
- Security level for player P_2
  - $\bar{C}_2 = \min_{a_2 \in A_2} \max_{a_1 \in A_1} C_2(a_1, a_2) = -\max_{a_2 \in A_2} \min_{a_1 \in A_1} C_1(a_1, a_2)$
- Example
  - $\bar{C}_1 = 2$, row 3
  - $\bar{C}_2 = 1$, column 1

		P2		
P1		(4, -4)	(-1, 1)	(-2, 2)
		(-1, 1)	(-2, 2)	(3, -3)
		(1, -1)	(2, -2)	(1, -1)

Game Theory: Zero Sum Games

- Minimax strategy for player P_1
  - $\bar{a}_1 = \underset{a_1 \in A_1}{\operatorname{argmin}} \max_{a_2 \in A_2} C_1(a_1, a_2)$
- Minimax strategy for player P_2
  - $\bar{a}_2 = \underset{a_2 \in A_2}{\operatorname{argmin}} \max_{a_1 \in A_1} C_2(a_1, a_2)$
- Example
  - $\bar{a}_1 = 3, \bar{a}_2 = 1$
  - $C_1(\bar{a}_1, \bar{a}_2) = 1$
- It holds that
  - $\bar{C}_1 \geq C_1(\bar{a}_1, \bar{a}_2) \geq -\bar{C}_2$
  - No guaranteed stability

		P2		
P1		(4, -4)	(-1, 1)	(-2, 2)
		(-1, 1)	(-2, 2)	(3, -3)
		(1, -1)	(2, -2)	(1, -1)

Game Theory: Non-Zero Sum Games

- Generalization of zero-sum games
  - $C_1 = -C_2$ no longer prerequisite
- Minimax strategy
  - Still guarantees least maximal costs
  - Not well motivated if opponent wants to minimize own costs and costs not directly antagonistic.

Game Theory: Nash Equilibrium

- Equilibrium (stable points)
  - (a_1^*, a_2^*) is Nash equilibrium if
    - $a_1^* = \underset{a_1 \in A_1}{\operatorname{argmin}} C_1(a_1, a_2^*)$
    - $a_2^* = \underset{a_2 \in A_2}{\operatorname{argmin}} C_2(a_1^*, a_2)$

No equilibrium

		P2	
P1	(4, -4)	(-1, 1)	(-2, 2)
	(-1, 1)	(-2, 2)	(3, -3)
	(1, -1)	(2, -2)	(1, -1)

Equilibrium

		P2	
P1	(4, -4)	(3, -3)	(-2, 2)
	(-1, 1)	(3, -3)	(3, -3)
	(1, -1)	(2, -2)	(1, -1)

Tobias Scheffer

Game Theory: Nash Equilibrium

- Equilibrium (stable points)
  - (a_1^*, a_2^*) is Nash equilibrium if
    - $a_1^* = \underset{a_1 \in A_1}{\operatorname{argmin}} C_1(a_1, a_2^*)$
    - $a_2^* = \underset{a_2 \in A_2}{\operatorname{argmin}} C_2(a_1^*, a_2)$
- In a Nash equilibrium, each player reacts optimally to their opponent‘s move.

Tobias Scheffer

Game Theory: Nash Equilibrium

- Equilibrium (stable points)
  - (a_1^*, a_2^*) is Nash equilibrium if
    - $a_1^* = \underset{a_1 \in A_1}{\operatorname{argmin}} C_1(a_1, a_2^*)$
    - $a_2^* = \underset{a_2 \in A_2}{\operatorname{argmin}} C_2(a_1^*, a_2)$
- In a Nash equilibrium, each player reacts optimally to their opponent‘s move.
- Simple algorithm: imagine any move; infer opponent‘s optimal reaction; infer own optimal reaction; reiterate until fixed point is reached.
- Fixed point is Nach equilibrium.

Tobias Scheffer

Game Theory: Nash Equilibrium

- When does it make sense to acto according to Nash equilibrium?
- Minimax is overly pessimistic:
  - Adversaries do not want to maximize your costs but want to minimize their own costs.
- When adversary will act according to Nash equilibrium, using this equilibrium too is optimal.
  - Only makes sense to assume if equilibrium exists.
- When players act according to different equilibria, then outcome can be arbitrarily bad for all players.
  - Acting according to Nash equilibrium makes most sense if equilibrium is unique.

Game Theory: Extensive-Form Games

- So far static games: players act simultaneously
  - No information about adversary‘s action exploitable.
- Extensive form games: players act in defined order.
  - Game takes the for of a tree.
- Simplest form: Stackelberg competition
  - First player acts first
  - Then, adversary acts (knowing the first player‘s move).
  - Then, costs are inferred.

Game Theory: Extensive-Form Games

- Simplest form: Stackelberg competition
  - First player acts first
  - Then, adversary acts (knowing the first player‘s move).
- Adversary solves simple minimization problem:
  - $a_2^* = \underset{a_2 \in \mathrm{A}_2}{\operatorname{argmin}} \quad V_2(a_1^*, a_2)\,.$
- First player has to account for adversary:
  - $a_1^* = \underset{a_1 \in \mathrm{A}_1}{\operatorname{argmin}} \max_{a_2 \in \mathrm{A}_2} V_1(a_1, a_2)$

Game Theory Basics

- Static (simultaneous actions) vs. extensive-form.
- Non-cooperative (no message passing)
- Zero-sum (antagonistic costs) vs non-zero-sum
- Zero-sum games: minimax strategies minimizes worst-case (over adversarial action space) costs.
- Nash equilibrium: deviating unilaterally increases costs for either player.
- Non-zero-sum games:
  - Nash equilibrium optimal if adversary will act according to the same equilibrium.
  - Does the game have an equilibrium? Is it unique? Will the adversary be rational enough to infer it?

Game-Theoretic Learning Models

Training distribution

Adversary ($v = +1$)

Test distribution

Costs for adversary

Interleaved optimization problems

Learner ($v = -1$)

Model parameters

Costs for learner

Game-Theoretic Learning Models

- Action of the data generator (adversary)
  - Theoretically: transform input distribution $p(\mathbf{x}, y) \rightarrow \bar{p}(\mathbf{x}, y)$
  - Empirically: transform data $D = \{(\mathbf{x}_i, y_i)\} \rightarrow D' = \{(\mathbf{x}'_i, y'_i)\}$

- Action of the learner:
  - Choose model parameters $f_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^{\mathrm{T}} \phi(\mathbf{x})$

Game-Theoretic Learning Models

- Empirical costs of the data generator (adversary)
  - $C_{+1}(\boldsymbol{\theta}, D') = \frac{1}{n}\sum_{i=1}^{n} \ell_{+1}(\boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x}_i'), y_i) + \rho_{+1}\Omega_{+1}(D, D')$

Adversary's loss function

Transformation costs

- Empirical costs of the learner
  - $C_{-1}(\boldsymbol{\theta}, D') = \frac{1}{n}\sum_{i=1}^{n} \ell_{-1}(\boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x}_i'), y_i) + \rho_{-1}\Omega_{-1}(\boldsymbol{\theta})$

Learner's loss function

Regularizer

Game-Theoretic Learning Models

- Empirical costs of the data generator (adversary)
  - $C_{+1}(\boldsymbol{\theta}, D') = \frac{1}{n}\sum_{i=1}^{n} \ell_{+1}(\boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x}_i'), y_i) + \rho_{+1}\Omega_{+1}(D, D')$

- Empirical costs of the learner
  - $C_{-1}(\boldsymbol{\theta}, D') = \frac{1}{n}\sum_{i=1}^{n} \ell_{-1}(\boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x}_i'), y_i) + \rho_{-1}\Omega_{-1}(\boldsymbol{\theta})$

Interleaved cost functions: each player‘s costs depend also on other player‘s action. No such thing as cost-minimizing parameters $\boldsymbol{\theta}$.

Game-Theoretic Learning Models

- Empirical costs of the data generator (adversary)
  - $C_{+1}(\boldsymbol{\theta}, D') = \frac{1}{n}\sum_{i=1}^{n} \ell_{+1}(\boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x}'_i), y_i) + \rho_{+1}\Omega_{+1}(D, D')$
- Empirical costs of the learner
  - $C_{-1}(\boldsymbol{\theta}, D') = \frac{1}{n}\sum_{i=1}^{n} \ell_{-1}(\boldsymbol{\theta}^{\mathrm{T}}\phi(\mathbf{x}'_i), y_i) + \rho_{-1}\Omega_{-1}(\boldsymbol{\theta})$
- Empirical costs cannot be minimized directly because opponent‘s action is not known.

Game-Theoretic Learning Models

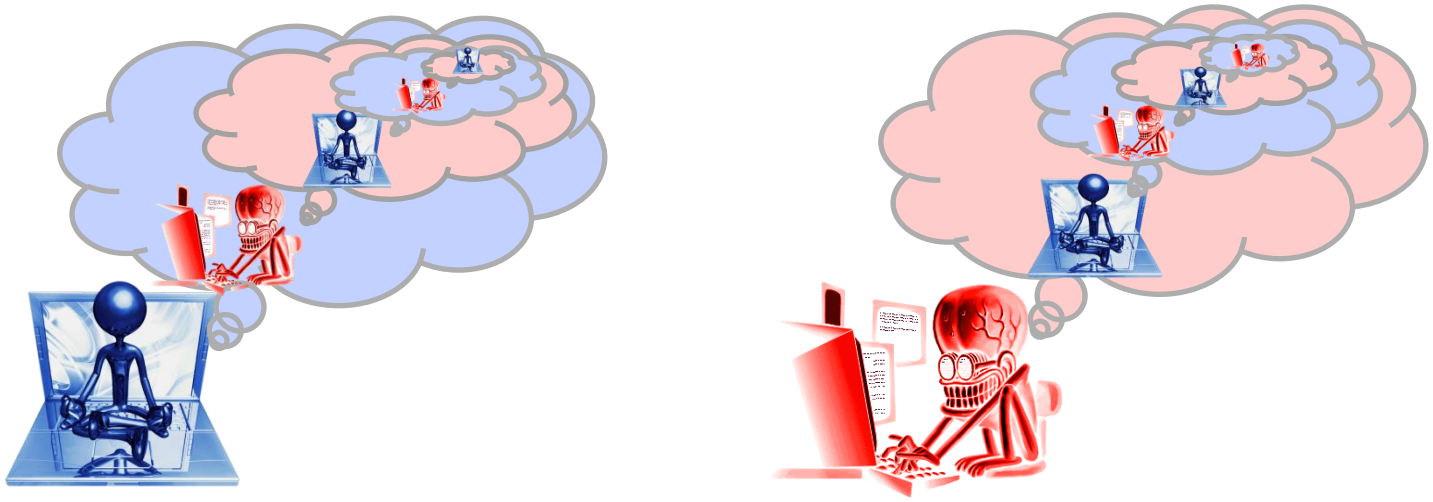
- Worst-case solution for learner:
  - $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \max_{D'} C(\boldsymbol{\theta}, D')$
  - = learning with invariances
  - Minimizes $C_{-1}(\boldsymbol{\theta}, D')$ if $D=D'$.
  - = Adversary will do nothing; iid assumption

- Unrealistically optimistic.

Game-Theoretic Learning Models

- Naive solution for learner:
  - $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta}, D) = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} \ell_{-1}(\boldsymbol{\theta}^{\mathrm{T}} \phi(\mathbf{x}_i), y_i) + \rho_{-1} \Omega_{-1}(\boldsymbol{\theta})$
  - = learn from training data
- Minimizes $C_{-1}(\boldsymbol{\theta}, D')$ if adversary will try to maximize learner's costs (as opposed to minimizing own costs).
- Too pesstimic.

Nash Equilibrial Prediction Models



- Learner‘s action is optimal reaction to adversary‘s action which is optimal reaction…
- Fixed point of *optimal reaction* is a Nash equilibrium.
- Probably not quite realistic for most applications, but perhaps more realistic than previous solutions.

Nash-Equilibrial Prediction Models

- Theorem: Nash equilibrium for prediction games exist and is unique if
  - Both loss functions have identical curvature, and
  - Regularization parameters are sufficiently large

[Brückner, Kanzow, Scheffer. Static Prediction Games for Adversarial Learning Problems. JMLR, 13:2617-2654, 2012]

Nash-Equilibrial Prediction Models

- Nach equilibrium meets:
  - $C_{-1}(\boldsymbol{\theta}^*, D^*) = \min_{\mathbf{w}} C_{-1}(\boldsymbol{\theta}, D^*)$
  - $C_{+1}(\boldsymbol{\theta}^*, D^*) = \min_{D'} C_{+1}(\boldsymbol{\theta}^*, D')$
- Nikaido-Isoda function: summed improvement that players can enjoy by changing their action
  - $\vartheta(\boldsymbol{\theta}, D', \dot{\boldsymbol{\theta}}, \dot{D}') = C_{-1}(\boldsymbol{\theta}, D') - C_{-1}(\dot{\boldsymbol{\theta}}, D') + C_{-1}(\dot{\boldsymbol{\theta}}, D') - C_{-1}(\dot{\boldsymbol{\theta}}, \dot{D}')$
  - Has to be zero at Nash equilibrium
- Theorem: descent direction for Nikaido-Isoda function exists and can be calculated.
  - Leads to Nash-SVM that finds Nash equilibrial prediction model.

Exercise

- Loss / cost functions of spammer and email service provider?
  -
  -
- Regularizers for learner and spammer?
  -
  -
- Possible transformations of spammer?
  -

Exercise

- Loss / cost functions of spammer and email service provider?
  - $\ell_{learner}(y, y') = \ell_{0/1}(y, y') \begin{cases} 1 \text{ if } yy' < 0 \\ 0 \text{ otherwise} \end{cases}$
  - $\ell_{spammer}(y, y') = \ell_{0/1}(y, -1)$
- Regularizers for learner and spammer?
  - $\Omega_{learner}(\boldsymbol{\theta}) = \boldsymbol{\theta}^{\mathrm{T}}\boldsymbol{\theta}$
  - $\Omega_{spammer}(D, \dot{D}) = \mid D - \dot{D} \mid_F^2 = \sum_i \sum_j (D_{ij} - \dot{D}_{ij})^2$
- Possible transformations of spammer?
  - Can transform any training matrix into any other matrix.

Nash-Equilibrial Prediction Models

- Learner and adversary act simultaneously.
- Both have distinct, conflicting but not necessarily antagonistic cost functions
- Unique Nash equilibrium exist if curvature of loss function matches and if regularizers are sufficiently large
- Equilibrium point is optimal strategy for learner if adversary also acts according to it.
- Nash-SVM finds equilibrium point
- Empirically, more robust for spam filtering than SVM and SVM with inveriances.

Stackelberg Prediction Games

- Two-stage game:
  - Learner fixes parameters of predictive model
  - Adversary then transforms input distribution.
- Learner chooses optimal parameters under the assumption that adversary will minimize his costs.
- Bilevel optimization problem:
  - $$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \max_{D^*} \; C_{-1}(\boldsymbol{\theta}, D^*)$$
    $$s.t. \qquad C_{+1}(\boldsymbol{\theta}, D^*) = \min_{D'} C_{+1}(\boldsymbol{\theta}, D')$$

Stackelberg Prediction Games

- Unique Stackelberg equilibrium always exists.
- Can be rephrased as constraint optimization problem (KKT conditions of lower-level OP).
- Locally optimal solution by SQP solver or interior-points method.

[Brückner, Scheffer. Stackelberg games for adversarial prediction problems. ACM KDD 2011.]

Game-Theoretic Learning Models

- Interests of learner and adversary modeled as loss functions.
- Transformation costs and prior on model parameteres modeled as regularizers.
- Interleaved optimization problems constitute a game between learner and adversary.
- Ragularizers and instance-specific costs make costs non-antagonistic.
  - Minimax solution too pessimistic
- Nash-SVM chooses predictive model according to Nash equilibrium.
- Stackelberg-SVM: learner acts first, then adversary.