Introduction to Machine Learning

Dale Schuurmans

University of Alberta

Learning phenomena

Reading Instruction	*gaining knowledge*
Observing Experiencing	*discovering patterns*
Experimenting Exploring	*understanding causality*
Practicing Adapting	*acquiring skills*
Growing Maturing	*development*
Evolving	*adaptation*

Machine learning

Automating learning phenomena

Systems that improve with experience

Central question

How to achieve useful improvement within reasonable amount of experience?

Answer

- Not by magic!
- Exist fundamental limits to learning
- Core trade-off
  - amount/quality of experience
  - prior knowledge/constraints

No such thing as "universal" learning

Human beings are

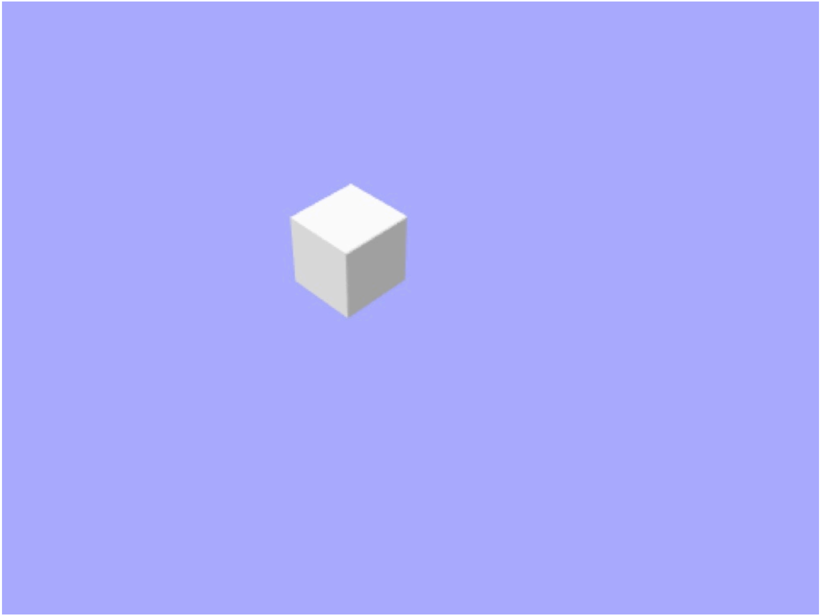
- heavily constrained
- extremely structured

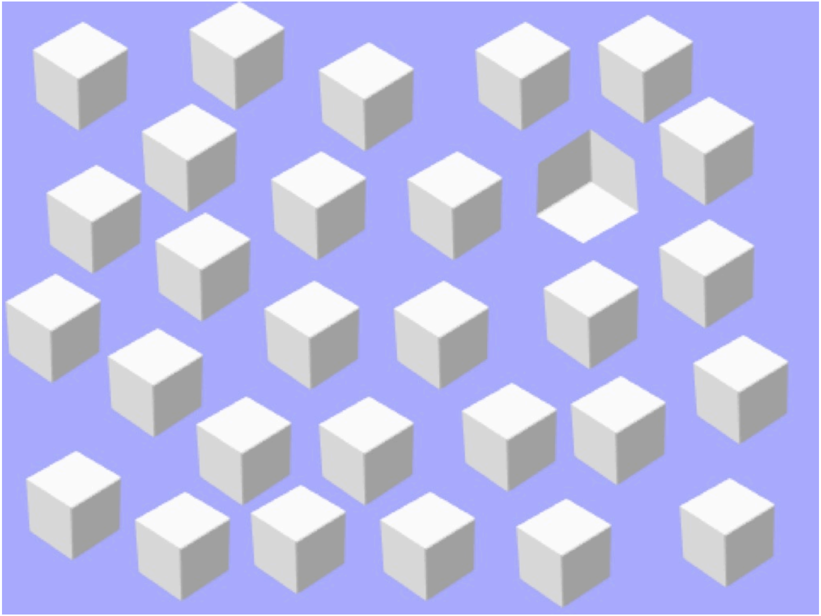
in their

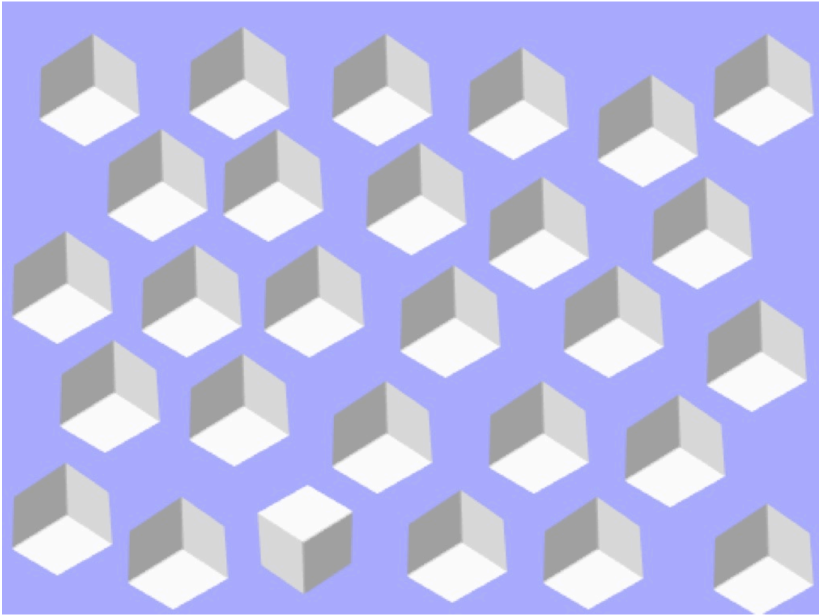
- learning
- perception
- cognition

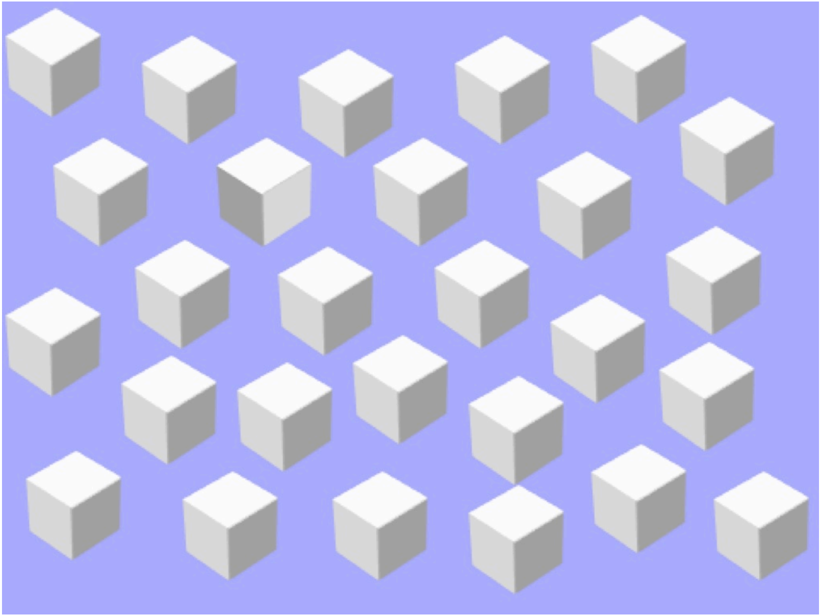
It takes serious scientific investigation

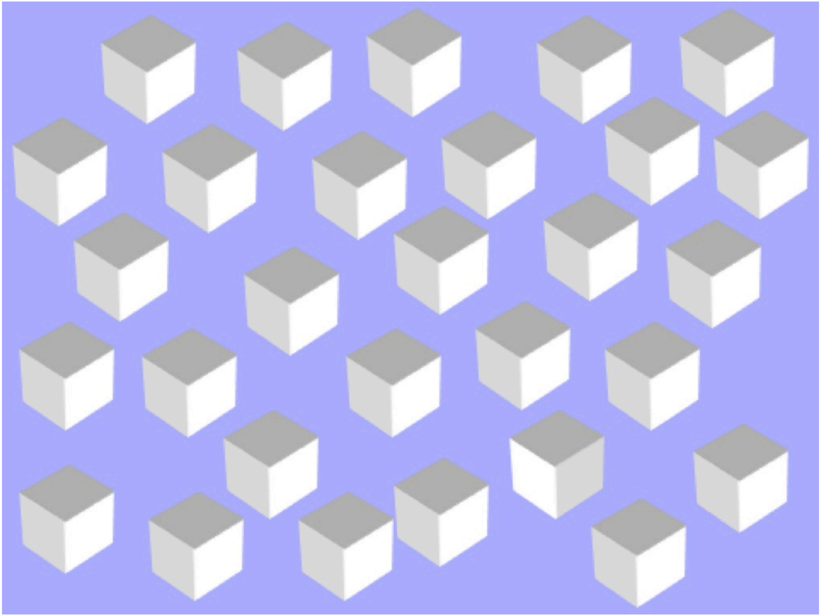
to ascertain exactly what those constraints/structures are











“time flies like an arrow”

Why the growing interest in machine learning?

Obviously

- data is everywhere
- data is increasingly captured
- data is increasingly comprehensive

- storage, communication, processing cheap & ubiquitous

Data is important

Machine learning provides an effective development methodology:

- when you cannot program a solution by hand
- but data is available

let the data determine the program

Machine learning is having an impact

language translation	personalization	automated driving
web search	surveillance	intrusion detection
spam filtering	ad selection	recommenders
speech recognition	handwriting recognition	text analysis
speaker recognition	game playing	non-player characters
face detection	car braking	information extraction
face recognition	engine control	product pricing

All major companies with large data sets have an interest

Lecture plan

Problem: Learning a function from data

Domain $\mathcal{X}$ Range $\mathcal{Y}$

$$\begin{matrix} \langle x_1, y_1 \rangle \\ \langle x_2, y_2 \rangle \\ \vdots \\ \langle x_t, y_t \rangle \end{matrix} \implies \boxed{\text{Learner}} \implies h : \mathcal{X} \to \mathcal{Y}$$

Idea

extrapolate y values over all x

Hope

predict well on unseen xs

Problem: Learning a function from data

One of the most studied problems in machine learning

Examples

image	$\rightarrow$	person
acoustic signal	$\rightarrow$	phonemes
transaction history	$\rightarrow$	fraud warning
English sentence	$\rightarrow$	French sentence

Complex data interpretation
Classification
Prediction/regression

Powerful idea
But how to do it?

To get started

Need

- Paired data, and representations for x, y, h
- Algorithm for computing h given $\langle x_1, y_1 \rangle, \ldots, \langle x_t, y_t \rangle$

Initial strategy: "empirical error minimization"

- Fix hypothesis space H
- Fix prediction error function $L(\hat{y}; y)$ (also called a loss function)

Then given data $\langle \mathbf{x}_1, y_1 \rangle, \ldots, \langle \mathbf{x}_t, y_t \rangle$, compute

$$\hat{h} = \arg\min_{h \in H} \frac{1}{t} \sum_{i=1}^{t} L(h(\mathbf{x}_i); y_i)$$

Simple example

Learning a **linear function**

$\mathbf{x} = \text{ vector } \in \mathbb{R}^n$

$y = \text{ scalar } \in \mathbb{R}$

$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}'\mathbf{x}$ for some $\mathbf{w} \in \mathbb{R}^n$

$H = \{h_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^n\}$

Prediction error

Let's choose, say, $L(\hat{y}; y) = |\hat{y} - y|$

Given X, $\mathbf{y}$, compute

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \frac{1}{t} \sum_{i=1}^{t} |X_{i:}\mathbf{w} - y_i|$$

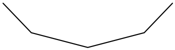
Get predictor

$\mathbf{x}' \mapsto \hat{y} = \mathbf{x}'\hat{\mathbf{w}}$

Learning a linear function

Note

The training problem in this example is a nonsmooth, piecewise linear, convex minimization



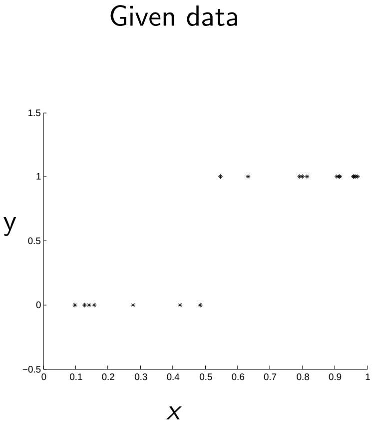
$$\min_{\mathbf{w}} \hat{\ell}(\mathbf{w}) \text{ where } \hat{\ell}(\mathbf{w}) = \frac{1}{t}\sum_{i=1}^{t} L(X_{i:}\mathbf{w}; y_i) = \frac{1}{t}\sum_{i=1}^{t} |X_{i:}\mathbf{w} - y_i|$$

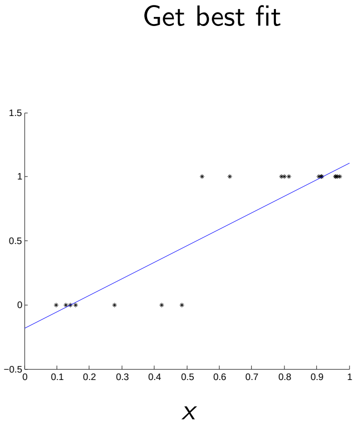
Still easy to solve

E.g. with a linear program

$$\min_{\mathbf{w},\boldsymbol{\delta}} \tfrac{1}{t}\boldsymbol{\delta}'\mathbf{1} \text{ subject to } y_i - \delta_i \le X_{i:}\mathbf{w} \le y_i + \delta_i$$

Learning a linear function





Question

Does it generalize?

- implicitly assuming independent identically distributed (iid) training pairs; i.e. fixed $P_{\mathcal{X}\mathcal{Y}} = P_{\mathcal{Y}|\mathcal{X}} P_{\mathcal{X}}$

Still, even given iid:

- $P_{\mathcal{Y}|\mathcal{X}}$ might not be well modeled by linear function
- Empirical error might be inaccurate: $E[\hat{\ell}(\hat{h})] \leq E[\ell(\hat{h})]$ where $\ell(\hat{h}) = E[L(\hat{h}(\mathbf{x}); y)]$, expected test error; i.e. minimum training loss underestimates test loss

Conclude that learning a linear function

Might be a good idea because

- compact representation
- efficient training
- efficient prediction

Might be a bad idea because

- linear too restrictive (underfits)
- linear not restrictive enough (overfits)

Preview

I will focus on **linear** function learning techniques

Unifies almost all current, tractable approaches to function learning

Much more powerful than you think

- *generalize* input representations via nonlinear features
- *generalize* output predictions via nonlinear transfers
- incorporate latent structure

All still allow efficient algorithms

(except latent structure—that's still research)

Generalized linear modeling

Quickest way to:

- get up to speed on much of the field
- empower you to implement interesting, useful methods

Plan

Generalized linear modeling

Part 1: Generalized domain representations and regularization
today

Part 2: Generalized range representations and structure
tomorrow

Part 3: Latent representations and unsupervised training
(some current research)
Wednesday

Themes

Modeling

Flexible representations

Computation

Efficient training and prediction algorithms

Generalization

Capacity control—overfitting avoidance

Part 1: Generalized domain representations and regularization

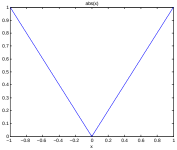
Dale Schuurmans

University of Alberta

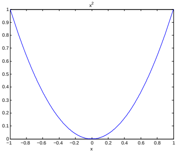
Warm up: loss functions

What prediction loss function $L(\hat{y}, y)$ to use?

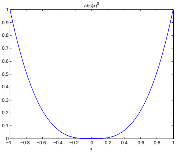
Absolute loss (L_1) $\quad |\hat{y} - y|$



Squared loss (L_2^2) $\quad (\hat{y} - y)^2$



L_p^p loss $\quad (\hat{y} - y)^p$



Loss functions

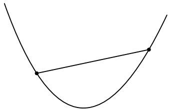
Convexity

ℓ convex if $\quad \ell(\rho \mathbf{w}_1 + (1-\rho)\mathbf{w}_2) \leq \rho\ell(\mathbf{w}_1) + (1-\rho)\ell(\mathbf{w}_2)$

for $0 \leq \rho \leq 1 \quad \ell$ at mean $\leq$ mean of ℓs

Properties

- ℓ convex, $\mathbf{w}$ local minima $\Rightarrow$ $\mathbf{w}$ global minima
- nonnegative weighted sum of convex is convex
- max of convex is convex
- ℓ convex $\Rightarrow$ $\ell(X\mathbf{w})$ convex in $\mathbf{w}$
- L_p^p loss convex if $p \geq 1$



Note

convex loss will generally result in tractable training problem

nonconvex loss will generally result in intractable training problem

(* we will see exceptions, but these will be somewhat special)

Loss functions

Smoothness

L_p^p loss differentiable for $p > 1$
L_1 loss not differentiable, but still convex

Properties

Nonsmooth optimization generally more expensive than smooth
But convexity still generally results in tractable training problems
(as we saw for L_1 loss)

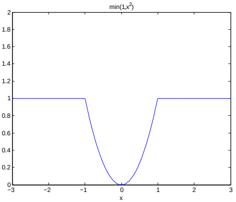
Other lecturers might explain algorithmic ideas behind efficient smooth/nonsmooth minimization.

Loss functions

Robust loss

$\min(1, (\hat{y} - y)^2)$

"gives up" on outliers



L_1 more robust than L_2^2

L_p^p more robust than L_q^q for $p \leq q$

These are iid losses

$L(\hat{\mathbf{y}}; \mathbf{y}) = \frac{1}{t} \sum_{i=1}^{t} L(\hat{y}_i; y_i)$

Non-iid losses

e.g., F-measure

Let us assume **iid losses**

Shorthand notation

$\hat{\ell}(\mathbf{w}) = L(X\mathbf{w}; \mathbf{y}) = \frac{1}{t} \sum_{i=1}^{t} L(X_{i:}\mathbf{w}; y_i)$

Loss functions

Today in Part 1

Just assume we've picked a convex loss $L(\hat{y}; y)$ (say L_1 or L_2^2)

Tomorrow in Part 2

Will show how loss function can be derived from other considerations

Wednesday in Part 3

Will show how robust loss can be expressed as a convex loss plus latent outlier indicators

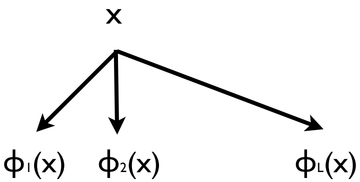
Generalizing the domain representation

Generalized domain representations

Simple idea: feature expansion

- expand representation $\mathbf{x} \mapsto \phi(\mathbf{x})$

New features are (nonlinear) function of original features



"basis functions", "features", "feature functions"

Feature expansion

Expand training set $X \mapsto \Phi$

$$\begin{bmatrix} X_{11} & \cdots & X_{1n} \\ \vdots & & \vdots \\ X_{t1} & \cdots & X_{tn} \end{bmatrix} \mapsto \begin{bmatrix} \phi_1(X_{1:}) & \cdots & \phi_L(X_{1:}) \\ \vdots & & \vdots \\ \phi_1(X_{t:}) & \cdots & \phi_L(X_{t:}) \end{bmatrix}$$

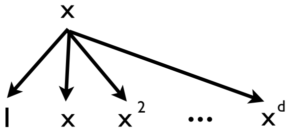
Learn a linear function over extended features
(a nonlinear function of the original features)

Generalized predictor

After learning an extended $L \times 1$ weight vector $\mathbf{w}$
get a nonlinear predictor

$$\mathbf{x} \;\mapsto\; \hat{y} = \sum_{j=1}^{L} w_j \phi_j(\mathbf{x}) = \mathbf{w}' \phi(\mathbf{x})$$

Example: polynomial basis



Assume $x_i \in \mathbb{R}$ (scalar)

Training data expansion $\begin{bmatrix} x_1 \\ \vdots \\ x_t \end{bmatrix} \mapsto \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & & & & \vdots \\ 1 & x_t & x_t^2 & \cdots & x_t^d \end{bmatrix}$

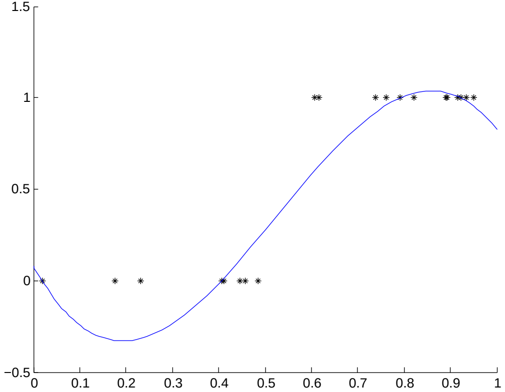
Training

Train $(d+1) \times 1$ vector of coefficients $\mathbf{w}$ using any desired loss

Learned predictor

$x \mapsto \hat{y} = \mathbf{w}'\phi(x) = \sum_{j=0}^{d} w_j x^j$

Example: polynomial basis



Example: trigonometric basis

Assume $x_i \in \mathbb{R}$ (scalar)

Training data expansion $\begin{bmatrix} x_1 \\ \vdots \\ x_t \end{bmatrix} \mapsto \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_t(x_1) \\ \vdots & & \vdots \\ \phi_1(x_t) & \cdots & \phi_t(x_t) \end{bmatrix}$

Use t basis functions assuming $t = 2n + 1$ for some n

1 constant basis function	$\phi_1 = \frac{a_0}{2}$
n cosine basis functions	$\phi_{1+j} = \cos(jx)$ for $j = 1...n$
n sine basis functions	$\phi_{n+1+j} = \sin(jx)$ for $j = 1...n$

If data points happen to be evenly spaced

$x_i - x_{i-1} = \Delta$ constant

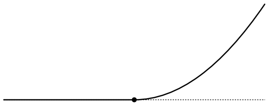
Then columns of Φ are orthonormal and Φ square

Hence exact fit of $\mathbf{y}$ given by $\mathbf{w}^* = \Phi' \mathbf{y}$ (discrete Fourier transform)

Example: basis splines

Assume $x_i \in \mathbb{R}$ (scalar)

$\phi_1(x) = 1$
$\phi_2(x) = x$
$\phi_3(x) = x^2$
$\phi_4(x) = x^3$
$\phi_5(x) = (x - x_1)_+^3 \cdots$
$\phi_j(x) = (x - x_{j-4}) +^3 \cdots$
$\phi_t(x) = (x - x_{t-4}) +^3$

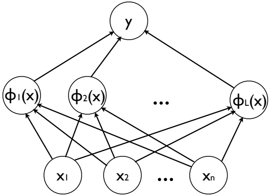


Linear combination is piecewise cubic

$x \mapsto \hat{y} = \sum_{j=1}^{t} w_j \phi_j(x)$
Predictor is continuous and has continuous 1st and 2nd derivatives

Example: multilayer neural network

Feedforward neural network with a fixed preprocessing layer



E.g. $\phi_j(\mathbf{x}) = \mathrm{sign}(\mathbf{u}_j'\mathbf{w})$ for some $\mathbf{u}_j$

Given intermediate representation

Learn $\mathbf{w}$, get predictor $\mathbf{x} \mapsto \hat{y} = \sum_{j=1}^{L} w_j \phi_j(\mathbf{x})$

Local basis functions

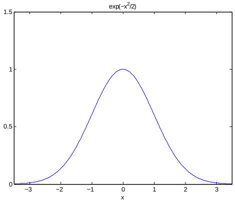
Local basis function

Choose a similarity function κ

$\phi_j(\mathbf{x}) = \kappa(\mathbf{x}, \hat{\mathbf{x}}_j)$ at $\hat{\mathbf{x}}_j$

$\kappa \geq 0$, maximized at $\mathbf{x} = \hat{\mathbf{x}}_j$

$\kappa(\mathbf{x}, \hat{\mathbf{x}}_j)$ decreasing in $\|\mathbf{x} - \hat{\mathbf{x}}_j\|$



$\hat{\mathbf{x}}_j$

Fixing prototype centers $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_L$

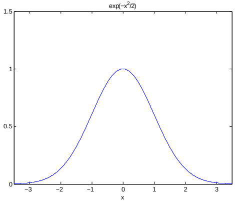
defines basis $\phi_1, \ldots, \phi_L$

Expand training set

$X \mapsto \Phi$

Learn weights $\mathbf{w}$ over expanded feature representation

Example: radial basis functions (rbfs)



$\hat{\mathbf{x}}_j$

$$\kappa(\mathbf{x}, \hat{\mathbf{x}}_j) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x} - \hat{\mathbf{x}}_j\|)$$

σ is a “width” parameter

Fully local methods

Locate a prototype center $\hat{\mathbf{x}}_i$ at *every* training point $X_{i:}$

$$X \mapsto \begin{bmatrix} \kappa(X_{1:}, X_{1:}) & \cdots & \kappa(X_{1:}, X_{t:}) \\ \vdots & & \vdots \\ \kappa(X_{t:}, X_{1:}) & \cdots & \kappa(X_{t:}, X_{t:}) \end{bmatrix} = K$$

Interpolation

For most local basis functions κ this enables interpolation

I.e. K is $t \times t$ square matrix

usually invertible (if training examples $X_{i:}$ not duplicated)

$\Rightarrow$ can solve $K\mathbf{w} = \mathbf{y}$ for $\mathbf{w}$

Example: for RBFs

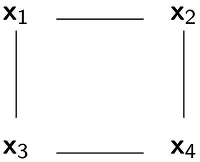
K is symmetric, diagonally dominant, invertible

Example: k nearest neighbors

k nearest neighbor basis function depends on entire training set X

$$\kappa(\mathbf{x}, X_{j:}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ closer to } X_{j:} \text{ than all but } k \text{ points in } X \\ 0 & \text{otherwise} \end{cases}$$

E.g. consider 2-nearest neighbors



$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Note

K is invertible provided each $X_{i:}$ sufficiently connected
(and k not too large nor too small)

General feature representations

Can construct feature representations for *arbitrary* objects

E.g. strings, graphs, documents

Each feature

- just computes some aspect of the object that hopefully is important for prediction
- map general objects into a feature vector representation

In practice

features are the main source of prior knowledge/constraints
—carefully engineered

E.g.

image processing	—	edge filters, line filters, SIFTs
document processing	—	bag of words, TF-IDF, *n*-grams
network processing	—	degree distribution, friend-of-friend dist'n

The elephant in the room

Dilemma

For a given problem, which features to use?

If $P_{\mathcal{Y}|\mathcal{X}}$ not known

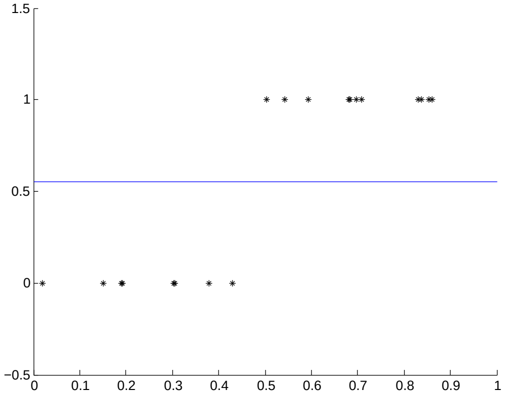
- why not try to be as expressive as possible?
- can represent any target function $f : \mathcal{X} \to \mathcal{Y}$ that way

Fundamental dilemma

underfitting versus overfitting

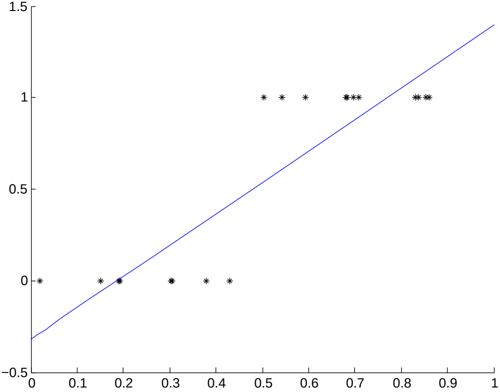
Dilemma

Example: polynomial fitting



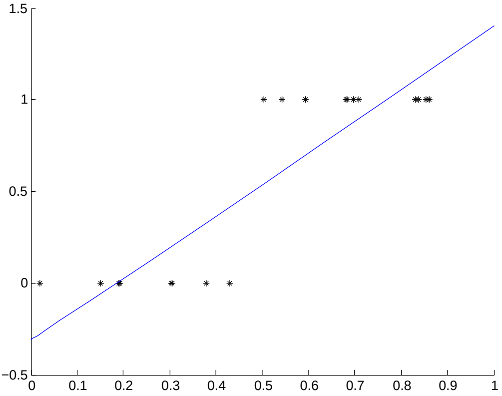
Dilemma

Example: polynomial fitting



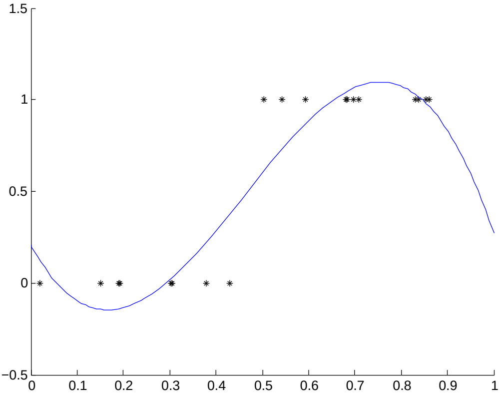
Dilemma

Example: polynomial fitting



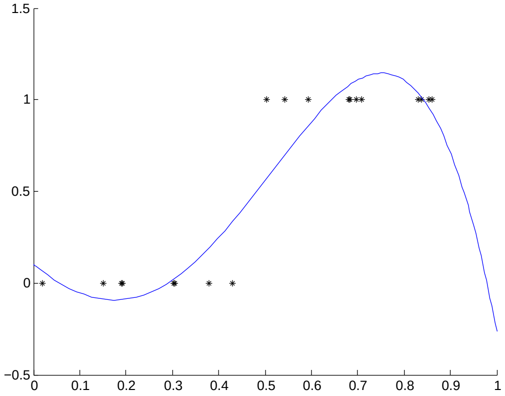
Dilemma

Example: polynomial fitting



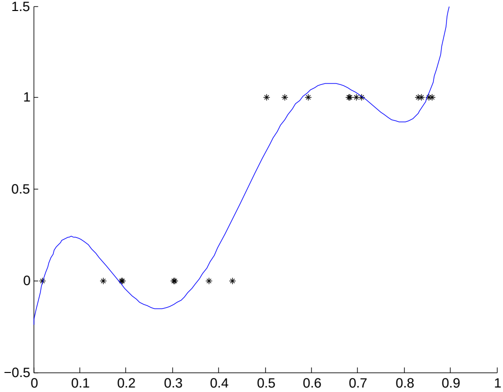
Dilemma

Example: polynomial fitting



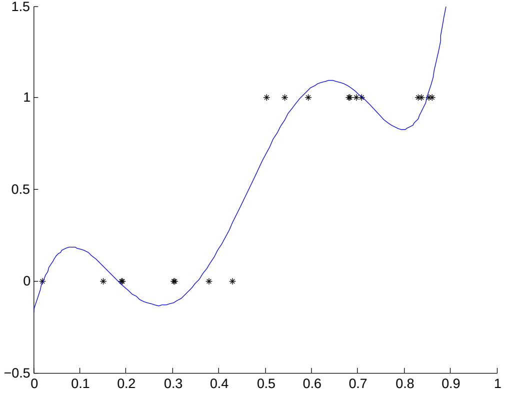
Dilemma

Example: polynomial fitting



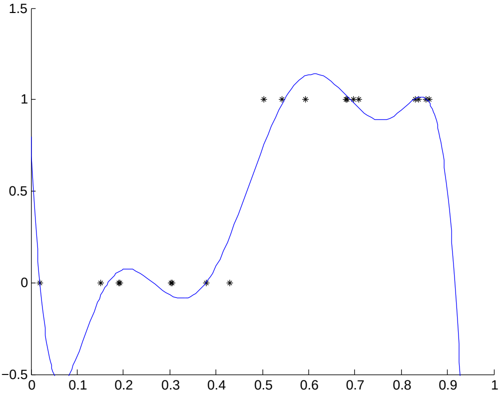
Dilemma

Example: polynomial fitting



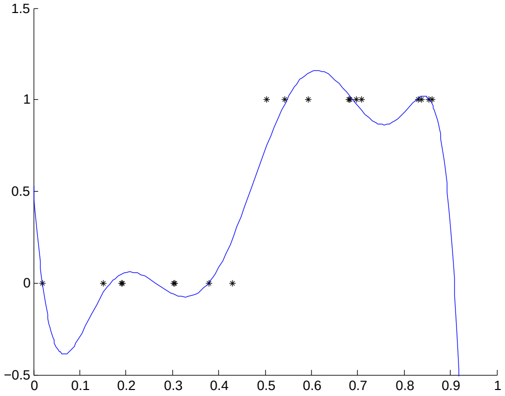
Dilemma

Example: polynomial fitting



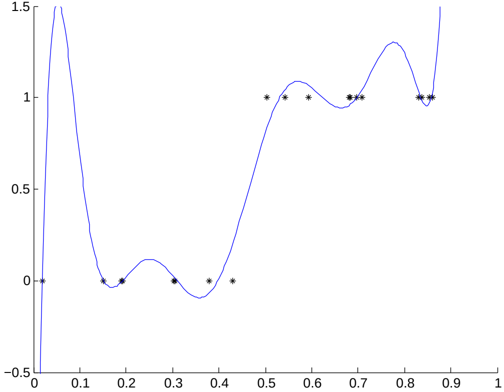
Dilemma

Example: polynomial fitting



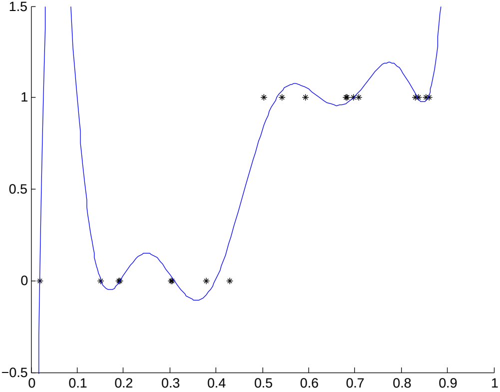
Dilemma

Example: polynomial fitting



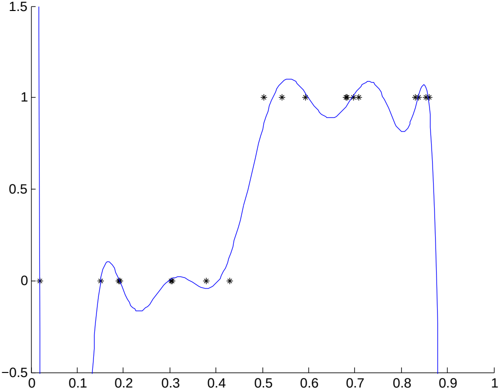
Dilemma

Example: polynomial fitting



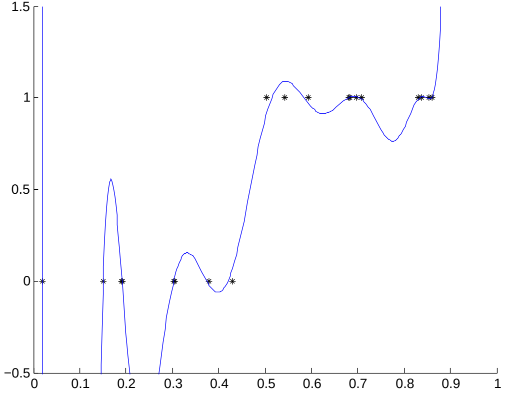
Dilemma

Example: polynomial fitting



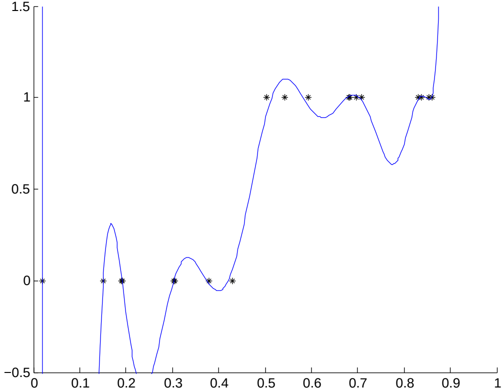
Dilemma

Example: polynomial fitting



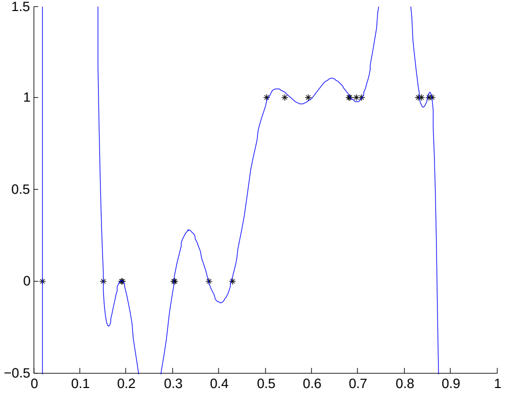
Dilemma

Example: polynomial fitting



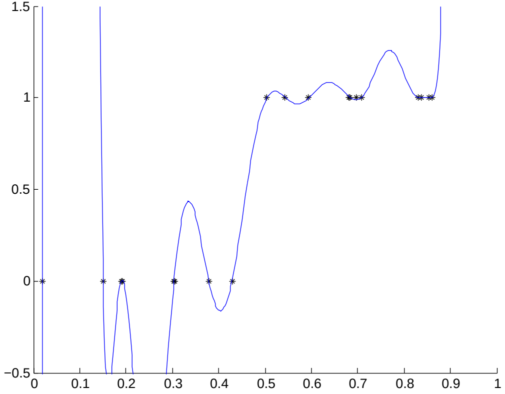
Dilemma

Example: polynomial fitting



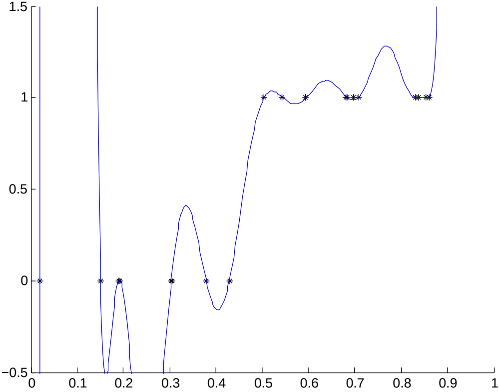
Dilemma

Example: polynomial fitting



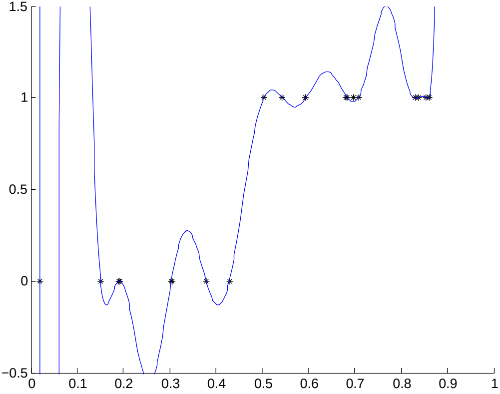
Dilemma

Example: polynomial fitting



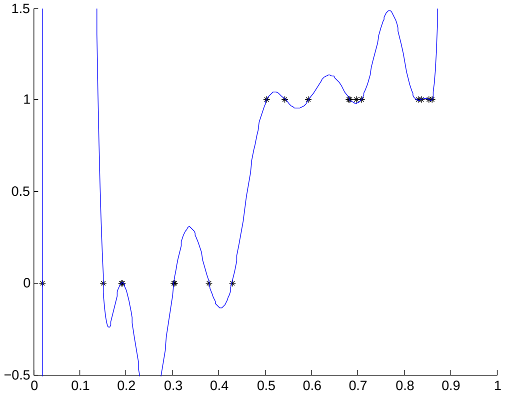
Dilemma

Example: polynomial fitting



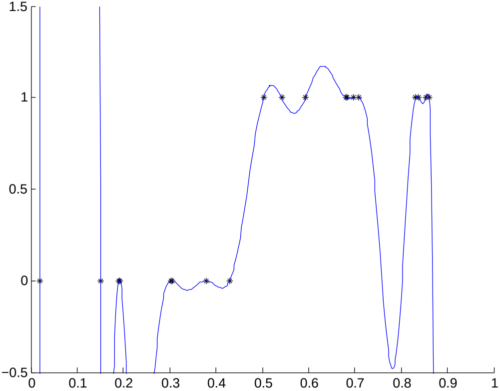
Dilemma

Example: polynomial fitting



Dilemma

Example: polynomial fitting



Overfitting versus underfitting

too many features	risks	overfitting
too few features	risks	underfitting

Strategies

Feature selection

- choose "right" set of basis functions

Regularization

- "smooth" functions by limiting size of weights

Overfitting versus underfitting

Regularization

"smoothing"
Limit slope of hypothesis function

$$\min_{\mathbf{w}} L(\Phi\mathbf{w};\mathbf{y}) + \beta\|\mathbf{w}\| \text{ where } \beta \geq 0 \text{ is a regularization parameter}$$

Tradeoff between minimizing error and size of **w**

How to measure size of **w**?

L_2^2 norm	$\rightarrow$	leads to kernels
L_1 norm	$\rightarrow$	leads to sparsity

Euclidean regularization

Euclidean regularization

Penalize $\mathbf{w}$ by its (squared) Euclidean norm

$$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \frac{\beta}{2}\|\mathbf{w}\|_2^2$$

$\beta > 0$ a regularization parameter

The L_2^2 regularizer is a convenient choice because

$\frac{1}{2}\|\mathbf{w}\|_2^2$ is

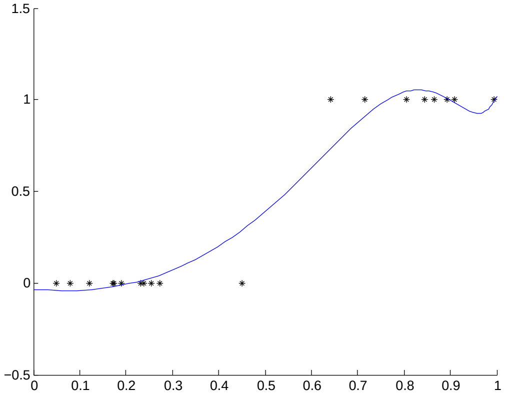
- convex
- smooth
- simple (e.g. $\nabla_{\mathbf{w}} = \mathbf{w}$)

More importantly

Euclidean regularization leads to an amazing generalization beyond finite dimensional feature vectors

Euclidean regularization

Example: polynomial fitting



Important property of Euclidean regularization

Simple representer theorem

For any L and any increasing R, if

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + R(\|\mathbf{w}\|_2^2)$$

exists, then $\mathbf{w}^* = \Phi'\mathbf{a}^*$ for some $\mathbf{a}^*$

Proof

Since $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*$ for $\mathbf{w}_1^* \in \text{rowspan}(\Phi)$ and $\mathbf{w}_0^* \perp \text{rowspan}(\Phi)$

$$\begin{aligned} \mathbf{w}_1^* &= \Phi'\mathbf{a}^* \text{ for some } \mathbf{a}^* \\ \Phi\mathbf{w}^* &= \Phi\mathbf{w}_0^* + \Phi\mathbf{w}_1^* = \Phi\mathbf{w}_1^* \\ \|\mathbf{w}^*\|_2^2 &= \|\mathbf{w}_0^* + \mathbf{w}_1^*\|_2^2 = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2 \end{aligned}$$

If $\mathbf{w}_0^* \neq 0$ then

$$\begin{aligned} L(\Phi\mathbf{w}^*; \mathbf{y}) + R(\|\mathbf{w}^*\|_2^2) &= L(\Phi\mathbf{w}_1^*; \mathbf{y}) + R(\|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2) \\ &> L(\Phi\mathbf{w}_1^*; \mathbf{y}) + R(\|\mathbf{w}_1^*\|_2^2) \text{ contradiction } \blacksquare \end{aligned}$$

Important property of Euclidean regularization

Equivalent adjoint formulation

Can instead solve for example weights $\mathbf{a}$, where $\mathbf{w} = \Phi'\mathbf{a}$

Original training	$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \frac{\beta}{2}\mathbf{w}'\mathbf{w}$
Original prediction	$\mathbf{x} \mapsto \hat{y} = \mathbf{w}^{*\prime}\phi(\mathbf{x})$
Adjoint training	$\min_{\mathbf{a}} L(\Phi\Phi'\mathbf{a}; \mathbf{y}) + \frac{\beta}{2}\mathbf{a}'\Phi\Phi'\mathbf{a}$
Adjoint prediction	$\mathbf{x} \mapsto \hat{y} = \mathbf{a}^{*\prime}\Phi\phi(\mathbf{x})$

Equivalent! by simple representer theorem

Key observation

Adjoint formulation does not require feature vectors
only inner products between feature vectors

Important property of Euclidean regularization

Equivalent kernel formulation

Assume a function $\kappa(\cdot,\cdot)$ that computes inner products $\kappa(\Phi_{i:},\Phi_{j:}) = \Phi_{i:}\Phi_{j:}'$

Kernel training	$\min_{\mathbf{a}} L(K\mathbf{a};\mathbf{y}) + \frac{\beta}{2}\mathbf{a}'K\mathbf{a}$
	where $K_{ij} = \kappa(\Phi_{i:},\Phi_{j:})$
Kernel prediction	$\mathbf{x} \mapsto \hat{y} = \mathbf{a}^{*\prime}\mathbf{k}$
	where $\mathbf{k}_i = \kappa(\Phi_{i:},\phi(\mathbf{x})')$

Example

Polynomial feature vector

$$\phi(x) = \left(\sqrt{\binom{d}{0}}, \sqrt{\binom{d}{1}}x, \sqrt{\binom{d}{2}}x^2, ..., \sqrt{\binom{d}{d}}x^d\right)$$

Corresponding kernel

$$\kappa(x_1,x_2) = (x_1x_2+1)^d = \sum_{i=0}^{d}\binom{d}{i}x_1^i x_2^i = \phi(x_1)'\phi(x_2)$$

Direct computation can be arbitrarily more efficient

Kernels

Similarity measure on a set of objects $\mathcal{X}$

vectors, strings, sentences, documents, trees, graphs

Instead of features, choose a kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

symmetric: $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa(\mathbf{x}_2, \mathbf{x}_1)$
semidefinite:
for any finite set $\{\mathbf{x}_1, ..., \mathbf{x}_t\} \subset \mathcal{X}$

$$\begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & & \vdots \\ \kappa(\mathbf{x}_t, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} \succeq 0$$

Strictly generalizes finite dimensional feature vectors

Example

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2)$
does not have finite dimensional feature representation

Kernels

Reproducing kernel Hilbert space

Given a symmetric, semidefinite operator $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ coherently defines a Hilbert space

- Basis given by features $\phi_{\hat{\mathbf{x}}}$ for all $\hat{\mathbf{x}} \in \mathcal{X}$
- $\mathcal{H}_0$ = finite linear combinations of $\phi_{\hat{\mathbf{x}}}$
- Define $\langle \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i}, \sum_{j=1}^m b_i \phi_{\hat{\mathbf{x}}_j} \rangle_{\mathcal{H}} = \sum_{i=1}^n \sum_{j=1}^m a_i b_j \kappa(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$
- Define $\| \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i} \|_{\mathcal{H}} = \langle \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i}, \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i} \rangle_{\mathcal{H}}^{1/2}$
- $\mathcal{H}$ = completion of $\mathcal{H}_0$ under $\| \cdot \|_{\mathcal{H}}$

Representer theorem still holds

For any L and any increasing R

$$h^* = \arg\min_{h \in \mathcal{H}} L(h(X); \mathbf{y}) + R(\|h\|_{\mathcal{H}})$$

can be written $h^*(\cdot) = \sum_{i=1}^t \mathbf{a}_i^* \phi_{X_{i:}}(\cdot) = \sum_{i=1}^t \mathbf{a}_i^* \kappa(X_{i:}, \cdot)$ for some $\mathbf{a}^*$

Feature selection

Feature selection

Problem

Choose a subset of feature functions to use

- i.e. choose a subset of $\{\phi_1, ..., \phi_L\}$

Difficulty

- 2^L subsets
- Intractable to enumerate
- Finding a bounded feature subset that minimizes training error NP-hard in general

Idea

Use a convex relaxation of feature selection

- L_1 regularization

L_1 regularization

L_1 regularized training problem

$$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \beta\|\mathbf{w}\|_1$$

$\beta \geq 0$ regularization parameter

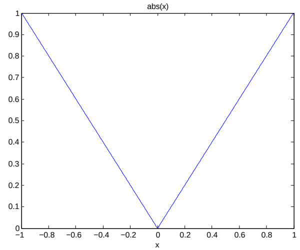
Properties

- Convex in $\mathbf{w}$
- Nonsmooth
- Implicitly encourages sparsity (i.e. $w_j = 0$ for some j)
- Provides a tractable relaxation of feature selection

L_1 regularization

Why does L_1 regularization yield sparse solutions?

$\|\mathbf{w}\|_1 = \sum_{j=1}^{L} |w_j|$

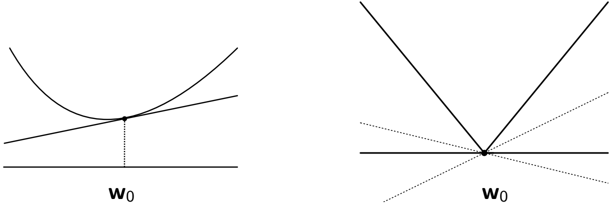


$$\frac{\partial}{\partial w_j} = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ \text{undef} & \text{if } w_j = 0 \end{cases}$$

L_1 regularization

Subgradients

For a differentiable convex function ℓ, always have
$$\ell(\mathbf{w}) \geq \ell(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)'\nabla\ell(\mathbf{w}_0)$$



A **subgradient** at $\mathbf{w}_0$

is any $\mathbf{d}_0$ such that $\ell(\mathbf{w}) \geq \ell(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)'\mathbf{d}_0$

Theorem

If ℓ differentiable at $\mathbf{w}_0$ then $\mathbf{d}_0$ is unique and $\mathbf{d}_0 = \nabla\ell(\mathbf{w}_0)$.

What if ℓ not differentiable at $\mathbf{w}_0$?
Then $\mathbf{d}_0$ is not unique

L_1 regularization

Implicit feature selection

Consider a descent step from a current $\mathbf{w}$

$$\frac{\partial}{\partial w_j} = \beta \mathrm{sign}(w_j) + \sum_{i=1}^{t} L'(\Phi_{i:}\mathbf{w}; y_i)\Phi_{ij} \quad \text{if } w_j \neq 0$$

What if current value of $w_j = 0$?

if $\quad |\sum_{i=1}^{t} L'(\Phi_{i:}\mathbf{w}; y_i)\Phi_{ij}| < \beta$

w_j stays at 0; that is, no local descent from $w_j = 0$

else

can reduce the objective
by moving w_j in direction of $-\sum_{i=1}^{t} L'(\Phi_{i:}\mathbf{w}; y_i)\Phi_{ij}$

L_1 regularization

Efficiently solvable if L convex

$$\min_{\mathbf{w}} L(\Phi\mathbf{w};\mathbf{y}) + \beta\|\mathbf{w}\|_1$$
$$= \min_{\mathbf{w},\boldsymbol{\xi}} L(\Phi\mathbf{w};\mathbf{y}) + \beta\mathbf{1}'\boldsymbol{\xi} \text{ subject to } \boldsymbol{\xi} \geq \mathbf{w},\ \boldsymbol{\xi} \geq -\mathbf{w}$$

- convex objective (if L convex)
- linear constraints

E.g.

If $L(\hat{y};y) = (\hat{y}-y)^2$ get a quadratic program
If $L(\hat{y};y) = |\hat{y}-y|$ get a linear program

Problem

L_1 regularization blocks the representer theorem!

How to combine kernels and feature selection?

Naive approach does not work:

$$\beta_1 \|\mathbf{w}\|_1 + \beta_2 \|\mathbf{w}\|_2^2$$

blocks representer theorem—no equivalent adjoint form

—hence no equivalent kernel form

Fortunately

It is possible to combine L_1 and L_2^2 keeping kernels,
but requires an indirect approach:

- introduce separate feature selection variables $\boldsymbol{\mu}$
- exploit Fenchel conjugate of L

Kernel selection

Kernel selection

Relating feature and kernel selection

Consider a feature representation Φ, a $t \times L$ matrix

Get kernel matrix

$$K = \Phi\Phi' = \sum_{j=1}^{L} \Phi_{:j}\Phi_{:j}' = \sum_{j=1}^{L} K_j$$

I.e. each basis feature $\Phi_{:j}$ corresponds to a rank 1 kernel matrix

$$K_j = \Phi_{:j}\Phi_{:j}'$$

Kernel selection

Introduce auxiliary feature/kernel selection variables

Let $1 \geq \boldsymbol{\mu} \geq 0$ be a vector of selection weights

Consider $\tilde{\Phi} = \Phi\Delta(\boldsymbol{\mu})^{1/2}$
($\Delta(\boldsymbol{\mu})$ denotes putting $\boldsymbol{\mu}$ on main diagonal of square matrix)

Get

$$\tilde{K} = \tilde{\Phi}\tilde{\Phi}' = \Phi\Delta(\boldsymbol{\mu})\Phi' = \sum_{j=1}^{L} \mu_j \Phi_{:j}\Phi'_{:j} = \sum_{j=1}^{L} \mu_j K_j$$

Will use use $\boldsymbol{\mu}$ to select features/kernels

Aside: Fenchel duality

Given a function $\ell(\mathbf{w})$

Define its Fenchel conjugate as

$$\ell^*(\boldsymbol{\alpha}) = \sup_{\mathbf{w}} \boldsymbol{\alpha}'\mathbf{w} - \ell(\mathbf{w})$$

Guaranteed to be convex in $\boldsymbol{\alpha}$ (since max of linear is convex)

Strong duality property

If $\ell(\mathbf{w})$ is a closed, convex function then $\ell^{**}(\mathbf{w}) = \ell(\mathbf{w})$

That is

$$\ell(\mathbf{w}) = \sup_{\boldsymbol{\alpha}} \boldsymbol{\alpha}'\mathbf{w} - \ell^*(\boldsymbol{\alpha})$$

Fenchel duality

Equivalent dual problem

Can get an equivalent reformulation of L_2^2 regularized training

$$\min_{\mathbf{w}} L(\Phi\mathbf{w};\mathbf{y}) + \frac{\beta}{2}\|\mathbf{w}\|_2^2 \qquad \text{primal problem}$$

$$= \max_{\boldsymbol{\alpha}} -L^*(\boldsymbol{\alpha};\mathbf{y}) - \frac{1}{2\beta}\boldsymbol{\alpha}'K\boldsymbol{\alpha} \qquad \text{dual problem}$$

where

$$L^*(\boldsymbol{\alpha};\mathbf{y}) = \sup_{\hat{\mathbf{y}}} \boldsymbol{\alpha}'\hat{\mathbf{y}} - L(\hat{\mathbf{y}};\mathbf{y}) \quad \text{and} \quad K = \Phi\Phi'$$

Important

The representer theorem holds so expressible in terms of a kernel

(* some technical conditions apply on L)

Kernel selection

Putting the pieces together

Add an L_1 regularizer on $\boldsymbol{\mu}$ and jointly optimize

$$\min_{0 \leq \boldsymbol{\mu} \leq 1} \min_{\mathbf{w}} L(\Phi \Delta(\boldsymbol{\mu})^{1/2} \mathbf{w}; \mathbf{y}) + \frac{\beta_1}{2} \|\mathbf{w}\|_2^2 + \beta_2 \mathbf{1}' \boldsymbol{\mu}$$

$$= \min_{0 \leq \boldsymbol{\mu} \leq 1} \max_{\boldsymbol{\alpha}} -L^*(\boldsymbol{\alpha}; \mathbf{y}) - \frac{1}{2\beta_1} \sum_{j=1}^{t} \mu_j \boldsymbol{\alpha}' K_j \boldsymbol{\alpha} + \beta_2 \mathbf{1}' \boldsymbol{\mu}$$

The latter form is a concave-convex program—no local minima

Various computational strategies exist

equivalent convex reformulation of latter form above

Boosting

Boosting

Incremental training for large or infinite bases

Saw previously that kernels could be used to implicitly train with large or infinite bases

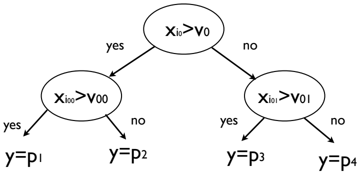
But what if you have a large basis but not corresponding efficient kernel?

Two classical examples of training with infinite bases:

- decision trees
- feedforward neural networks

Decision trees

Special generalized linear function



Each path corresponds to a single feature

$$\phi_1(\mathbf{x}) = 1_{(x_{i_0} > v_0 \text{ and } x_{i_{00}} > v_{00})} \in \{0, 1\}$$

$$\phi_2(\mathbf{x}) = 1_{(x_{i_0} > v_0 \text{ and } x_{i_{00}} \leq v_{00})} \in \{0, 1\}$$

$$\phi_3(\mathbf{x}) = 1_{(x_{i_0} \leq v_0 \text{ and } x_{i_{01}} > v_{01})} \in \{0, 1\}$$

$$\phi_3(\mathbf{x}) = 1_{(x_{i_0} \leq v_0 \text{ and } x_{i_{01}} \leq v_{01})} \in \{0, 1\}$$

Same predictor can be represented by a generalized linear function

$$\hat{y} = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \phi_3(\mathbf{x}) \\ \phi_4(\mathbf{x}) \end{bmatrix}' \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

Decision trees

Linear predictor plus tree constraint over bases

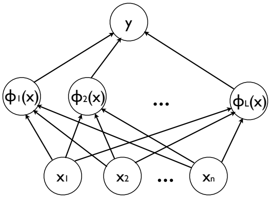
- NP-hard to find best tree of bounded size
- Standard training algorithms are heuristics
- Linear predictor = weighted forest of decision trees
- Could just learn a linear predictor over same basis

Difficulty

- Set of bases is infinite
- How to learn a linear model in such a case?
  (don't have an equivalent kernel)

Multilayer neural network

Two-layer feedforward neural networks



Difficulty

- Optimally training a 2-layer neural network is NP-hard
- Fixing # bases creates intractable feature selection problem
- Backpropagation training = local optimization heuristic

Can 2-layer neural network be trained efficiently?

- Idea: use L_1 regularization instead of feature selection
- Set of bases is infinite

Boosting

Incremental strategy for training a linear model

over a large or even infinite feature set

Strategy

- Do not enumerate basis
- Grow basis one function at a time by greedy procedure

Maintain a sparse model

At stage k have selected $k-1$ bases

$$h_{k-1}(\mathbf{x}) = \sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x})$$

Boosting

Greedy coordinate descent

Let

$$\ell(\mathbf{w}^{(k-1)}) = \sum_{i=1}^{t} L(h_{k-1}(\mathbf{x}_i); y_i) = \sum_{i=1}^{t} L\Big(\sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x}_i); y_i\Big)$$

Score of a new candidate feature ϕ_k

$$\frac{\partial \ell}{\partial w_k}\bigg|_{\mathbf{w}=\mathbf{w}^{(k-1)}} = \sum_{i=1}^{t} L'\Big(\sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x}_i); y_i\Big)\phi_k(\mathbf{x}_i) = \mathbf{L}'_{k-1}\phi_k$$

Boosting

Weak learning problem

Find steepest coordinate descent direction

$$\min_{\phi_k \in \Phi} \mathbf{L}'_{k-1}\phi_k$$

Behaves like weighted misclassification error

If $\mathbf{L}'_{k-1}\phi_k \geq 0$, halt

Once ϕ_k selected, solve for w_k by line search

$$\min_{w_k} \sum_{i=1}^{t} L(h_{k-1}(\mathbf{x}_i) + w_k \phi_k(\mathbf{x}_i); y_i)$$

Boosting

Convergence theorem

If

- L convex
- L' is b-Lipschitz continuous:

$$\|\mathbf{L}'(h) - \mathbf{L}'(g)\| \leq b\|h - g\| \quad \text{for some } b < \infty$$

- $\|\phi_k\| \leq B$ for some $B < \infty$
- Φ negation closed: $\phi \in \Phi \Rightarrow -\phi \in \Phi$
- Weak learner is approximately optimal: $\exists 0 < \gamma \leq 1$ such that

$$\mathbf{L}'_{k-1}\phi_k \leq \gamma \mathbf{L}'_{k-1}\phi^*_k \quad \text{for all } k$$

Then

- h_k converges to a global minimizer of L

(Mason et al. 2000)

Boosting

Adding regularization

Can use same strategy to converge to minimizer of

$$\min_{h \in \mathrm{span}(\Phi)} L(h(X); \mathbf{y}) + \beta \|\mathbf{w}\|_1$$

or

$$\min_{h \in \mathrm{span}(\Phi)} L(h(X); \mathbf{y}) + \frac{\beta}{2} \|\mathbf{w}\|_2^2$$

provided totally corrective weight update used.
That is, given ϕ_k, solve

$$\min_{w_1, \ldots, w_k} \sum_{i=1}^{t} L\left(\sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x}_i); y_i \right) + R([w_1, \ldots, w_k])$$

I.e. jointly re-optimize w_k with all previous weights

Boosting

Catch

Requires approximately optimal weak learner
(Warning: many papers sweep this little detail under the rug)

Good news

Tractable for some cases

- E.g. Φ = "decision stumps", $\phi(\mathbf{x}) = 1_{(x_j < c)}$ or $1_{(x_j \geq c)}$

Bad news

Intractable for almost all interesting bases

- E.g. Φ = "perceptrons" (linear threshold classifiers)
  NP-hard, even to approximate (Höffgen & Simon 1992)

Boosting

Crazy idea: sample bases randomly!

Can still guarantee a near optimal hypothesis with high probability

Set up

Let $H = \{h(\mathbf{x}) : \int w(\theta)\phi_\theta(\mathbf{x})p(\theta)d\theta$ such that $\|w(\theta)\| \leq c \ \forall\theta\}$

Assume $\|\phi\| \leq 1$ for all $\phi \in \Phi$

Sample $\theta_1, ..., \theta_K \sim p(\theta)$

Let $\hat{H} = \{h(\mathbf{x}) = \sum_j w_j \phi_{\theta_j}(\mathbf{x}) : |w_j| \leq c \ \forall j\}$

Theorem

For any $h \in H$ with probability at least $1 - \delta$
there exists some $\hat{h} \in \hat{H}$ such that

$$L(\hat{h}(X); \mathbf{y}) \leq L(h(X); \mathbf{y}) + \frac{bc}{\sqrt{Kt}}\left(1 + \sqrt{8 \log \frac{1}{\delta}}\right)$$

Part 2: Generalized output representations and structure

Dale Schuurmans

University of Alberta

Output transformation

Output transformation

What if targets y special?

E.g. what if y nonnegative $y \geq 0$
y probability $y \in [0, 1]$
y class indicator $y \in \{\pm 1\}$

Would like predictions $\hat{y}$ to respect same constraints

Cannot do this with linear predictors

Consider a new extension

Nonlinear output transformation f such that $\mathrm{range}(f) = \mathcal{Y}$

Notation and terminology

$\hat{y} = f(\hat{z})$ where $\hat{z} = \mathbf{x}'\mathbf{w}$
$\hat{z} = \mathbf{x}'\mathbf{w}$ "pre-prediction"
$\hat{y} = f(\hat{z})$ "post-prediction"

Nonlinear output transformation: Examples

Exponential

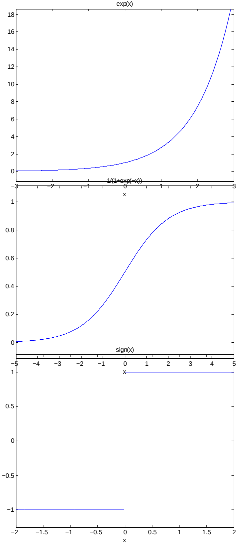
If $y \geq 0$ use $\hat{y} = f(\hat{z}) = \exp(\hat{z})$

Sigmoid

If $y \in [0, 1]$ use $\hat{y} = f(\hat{z}) = \frac{1}{1+\exp(-\hat{z})}$

Sign

If $y \in \{\pm 1\}$ use $\hat{y} = f(\hat{z}) = \text{sign}(\hat{z})$



Nonlinear output transformation: Risk

Combining arbitrary f with L can create local minima

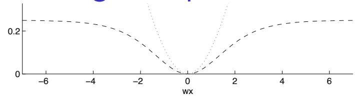
E.g.

$$L(\hat{y}; y) = (\hat{y} - y)^2$$

$$f(\hat{z}) = \sigma(\hat{z}) = (1 + \exp(-\hat{z}))^{-1}$$

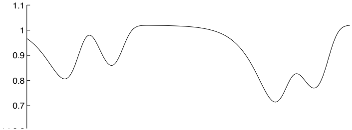
Objective $\sum_i (\sigma(X_{i:}\mathbf{w}) - y_i)^2$ is not convex in $\mathbf{w}$

Consider one training example



(Auer et al. NIPS-95)

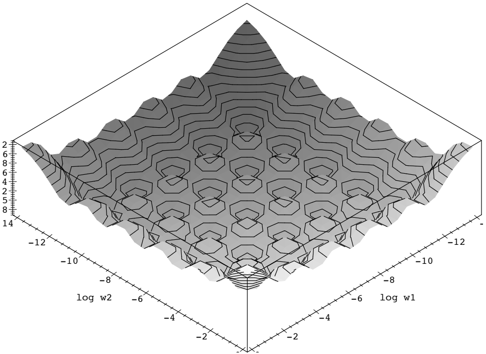
Local minima can combine



Nonlinear output transformation

Possible to create exponentially many local minima

t training examples can create $(t/n)^n$ local minima in n dimensions
—just local t/n training examples along each dimension



From (Auer et al., NIPS-95)

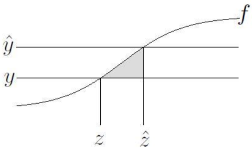
Important idea: matching loss

Assume f is continuous, differentiable, and strictly increasing

Want to define $L(\hat{y}; y)$ so that $L(f(\hat{z}); y)$ is convex in $\hat{z}$

Define matching loss by

$$
\begin{aligned}
L(f(\hat{z}); f(z)) &= \int_z^{\hat{z}} f(\theta) - f(z)\, d\theta \\
&= F(\theta)\big|_z^{\hat{z}} - f(z)\theta\big|_z^{\hat{z}} \\
&= F(\hat{z}) - F(z) - f(z)(\hat{z} - z)
\end{aligned}
$$



where $F'(z) = f(z)$; defines a Bregman divergence

Important idea: matching loss

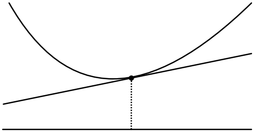
Properties

$F''(z) = f'(z) > 0$ since f strictly increasing

$\Rightarrow F$ strictly convex

$\Rightarrow F(\hat{z}) \geq F(z) + f(z)(\hat{z} - z)$ (convex function lies above tangent)

$\Rightarrow L(f(\hat{z}); f(z)) \geq 0$ and $L(f(\hat{z}); f(z)) = 0$ iff $\hat{z} = z$



Matching loss: examples

Identity transfer

$f(z) = z$, $F(z) = z^2/2$, $y = f(z) = z$
Squared error
$L(\hat{y}; y) = (\hat{y} - y)^2/2$

Exponential transfer

$f(z) = e^z$, $F(z) = e^z$, $y = f(z) = e^z$
Unnormalized entropy error
$L(\hat{y}; y) = y \ln \frac{y}{\hat{y}} + \hat{y} - y$

Sigmoid transfer

$f(z) = \sigma(z) = 1/(1 + e^{-z})$, $F(z) = \ln(1 + e^z)$, $y = f(z) = \sigma(z)$
Cross entropy error
$L(\hat{y}; y) = y \ln \frac{y}{\hat{y}} + (1 - y) \ln \frac{1-y}{1-\hat{y}}$

Matching loss

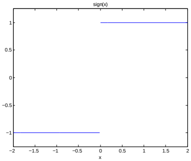
Given suitable f
Can derive a matching loss that ensures convexity of $L(f(X\mathbf{w}); \mathbf{y})$

Retain everything from before

- efficient training
- basis expansions
- L_2^2 regularization $\rightarrow$ kernels
- L_1 regularization $\rightarrow$ sparsity

Major problem remains: Classification

If, say, $y \in \{\pm 1\}$ class indicator, use $\hat{y} = \mathrm{sign}(\hat{z})$



Not continuous, differentiable, strictly increasing
Cannot use matching loss construction

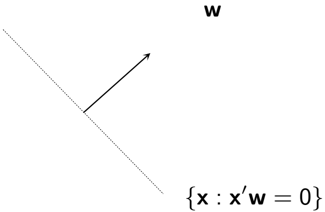
Misclassification error

$$L(\hat{y}; y) = 1_{(\hat{y} \neq y)} = \begin{cases} 0 & \text{if } \hat{y} = y \\ 1 & \text{if } \hat{y} \neq y \end{cases}$$

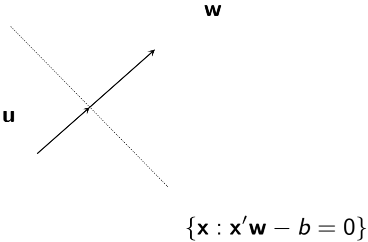
Classification

Classification

Consider geometry of linear classifiers $\hat{y} = \mathrm{sign}(\mathbf{x}'\mathbf{w})$



Linear classifiers with offset $\hat{y} = \mathrm{sign}(\mathbf{x}'\mathbf{w} - b)$



$u = \frac{b}{\|\mathbf{w}\|_2^2}\mathbf{w}$ since $\mathbf{u}'\mathbf{w} = b$, $\mathbf{u}'\mathbf{w} - b = 0$

Classification

Question

Given training data X, $\mathbf{y} \in \{\pm 1\}^t$ can minimum misclassification error $\mathbf{w}$ be computed efficiently?

Answer

Depends

Classification

Good news

Yes, if data is linearly separable

Linear program

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathbf{1}'\boldsymbol{\xi} \text{ subject to } \Delta(\mathbf{y})(X\mathbf{w} - \mathbf{1}b) \geq \mathbf{1} - \boldsymbol{\xi},\ \boldsymbol{\xi} \geq 0$$

Returns $\boldsymbol{\xi} = \mathbf{0}$ if data linearly separable
Returns some $\xi_i > 0$ if data not linearly separable

Classification

Bad news

No, if data not linearly separable

NP-hard to solve

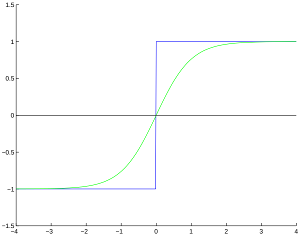
$$\min_{\mathbf{w}} \sum_{i} 1_{(\mathrm{sign}(X_{i:}\mathbf{w}-b)\neq y_i)} \qquad \text{in general}$$

NP-hard even to approximate (Höffgen et al. 1995)

How to bypass intractability of learning linear classifiers?

Two standard approaches

1. Use a matching loss to approximate sign (e.g. tanh transfer)



2. Use a surrogate loss for training, sign for test

Approximating classification with a surrogate loss

Idea

Use a different loss $\tilde{L}$ for training
than the loss L used for testing

Example

Train on $\tilde{L}(\hat{y}; y) = (\hat{y} - y)^2$
even though test on $L(\hat{y}; y) = 1_{(\hat{y} \neq y)}$

Obvious weakness

Regression losses like least squares penalize predictions
that are "too correct"

Tailored surrogate losses for classification

Margin losses

For a given target y and pre-prediction $\hat{z}$

Definition

The prediction margin is $m = \hat{z}y$

Note

if $\hat{z}y = m > 0$ then $\mathrm{sign}(\hat{z}) = y$, zero misclassification
if $\hat{z}y = m \leq 0$ then $\mathrm{sign}(\hat{z}) \neq y$, misclassification error 1

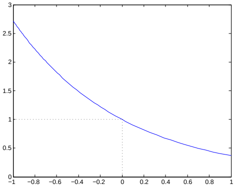
Definition

a margin loss is a decreasing (nonincreasing) function of the margin

Margin losses

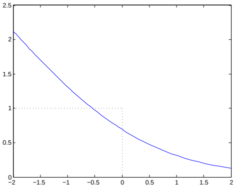
Exponential margin loss

$\tilde{L}(\hat{z}; y) = e^{-\hat{z}y}$



Binomial deviance

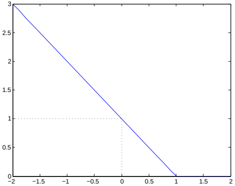
$\tilde{L}(\hat{z}; y) = \ln(1 + e^{-\hat{z}y})$



Margin losses

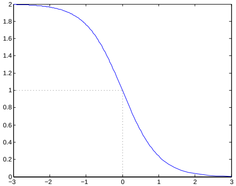
Hinge loss (support vector machines)

$$\tilde{L}(\hat{z}; y) = (1 - \hat{z}y)_+ = \max(0, 1 - \hat{z}y)$$



Robust hinge loss (intractable training)

$$\tilde{L}(\hat{z}; y) = 1 - \tanh(\hat{z}y)$$



Margin losses

Note

Convex margin loss can provide efficient upper bound minimization for misclassification error

Retain all previous extensions

- efficient training
- basis expansion
- L_2^2 regularization $\to$ kernels
- L_1 regularization $\to$ sparsity

Multivariate prediction

Multivariate prediction

What if prediction targets $\mathbf{y}'$ are **vectors**?

For linear predictors, use a weight **matrix** W

Given input $\mathbf{x}'$, predict a vector

$$\underset{1 \times k}{\hat{\mathbf{y}}'} = \underset{1 \times n}{\mathbf{x}'} \underset{n \times k}{W}$$

On training data, get prediction matrix

$$\underset{t \times k}{\hat{Y}} = \underset{t \times n}{X} \underset{n \times k}{W}$$

$W_{:j}$ is the weight vector for jth output column
$W_{i:}$ is vector of weights applied to ith feature

Try to approximate target matrix Y

Multivariate linear prediction

Need to define loss function between **vectors**

E.g. $L(\hat{\mathbf{y}}; \mathbf{y}) = \sum_\ell (\hat{y}_\ell - y_\ell)^2$

Given X, Y, compute

$$\min_W \sum_{i=1}^{t} L(X_{i:}W; Y_{i:})$$
$$= \min_W L(XW; Y)$$

Note: using shorthand $L(XW; Y) = \sum_{i=1}^{t} L(X_{i:}W; Y_{i:})$

Feature expansion

$X \mapsto \Phi$

- Doesn't change anything, can still solve same way as before
- Will just use X and Φ interchangeably from now on

Multivariate prediction

Can recover all previous developments

- efficient training
- feature expansion
- L_2^2 regularization $\rightarrow$ kernels
- L_1 regularization $\rightarrow$ sparsity
- output transformations
- matching loss
- classification—surrogate margin loss

L_2^2 regularization—kernels

$$\min_W L(XW; Y) + \frac{\beta}{2}\mathrm{tr}(W'W)$$

Still get representer theorem

Solution satisfies $W^* = X'A^*$ for some A^*

Therefore still get kernels

$$\begin{aligned}&\min_W L(XW; Y) + \frac{\beta}{2}\mathrm{tr}(W'W)\\ =\ &\min_A L(XX'A; Y) + \frac{\beta}{2}\mathrm{tr}(A'XX'A)\\ =\ &\min_A L(KA; Y) + \frac{\beta}{2}\mathrm{tr}(A'KA)\end{aligned}$$

Note

We are actually regularizing using a matrix norm

Frobenius norm $\|W\|_F^2 = \sum_{ij} W_{ij}^2 = \mathrm{tr}(W'W)$

$\|W\|_F = \sqrt{\sum_{ij} W_{ij}^2} = \sqrt{\mathrm{tr}(W'W)}$

Brief background: Recall matrix trace

Definition

For a square matrix A, $\mathrm{tr}(A) = \sum_i A_{ii}$

Properties

$$\begin{aligned}
\mathrm{tr}(A) &= \mathrm{tr}(A') \\
\mathrm{tr}(aA) &= a\mathrm{tr}(A) \\
\mathrm{tr}(A+B) &= \mathrm{tr}(A) + \mathrm{tr}(B) \\
\mathrm{tr}(A'B) &= \mathrm{tr}(B'A) = \textstyle\sum_{ij} A_{ij}B_{ij} \\
\mathrm{tr}(A'A) &= \mathrm{tr}(AA') = \textstyle\sum_{ij} A_{ij}^2 \\
\mathrm{tr}(ABC) &= \mathrm{tr}(CAB) = \mathrm{tr}(BCA) \\
\tfrac{d}{dW}\mathrm{tr}(C'W) &= C \\
\tfrac{d}{dW}\mathrm{tr}(W'AW) &= (A+A')W
\end{aligned}$$

L_1 regularization—sparsity?

We want sparsity in rows of W, not columns
(that is, we want feature selection, not output selection)
To achieve our goal need to select the right regularizer

Consider the following matrix norms

L_1 norm	$\|W\|_1 = \max_j \sum_i |W_{ij}|$
L_∞ norm	$\|W\|_\infty = \max_i \sum_j |W_{ij}|$
L_2 norm	$\|W\|_2 = \sigma_{max}(W)$ (maximum singular value)
trace norm	$\|W\|_{tr} = \sum_j \sigma_j(W)$ (sum of singular values)
$2,1$ block norm	$\|W\|_{2,1} = \sum_i \|W_{i:}\|$
Frobenius norm	$\|W\|_F = \sqrt{\sum_{ij} W_{ij}^2} = \sqrt{\sum_j \sigma_j(W)^2}$

Which, if any, of these yield the desired sparsity structure?

Matrix norm regularizers

Consider examples

$$U = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \; V = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \; W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

We want to favor a structure like U over V and W

	U	V	W
L_1 norm	1	2	1
L_∞ norm	2	1	1
L_2 norm	$\sqrt{2}$	$\sqrt{2}$	1
trace norm	$\sqrt{2}$	$\sqrt{2}$	2
$2,1$ norm	$\sqrt{2}$	2	2
Frobenius norm	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$

Use $2,1$ norm for feature selection: favors null rows

Use trace norm for subspace selection: favors lower rank

All norms are convex in W

L_1 regularization—sparsity?

To train for feature selection sparsity:

$$\min_W L(XW, Y) + \beta \|W\|_{2,1}$$

$$\text{or} \quad \min_W L(XW, Y) + \frac{\beta}{2} \|W\|_{2,1}^2$$

To train for subspace selection:

$$\min_W L(XW, Y) + \beta \|W\|_{tr}$$

$$\text{or} \quad \min_W L(XW, Y) + \frac{\beta}{2} \|W\|_{tr}^2$$

When do we still get a representer theorem?

Obvious in vector case

Regularizer R a nondecreasing function of $\|\mathbf{w}\|_2^2 = \mathbf{w}'\mathbf{w}$

But in matrix case?

Theorem (Argyriou et al. JMLR 2009)

Regularizer R yields representer theorem

iff

R is a matrix-nondecreasing function of $W'W$

That is, $R(W) = T(W'W)$ for some function T
where $T(A) \geq T(B)$ for all $A, B \in S_+$ such that $A \succeq B$

Examples

$\|W\|_F$, $\|W\|_{tr}$

Schatten p-norms: $\|W\|_p \triangleq \|\sigma(W)\|_p$

Multivariate output transformations

Multivariate output transformations

Use transformation $\mathbf{f} : \mathbb{R}^k \to \mathbb{R}^k$ to map pre-predictions into range

$$\hat{\mathbf{y}}' = \mathbf{f}(\hat{\mathbf{z}}')$$

Exponential

$\mathbf{y} \geq 0$ nonnegative, use $\mathbf{f}(\hat{\mathbf{z}}) = \exp(\hat{\mathbf{z}})$ componentwise

Softmax

$\mathbf{y} \geq 0$, $\mathbf{1}'\mathbf{y} = 1$ probability vector, use $\mathbf{f}(\hat{\mathbf{z}}) = \frac{\exp(\hat{\mathbf{z}})}{\mathbf{1}' \exp(\hat{\mathbf{z}})}$

Indmax

$\mathbf{y} = \mathbf{1}_c$ class indicator, use $\mathbf{f}(\hat{\mathbf{z}}) = \mathrm{indmax}(\hat{\mathbf{z}})$
(all 0s except 1 in position of max of $\hat{\mathbf{z}}$)

For nice output transformations can use matching loss

Choose

$F : \mathbb{R}^k \to \mathbb{R}$ such that F strongly convex and $\nabla F(\hat{\mathbf{z}}) = \mathbf{f}(\hat{\mathbf{z}})$

Then define

$$
\begin{aligned}
L(\hat{\mathbf{y}}'; \mathbf{y}') &= L(\mathbf{f}(\hat{\mathbf{z}}'); \mathbf{f}(\mathbf{z}')) \\
&= F(\hat{\mathbf{z}}) - F(\mathbf{z}) - \mathbf{f}(\mathbf{z})'(\hat{\mathbf{z}} - \mathbf{z})
\end{aligned}
$$

Recall

Since F strongly convex we have: $F(\hat{\mathbf{z}}) \geq F(\mathbf{z}) + \mathbf{f}(\mathbf{z})'(\hat{\mathbf{z}} - \mathbf{z})$
Hence $L(\hat{\mathbf{y}}'; \mathbf{y}') \geq 0$ and $L(\hat{\mathbf{y}}'; \mathbf{y}') = 0$ iff $\mathbf{f}(\hat{\mathbf{z}}) = \mathbf{f}(\mathbf{z})$

Bregman divergence on vectors

(Kivinen & Warmuth 2001)

Multivariate matching loss examples

Exponential

$F(\mathbf{z}) = \mathbf{1}' \exp(\mathbf{z})$, $\nabla F(\mathbf{z}) = \mathbf{f}(\mathbf{z}) = \exp(\mathbf{z})$ componentwise

Matching loss is unnormalized entropy

$$L(\hat{\mathbf{y}}'; \mathbf{y}') = \mathbf{y}'(\ln \mathbf{y} - \ln \hat{\mathbf{y}}) + \mathbf{1}'(\mathbf{y} - \hat{\mathbf{y}})$$

Softmax

$F(\mathbf{z}) = \ln(\mathbf{1}' \exp(\mathbf{z}))$, $\nabla F(\mathbf{z}) = \mathbf{f}(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\mathbf{1}' \exp(\mathbf{z})}$

Matching loss is cross entropy, or Kullback-Leibler divergence

$$L(\hat{\mathbf{y}}'; \mathbf{y}') = \mathbf{y}'(\ln \mathbf{y} - \ln \hat{\mathbf{y}})$$

Multivariate classification

For classification need to use a surrogate loss

$$\hat{\mathbf{y}} = \text{indmax}(\hat{\mathbf{z}}) \qquad \mathbf{y} = \mathbf{1}_c \text{ class indicator vector}$$

Multivariate margin loss

- Depends only on $\mathbf{y}'\hat{\mathbf{z}}$ and $\hat{\mathbf{z}}$

Example: multinomial deviance

$$\tilde{L}(\hat{\mathbf{z}}; \mathbf{y}) = \ln(\mathbf{1}' \exp(\hat{\mathbf{z}})) - \mathbf{y}'\hat{\mathbf{z}}$$

Example: multiclass SVM loss

$$\tilde{L}(\hat{\mathbf{z}}; \mathbf{y}) = \max(\mathbf{1} - \mathbf{y} + \hat{\mathbf{z}} - \mathbf{1}\mathbf{y}'\hat{\mathbf{z}})$$

Idea: If c correct class, try to push $\hat{z}_c > \hat{z}_{c'} + 1$ for $c' \neq c$

Multivariate classification

Example: multiclass SVM

$$\min_{W} \frac{\beta}{2}\|W\|_F^2 + \sum_{i=1}^{t} \max(\mathbf{1}' - Y_{i:} + X_{i:}W - X_{i:}WY_{i:}'\mathbf{1}')$$

$$= \min_{W,\boldsymbol{\xi}} \frac{\beta}{2}\|W\|_F^2 + \mathbf{1}'\boldsymbol{\xi}$$

$$\text{subject to } \boldsymbol{\xi}\mathbf{1}' \geq \mathbf{1}\mathbf{1}' - Y + XW - \boldsymbol{\delta}(XWY')\mathbf{1}'$$

where $\boldsymbol{\delta}$ means extracting main diagonal into a vector

Get a quadratic program

Note

Representer theorem applies because regularizing by $\|W\|_F^2$

Classification

$$\mathbf{x}' \mapsto \hat{\mathbf{y}}' = \mathrm{indmax}(\mathbf{x}'W)$$

Structured output prediction

Structured output prediction

Example: Optical character recognition

Map sequence of handwritten to recognized characters

```
The ncd qfple
Thc rcd apfle
The red apple
```

- Predicting character from handwriting is hard
- But: there are strong mutual constraints on the labels
- Idea: treat output as a joint label—try to capture constraints

Problem

Get an exponential number of joint labels

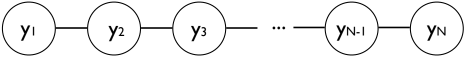
Structured output prediction

Assume structure

E.g. for output sequences assume a decomposition

$$\mathbf{w}'\phi(\mathbf{x},\mathbf{y}) = \sum_{\ell} \mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1})$$

Total response for sequence = sum of responses over local parts



Can now use a "message passing" algorithm

To efficiently compute answers for exponential sums and exponential maximizations

Computational problems

We would like to be able to efficiently compute

Sum over sequences

$$\sum_{\mathbf{y}} \exp(\mathbf{w}'\phi(\mathbf{x},\mathbf{y})) = \sum_{\mathbf{y}} \exp(\sum_{\ell} \mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1})) = \sum_{\mathbf{y}} \prod_{\ell} \exp(\mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1}))$$

Max over sequences

$$\max_{\mathbf{y}} \mathbf{w}'\phi(\mathbf{x},\mathbf{y}) = \max_{\mathbf{y}} \sum_{\ell} \mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1})$$

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} \mathbf{w}'\phi(\mathbf{x},\mathbf{y})$$

Exploit distributivity property

$$a \circ (f(x_1) * f(x_2)) = (a \circ f(x_1)) * (a \circ f(x_2))$$

sum-product:	$* = +$	$\circ = \times$
max-sum:	$* = \max$	$\circ = +$

Efficient computation

Example: max-sum

Note: $\max_x a + f(x) = a + \max_x f(x)$

Consider example

$$\max_{y_5,y_4,y_3,y_2,y_1} f_4(y_4,y_5) + f_3(y_3,y_4) + f_2(y_2,y_3) + f_1(y_1,y_2)$$

$$= \underbrace{\max_{y_5} \underbrace{\max_{y_4} f_4(y_4,y_5) + \underbrace{\max_{y_3} f_3(y_3,y_4) + \underbrace{\max_{y_2} f_2(y_2,y_3) + \underbrace{\max_{y_1} f_1(y_1,y_2)}_{m_1(y_2)}}_{m_2(y_3)}}_{m_3(y_4)}}_{m_4(y_5)}}_{m_5}$$

Reduced $O(|\mathcal{Y}|^k)$ computation to $O(k|\mathcal{Y}|^2)$

Max-sum message passing

Viterbi algorithm

$$
\begin{aligned}
m_1(y_2) &= \max_{y_1} \mathbf{w}'\psi(\mathbf{x}, y_1, y_2) \\
&\vdots \\
m_\ell(y_{\ell+1}) &= \max_{y_\ell} \mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1}) + m_{\ell-1}(y_\ell) \\
&\vdots \\
m_{k-1}(y_k) &= \max_{y_{k-1}} \mathbf{w}'\psi(\mathbf{x}, y_{k-1}, y_k) + m_{k-2}(y_{k-1}) \\
m &= \max_{y_k} m_{k-1}(y_k)
\end{aligned}
$$

Efficient computation

Example: sum-product

Note: $\sum_x af(x) = a\sum_x f(x)$

Consider example

$$\sum_{y_5,y_4,y_3,y_2,y_1} f_4(y_4,y_5)f_3(y_3,y_4)f_2(y_2,y_3)f_1(y_1,y_2)$$

$$= \underbrace{\sum_{y_5} \underbrace{\sum_{y_4} f_4(y_4,y_5) \underbrace{\sum_{y_3} f_3(y_3,y_4) \underbrace{\sum_{y_2} f_2(y_2,y_3) \underbrace{\sum_{y_1} f_1(y_1,y_2)}_{m_1(y_2)}}_{m_2(y_3)}}_{m_3(y_4)}}_{m_4(y_5)}}_{m_5}$$

Reduced $O(|\mathcal{Y}|^k)$ computation to $O(k|\mathcal{Y}|^2)$

Sum-product message passing

Forward-backward algorithm

$$m_1(y_2) = \sum_{y_1} \mathbf{w}'\psi(\mathbf{x}, y_1, y_2)$$

$$\vdots$$

$$m_\ell(y_{\ell+1}) = \sum_{y_\ell} \mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1}) m_{\ell-1}(y_\ell)$$

$$\vdots$$

$$m_{k-1}(y_k) = \sum_{y_{k-1}} \mathbf{w}'\psi(\mathbf{x}, y_{k-1}, y_k) m_{k-2}(y_{k-1})$$

$$m = \sum_{y_k} m_{k-1}(y_k)$$

Conditional random fields

$$\min_{\mathbf{w}} \sum_i \ln\left(\sum_{\tilde{\mathbf{y}}} \prod_\ell \exp(\mathbf{w}'\psi(\mathbf{x}_i, \tilde{y}_\ell, \tilde{y}_{\ell+1})\right) - \sum_\ell \mathbf{w}'\psi(\mathbf{x}_i, y_{i\ell}, y_{i\ell+1})$$

$$\frac{d}{d\mathbf{w}} = \sum_{i,\tilde{\mathbf{y}},\ell} \psi(\mathbf{x}_i, \tilde{y}_\ell, \tilde{y}_{\ell+1}) \frac{\prod_\ell \exp(\mathbf{w}'\psi(\mathbf{x}_i, \tilde{y}_\ell, \tilde{y}_{\ell+1}))}{Z(\mathbf{w}, \mathbf{x}_i)} - \psi(\mathbf{x}_i, y_{i\ell}, y_{i\ell+1})$$

where

$$Z(\mathbf{w}, \mathbf{x}_i) = \sum_{\tilde{\mathbf{y}}} \prod_\ell \exp(\mathbf{w}'\psi(\mathbf{x}_i, \tilde{y}_\ell, \tilde{y}_{\ell+1}))$$

Use the sum-product algorithm to efficiently compute $\sum_{\mathbf{y}} \prod_\ell$

Classification

$\mathbf{x}' \mapsto \hat{\mathbf{y}}' = \arg\max_{\mathbf{y}} \sum_\ell \mathbf{w}'\psi(\mathbf{x}, y_\ell, y_{\ell+1})$

(Lafferty et al. 2001)

Maximum margin Markov networks

$$\min_{\mathbf{w}} \sum_i \max_{\tilde{\mathbf{y}}} \sum_\ell \delta(y_{i\ell} y_{i\ell+1}; \tilde{y}_\ell \tilde{y}_{\ell+1}) + \mathbf{w}'(\boldsymbol{\psi}(\mathbf{x}_i, \tilde{y}_\ell \tilde{y}_{\ell+1}) - \boldsymbol{\psi}(\mathbf{x}_i, y_{i\ell} y_{i\ell+1}))$$

$$= \min_{\mathbf{w},\boldsymbol{\xi}} \mathbf{1}'\boldsymbol{\xi} \text{ s.t. } \xi_i \geq \sum_\ell \delta(y_{i\ell} y_{i\ell+1}; \tilde{y}_\ell \tilde{y}_{\ell+1}) + \mathbf{w}'(\boldsymbol{\psi}(\mathbf{x}_i, \tilde{y}_\ell \tilde{y}_{\ell+1}) - \boldsymbol{\psi}(\mathbf{x}_i, y_{i\ell} y_{i\ell+1}))$$

$$= \min_{\mathbf{w},\boldsymbol{\xi}} \mathbf{1}'\boldsymbol{\xi} \text{ s.t. } \xi_i \geq \sum_\ell C(\mathbf{w}, \mathbf{x}_i, y_{i\ell} y_{i\ell+1}, \tilde{y}_\ell \tilde{y}_{\ell+1}) \qquad \text{for all } i \text{ and } \tilde{\mathbf{y}}$$

Exponential number of constraints!
Encode messages from efficient max-sum with auxiliary variables

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},\mathbf{m}} \mathbf{1}'\boldsymbol{\xi} \text{ s.t. } & \xi_i \geq m_{ik-1}(\tilde{y}_k) \\ & m_{ik-1}(\tilde{y}_k) \geq C(\mathbf{w}, \mathbf{x}_i, y_{ik-1} y_{ik}, \tilde{y}_{k-1} \tilde{y}_k) + m_{ik-2}(\tilde{y}_{k-1}) \\ & \vdots \\ & m_{i\ell}(\tilde{y}_{\ell+1}) \geq C(\mathbf{w}, \mathbf{x}_i, y_{i\ell} y_{i\ell+1}, \tilde{y}_\ell \tilde{y}_{\ell+1}) + m_{i\ell-1}(\tilde{y}_\ell) \\ & \vdots \end{aligned}$$

Classification: same as for CRFs (Taskar et al. 2004a)

Extensions

These algorithms have been generalized to cases where:

y is a tree of fixed structure

y is a context-free parse

y is a graph matching

y is a planar graph

I.e. any structure where an efficient algorithm exists for

$$\sum_{\mathbf{y}} \prod_{\ell} \qquad \max_{\mathbf{y}} \sum_{\ell}$$

Has led to some nice advances in

natural language processing

speech processing

image processing

Conditional probability modeling

Conditional probability modeling

Up to now we have focused on point predictors

$$\hat{y} = \mathbf{x}'\mathbf{w}$$
$$\hat{y} = f(\mathbf{x}'\mathbf{w})$$
$$\hat{y} = \mathrm{sign}(\mathbf{x}'\mathbf{w})$$

Now want a conditional distribution over y given $\mathbf{x}$

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

represents a point predictor and uncertainty about the prediction

Optimal point predictor

Given $p(y|\mathbf{x})$ what is optimal point predictor?

Depends on the loss function

Example: squared error

$$L(\hat{y}; y) = (\hat{y} - y)^2$$
$$\min_{\hat{y}} E[(\hat{y} - y)^2|\mathbf{x}] = \min_{\hat{y}} \int (\hat{y} - y)^2 p(y|\mathbf{x})\, dy$$
$$\frac{d}{d\hat{y}} = 0 \Rightarrow \hat{y} = E[y|\mathbf{x}]$$

Example: matching loss

$L(\hat{y}; y) = F(f^{-1}(\hat{y})) - F(f^{-1}(y)) - y(f^{-1}(\hat{y}) - f^{-1}(y))$

Let $\bar{y} = E[y|\mathbf{x}]$ and consider

$$\begin{aligned}
&E[L(\hat{y}; y)|\mathbf{x}] - E[L(\hat{y}; y)|\mathbf{x}] \\
=&E[F(f^{-1}(\hat{y})) - F(f^{-1}(\bar{y})) - \bar{y}(f^{-1}(\hat{y}) - f^{-1}(\bar{y}))|\mathbf{x}] \\
=&L(\hat{y}; \bar{y}) \geq 0
\end{aligned}$$

Minimized by setting $\hat{y} = \bar{y} = E[y|\mathbf{x}]$

Optimal point predictor

Example: absolute error

$L(\hat{y}; y) = |\hat{y} - y|$
$\min_{\hat{y}} E[|\hat{y} - y|\,|\mathbf{x}] = \min_{\hat{y}} \int |\hat{y} - y| p(y|\mathbf{x})\, dy$
$\hat{y} =$ conditional median of y given $\mathbf{x}$
(Therefore cannot be a matching loss!)

Example: misclassification error

$L(\hat{y}; y) = 1_{(\hat{y} \neq y)}$
$\min_{\hat{y}} E[1_{(\hat{y} \neq y)}|\mathbf{x}] = \min_{\hat{y}} P(\hat{y} \neq y|\mathbf{x})$
$\hat{y} = \arg\max_y P(y|\mathbf{x})$

But with a full conditional model $p(y|\mathbf{x})$

we would also have uncertainty in the predictions
E.g. $\mathrm{Var}(y|\mathbf{x})$ or $H(y|\mathbf{x})$

Aside: Bregman divergences

Transfers and inverses

$$\mathbf{y} = f(\mathbf{z}) \qquad \mathbf{z} = f^{-1}(\mathbf{y})$$
$$\hat{\mathbf{y}} = f(\hat{\mathbf{z}}) \qquad \hat{\mathbf{z}} = f^{-1}(\hat{\mathbf{y}})$$

Convex potentials and conjugates

$$F^*(\mathbf{y}) = \sup_{\mathbf{z}} \mathbf{y}'\mathbf{z} - F(\mathbf{z}) \;\; = \mathbf{y}' f^{-1}(\mathbf{y}) - F(f^{-1}(\mathbf{y}))$$
$$F(\hat{\mathbf{z}}) = \sup_{\hat{\mathbf{y}}} \hat{\mathbf{y}}'\hat{\mathbf{z}} - F^*(\hat{\mathbf{y}}) = \hat{\mathbf{z}}' f(\hat{\mathbf{z}}) - F^*(f(\hat{\mathbf{z}}))$$

Get equivalent divergences

$$\begin{aligned} D_F(\hat{\mathbf{z}} \| \mathbf{z}) &= F(\hat{\mathbf{z}}) - F(\mathbf{z}) - f(\mathbf{z})'(\hat{\mathbf{z}} - \mathbf{z}) \\ &= F(\hat{\mathbf{z}}) - \hat{\mathbf{z}}'\mathbf{y} + F^*(\mathbf{y}) \\ &= F^*(\mathbf{y}) - F^*(\hat{\mathbf{y}}) - f^{-1}(\hat{\mathbf{y}})(\mathbf{y} - \hat{\mathbf{y}}) \;=\; D_{F^*}(\mathbf{y} \| \hat{\mathbf{y}}) \end{aligned}$$

Aside: Bregman divergences

Nonlinear predictor

$$D_{F^*}(\mathbf{y} \| f(\hat{\mathbf{z}})) = D_F(\hat{\mathbf{z}} \| f^{-1}(\mathbf{y}))$$

Linear predictor

$$D_{F^*}(\mathbf{y} \| \hat{\mathbf{y}}) = D_F(f^{-1}(\hat{\mathbf{y}}) \| f^{-1}(\mathbf{y}))$$

Exponential family model

$$p(\mathbf{y}|\hat{\mathbf{z}}) = \exp(\mathbf{y}'\hat{\mathbf{z}} - F(\hat{\mathbf{z}}))p_0(\mathbf{y})$$
$$F(\hat{\mathbf{z}}) = \log \int \exp(\mathbf{y}'\hat{\mathbf{z}})p_0(\mathbf{y})\, d\mathbf{y}$$

Note

$\int p(\mathbf{y}|\hat{\mathbf{z}})\, d\mathbf{y} = 1$ is assured by $F(\hat{\mathbf{z}})$

$F(\hat{\mathbf{z}})$ convex (log-sum-exp is convex)

$E[\mathbf{y}|\hat{\mathbf{z}}] = f(\hat{\mathbf{z}}) = \hat{\mathbf{y}}$

Connection to Bregman divergences

Recall: $D_F(\hat{\mathbf{z}}\|f^{-1}(\mathbf{y})) = F(\hat{\mathbf{z}}) - \hat{\mathbf{z}}'\mathbf{y} + F^*(\mathbf{y}) = D_{F^*}(\mathbf{y}\|f(\hat{\mathbf{z}}))$

So

$$\begin{aligned} p(\mathbf{y}|\hat{\mathbf{z}}) &= \exp(\mathbf{y}'\hat{\mathbf{z}} - F(\hat{\mathbf{z}}))p_0(\mathbf{y}) \\ &= \exp(-D_F(\hat{\mathbf{z}}\|f^{-1}(\mathbf{y})) + F^*(\mathbf{y}))p_0(\mathbf{y}) \end{aligned}$$

Bregman divergences and exponential families

Theorem

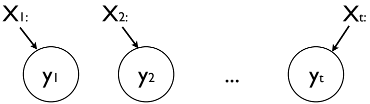
There is a bijection between regular Bregman divergences and regular exponential family models

(Banerjee et al. JMLR 2005)

Training conditional probability models

Training conditional probability models

Maximum conditional likelihood



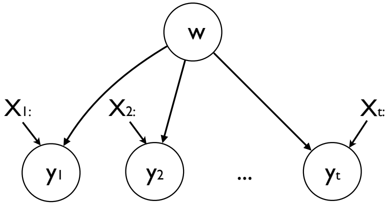
$$\max_{\mathbf{w}} \prod_{i=1}^{t} p(y_i | X_{i:} \mathbf{w})$$

$$\equiv \min_{\mathbf{w}} - \sum_{i=1}^{t} \log p(y_i | X_{i:} \mathbf{w})$$

$$= \min_{\mathbf{w}} \sum_{i=1}^{t} D_F(X_{i:} \mathbf{w} \| f^{-1}(y_i)) + \text{const}$$

Training conditional probability models

Maximum a posteriori estimation



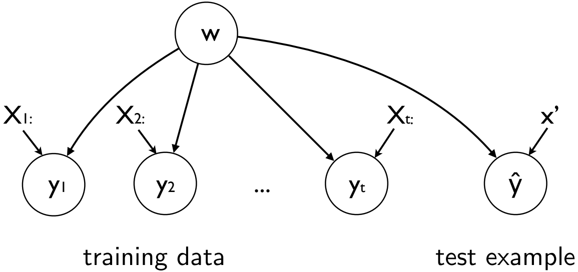
$$\max_{\mathbf{w}} p(\mathbf{w}) \prod_{i=1}^{t} p(y_i | X_{i:} \mathbf{w})$$

$$\equiv \min_{\mathbf{w}} - \log p(\mathbf{w}) - \sum_{i=1}^{t} \log p(y_i | X_{i:} \mathbf{w})$$

$$= \min_{\mathbf{w}} R(\mathbf{w}) + \sum_{i=1}^{t} D_F(X_{i:} \mathbf{w} \| f^{-1}(y_i)) + \text{const}$$

Training conditional probability models

Bayes



Do not just find single best $\mathbf{w}^*$, instead marginalize over $\mathbf{w}$

Predictive distribution

$$p(\hat{y}|\mathbf{x}', X, y)$$

Predictive distribution

$$
\begin{aligned}
p(\hat{y}|\mathbf{x}', X, y) &= \int p(\hat{y}, \mathbf{w}|\mathbf{x}', X, y)\, d\mathbf{w} \\
&= \int p(\hat{y}|\mathbf{x}'\mathbf{w}) p(\mathbf{w}|X, y)\, d\mathbf{w} \\
&= \int p(\hat{y}|\mathbf{x}'\mathbf{w}) \frac{p(\mathbf{w}) \prod_{i=1}^{t} p(y_i|X_{i:}\mathbf{w})}{\int p(\tilde{\mathbf{w}}) \prod_{i=1}^{t} p(y_i|X_{i:}\tilde{\mathbf{w}})\, d\tilde{\mathbf{w}}}\, d\mathbf{w}
\end{aligned}
$$

Bayesian model averaging

$$
\begin{aligned}
E[\hat{y}|\mathbf{x}', X, y] &= \int E[\hat{y}|\mathbf{x}'\mathbf{w}] p(\mathbf{w}|X, y)\, d\mathbf{w} \\
&= \int f(\mathbf{x}'\mathbf{w}) p(\mathbf{w}|X, y)\, d\mathbf{w}
\end{aligned}
$$

weighted average prediction

Bayesian learning

Difficulty

The integrals are usually very hard to compute

$$\int f(\mathbf{x}'\mathbf{w})p(\mathbf{w}|X,y)\,d\mathbf{w}$$

$$\int p(\tilde{\mathbf{w}})\prod_{i=1}^{t} p(y_i|X_{i:}\tilde{\mathbf{w}})\,d\tilde{\mathbf{w}}$$

Resort to MCMC techniques in general

Important special case: Gaussian process regression

Assume

$$y|\mathbf{x}'\mathbf{w} \sim N(\mathbf{x}'\mathbf{w};\, \sigma^2)$$

$$\mathbf{w} \sim N(0;\, \tfrac{\sigma^2}{\beta} I)$$

Assume $\mathbf{w}$ independent of $\mathbf{x}$, σ^2 and β known; given X, $\mathbf{y}$

Want predictive distribution: $\hat{y}|\mathbf{x}', X, \mathbf{y}$

1. Form $\left[\begin{array}{c} \mathbf{w} \\ \mathbf{y} \end{array}\right] \Big| X$ by combining $\mathbf{w}$ and $\mathbf{y}|X, \mathbf{w}$ to get joint
2. Form $\mathbf{w}|X, \mathbf{y}$ by conditioning
3. Form $\left[\begin{array}{c} \mathbf{w} \\ \hat{y} \end{array}\right] \Big| \mathbf{x}', X, \mathbf{y}$ by combining $\mathbf{w}|X, \mathbf{y}$ and $\hat{y}|\mathbf{x}', \mathbf{w}$ to get joint
4. Recover $\hat{y}|\mathbf{x}', X, \mathbf{y}$ by marginalizing

All using standard closed form operations on Gaussians

(E.g. (Rasmussen & Williams 2006))

Gaussian process regression

Get closed form for predictive distribution:

$$\hat{y}|\mathbf{x}', X, \mathbf{y} \sim N(\mathbf{x}'\mu_{\mathbf{w}};\ \sigma^2 + \mathbf{x}'\Sigma_{\mathbf{w}}\mathbf{x})$$
$$= N(\mathbf{x}'X'(K+\beta I)^{-1}\mathbf{y};\ \sigma^2 + \tfrac{\sigma^2}{\beta}\mathbf{x}'(I - X'(K+\beta I)^{-1}X)\mathbf{x})$$
$$= N(\mathbf{k}'(K+\beta I)^{-1}\mathbf{y};\ \sigma^2(1 + \tfrac{1}{\beta}\kappa - \tfrac{1}{\beta}\mathbf{k}'(K+\beta I)^{-1}\mathbf{k})$$

where $\kappa = \mathbf{x}'\mathbf{x}$, $\mathbf{k} = X\mathbf{x}$, $K = XX'$

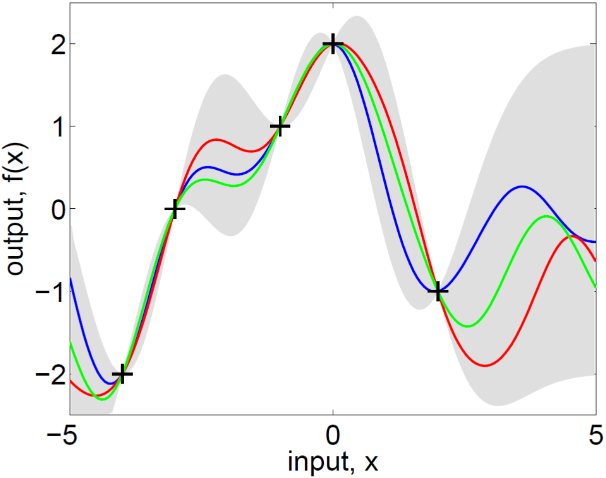
Optimal point predictor and variance

$$E[\hat{y}|\mathbf{x}', X, \mathbf{y}] = \mathbf{k}'(K+\beta I)^{-1}\mathbf{y}$$
$$\mathrm{Var}(\hat{y}|\mathbf{x}', X, \mathbf{y}) = \sigma^2(1 + \frac{1}{\beta}\kappa - \frac{1}{\beta}\mathbf{k}'(K+\beta I)^{-1}\mathbf{k})$$

Same point predictor as L_2^2 regularized least squares
But now get uncertainty in $\hat{y}$ that is affected by $\mathbf{x}'$

Gaussian process regression example

Samples from the posterior distribution



Picture is taken from Rasmussen and Williams

Part 3: Latent representations and unsupervised training

Dale Schuurmans

University of Alberta

Outline

Reverse prediction

Unsupervised training

Robust training

Latent structure training

Relaxations and global solution methods

References and Readings

Dale Schuurmans

University of Alberta

Papers

- Argyriou, A., Evgeniou, T., and Pontil, M. (2008).
  Convex multi-task feature learning.
  Machine Learning, 73(3):243–272.

- Argyriou, A., Micchelli, C., and Pontil, M. (2009).
  When is there a representer theorem? Vector versus matrix regularizers.
  Journal of Machine Learning Research, 10:2507–2529.

- Auer, P., Herbster, M., and Warmuth, M. K. (1996).
  Exponentially many local minima for single neurons.
  In *Advances in Neural Information Processing Systems 9*.

- Bach, F., Mairal, J., and Ponce, J. (2008).
  Convex sparse matrix factorizations.
  arXiv:0812.1869v1.

Papers

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005).
Clustering with Bregman divergences.
Journal of Machine Learning Research, 6:1705–1749.

Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O., and Marcotte, P. (2005).
Convex neural networks.
In *Advances in Neural Information Processing Systems 19*.

Bradley, D. and Bagnell, A. (2009).
Convex coding.
In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*.

Caetano, T., McAuley, J., Cheng, L., Le, Q., and Smola, A. (2009).
Learning graph matching.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(6):1048–1058.

Papers

Candès, E. and Recht, B. (2009).
Exact matrix completion via convex optimization.
Foundations of Computational Mathematics, 9(6):717–772.

Candes, E. and Wakin, M. (2008).
An introduction to compressive sampling.
IEEE Signal Processing Magazine, pages 21–30.

Crammer, K. and Singer, Y. (2001).
On the algorithmic implementation of multiclass kernel-based vector machines.
Journal of Maching Learning Research, 2:265–292.

Freund, Y. and Schapire, R. E. (1997).
A decision-theoretic generalization of on-line learning and an application to boosting.
Journal of Computer and System Sciences, 55(1):119–139.

Papers

Friedman, J., Hastie, T., and Tibshirani, R. (2000).
Additive logistic regression: a statistical view of boosting.
Annals of Statistics, 28(2):337–407.

Goldberg, A. B., Zhu, X., Recht, B., Xu, J.-M., and Nowak, R. (2010).
Transduction with matrix completion: Three birds with one stone.
In *Advances in Neural Information Processing Systems 23*.

Guyon, I. and Elisseeff, A. (2003).
An introduction to variable and feature selection.
Journal of Machine Learning Research, 3:1157–1182.

Hancock, T., Jiang, T., Li, M., and Tromp, J. (1996).
Lower bound on learning decision lists and trees.
Information and Computation, 126(2):114–122.

Papers

Höffgen, K.-U., Simon, H.-U., and Van Horn, K. S. (1995).
Robust trainability of single neurons.
Journal of Computer and System Science, 50(1):114–125.

Joachims, T. (2005).
A support vector method for multivariate performance measures.
In *Proceedings of the International Conference on Machine Learning (ICML)*.

Jojic, V., Saria, S., and Koller, D. (2011).
Convex envelopes of complexity controlling penalties: The case against premature envelopment.
In *Proceedings of the Conference on Artificial Intelligence and Statistics (AISTATS)*.

Papers

Kimeldorf, G. S. and Wahba, G. (1970).
A correspondence between Bayesian estimation on stochastic processes and smoothing by splines.
Annals of Mathematical Statistics, 41:495–502.

Kivinen, J. and Warmuth, M. K. (2001).
Relative loss bounds for multidimensional regression problems.
Machine Learning, 45(3):301–329.

Kloft, M., Rückert, U., and Bartlett, P. (2010).
A unifying view of multiple kernel learning.
Technical Report UCB/EECS-2010-49, EECS Department, University of California, Berkeley.

Lafferty, J. D., McCallum, A., and Pereira, F. (2001).
Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data.
In *Proceedings of the International Conference on Machine Learning (ICML)*.

Papers

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004).
Learning the kernel matrix with semi-definite programming.
Journal of Machine Learning Research, 5:27–72.

Mason, L., Baxter, J., Bartlett, P. L., and Frean, M. (2000).
Functional gradient techniques for combining hypotheses.
In *Advances in Large Margin Classifiers*, pages 221–246. MIT Press.

Neal, R. M. (1993).
Probabilistic inference using Markov chain Monte Carlo methods.
Technical report, Dept. of Computer Science, University of Toronto.
CRG-TR-93-1.

Papers

Pong, T. K., Tseng, P., Ji, S., and Ye, J. (2010).
Trace norm regularization: Reformulations, algorithms, and multi-task learning.
SIAM Journal on Optimization.

Rahimi, A. and Recht, B. (2008).
Random features for large-scale kernel machines.
In *Advances in Neural Information Processing Systems 20*.

Rahimi, A. and Recht, B. (2009).
Weighted sums of random kitchen sinks: Replacing optimization with randomization in learning.
In *Advances in Neural Information Processing Systems 21*.

Papers

Schapire, R., Freund, Y., Bartlett, P. L., and Lee, W. S. (1998).
Boosting the margin: A new explanation for the effectiveness of voting methods.
Annals of Statistics, 26:1651–1686.

Taskar, B., Guestrin, C., and Koller, D. (2004a).
Max-margin Markov networks.
In *Advances in Neural Information Processing Systems 16*.

Taskar, B., Klein, D., Collins, M., Koller, D., and Manning, C. (2004b).
Max-margin parsing.
In *Empirical Methods in Natural Language Processing (EMNLP)*.

Papers

- Tibshirani, R. (1996).
  Regression shrinkage and selection via the lasso.
  Journal of the Royal Statistical Society Series B, 58:267–288.
- Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005).
  Large margin methods for structured and interdependent output variables.
  Journal of Machine Learning Research, 6:1453–1484.
- Wainwright, M. J. and Jordan, M. I. (2003).
  Graphical models, exponential families, and variational inference.
  Technical Report 649, UC Berkeley, Department of Statistics.

Papers

Yu, Y., Yang, M., Xu, L., White, M., and Schuurmans, D. (2010).
Relaxed clipping: A global training method for robust regression and classification.
In *Advances in Neural Information Processing Systems 23*.

Zhang, X., Yu, Y., White, M., Huang, R., and Schuurmans, D. (2011).
Convex sparse coding, subspace learning, and semi-supervised extensions.
In *Proceedings of the Annual Conference on Artificial Intelligence (AAAI)*.

Zou, H. and Hastie, T. (2005).
Regularization and variable selection via the elastic net.
Journal of Royal Statistics Society B, 67(2):301–320.

Books

Bertsekas, D. P. (1995).
Nonlinear Programming.
Athena Scientific.

Bishop, C. (2006).
Pattern Recognition and Machine Learning.
Springer.

Borwein, J. M. and Lewis, A. S. (2000).
Convex Analysis and Nonlinear Optimization: Theory and Examples.
Canadian Mathematical Society.

Boyd, S. and Vandenberghe, L. (2004).
Convex Optimization.
Cambridge University Press.

Books

Duda, R. O., Hart, P. E., and Stork, D. G. (2001).
Pattern Classification and Scene Analysis.
John Wiley and Sons.
Second edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009).
The Elements of Statistical Learning.
Springer, 2nd edition.

Horn, R. A. and Johnson, C. R. (1985).
Matrix Analysis.
Cambridge University Press.

Koller, D. and Friedman, N. (2009).
Probabilistic Graphical Models: Principles and Techniques.
MIT Press.

Books

- Neal, R. (1996).
  Bayesian Learning in Neural Networks.
  Springer.
- Quinlan, J. R. (1993).
  C4.5: Programs for Machine Learning.
  Morgan Kaufmann Publishers.
- Rasmussen, C. E. and Williams, C. K. I. (2006).
  Gaussian Processes for Machine Learning.
  MIT Press.
- Rockafellar, R. T. (1970).
  Convex Analysis.
  Princeton University Press.

Books

- Schölkopf, B. and Smola, A. (2002).
  Learning with Kernels.
  MIT Press.
- Shawe-Taylor, J. and Cristianini, N. (2004).
  Kernel Methods for Pattern Analysis.
  Cambridge University Press.
- Vapnik, V. (1998).
  Statistical Learning Theory.
  John Wiley and Sons.