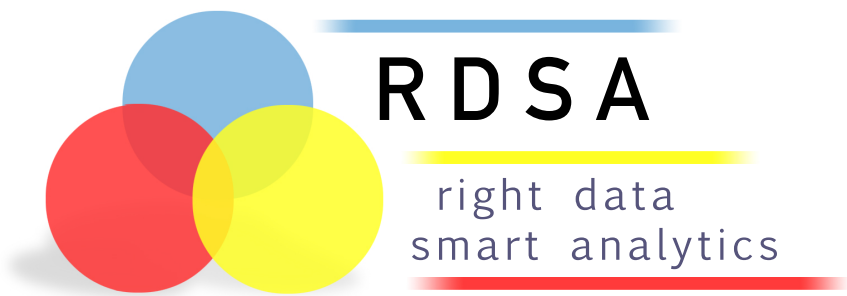
Machine Learning, Artificial Intelligence, and Precision Medicine

Haoda Fu, Ph.D.

Research Advisor
Eli Lilly and Company
Email: fu_haoda@lilly.com

Oct. 8, 2020



Outlines

1. Precision Medicine
2. Individualized Treatment Recommendation Framework
3. Support Vector Machines, Angel Based Classifiers, and Outcome Weighted Learning in Reproducing Kernel Hilbert Spaces
4. Reinforcement Learning and Multi-Stage Decision Making

Definition of Precision Medicine

Precision Medicine(Wiki)

Precision medicine is a medical model that proposes the customization of healthcare, with medical decisions, practices, and/or products being tailored to the individual patient. In this model, diagnostic testing is often employed for selecting appropriate and optimal therapies based on the context of a patient's genetic content or other molecular or cellular analysis. Tools employed in precision medicine can include molecular diagnostics, imaging, and analytics/software.

Summary

Making optimal healthcare decision for each individual patient based on this subject's context information.

Illustration Data

Table 1: An illustration dataset

ID	Y	A	X_1	X_2	X_3	$\cdots$
1	1.5	1	F	26	7.8	$\cdots$
2	1.2	2	M	28	8.2	$\cdots$
3	2.3	3	M	31	8.9	$\cdots$
4	0.9	2	F	35	9.4	$\cdots$
5	1.7	1	M	22	7.3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Research Question

Based on these data, how can we treat a new patient?
In other words, how can we learn a treatment assignment rule that, if followed by the entire population of certain patients, would lead to the best outcome on average?

Three Key Components for Precision Medicine

Context based decision learning has data in 3 components:

- $X_1, X_2, \cdots, X_p$ is context information.
- A is a context action.
- Y is a reward.

Notes:

- This data structure differs from data for typical supervised and unsupervised learning.
- Examples on common mistakes about data collection for precision medicine ...

Other Examples: Car Purchase

Table 2: My Friends' Rating of Their First Cars

ID	Satisfaction	Car Type	Gender	Age	Mileage per Day	⋯
1	90%	Focus	F	26	7.8	⋯
2	85%	Corolla	M	28	8.2	⋯
3	70%	Civic	M	31	8.9	⋯
4	75%	Corolla	F	35	9.4	⋯
5	60%	Civic	M	22	7.3	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Learning from these data, what car should I purchase?

Other Examples: Business Investment

Table 3: Previous Commercial Investments and Returns

Case ID	Return	Type	Month	Location	Share of Market	$\cdots$
1	1.2	TV	Jan	MW	12.5	$\cdots$
2	0.9	Radio	Oct	NE	18.2	$\cdots$
3	1.4	Web	Nov	WE	12.9	$\cdots$
4	1.3	Web	Dec	MW	10.4	$\cdots$
5	1.2	Radio	Feb	SE	11.3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Learning from these data, what is our best way to invest in New England area if our product has 12% market share in this March?

Other Examples: Connected Care Device

Table 4: Sending Out a Reminder at Right Time for Right Patients

ID	Cost	Send Reminder	FBG	3 Hypo	SU	$\cdots$
1	\$875	0	159	Y	Y	$\cdots$
2	\$475	0	170	Y	N	$\cdots$
3	\$150	1	160	N	N	$\cdots$
4	\$375	1	182	Y	Y	$\cdots$
5	\$525	1	110	N	Y	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Learning from these data, how can we develop a smart reminder to recommend patients to see their doctor within the next 3 weeks?

Other Examples: Choice of Digital Biomarkers

Table 5: Choose Right Digital Biomarker for Alzheimer's Disease

ID	Accuracy	Digital Biomarker	State	Age	Gender	⋯
1	70%	App No.1	Mild	63	F	⋯
2	83%	App No.2	Moderate	72	F	⋯
3	77%	App No.1	Mild	65	M	⋯
4	62%	App No.3	Severe	86	M	⋯
5	53%	App No.2	Moderate	77	F	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

Learning from these data, which is the most accurate digital biomarker that we need to choose for a new patient based on this subject's characteristics? If we can only choose one digital biomarker for patients with mild Alzheimer's Disease which one we need to utilize?

Making Optimal Decision Based on Data

Broad applications, some examples:

- Treatment selection: which treatment is the best for this patient?
- Treatment transition: should we keep using the current treatment or consider an intensification?
- Business analytic: how to invest (among a few choices) to maximize the return?
- Recommendation system: which item should a system recommend to a customer to maximize profit?

All these problems are similar in terms of data format and analytic solutions. **Essentially, we focus on a problem of making the optimal decision based on data.**

So, what is a general framework to solve this?

Reinforcement Learning Framework

Later you will see that:

- This problem is a special case in reinforcement learning framework which is different from supervised learning (e.g. classification) and unsupervised learning (e.g. clustering).
- Traditional alternatives (e.g. linear regression) are not efficient to solve these problems.
- It is connected with supervised learning methods (e.g. support vector machines).
- It can be extended to multiple stage decision making to optimize treatment sequences (e.g. dynamic treatment regimes).

Outlines

Notations

- There are N subjects from a large population.
- A_i is the treatment assignment (actions), where $i = 1, \cdots, N$.
- Y_i is the response assuming that larger Y_i is better (rewards).
- X_i is a vector of covariates.
- (Y, A, X) is the generic random variable of $\{(Y_i, A_i, X_i)\}$.
- $\mathcal{P}$ is the distribution of (Y, A, X).
- E is the expectation with respect to $\mathcal{P}$.
- Population space $\mathcal{X}$, i.e. $X_i \in \mathcal{X}$.
- $\mathcal{D}(\cdot)$ is a treatment recommendation based on covariates, i.e. $\mathcal{D}(\cdot) : \mathcal{X} \to \mathcal{A}$.
- $\mathcal{P}^{\mathcal{D}}$ is the distribution of (Y, A, X) given that $A = \mathcal{D}(X)$.

Modeling Assumptions

Assumption 1: Positivity $\exists \epsilon > 0, P\{P(A = a|X) \geq \epsilon, \forall a \in \mathcal{A}\} = 1.$

Assumption 2: Strong ignorability $\{Y^*(a) : a \in \mathcal{A}\} \perp A|X.$

Assumption 3: SUTVA $Y = \sum_{a=1}^{k} Y^*(a) I(A = a).$

SUTVA: Stable Unit Treatment Value Assumption. $Y^*(a)$ is the potential outcome if patient X takes treatment a.

Value Function

Define,

$$E^{\mathcal{D}}(Y) = \int Y d\mathcal{P}^{\mathcal{D}} = \int Y \frac{d\mathcal{P}^{\mathcal{D}}}{d\mathcal{P}} d\mathcal{P} = E\left[\frac{I\{A = \mathcal{D}(X)\}}{p(A|X)} Y\right],$$

where we use the fact that,

$$\frac{d\mathcal{P}^{\mathcal{D}}}{d\mathcal{P}} = \frac{p(y|x,a) I\{a = \mathcal{D}(x)\} p(x)}{p(y|x,a)p(a|x)p(x)} = \frac{I\{a = \mathcal{D}(x)\}}{p(a|x)}.$$

Our objective is to find $\mathcal{D}(\cdot)$ to maximize the following value function:

Value function

$$\mathcal{D}_o \in \underset{\mathcal{D} \in R}{\operatorname{argmax}} E^{\mathcal{D}}(Y) = E\left[\frac{I\{A = \mathcal{D}(X)\}}{p(A|X)} Y\right], \quad (1)$$

where R is a space of possible treatment recommendations.

Advantages of This Framework

- Y is able to handle binary, continuous, time to event data type.
- A is able to handle multiple treatments.
- X is able to incorporate variety of variables. For example, if X includes study ID, the framework can be used for meta analysis.
- $P(A|X)$ allows treatment assignments depending on covariates. So it can handle both randomized control trials and observational studies.
- It has an objective function to evaluate different treatment assignments.

An Example to Build Intuition

Table 6: Example Data

ID	Y	A	X	$P(A\|X)$
1	1	1	1	0.5
2	2	1	2	0.5
3	3	1	3	0.5
4	4	1	4	0.5
5	5	1	5	0.5
6	3	2	1	0.5
7	3	2	2	0.5
8	3	2	3	0.5
9	3	2	4	0.5
10	3	2	5	0.5

Questions to think about: why is $P(A|X) = 0.5$? what do the responses look like?

Which Doctor is Better

Suppose we have two doctors and each of them has a treatment rule. Which doctor is a better one?

- Doctor Adam: give patients treatment 1 if $X \geq 2$, and treatment 2 otherwise, denoted as $\mathcal{D}_A(X)$.
- Doctor Barry: give patients treatment 1 if $X \geq 3$, and treatment 2 otherwise, denoted as $\mathcal{D}_B(X)$.

Example Continued

Table 7: Calculation Based on Table 6

ID	Y	A	X	$P(A\|X)$	$\mathcal{D}_A$	$\mathcal{D}_B$	$\mathcal{D}_A = A$	$\mathcal{D}_B = A$
1	1	1	1	0.5	2	2	0	0
2	2	1	2	0.5	1	2	1	0
3	3	1	3	0.5	1	1	1	1
4	4	1	4	0.5	1	1	1	1
5	5	1	5	0.5	1	1	1	1
6	3	2	1	0.5	2	2	1	1
7	3	2	2	0.5	1	2	0	1
8	3	2	3	0.5	1	1	0	0
9	3	2	4	0.5	1	1	0	0
10	3	2	5	0.5	1	1	0	0

Example Continued

Doctor Adam:

$$E^{\mathcal{D}_A}(Y) = \frac{1}{10}\left(\frac{0}{0.5}\times 1+\frac{1}{0.5}\times 2+\frac{1}{0.5}\times 3+\frac{1}{0.5}\times 4+\frac{1}{0.5}\times 5+\frac{1}{0.5}\times 3+\frac{0}{0.5}\times 3+\frac{0}{0.5}\times 3+\frac{0}{0.5}\times 3+\frac{0}{0.5}\times 3\right)$$

$$= 3.4$$

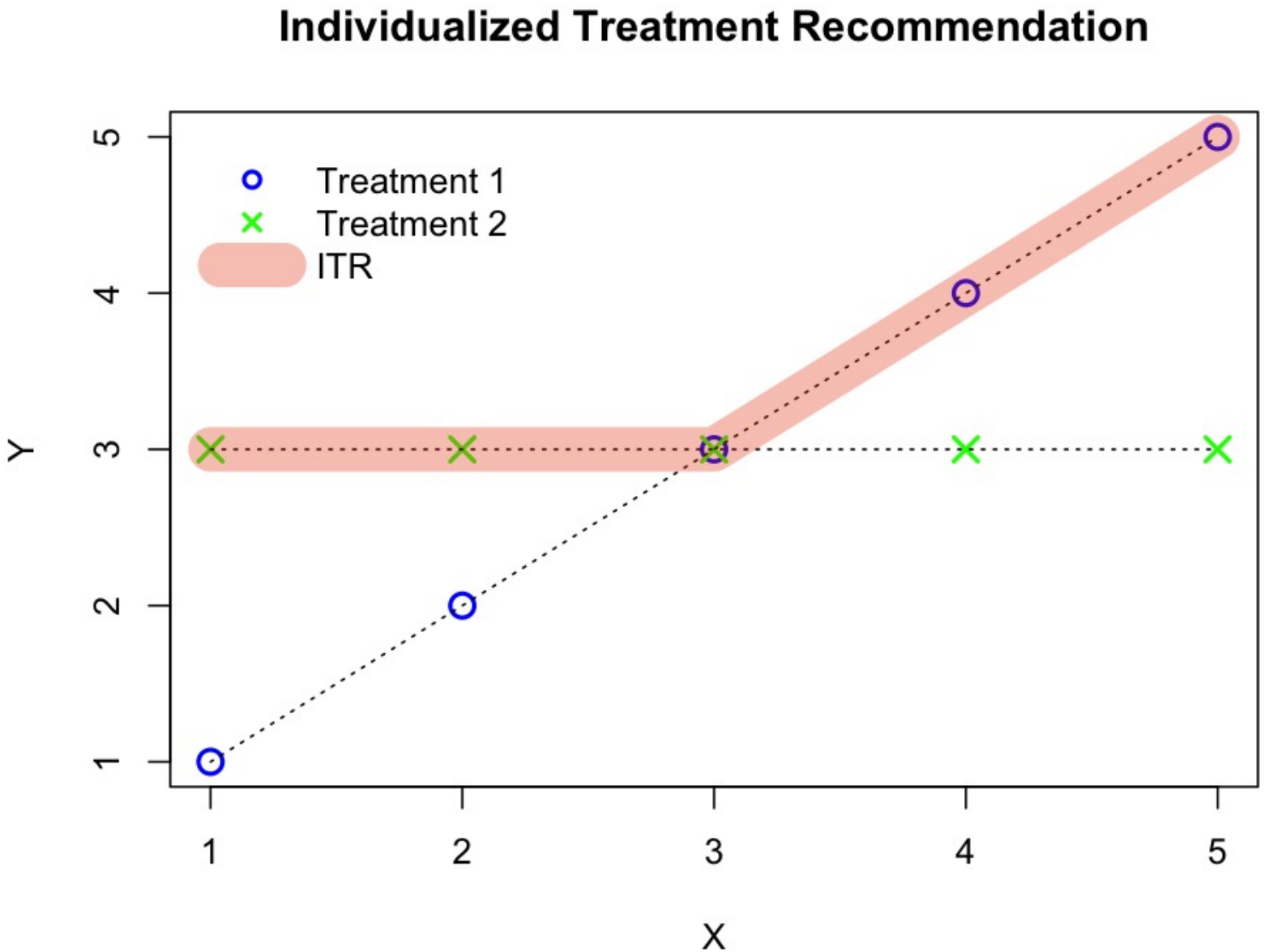
Doctor Barry:

$$E^{\mathcal{D}_B}(Y) = \frac{1}{10}\left(\frac{0}{0.5}\times 1+\frac{0}{0.5}\times 2+\frac{1}{0.5}\times 3+\frac{1}{0.5}\times 4+\frac{1}{0.5}\times 5+\frac{1}{0.5}\times 3+\frac{1}{0.5}\times 3+\frac{0}{0.5}\times 3+\frac{0}{0.5}\times 3+\frac{0}{0.5}\times 3\right)$$

$$= 3.6$$

Conclusion: Doctor Barry's rule is better than Doctor Adam's. Can we improve Doctor Barry's rule? How can we find the best rule?

Graphic Illustration



Thought Provoking Questions

- Both treatment 1 and treatment 2 have an average treatment effect as 3.0. But ITR generates average benefit value 3.6. Can algorithm beat a new molecule entity?

Thought Provoking Questions

- Both treatment 1 and treatment 2 have an average treatment effect as 3.0. But ITR generates average benefit value 3.6. Can algorithm beat a new molecule entity?
- Treatment 1 should not be only better than treatment 2. It has to be better with a non-trivial benefit margin. How can we handle this case?

Thought Provoking Questions

- Both treatment 1 and treatment 2 have an average treatment effect as 3.0. But ITR generates average benefit value 3.6. Can algorithm beat a new molecule entity?
- Treatment 1 should not be only better than treatment 2. It has to be better with a non-trivial benefit margin. How can we handle this case?
- What if the treatment randomization ratio is not 1:1?

Thought Provoking Questions

- Both treatment 1 and treatment 2 have an average treatment effect as 3.0. But ITR generates average benefit value 3.6. Can algorithm beat a new molecule entity?
- Treatment 1 should not be only better than treatment 2. It has to be better with a non-trivial benefit margin. How can we handle this case?
- What if the treatment randomization ratio is not 1:1?
- What if we have multiple covariates? The rule can be complicated.
- What if we have multiple treatments?

Analysis results: how ITR creates more value.

This data analysis shows how ITR creates additional value for patients. We have 1978 patients from two treatment arms, and 2 important biomarkers are selected from 35 biomarkers.

Table 8: *HbA1c Reduction Before and After Following ITR.* Patients with baseline fasting insulin $\geq$ 61.12pmol/L and baseline HbA1c $\geq$ 8.1% (A_o^1) are recommended to take Pioglitazone, otherwise (A_o^0) patients are recommended to take Gliclazide. After following ITR, the overall HbA1c reduction changes from -1.287% to -1.473%. Notes: ITR is our proposed method which is referred to as Individualized Treatment Recommendation.

Original			Follow ITR		
-1.287			-1.473		
	Gliclazide	Pioglitazone		Gliclazide	Pioglitazone
Mean	-1.271	-1.303	A_o^1	-1.394	-1.864
			A_o^0	-1.19	-0.932

Research Article

Statistics in Medicine

Received 25 November 2014, Accepted 1 February 2016 Published online in Wiley Online Library

(wileyonlinelibrary.com) DOI: 10.1002/sim.6920

Estimating optimal treatment regimes via subgroup identification in randomized control trials and observational studies

Haoda Fu,[a*†] Jin Zhou[b] and Douglas E. Faries[a]

Note: Illustration Codes Are Based on This Paper.

To be completed ...

- write down data generation model to generate training data sets.
- write down a Rcpp package for students to install and try.
- generate the results.

Key Insights on Solving ITR

Three connections:

1. Maximization and minimization of the value function.
2. Classification and loss functions.
3. ITR and weighted classifications.

From Maximization to Minimization

Original objective function

$$\mathcal{D}_o \in \underset{\mathcal{D}\in R}{\operatorname{argmax}}\, E^{\mathcal{D}}(Y) = E\left[\frac{I\{A=\mathcal{D}(X)\}}{p(A|X)}Y\right]. \tag{2}$$

Making connections:

$$E\left\{\frac{Y}{p(A|X)}\right\} - E\left[\frac{I\{A=\mathcal{D}(X)\}}{p(A|X)}Y\right] = E\left[\frac{I\{A\neq\mathcal{D}(X)\}}{p(A|X)}Y\right],$$

New objective function

$$\mathcal{D}_o \in \underset{\mathcal{D}\in R}{\operatorname{argmin}}\, E^{\mathcal{D}}(Y) = E\left[\frac{I\{A\neq\mathcal{D}(X)\}}{p(A|X)}Y\right]. \tag{3}$$

Empirical Evaluation

Objective function

$$\mathcal{D}_o \in \underset{\mathcal{D}\in R}{\operatorname{argmin}}\, E^{\mathcal{D}}(Y) = E\left[\frac{I\{A \neq \mathcal{D}(X)\}}{p(A|X)} Y\right].$$

When we have data, we can evaluate the objective function as,

Empirical evaluation

$$D_o = \underset{D\in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{Y_i}{p(A_i|X_i)} I\{A_i \neq \mathcal{D}(X_i)\}. \tag{4}$$

Classification Problems

A classification problem is to train a rule $\mathcal{D}(X)$ on a dataset to predict new subject membership. A simple dataset can be as below,

Table 9: An illustration dataset

ID	A	X_1	X_2	X_3	$\cdots$
1	1	F	26	7.8	$\cdots$
2	2	M	28	8.2	$\cdots$
3	1	M	31	8.9	$\cdots$
4	3	F	35	9.4	$\cdots$
5	1	M	22	7.3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Classification and Loss Function

Roughly speaking, A good classifier has smaller errors (we will discuss regularization later).

Classification objective function

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

Classification and Loss Function

Roughly speaking, A good classifier has smaller errors (we will discuss regularization later).

Classification objective function

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

If we compare our ITR objective function as below,

ITR objective function

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{Y_i}{p(A_i|X_i)} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

Important Implications and Next Steps

- We can solve the original reinforcement learning problem (ITR) as a weighted supervised learning problems.
- There are vast amount of methods and literatures on solving classification problems, in particular for binary classifications.
- With some modifications, we can leverage these existing algorithms to develop our ITR algorithms.
- In the next section, we will focus on support vector machines (SVM) theories and implementations for binary classification and binary treatment ITR, and extends it to multicategory ITR through angle based classifiers.

Outlines

Mathematical Optimization

- Constrained optimization has the form

$$\begin{aligned} \text{minimize} \quad & Q(\theta) \\ \text{subject to} \quad & \theta \in \mathcal{S} \subset \mathbb{R}^d \end{aligned}$$

- $\theta = (\theta_1, \theta_2, \cdots, \theta_d)$: optimization variables.
- $Q(\theta) : \mathbb{R}^d \to \mathbb{R}$: objective function.
- $\mathcal{S}$: feasible set.
- θ^*: optimal solution which has the smallest value of $Q(\theta)$ among all vectors that are in the feasible set $\mathcal{S}$.
- Convex optimization: both objective function and feasible set are convex.

Equality Constrained Minimization

- Consider

$$\begin{aligned} \text{minimize} \quad & Q(\theta) \\ \text{subject to} \quad & R(\theta) = 0 \end{aligned}$$

- $\mathcal{S} = \{\theta : R(\theta) = 0\}$ is a $(d-1)$-dimensional surface in $\mathbb{R}^d$.
- For every θ such that $R(\theta) = 0$, $\nabla R(\theta)$ is orthogonal to the surface.
- If θ^* is a local minimum, then ∇Q is orthogonal to the surface at θ^*.

Lagrange Multiplier

- Conclusion: at a local minimum, there exists $\lambda \in \mathbb{R}$ such that

$$\nabla Q(\theta^*) = \lambda \nabla R(\theta^*)$$

- This leads us to introduce the Lagrangian

$$L(\theta, \lambda) = Q(\theta) + \lambda R(\theta)$$

  where λ is the Lagrange multiplier.
- We have argued that a local minimum corresponds to a stationary point of the Lagrangian. Furthermore, we can reverse our logic to deduce that a stationary point of the Lagrangian is a local optimum.

Inequality Constrained Minimization

Now consider the (primal) problem

$$\begin{array}{ll} \text{minimize} & Q(\theta) \\ \text{subject to} & R(\theta) \preceq 0 \end{array}$$

Suppose θ^* is a local minimum. There are two cases:

- Inactive constraint: $R(\theta^*) \preceq 0 \Rightarrow \nabla Q(\theta^*) = 0 \Rightarrow$ stationary point of $L(\theta, \lambda)$ with $\lambda = 0$
- Active constraint: $R(\theta^*) = 0 \Rightarrow$ same as equality constraint except we require $\lambda > 0$.

Karush-Kuhn-Tucker Conditions

In either case, we have $\lambda R(\theta^*) = 0$. Therefore, a local minimum satisfies (Karush-Kuhn-Tucker conditions)

$$\begin{aligned} \nabla L(\theta^*) &= \nabla Q(\theta^*) + \lambda \nabla R(\theta^*) = 0 \\ R(\theta^*) &\preceq 0 \\ \lambda R(\theta^*) &= 0 \\ \lambda &\geq 0. \end{aligned}$$

- Often the KKT conditions may be used to transform the primal problem to an equivalent dual problem, where the variables being optimized are the Lagrange multipliers.
- Reference: Boyd and Vandenberghe (2009) Convex Optimization.

Classification

- Observe a collection of i.i.d. training data $(X_1, a_1), (X_2, a_2), \ldots, (X_n, a_n)$ from $\mathcal{P}$.
- Covariates (inputs, features, prediction variables): $X_i = (X_{i1}, \ldots, X_{ip})$
- Response variable (class label, output):

$$a_i \in \{c_1, c_2, \ldots, c_K\}$$

.

- We want to build a model $\mathcal{D}(X)$ (using the training data), so that when seeing a new input vector X, we can predict the output $\widehat{a}$.

Classification Errors and Loss Function

- Loss function (0/1):

$$L\{A, \mathcal{D}(X)\} = \begin{cases} 0 & \text{if } A = \mathcal{D}(X) \\ 1 & \text{if } A \neq \mathcal{D}(X) \end{cases}$$

- Misclassification error

$$\begin{aligned} R(\mathcal{D}) &= E_{\mathcal{P}} L\{A, \mathcal{D}(X)\} \\ &= P_{\mathcal{P}}[I\{A \neq \mathcal{D}(X)\}]. \end{aligned}$$

- For binary class case, Bayes optimal classifier ($A \in \{-1, 1\}$):

$$\begin{aligned} \mathcal{D}^*(X_i) &= \underset{\mathcal{D}}{\text{argmin}}\, R(\mathcal{D}) \\ &= \text{sign}\left\{\Pr(A = 1 | X = X_i) - \Pr(A = -1 | X = X_i)\right\}. \end{aligned}$$

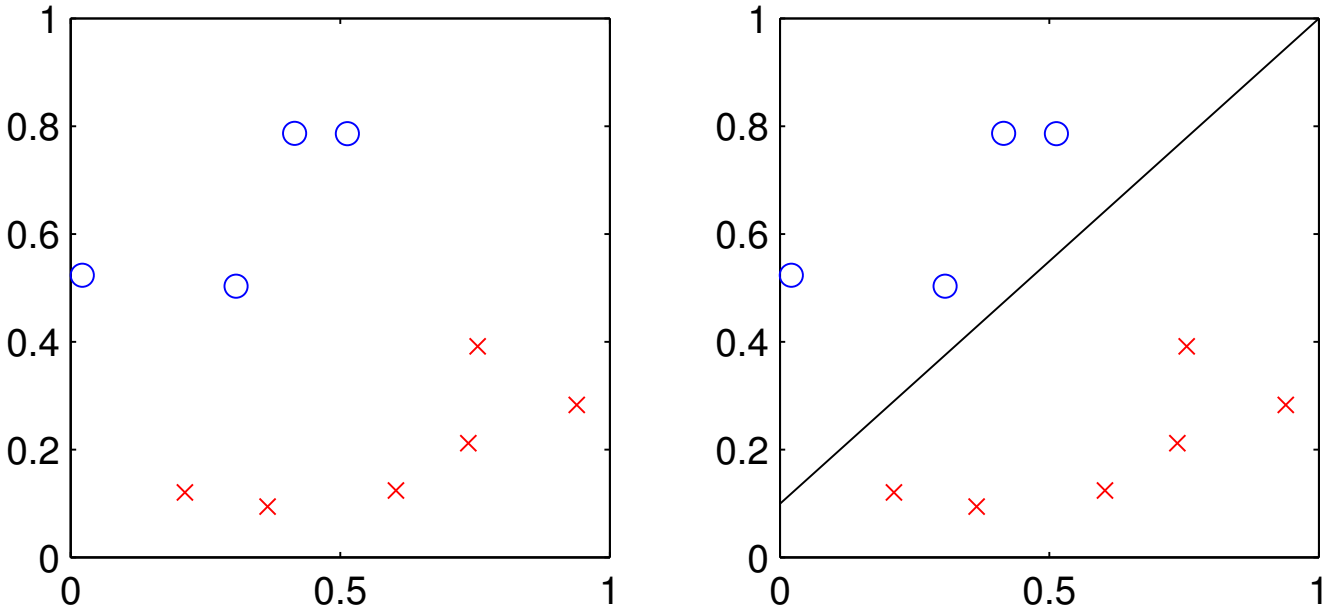
- Bayes error: $R(\mathcal{D}^*)$.

Binary Large-Margin Classifier

- $a \in \{\pm 1\}$;
  Estimate $f(X)$ with classification rule $\mathrm{sign}\{f(X)\} : \mathbb{R}^d \to \{\pm 1\}$,
  $\widehat{a} = +1$ if $f(X) \geq 0$ and $\widehat{a} = -1$ if $f(X) < 0$.
- $A_i f(X_i)$: functional margin.
- Correction classification if $A_i f(X_i) > 0$.
- The 0–1 loss: $I\{A_i f(X_i) \leq 0\}$.

Support Vector Machine (SVM)

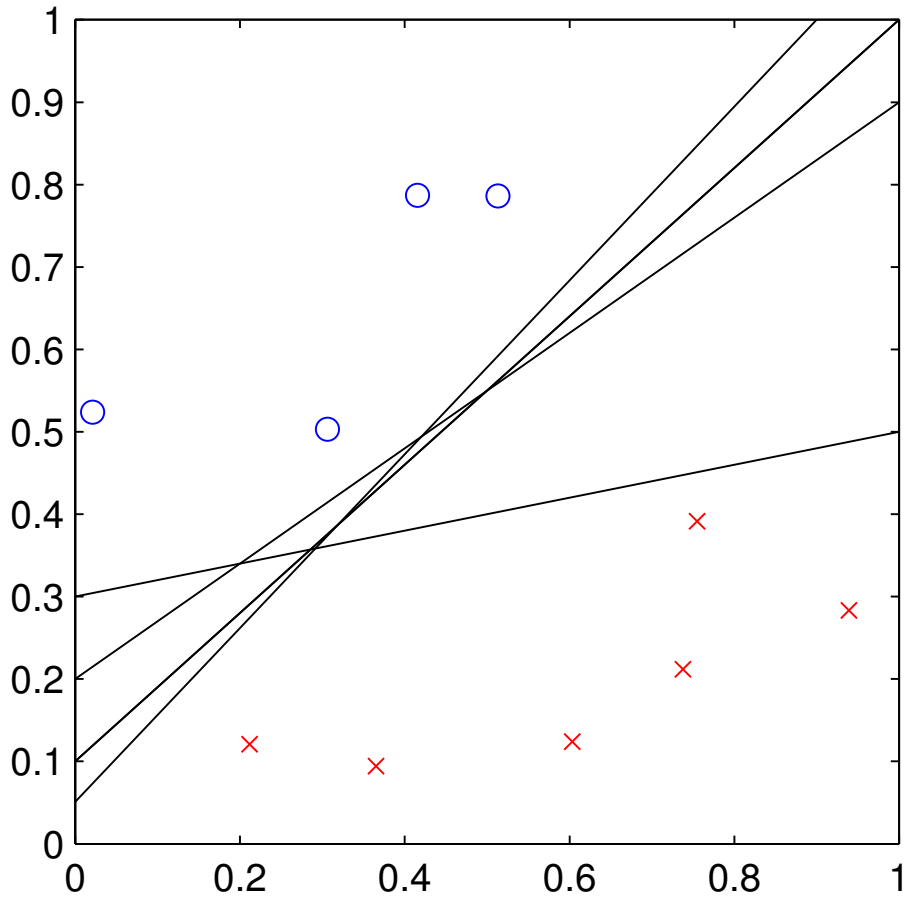
Linearly separable: Find $f(X) = \beta_0 + X^\top \beta$ to separate two groups of points.



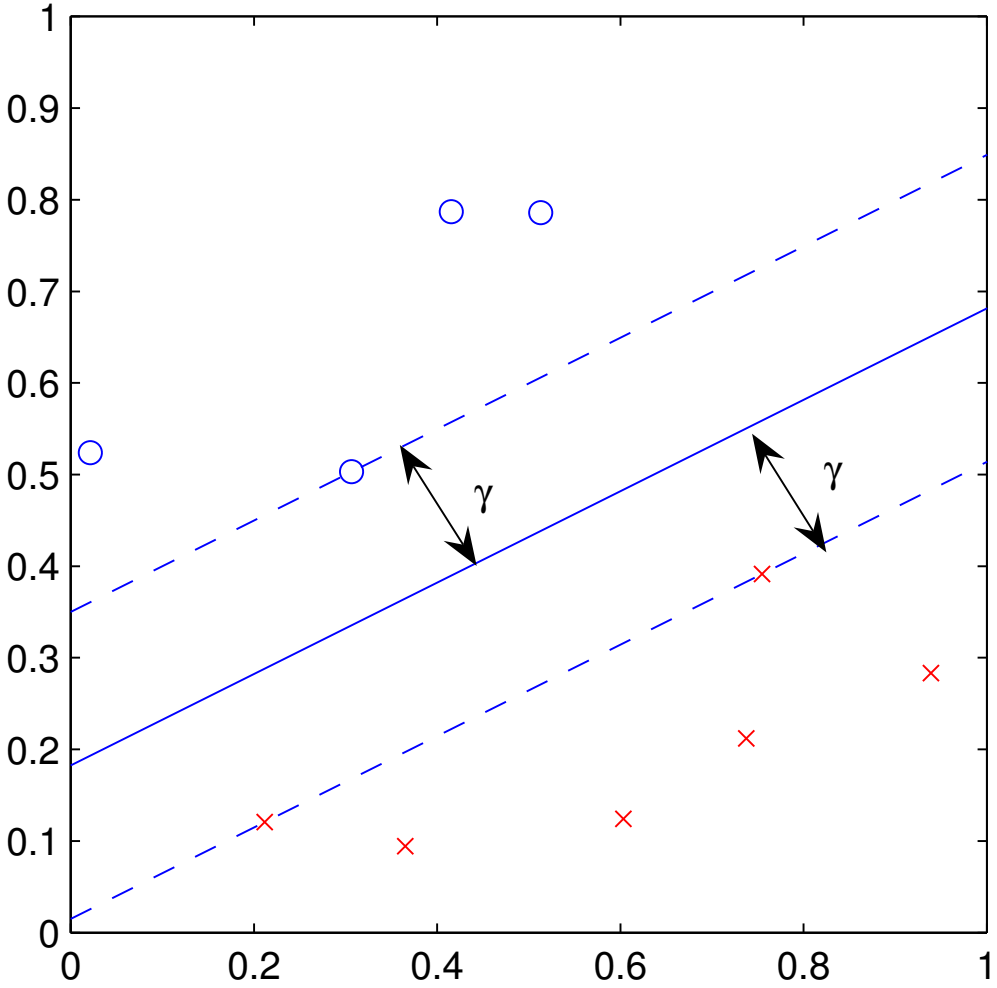
Note:

- red cross $\longleftrightarrow +1$; blue circle $\longleftrightarrow -1$.

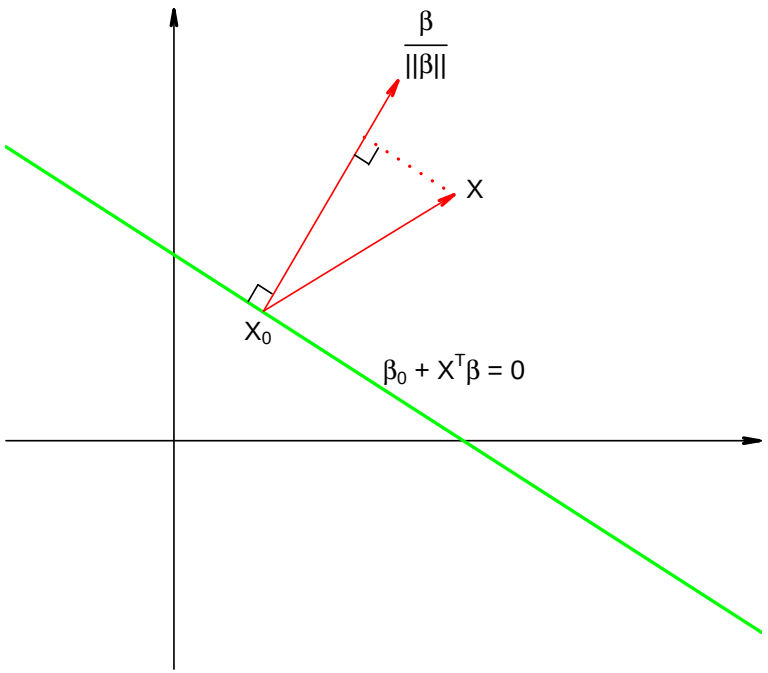
Which one is the best?



SVM: Maximum Separation



Signed Distance to Hyperplanes



- Hyperplane is defined by $\{X : \beta_0 + X^\top \beta = 0\}$.
- For any point X_0 in the hyperplane, $X_0^\top \beta = -\beta_0$.
- Signed distance of point X to the plane is $\left\langle \frac{\beta}{\|\beta\|}, X - X_0 \right\rangle$, where X_0 is any point in the plane.

Properties of Hyperplane

The affine set $L : \{X | f(X) = \beta_0 + \beta^\top X = 0\}$

- The normal vector of L is $\beta^* = \beta / \|\beta\|$.
- For any $X_0 \in L$, we have,

$$\beta^\top X_0 = -\beta_0.$$

- The signed distance of any X to L is

$$\beta^{*\top}(X - X_0) = \frac{f(X)}{\|\beta\|}.$$

Thus $f(X)$ is proportional to the signed distance from X to L.

Maximum Margin Classifier

Goal: Separate two classes and maximizes the distance to the closest points from either class (Vapnik 1996)

- Unique solutions
- Better classification performance on the training data

$$\begin{aligned} \underset{\beta,\beta_0}{\text{maximize}} \quad & \gamma \\ \text{subject to} \quad & A_i(\beta_0 + X_i^\top \beta) \geq \gamma, \\ & \|\beta\| = 1. \end{aligned}$$

All the points are at least a signed distance γ from the decision boundary

- Maximize the minimum distance
- Need constraint $\|\beta\| = 1$

Equivalent Problem

Try to get rid of the constraint $\|\beta\| = 1$

$$\frac{1}{\|\beta\|} A_i(X_i^\top \beta + \beta_0) \geq \gamma,$$

or equivalently

$$A_i(X_i^\top \beta + \beta_0) \geq \gamma \|\beta\|$$

Any positively scaled (β, β_0) also satisfies this inequality. We set $\|\beta\| = \frac{1}{\gamma}$. Then the objective function $\gamma = 1/\|\beta\|$, and

$$\begin{aligned} \underset{\beta, \beta_0}{\text{minimize}} \quad & \frac{1}{2}\|\beta\| \\ \text{subject to} \quad & A_i(X_i^\top \beta + \beta_0) \geq 1, \quad \forall i = 1, ..., n. \end{aligned}$$

Linear SVM for perfectly separable cases.
Note: by definition $1/\|\beta\|$ is the width of margin.

Maximal Margin Classifier

Geometrical **Margin**:
Defined as $d_+ + d_-$ where $d_+(d_-)$ is the shortest distance from the separating hyperplane to the closest positive (negative) training data point.

- Margin is bounded below by $\frac{2}{\|\beta\|}$.
- Use squared margin for computation convenience.
- A large margin on the training data will lead to good separation on the test data.

Defined validly only for separable cases.

Optimal Hyperplane of SVM

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\beta\|^2 \\ \text{subject to} \quad & A_i(\langle \beta, X_i \rangle + \beta_0) - 1 \geq 0, \quad \forall i = 1, 2, \cdots, n. \end{aligned}$$

- Lagrange function is :

$$L_P(\beta, \beta_0, \alpha) = \frac{1}{2}\|\beta\|^2 - \sum_{i=1}^{n} \alpha_i \{A_i(\langle \beta, X_i \rangle + \beta_0) - 1\}$$

- For any fixed α:

$$\begin{cases} \frac{\partial L(\beta, \beta_0, \alpha)}{\partial \beta_j} = 0, \quad j = 1, 2, \cdots, p \\ \frac{\partial L(\beta, \beta_0, \alpha)}{\partial \beta_0} = 0 \end{cases} \Longrightarrow \begin{cases} \beta = \sum_{i=1}^{n} \alpha_i A_i X_i \\ 0 = \sum_{i=1}^{n} \alpha_i A_i \end{cases}$$

The Dual Problem

$$\begin{aligned} \text{maximize} \quad & L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j A_i A_j \langle X_i, X_j \rangle \\ \text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, 2, \cdots, n \\ & \sum_{i=1}^{n} \alpha_i A_i = 0. \end{aligned}$$

This optimization is a quadratic programming problem and can be solved using classical optimization software. We are going to provide details on implementation in the R hands on example.

Primal vs. Dual

- Minimize L_P with respect to primal variables β_0, β
- Maximize L_D with respect to dual variables α_i
- Maximizing the dual is often a simpler convex QP than the primal, in particular when $p \gg n$.

Recovering the Optimal Hyperplane

- The optimizer of the dual: α^*
- β^* is given by:

$$\beta^* = \sum_{i=1}^{n} \alpha_i^* A_i X_i.$$

- β_0^* ???
- Decision function:

$$f(\mathbf{x}) = \langle \beta^*, X \rangle + \beta_0^*.$$

- Classification rule:

$$\text{sign}\{f(X)\}.$$

Support Vectors

The KKT conditions imply,

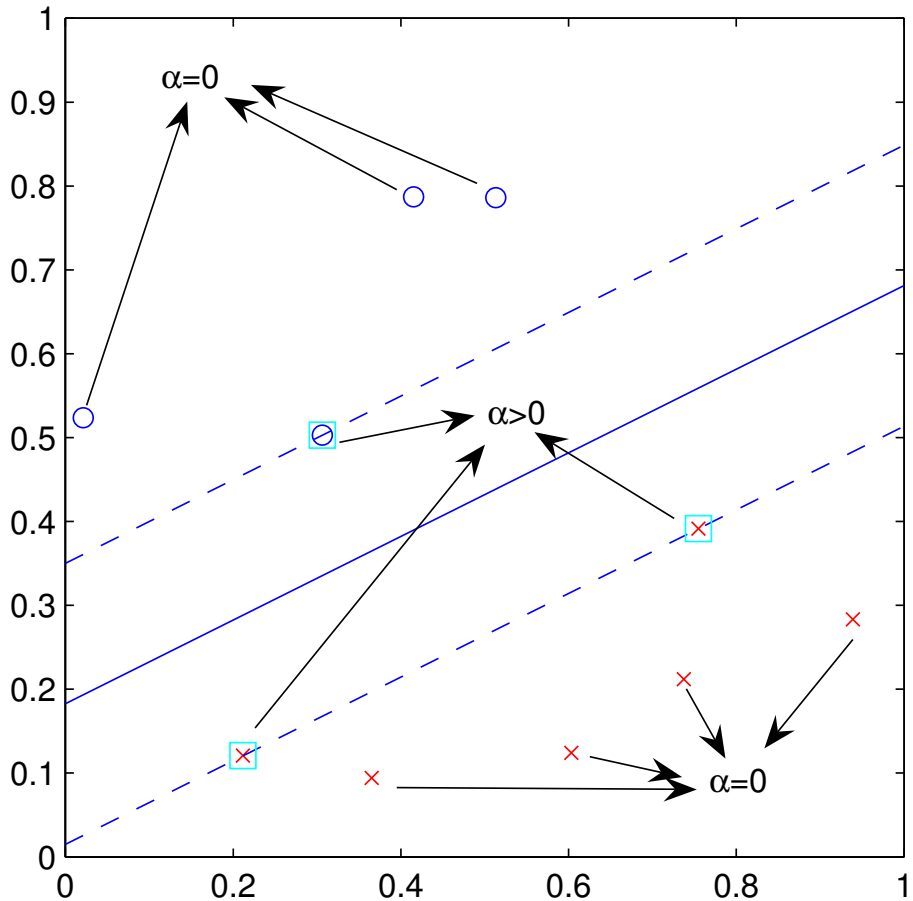
$$\alpha_i^* \left\{ A_i(\beta_0^* + X_i^\top \beta^*) - 1 \right\} = 0.$$

These imply

- If $A_i f^*(X_i) > 1$, then $\alpha_i^* = 0$.
- If $\alpha_i^* > 0$, then $A_i f^*(X_i) = 1$, or in other words, X_i is on the boundary of the "slab".
- The solution β^* is defined in terms of a linear combination of the support points.

Geometric Interpretation: Support Vectors

The i-th point is called a support vector if $\alpha_i > 0$



The i-th point is a support vector $\Longrightarrow A_i(\langle \beta^*, X_i \rangle + \beta_0) = 1 \Longrightarrow \beta_0^* = \ldots$

Comments on Various Classifiers

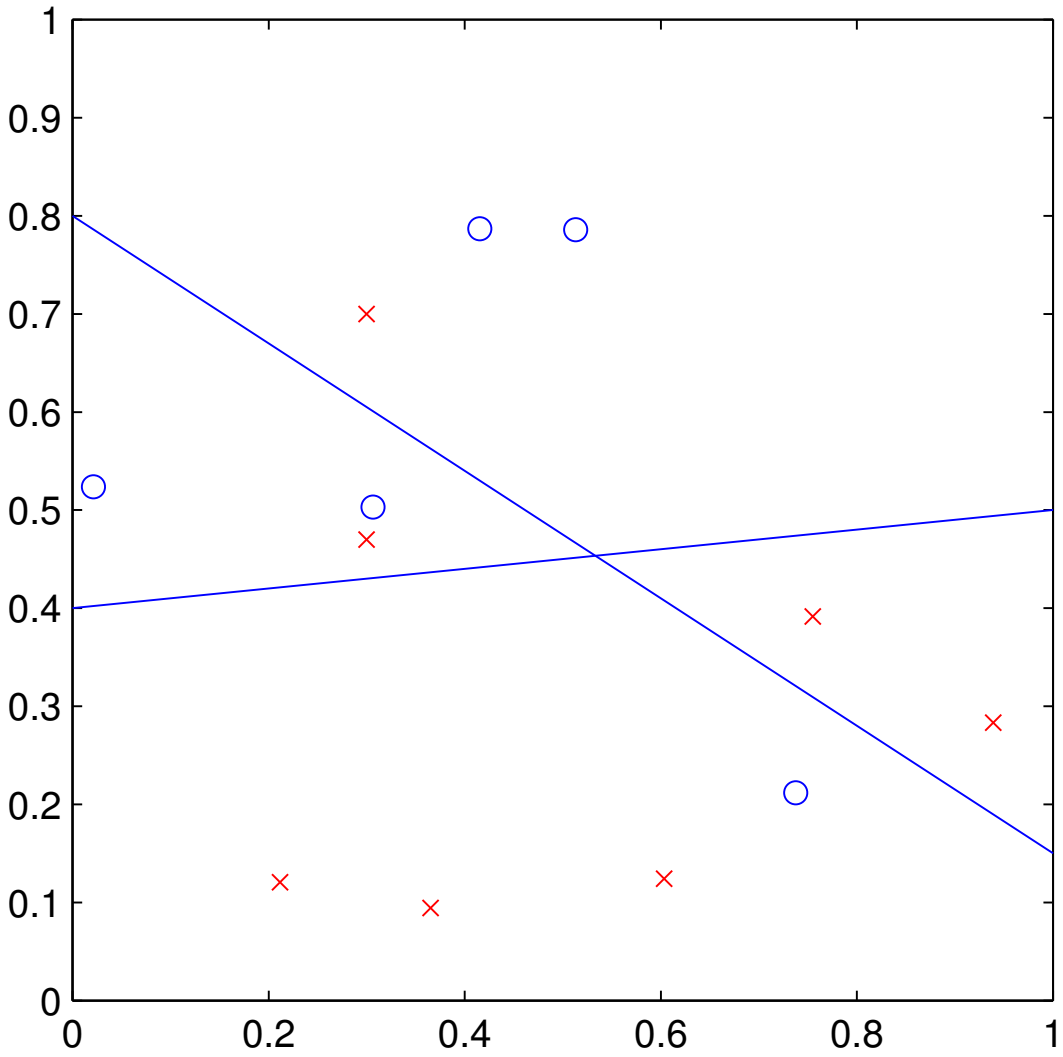
If the classes are really Gaussian, then

- the LDA is optimal.
- the separating hyperplane pays a price for focusing on the (noisier) data at the boundaries

Optimal separating hyperplane has less assumptions, thus more robust to model misspecification.

- The logistic regression solution can be similar to the separating hyperplane solution.
- For perfectly separable case, the likelihood solution can be infinity.

Linearly Non-Separable Case



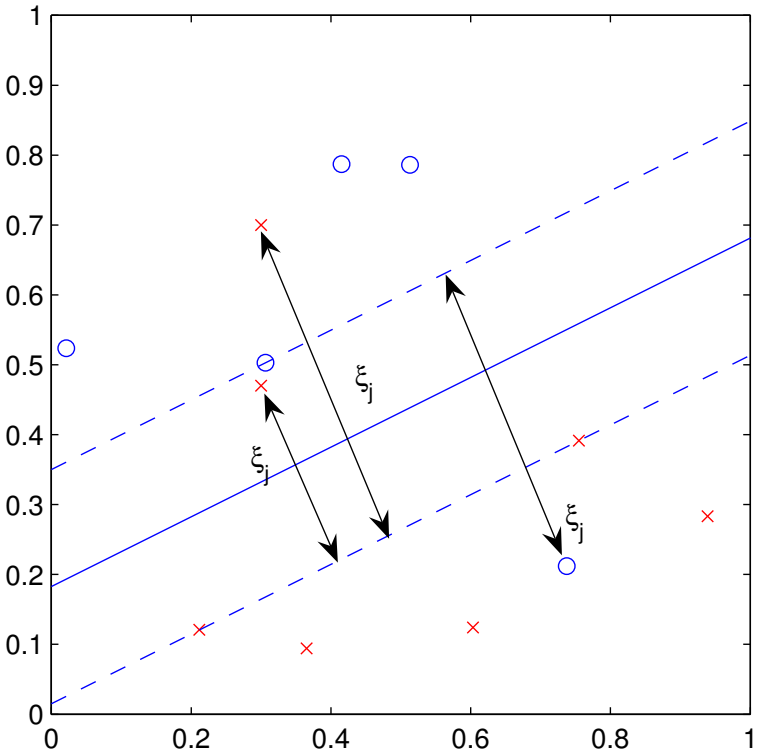
General Case for SVM

- Nonseparable: "zero"-error not attainable $\rightarrow$ "slack variables" $\{\xi_i\}_{i=1}^n$

$$\begin{aligned} \underset{\beta,\beta_0,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & A_i f(X_i) \geq (1-\xi_i), \quad i = 1,\ldots,n, \\ & \xi_i \geq 0, \quad i = 1,\ldots,n, \end{aligned}$$

where $C > 0$ is a tuning parameter.

Geometric Interpretation: Slack Variables



Slack variables ξ satisfies

$$A_i(\langle \beta, X_i \rangle + \beta_0) \geq 1 - \xi_i,$$
$$\xi_i \geq 0$$
$$i = 1, \ldots, n.$$

Interpretation of SVM

Objective function consists of two parts

- For an error to occur, $\xi_i > 1$. So $\sum_i \xi_i$ is an upper bound on the number of training errors.
- maximize the margin (minimize the inverse margin $\frac{1}{2}||\beta||^2$).

About C:

- Tuning parameter; balances the error and margin width
- For separable case, $C = \infty$. (why?)

Inequality constraints:

- Soft classification; allows some errors (misclassifications).

Quadratic Programming

Equivalently

$$\begin{aligned} \underset{\beta,\beta_0,\xi}{\text{minimize}} \quad & \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & A_i f(X_i) \geq (1-\xi_i), \quad i = 1,\ldots,n, \\ & \xi_i \geq 0, \quad i = 1,\ldots,n, \end{aligned}$$

The Lagrange primal is

$$L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left\{y_i(\beta_0 + X_i^\top\beta) - (1-\xi_i)\right\} - \sum_{i=1}^{n}\mu_i\xi_i$$

where $\alpha_i, \mu_i \geq 0$.

Quadratic Programming Continued

Setting the derivatives to zero, we get,

$$\begin{aligned}
\frac{\partial L_p}{\partial \beta} &: \beta = \sum_{i=1}^{n} \alpha_i A_i X_i \\
\frac{\partial L_p}{\partial \beta_0} &: 0 = \sum_{i=1}^{n} \alpha_i A_i \\
\frac{\partial L_p}{\partial \xi_i} &: \alpha_i = C - \mu_i
\end{aligned}$$

The Dual Problem for SVM

Substituting into the Lagrange primal, we obtain the Lagrange dual problem as

$$\begin{aligned} \text{minimize} \quad & L_D(\alpha) = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j A_i A_j \langle X_i, X_j \rangle - \sum_{i=1}^{n} \alpha_i \\ \text{subject to} \quad & 0 \le \alpha_i \le C, \quad i = 1, 2, \cdots, n \\ & \sum_{i=1}^{n} \alpha_i A_i = 0. \end{aligned}$$

- Can be solved by quadratic programming.
- Recover β: $\beta = \sum_{i=1}^{n} \alpha_i A_i X_i$; For given β, β_0 can be solved using KKT conditions or Linear Programming (LP).

Support Vectors

The KKT conditions include

$$\begin{aligned} \alpha_i^* \left\{ A_i(\beta_0^* + X_i^\top \beta^*) - (1 - \xi_i^*) \right\} &= 0 \\ \mu_i^* \xi_i^* &= 0 \end{aligned}$$

These imply

$$\begin{aligned} A_i f^*(X_i) > 1 &\Rightarrow \alpha_i^* = 0 \\ A_i f^*(X_i) < 1 &\Rightarrow \alpha_i^* = C \\ A_i f^*(X_i) = 1 &\Rightarrow 0 \leq \alpha_i^* \leq C \end{aligned}$$

Solution

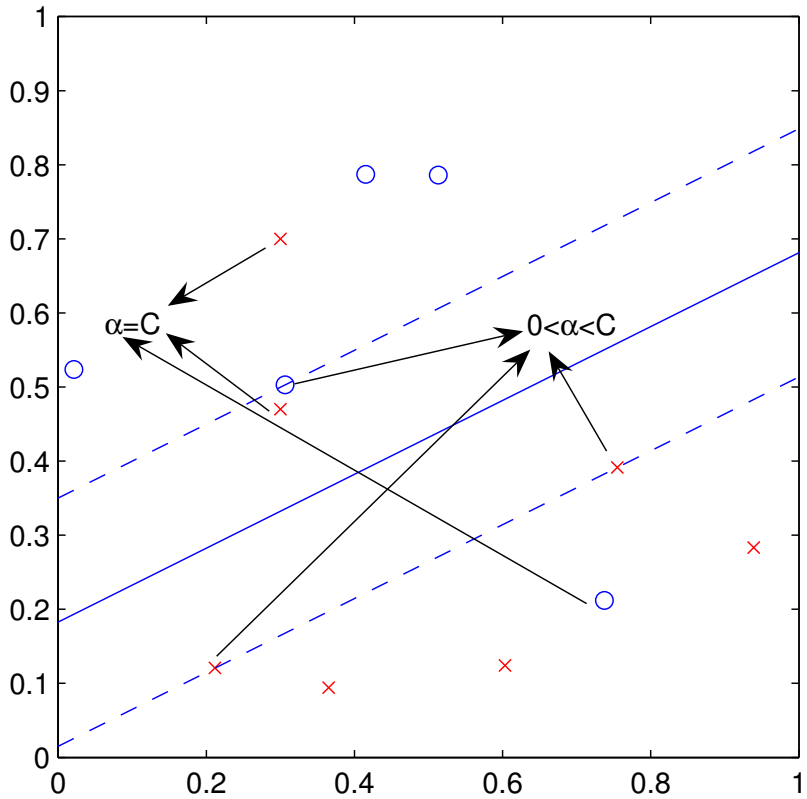
The solution is expressed in terms of fitted Lagrange multipliers $\widehat{\alpha}_i$:

$$\widehat{\beta} = \sum_{i=1}^{n} \widehat{\alpha}_i A_i X_i$$

Some fraction of $\widehat{\alpha}_i$ are exactly zero (from KKT conditions); the X_i for which $\widehat{\alpha}_i \neq 0$ are called support points $\mathcal{S}$.

$$\widehat{f}(X) = \widehat{\beta}_0 + X^\top \widehat{\beta} = \widehat{\beta}_0 + \sum_{i \in \mathcal{S}} \widehat{\alpha}_i A_i \langle X, X_i \rangle$$

Support Vectors



- $\alpha_i = 0 \rightarrow A_i f(X_i) > 1$; not needed in constructing $f(X)$. Support vectors:
- $0 < \alpha_i < C \rightarrow A_i f(X_i) = 1$ (Solve β_0).
- $\alpha_i = C \rightarrow A_i f(X_i) < 1$.
- Outliers are SVs!

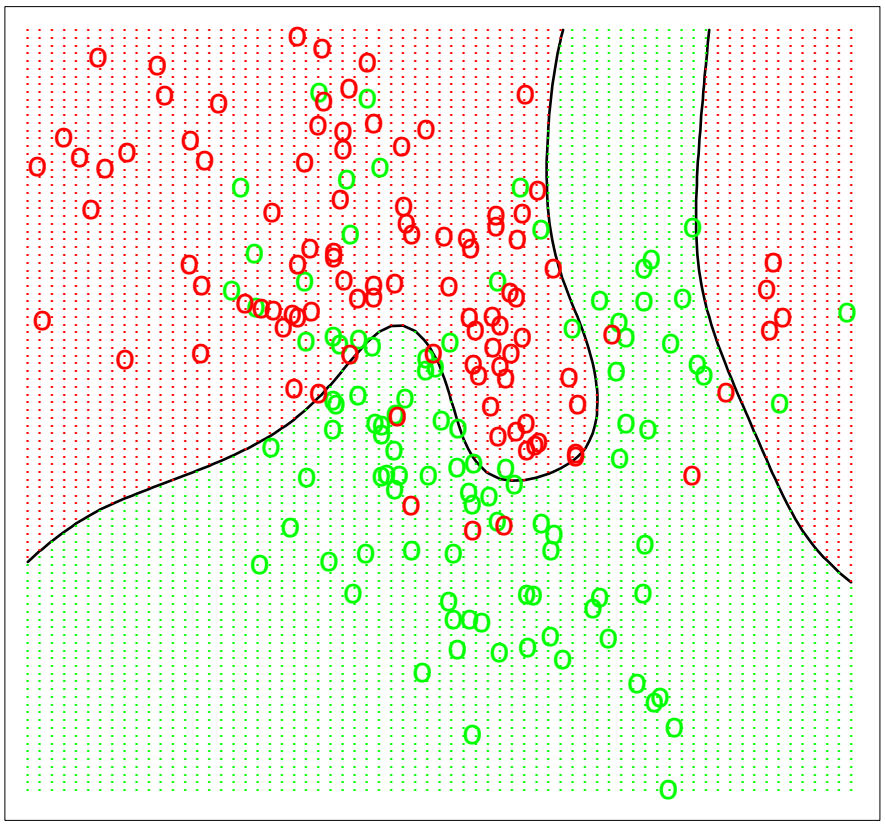
Tuning Parameter C

- large C puts more weight on misclassification rate than margin width – more attention on correctly classified points near the decision boundary (smaller bias)
- small C puts more weight on margin width than misclassification rate – more attention on data further away from the boundary (smaller variance)

Misclassified points are given weight, no matter how far away.
Tuning procedures:

- cross-validation; leave-one-out cross validation

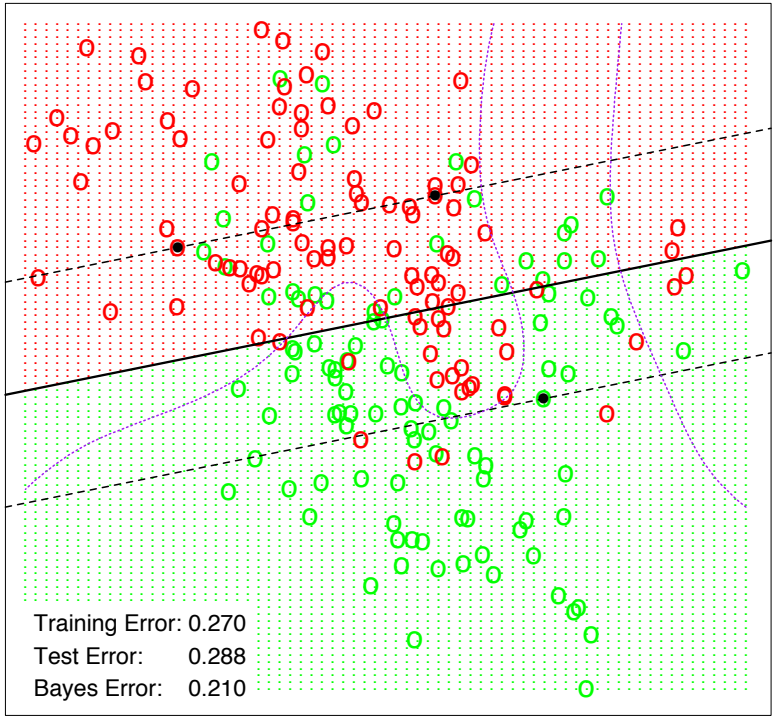
Example



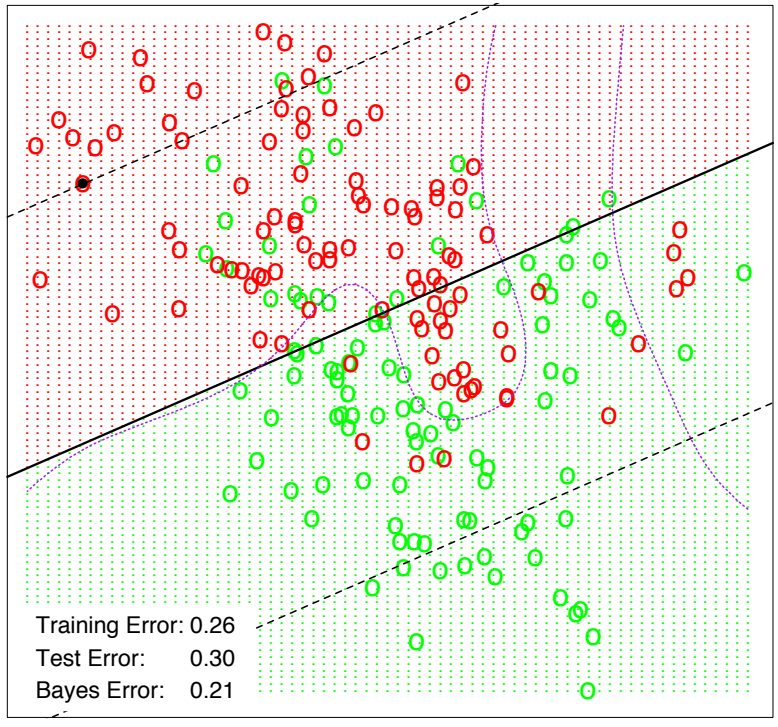
Mixture of Gaussian.

- Red class: 10 centers μ_k from $N\{(-1,1)^\top, I\}$; then randomly pick one center, and generate a data point from $N(\mu_k, I/5)$.
- Green class is similar, with $N\{(1,-1)^\top, I\}$.
- Bayes error: 0.21.

Linear SVMs



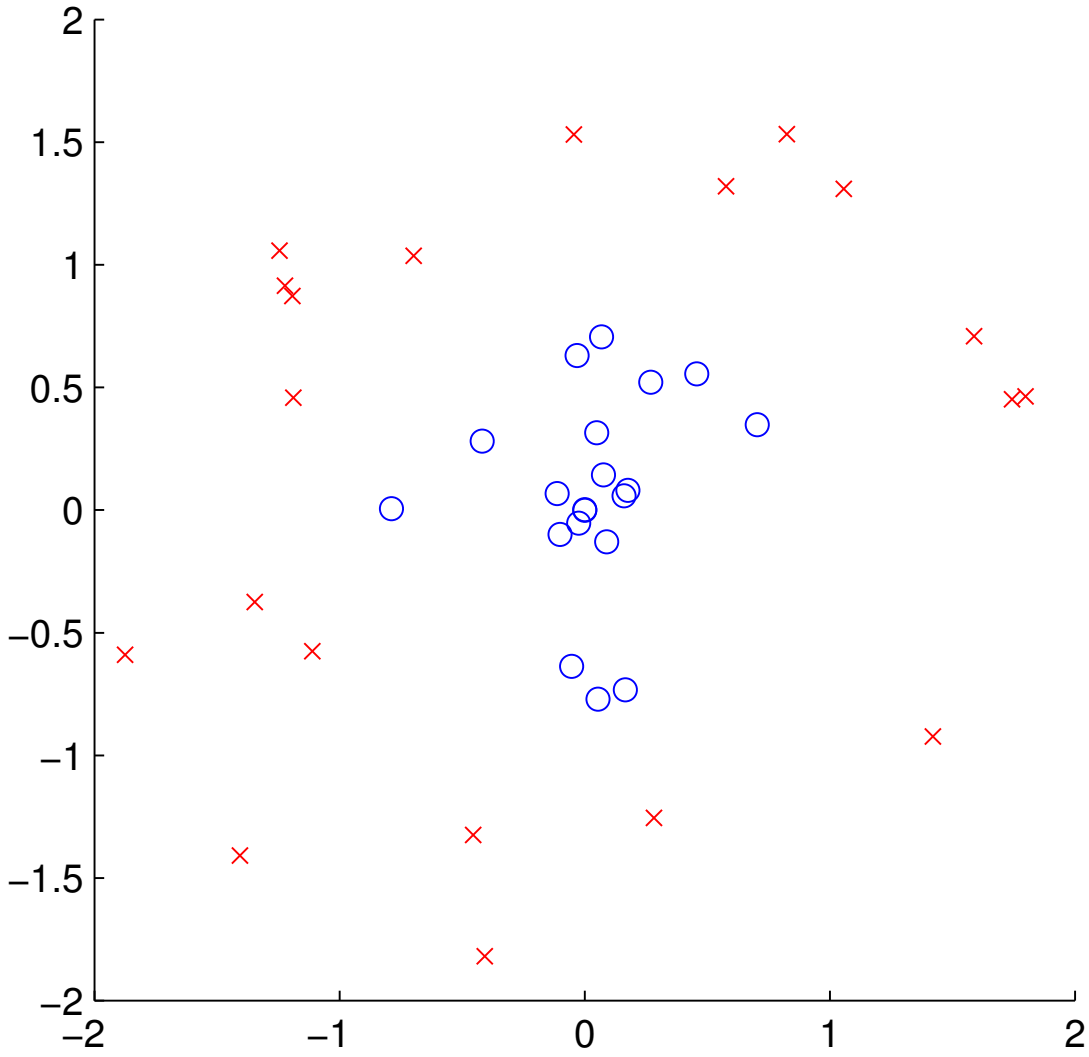
C = 10000



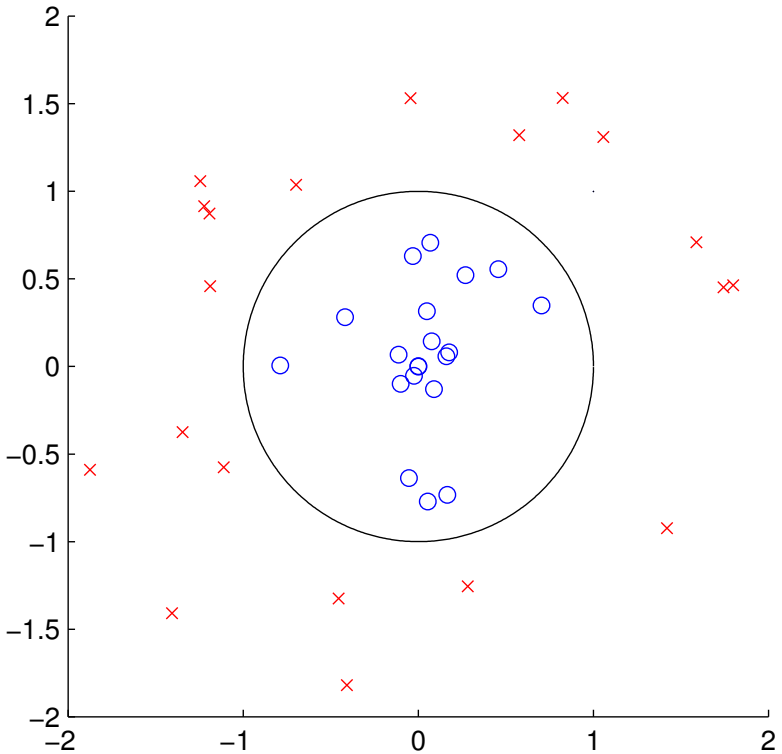
C = 0.01

Resulting classifier is $\widehat{\mathcal{D}}(X) = \text{sign}(\widehat{\beta}_0 + X^\top \widehat{\beta})$.

Non-Linear Boundaries???



Donut Example



- Decision function

$$f(X_i) = X_{i1}^2 + X_{i2}^2 - 1,$$

- Let $\phi(X_i) = (X_{i1}^2, X_{i2}^2)^\top$, $\beta = (1,1)^T$, and $\beta_0 = -1$.

$$f(X_i) = \langle \beta, \phi(X_i) \rangle + \beta_0$$

Extension to Non-Linear Boundaries

- Key idea: transform X_i into a higher dimensional space
  - Input space: the space the point X falls into.
  - Feature space: the space of $\phi(X)$ (or denote as $h(X)$)
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in the input space
  - Classification can become much easier with a proper transformation

Reformulation of SVM Optimization

- SVM solves

$$\underset{\beta_0,\beta_1}{\text{minimize}} \quad \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \ell_{\text{SVM}}\{A_i f(X_i)\}.$$

- $\ell_{\text{SVM}}(u) = (1-u)_+$ (Hinge Loss).
- Nonlinear learning can be achieved by basis expansion or kernel learning.
- Kernel Trick: Replace $\langle X_i, X_j \rangle$ by $K(X_i, X_j)$ and $f(X) = \sum_{i=1}^{n} A_i \alpha_i K(X_i, X) + \beta_0$.

Flexible Classifiers

- Enlarge the input space via basis expansion ($p \to q$):

$$h(X) = (h_1(X), h_2(X), \ldots, h_q(X)) .$$

- Lagrange dual and solution become

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i'=1}^{n} \alpha_i \alpha_{i'} A_i A_{i'} \langle h(X_i), h(X_{i'}) \rangle,$$

and

$$\widehat{f}(X) = \widehat{\beta}_0 + \sum_{i \in \mathcal{S}} \widehat{\alpha}_i A_i \langle h(X), h(X_i) \rangle.$$

Example

2nd degree polynomial in $\mathbb{R}^2$. We choose:

$$
\begin{aligned}
h_1(X_i) &= 1 \\
h_2(X_i) &= \sqrt{2}X_{i1} \\
h_3(X_i) &= \sqrt{2}X_{i2} \\
h_4(X_i) &= X_{i1}^2 \\
h_5(X_i) &= X_{i2}^2 \\
h_6(X_i) &= \sqrt{2}X_{i1}X_{i2}
\end{aligned}
$$

Kernel Trick and Decision function of kernel SVM

- $\beta = \sum_{i=1}^{n} \alpha_i A_i h(X_i)$
- The decision function is given by:

$$\begin{aligned} f(X) &= \langle \beta, h(X) \rangle + \beta_0 \\ &= \sum_{i=1}^{n} \alpha_i A_i K(X_i, X) + \beta_0 \end{aligned}$$

- The explicit computation of $h(X)$ is not necessary.
- It is enough to have kernel $K(X_i, X_j) = \langle h(X_i), h(X_j) \rangle$.
- Given a suitable kernel function $K(X, X')$, don't need $h(X)$ at all.

$$\widehat{f}(X) = \widehat{\beta}_0 + \sum_{i \in \mathcal{S}} \widehat{\alpha}_i a_i K(X, X_i)$$

Example Continued

If we choose

$$K(X_i, X_j) = (1 + \langle X_i, X_j \rangle)^2$$

then

$$\begin{aligned} K(X_i, X_j) &= (1 + X_{i1}X_{j1} + X_{i2}X_{j2})^2 \\ &= 1 + 2X_{i1}X_{j1} + 2X_{x2}X_{j2} + (X_{i1}X_{j1})^2 \\ &\quad +(X_{i2}X_{j2})^2 + 2X_{i1}X_{j1}X_{i2}X_{j2} \\ &= \langle h(X_i), h(X_j) \rangle \end{aligned}$$

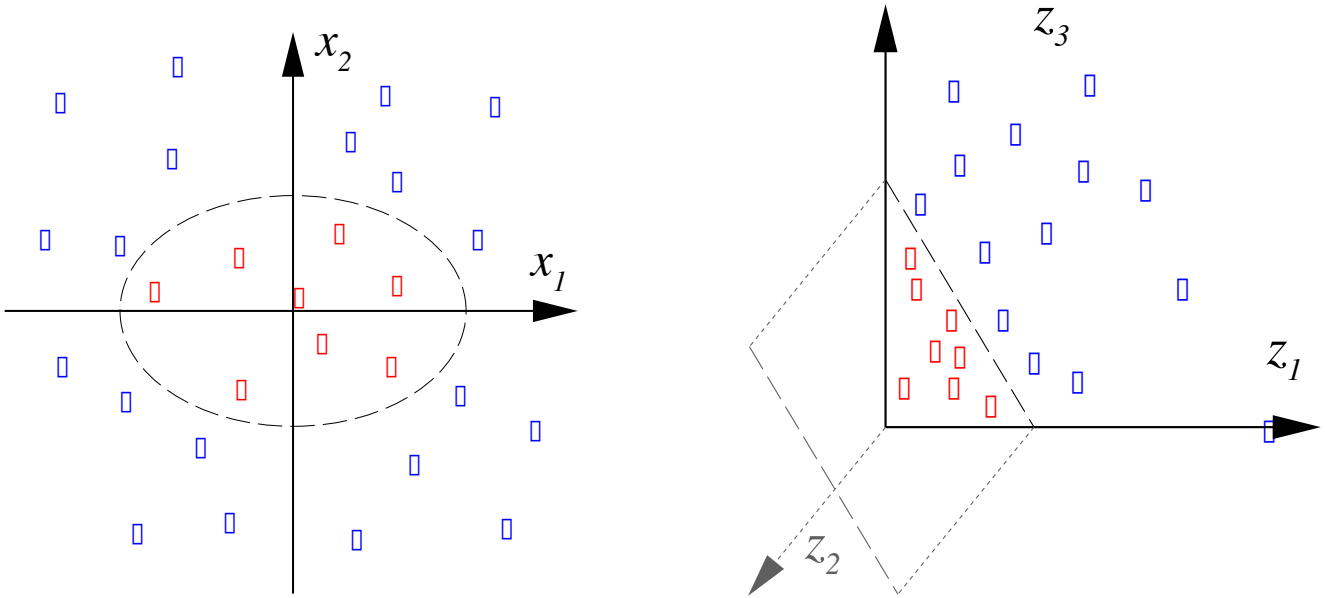
Popular Kernels

- Linear: $K(X_i, X_j) = \langle X_i, X_j \rangle = \sum_{k=1}^{p} X_{ik} X_{jk}$.
- Polynomial: $K(X_i, X_j) = (1 + \langle X_i, X_j \rangle)^d$.
- Gaussian (Radial Basis Function, i.e. RBF): $K(X_i, X_j) = \exp(-\sigma \|X_i - X_j\|^2) = \exp\left\{-\sigma \sum_{k=1}^{p} (X_{ik} - X_{jk})^2\right\}$.

$K(X_i, X_j)$ is a symmetric, positive (semi-) definite function: For every $n = 1, 2, \ldots$, and every set of real numbers $\{\alpha_1, \alpha_2, \ldots, a_n\}$ and $X_1, X_2, \ldots, X_n$, we have $\sum_{i,i'=1}^{n} \alpha_i \alpha_j K(X_i, X_j) \geq 0$.

Example: All Degree 2 Monomials

$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$



Role of Tuning Parameters

Large C

- discourage any positive ξ_i
- may lead to an overfit wiggly boundary in the original space

Small C

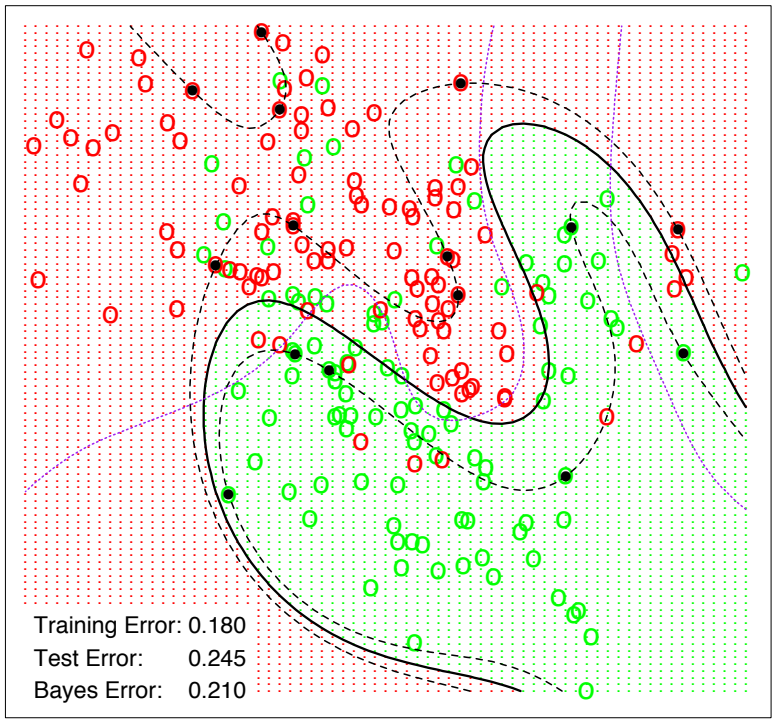
- encourage small value of $\|\beta\|$
- may lead to smoother boundary

Adaptive Tuning of Parameters

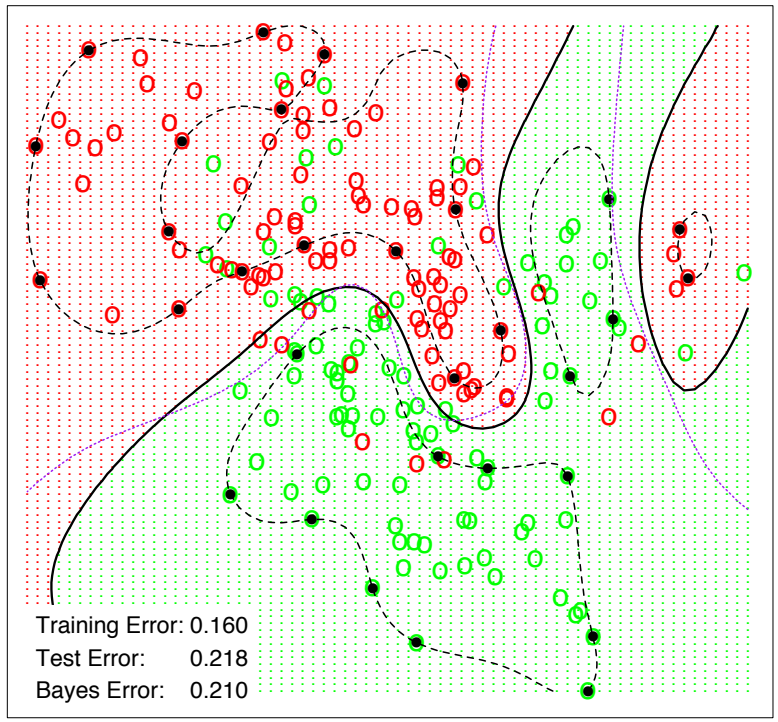
- cross validation
- minimizing test errors

Nonlinear SVMs

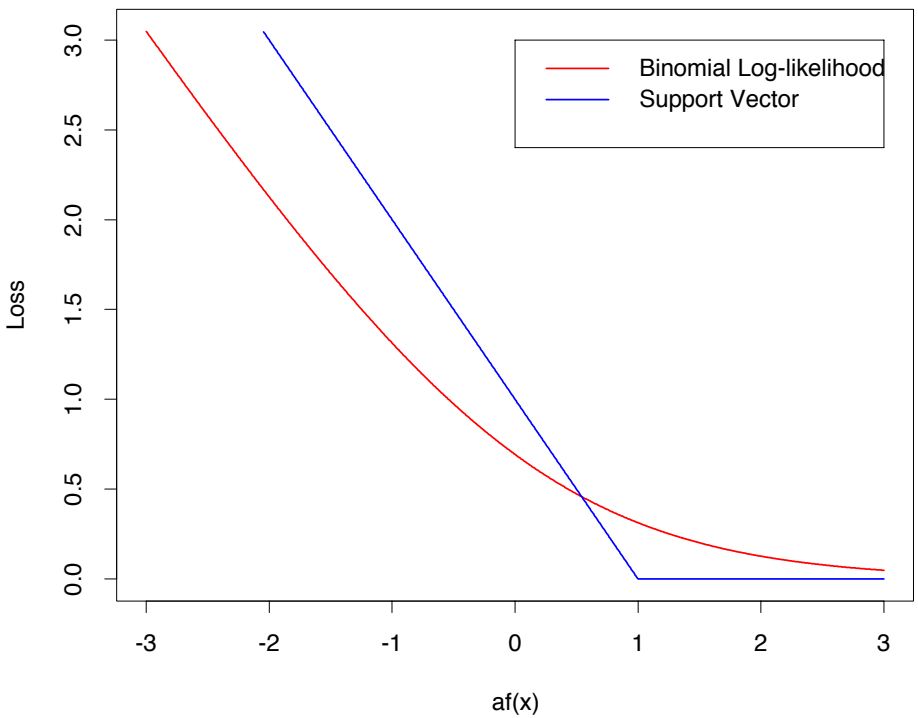
SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



SVM via Loss + Penalty



With $f(X) = \beta_0 + X^\top \beta$, consider

$$\underset{\beta_0,\beta}{\text{minimize}} \quad \sum_{i=1}^{n} \{1 - A_i f(X_i)\}_+ + \frac{\lambda}{2}\|\beta\|^2$$

Solution identical to SVM solution, with $\lambda = 1/C$.

SVM and Function Estimation

SVM with general kernel $K(\cdot, \cdot)$ minimizes:

$$\sum_{i=1}^{n} \{1 - A_i f(X_i)\}_+ + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}_K}$$

with $f \in \mathcal{H}_K$. $\mathcal{H}_K$ is the reproducing kernel Hilbert space (RKHS) of functions generated by the kernel $K(\cdot, \cdot)$.

Function space $\mathcal{H}_K$ generated by a positive (semi-) definite function $K(X_i, X_j)$. Eigen expansion (Mercer's theorem)

$$K(X_i, X_j) = \sum_{k=1}^{\infty} \gamma_k \phi_k(X_i) \phi_k(X_j)$$

where

$$\gamma_k \geq 0, \ \sum_{k=1}^{\infty} \gamma_k^2 < \infty$$

RKHS Continued

Define $\mathcal{H}_K$ to be the set of functions of the form

$$f(X) = \sum_{k=1}^{\infty} \theta_k \phi_k(X)$$

and define the inner product

$$\left\langle \sum_{k=1}^{\infty} \theta_k \phi_k(X), \sum_{k'=1}^{\infty} \delta_{k'} \phi_{k'}(X) \right\rangle_{\mathcal{H}_K} \stackrel{\text{def}}{=} \sum_{k=1}^{\infty} \frac{\theta_k \delta_k}{\gamma_k}$$

Then the squared norm of f is

$$\|f(X)\|^2_{\mathcal{H}_K} = \sum_{k=1}^{\infty} \theta_k^2 / \gamma_k$$

which is generally viewed as a roughness penalty.

Basis Representation

More generally we can optimize

$$\underset{f \in \mathcal{H}_K}{\text{minimize}} \quad \left[\sum_{i=1}^{n} \ell\{A_i, f(X_i)\} + \frac{\lambda}{2} ||f||^2_{\mathcal{H}_K} \right].$$

Equivalently

$$\underset{\theta_j}{\text{minimize}} \quad \left[\sum_{i=1}^{n} \ell\{A_i, \sum_{k=1}^{\infty} \theta_k \phi_k(X_i)\} + \frac{\lambda}{2} \sum_{k=1}^{\infty} \frac{\theta_k^2}{\gamma_k} \right].$$

The Representer Theorem

The solution has the finite form (Wahba 1990)

$$\widehat{f}(X) = \sum_{i=1}^{n} \widehat{\alpha}_i K(X, X_i)$$

a finite expansion in the representers $K(X, X_i)$. Example: smoothing spline and thin-plate spline.

Reproducing property

$$\langle K(X, X_i), f(X) \rangle_{\mathcal{H}_K} = f(X_i)$$

Hence

$$\|\widehat{f}\|^2_{\mathcal{H}_K} = \sum_{i=1}^{n} \sum_{j=1}^{n} K(X_i, X_j) \widehat{\alpha}_i \widehat{\alpha}_j$$

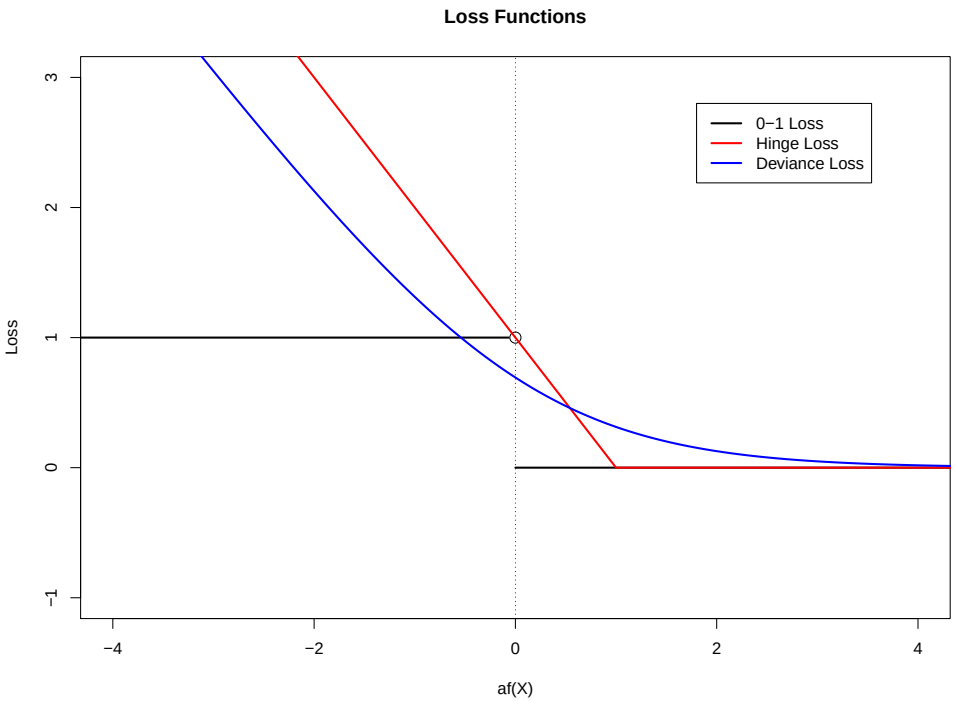
Equivalent finite dimensional criterion (in matrix notation):

$$\underset{\alpha}{\text{minimize}} \quad \ell(A, K\alpha) + \frac{\lambda}{2} \alpha^\top K \alpha,$$

where K is the $n \times n$ matrix with elements $K(X_i, X_j)$.

Loss Functions

To estimate the classifier (threshold), $\text{sign}\{\Pr(A = 1|X) - \Pr(A = -1|X)\}$



- 0-1 Loss:
  $\ell\{A, f(X)\} = I\{Af(X) < 0\}$.
- Hinge Loss:
  $\ell\{A, f(X)\} = \{1 - Af(X)\}_+$
- Deviance Loss: $\ell\{A, f(X)\} =$
  $\log[1 + \exp\{-Af(X)\}]$

Kernel Logistic Regression

Deviance loss is from binomial distribution:

$$\ell\{A, f(X)\} = \log[1 + \exp\{-Af(X)\}]$$

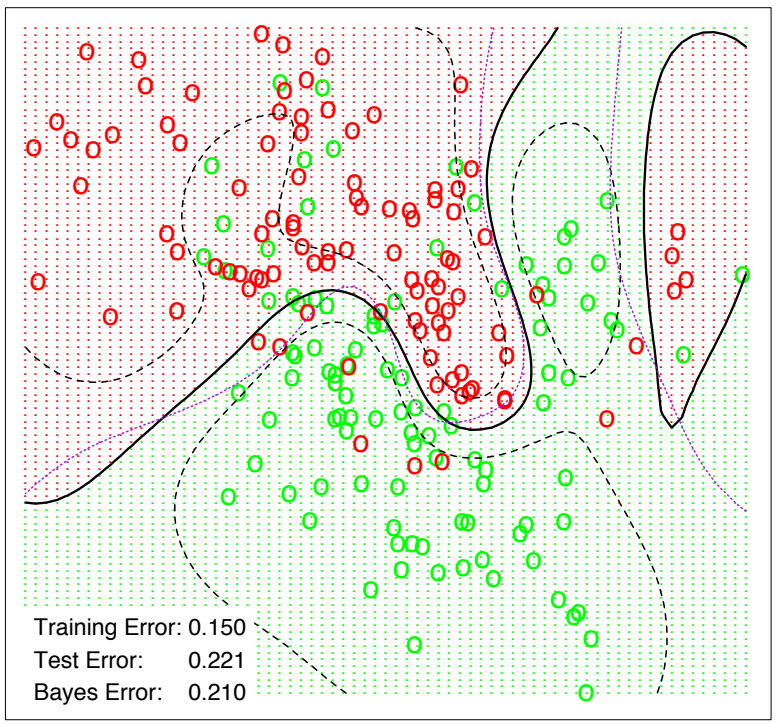
- binomial log-likelihood
- Estimates the logit

$$\log \frac{\Pr(A = +1|X)}{\Pr(A = -1|X)}.$$

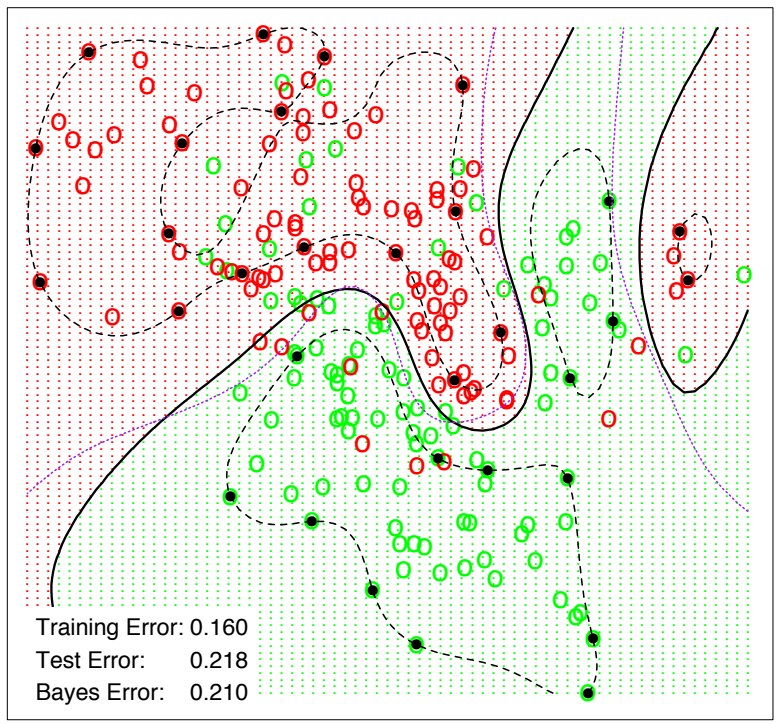
- Replace $(1 - Af)_+$ with $\log(1 + e^{-Af})$, the binomial deviance.
- Similar classification performance as the SVM.
- Provide estimates of class probabilities.
- Natural generalization to the multi-class case.

KLR vs SVM

LR - Radial Kernel in Feature Space



SVM - Radial Kernel in Feature Space



Remarks

- SVM can be viewed as regularized fitting with a particular loss function: hinge loss.
- The hinge loss allows for compression in terms of basis functions, from n to some fraction of n.
- Regularized logistic regression gives very similar fit, using binomial deviance as the loss.
- KLR does not have compression properties but it provides probability estimation.

Curse of Dimensionality

- True function quadratic in x_1 to x_4.
- Noise features x_5 to x_{10} included.
- SVMs can suffer in high dimensions.
- Sparse SVM such as L_1 SVM can be used.

	Test Error (SE)	
Method	No noise feature	6 noise feature
SVM/poly 1	0.423 (0.006)	0.466 (0.008)
SVM/poly 2	0.081 (0.016)	0.172 (0.015)
SVM/poly 5	0.212 (0.008)	0.393 (0.004)
SVM/poly 10	0.265 (0.011)	0.438 (0.006)

Robust SVM

- Hinge loss can be sensitive to outliers
- Truncated hinge loss
- Why not 0-1 loss?

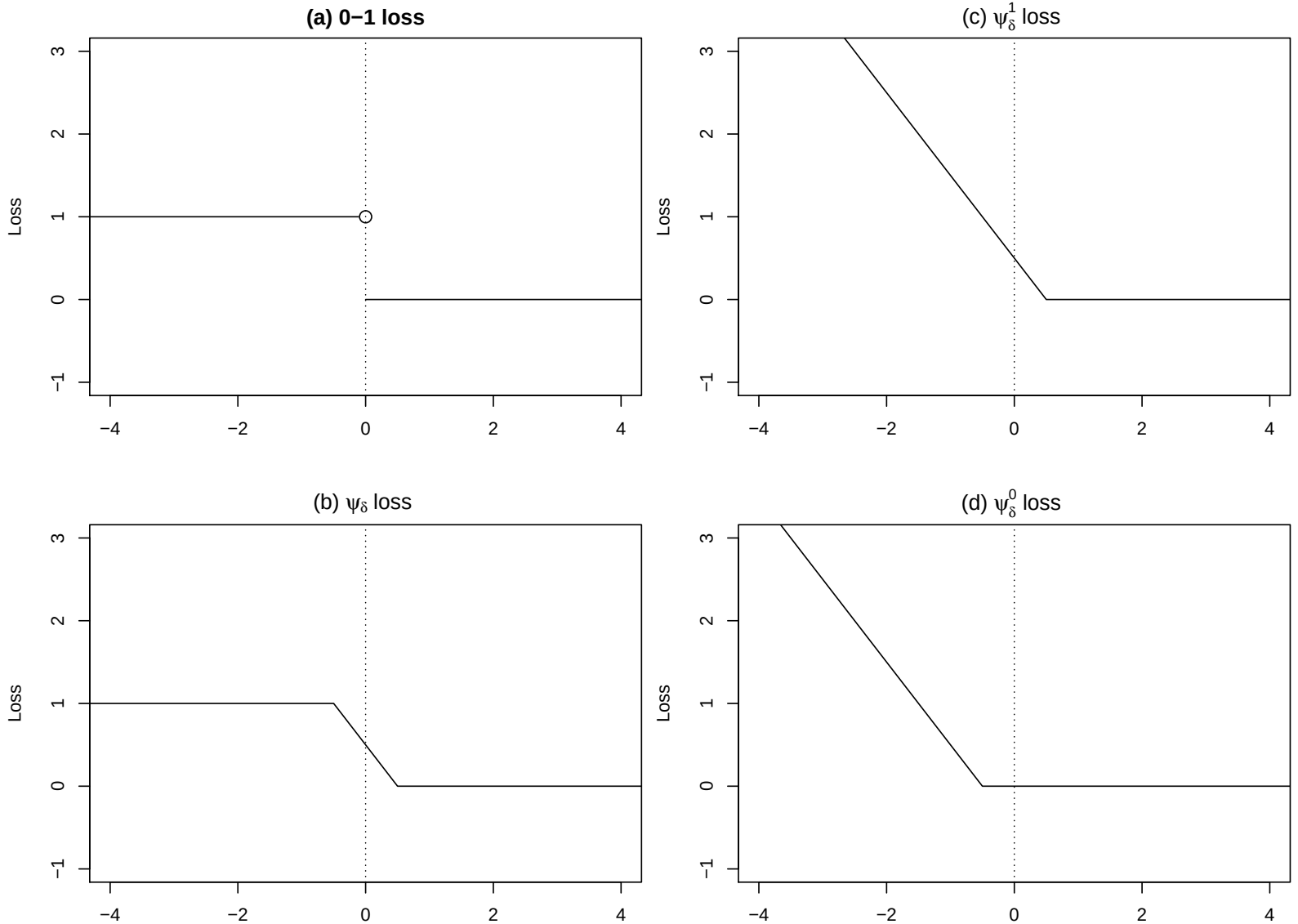
Robust Learning

- Robust Learning: Reduce the loss for outliers (Wu and Liu, JASA 2007).
- We approximate $I\{Af(X) < 0\}$ by the following ψ-loss:

$$\begin{aligned} \psi_\delta(X) &= \psi_\delta^1(X) - \psi_\delta^0(X) \\ &= (2\delta)^{-1}(\delta - X)_+ - (2\delta)^{-1}(-\delta - X)_+. \end{aligned}$$

- Challenge: the loss function is not a convex anymore.

DC Decomposition



DC Algorithm

DC Algorithm (Difference of Convex)

1. Initialize Θ_0.
2. Repeat $\Theta_{t+1} = \text{argmin}_{\Theta}(J_{vex}(\Theta) + \langle J'_{cav}(\Theta_t), \Theta - \Theta_t \rangle)$ until convergence of Θ_t.

- The algorithm converges in finite steps (Liu et al., JCGS, 2005).
- Choice of initial values: Use the original classifiers without truncation.
- The set of SVs is a only a SUBSET of the original one!

Classification and Loss Function

Roughly speaking, A good classifier has smaller errors (we will discuss regularization later).

Classification objective function

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

If we compare our ITR objective function as below,

ITR objective function

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{Y_i}{p(A_i|X_i)} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

ITR objective function

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{Y_i}{p(A_i|X_i)} I\{A_i \neq \mathcal{D}(X_i)\}.$$

ITR.SVM

$$D_o = \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{Y_i}{p(A_i|X_i)} \ell\{A_i, f(X_i)\} + \frac{\lambda}{2}\|f\|^2_{\mathcal{H}_K}.$$

Algorithm 1 ITR.SVM

1: Compute $\widehat{\alpha}$ by solving the following convex problem,

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -\frac{1}{2}\alpha^T \mathbf{H}\alpha + \underline{1}^T\alpha \\ \text{subject to} \quad & \begin{cases} \mathbf{0} \leq \alpha \leq \eta, \\ \alpha^T A^* = 0, \end{cases} \end{aligned}$$

where the $\mathbf{H}$, η, and A^* are defined in previous slides.

2: Compute $\widehat{\beta}_i, i = 1, \ldots, n$,

$$\widehat{\beta}_i \quad = \quad \widehat{\alpha}_i A_i^*, \quad \forall i = 1, \ldots, n.$$

3: Computer $\widehat{\beta}_0$ using,

$$\forall \alpha_i : 0 < \alpha_i < \eta_i \quad \Rightarrow \quad A_i^* \{\beta_0 + \sum_{j=1}^{n} \beta_j K(X_i, X_j)\} = 1.$$

4: **return** $\widehat{\beta}_{SVM} = \{\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_n\}$.

Algorithm 2 ITR.SVM.DC

1: Set initial value $\beta^{(0)} = \widehat{\beta}_{SVM}$ which is computed from IOWL-SVM (Algorithm 1).
2: **repeat**
3: Compute $I_i^{(l)} = I\left[A_i^* \{\beta_0^{(l)} + \sum_{j=1}^n \beta_j^{(l)} K(X_i, X_j)\} < -\delta\right]$.
4: Compute $\widehat{\alpha}$ by solving the following convex problem,

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\alpha^T \mathbf{H}\alpha + \delta \underline{1}^T \alpha \\ s.t. \quad & \begin{cases} -\eta_I \leq \alpha \leq \eta - \eta_I, \\ \alpha^T A^* = 0. \end{cases} \end{aligned}$$

5: Compute $\widehat{\beta}_i^{(l+1)}$,

$$\widehat{\beta}_i^{(l+1)} \quad = \quad \widehat{\alpha}_i A_i^*, \quad \forall i = 1, \cdots, n.$$

6: Compute $\widehat{\beta}_0^{(l+1)}$,

$$\forall \alpha_i : -\eta_{I_i} < \alpha_i < \eta_i - \eta_{I_i} \quad \Rightarrow \quad A_i^* \{\beta_0 + \sum_{j=1}^n \widehat{\beta}_j^{(l+1)} K(X_i, X_j)\} = \delta.$$

7: **until** $\| I^{(l+1)} - I^{(l)} \| = 0$.

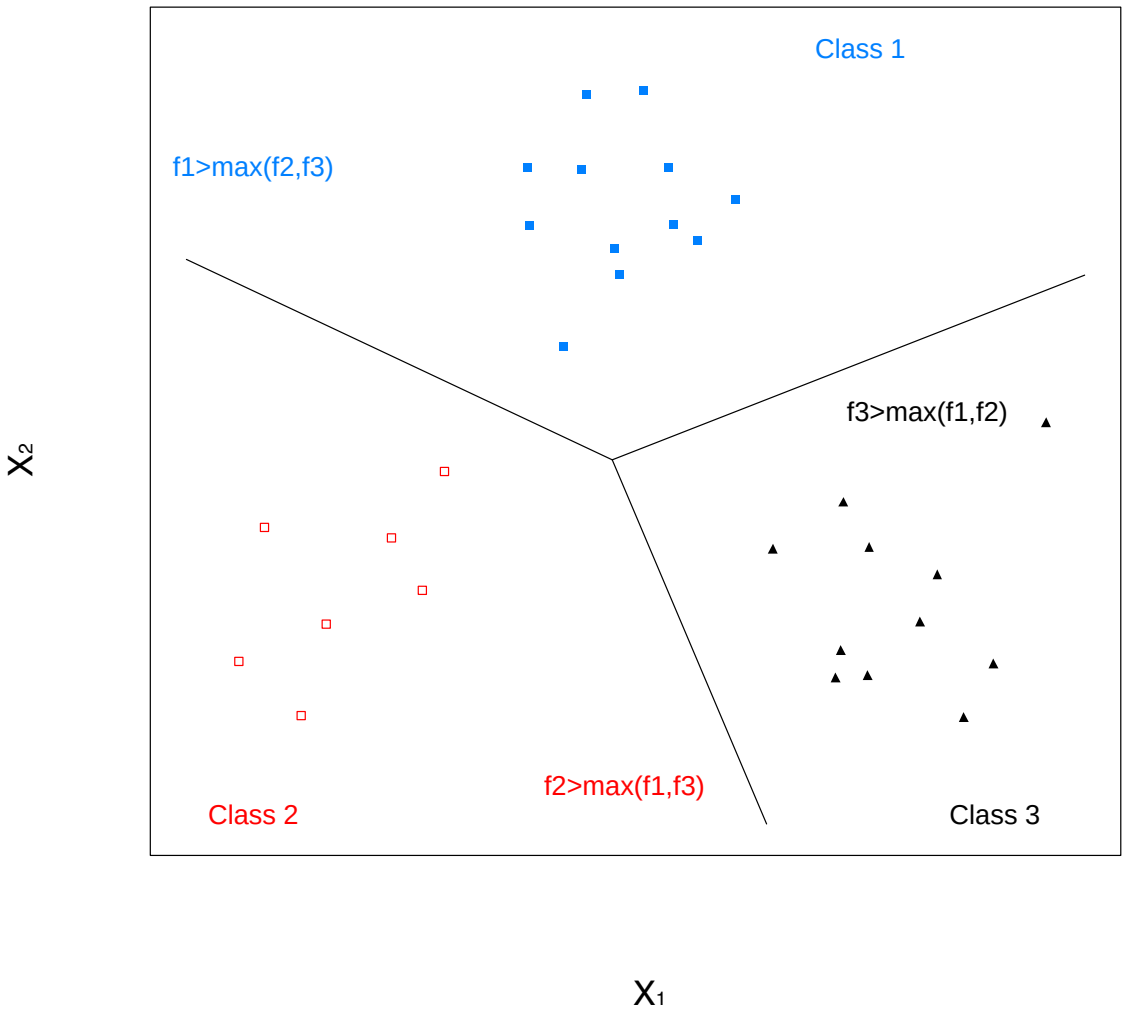
To be completed ...

- write down data generation model to generate training data sets.
- write down a Rcpp package for students to install and try.
- generate the results.

From Binary to Multicategory

- Require novel techniques.
- Label: $\{-1, +1\} \to \{1, 2, \ldots, k\}$.
- k-class
  - Construct decision function vector $f = (f_1, \ldots, f_k)$. ($k = 2$ only one f)
  - Classifier: $\text{argmax}_{j=1,\ldots,k} f_j(X)$. ($k = 2$: $\text{sign}(f)$).
- Sum-to-zero constraint $\sum_{j=1}^{k} f_j(X) = 0$.

Illustration



Find $f = (f_1, f_2, f_3)$ and use $\text{argmax}_j f_j(X)$ to do classification.

Existing Formulations

- There are many existing formulations and ad hoc approaches.
- One versus the rest or one versus the other.
- Vapnik (1998), Weston and Watkins (1999), Bredensteiner and Bennett (1999), Guermeur (2002)

$$\ell\{f(X), A\} = \sum_{j \neq A} [1 - (f_A - f_j)]_+$$

- Crammer and Singer (2001), Liu and Shen (2006)

$$V\{f(X), A\} = [1 - (f_A - \max_{j \neq A} f_j)]_+$$

- Multicategory SVM (Lee et al., 2014) and reinforced multicategory hinge loss (Liu and Yuan, JCGS, 2011)
- Many formulations are either not always Fisher consistent or computational inefficiency.

Multicategory Angle-Based Classification (ABC)

- A simplex based classification structure
- Advantages of ABC (Zhang and Liu, Biometrika, 2014)
  - General structure: binary $\rightarrow$ multicategory
  - Clear geometric interpretation
  - Free of sum-to-zero constraint $\Rightarrow$ faster computational speed
  - Theoretical advantages
  - Numerically competitive

A k-Regular Polyhedron in a $\mathbb{R}^{k-1}$ Euclidean Space

A simplex W with k vertices $\{W_1, \cdots, W_k\}$ in a $(k-1)$-dimensional space,

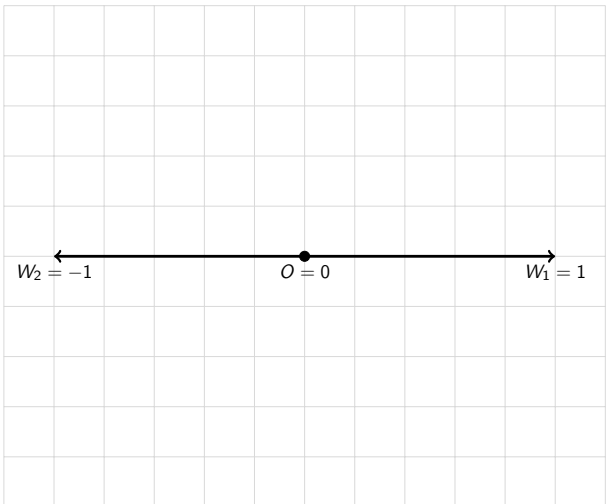
$$W_j = \begin{cases} (k-1)^{-1/2}\mathbf{1}_{k-1}, & j = 1, \\ -(1+k^{1/2})/\{(k-1)^{3/2}\}\mathbf{1}_{k-1} + \{k/(k-1)\}^{1/2} e_{j-1}, & 2 \le j \le k, \end{cases}$$

where $\mathbf{1}_i$ is a vector of 1 with length equal to i, and e_i is a vector in $\mathbb{R}^{k-1}$ such that its every element is 0, except the ith element is 1.

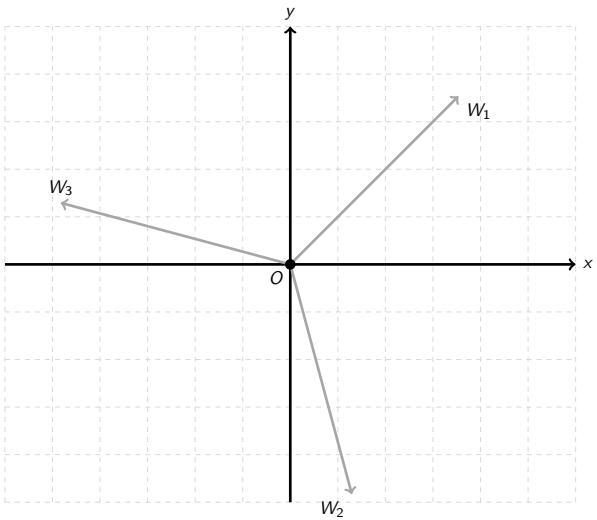
Properties:

- The centre of W is at the origin.
- Each W_j has Euclidean norm 1.
- The angles between any two directions $\angle(W_i, W_j), \forall i \neq j$ are equal.
- Every vector in $\mathbb{R}^{k-1}$ generate k different angles with respect to $\{W_1, \cdots, W_k\}$, and all these angles are in $[0, \pi]$.

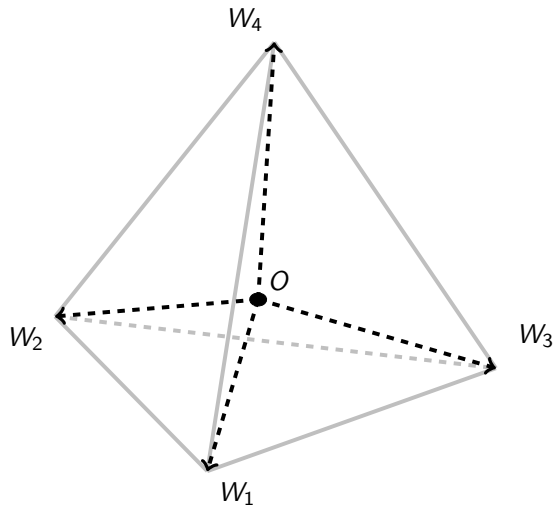
Illustration of $\{W_i\}$ When $k = 2, 3, 4$.



(a) $k = 2$



(b) $k = 3$



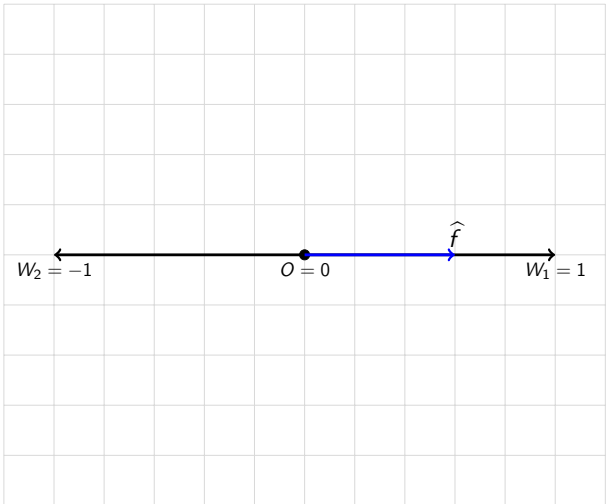
(c) $k = 4$

Remark: When $k = 3$, $\{W_i, i = 1, 2, 3\}$ are the vertices of an equilateral triangle, and when $k = 4$, $\{W_i, i = 1, 2, 3, 4\}$ are the vertices of a regular tetrahedron.

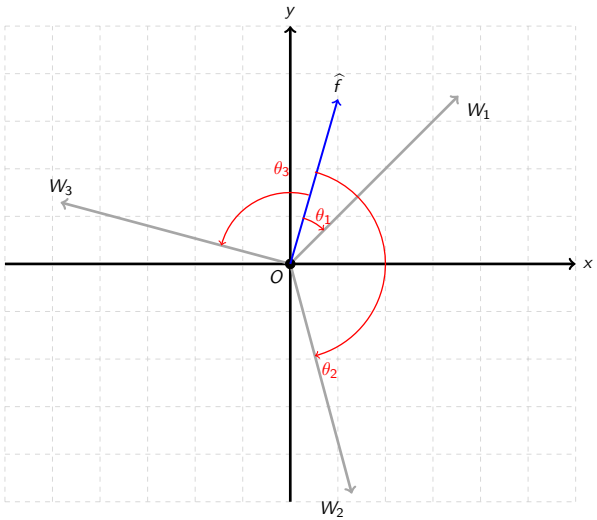
Angle Based Classifier

- Let W_j represent class j.
- Our method is to map x to $\widehat{f}(x) \in \mathbb{R}^{k-1}$.
- $\mathcal{A}$ is the class spaces as $\mathcal{A} = \{1, 2, \cdots, k\}$, and $a_i \in \mathcal{A}$ which is the class membership of subject i.
- We predict $\widehat{a}$ to be the class whose corresponding angle is the smallest, i.e. $\widehat{a} = \arg\min_j \angle(W_j, \widehat{f})$, where $\angle(\cdot, \cdot)$ denotes the angle between two vectors.
- Minimizing the angle is equivalent to maximize $\langle f(x_i), W_{a_i} \rangle$.

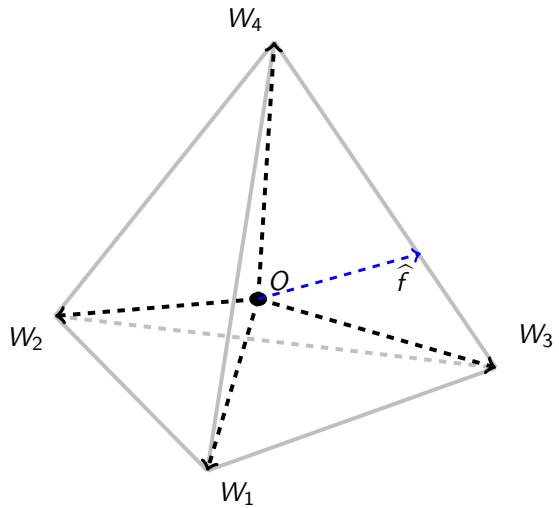
Angle Based Classifier Illustration



(a) $k = 2$



(b) $k = 3$



(c) $k = 4$

- $k = 2$, $W_1 = 1$ and $W_2 = -1$.
- $k = 3$ (equilateral triangle),
  $W_1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right), W_2 = \left(\frac{\sqrt{3}-1}{2\sqrt{2}}, -\frac{\sqrt{3}+1}{2\sqrt{2}}\right), W_3 = \left(-\frac{\sqrt{3}+1}{2\sqrt{2}}, \frac{\sqrt{3}-1}{2\sqrt{2}}\right).$
- $k = 4$ (regular tetrahedron),
  $W_1 = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right), W_2 = \left(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right), W_3 = \left(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right), W_4 = -\left(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right).$

Angle Based Classifier

With ℓ a convex monotone decreasing function, we have our angle based classifier as,

$$\underset{f \in \mathcal{F}}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} \ell\{\langle f(x_i), W_{a_i} \rangle\} + \lambda J(f). \tag{5}$$

Example ($k = 2$)

For a binary case, i.e. $k = 2$, $\langle f(x_i), W_{a_i} \rangle = a f(x_i)$,

- When $\ell(\cdot)$ is a deviance loss, $\ell(z) = \log\{1 + \exp(-z)\}$, equation (5) is a logistic regression.
- When $\ell(\cdot)$ is a hinge loss, $\ell(z) = (1 - z)_+$, equation (5) is the support vector machine.

Fisher Consistency

Let $f^*(\cdot)$ be a classifier, and a function $g(\cdot,\cdot)$ is a map $g\{f^*(x), i\}$ from $x \in \mathcal{X}$ and $i \in \mathcal{A}$ to $\mathbb{R}$. The classification of x is $\widehat{a} = \arg\max_{\forall i} g\{f^*(x), i\}$. In our angle based classifier, $g\{f^*(x), i\} = \langle f^*(x), W_i \rangle$.

Definition (Fisher consistency)

A classifier $f^*(\cdot)$ is called Fisher's consistence if it satisfies that, $\forall x$,

$$\underset{\forall i}{\operatorname{argmax}} \Pr(A = i | X = x) \quad = \quad \underset{\forall j}{\operatorname{argmax}}\, g\{f^*(x), j\}.$$

Theorem (Fisher consistency for ABC)

The angle-based classifier from is Fisher consistency if ℓ is a convex, the derivative ℓ' exists and $\ell'(x) < 0, \forall x$.

ITR.ABC

Original objective function,

$$D_o \quad = \quad \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{Y_i}{p(A_i|X_i)} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

ITR.ABC objective function,

$$\underset{f \in \mathcal{F}}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} \frac{Y_i}{\Pr(A_i|X_i)} \ell\{\langle f(x_i), W_{a_i} \rangle\} + \lambda J(f).$$

Theorem (Fisher consistency for ITR.ABC)

A classifier $f^*(\cdot)$ is called Fisher's consistence if it satisfies that, $\forall x$,

$$\underset{\forall j}{\operatorname{argmax}} \langle f^*(x), W_j \rangle \quad = \quad \underset{\forall j}{\operatorname{argmax}}\, E(Y|A = j, x)$$

ITR.ABC is Fisher consistency if ℓ is a convex, the derivative ℓ' exists and $\ell'(x) < 0, \forall x$.

Illustration Data: Survival Outcome with Censoring

Table 10: *An illustration dataset: with censoring.* $Y = T \wedge C$ and $\Delta = I(T \leq C)$.

ID	Y	Δ	Trt	X_1	X_2	X_3	$\cdots$
1	1.5	1	1	F	26	7.8	$\cdots$
2	1.0	0	2	M	28	8.2	$\cdots$
3	2.3	1	3	M	31	8.9	$\cdots$
4	0.8	0	2	F	35	9.4	$\cdots$
5	1.7	1	1	M	22	7.3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Research Question

When some survival times are censored, based on these data, how we can learn a treatment assignment rule that, if followed by the entire population of certain patients, would lead to the best outcome on average.

How it works

Because,

$$E(T|A,X) = E\left[E\left\{\frac{T\cdot I(C>T)}{S_C(T|A,X)}\middle|A,X,T\right\}\right] = E\left\{\frac{\Delta\cdot Y}{S_C(Y|A,X)}\middle|A,X\right\},$$

we have,

$$\begin{aligned} L(\mathcal{D}) &\triangleq E\left[\frac{I\{A\neq\mathcal{D}(X)\}}{p(A|X)}T\right] \\ &= E\left[\frac{I\{A\neq\mathcal{D}(X)\}}{p(A|X)}E(T|A,X)\right] \\ &= E\left[\frac{I\{A\neq\mathcal{D}(X)\}\,\Delta Y}{p(A|X)S_C(Y|A,X)}\right]. \end{aligned} \tag{6}$$

Intuitive motivations

What if my model is wrong ...

- We often assume independent and noninformative censoring in analyzing survival outcomes (e.g. Cox model).
- Therefore, censoring event time and survival event time can be modeled separately.
- When event times or censoring times are rare, we may not be very confident to model both event time and censoring time right.
- It will be great that my method is right, if I can get at least one model is correct although I am not sure which one is correct.
- Let us use notations with superscript m to denote a proposed model (which may not be the true model).
- Now we propose our doubly robust estimator as ...

Doubly Robust Estimator

Doubly Robust Estimator

$$L^m(\mathcal{D}) \triangleq E\left(\left[\frac{\Delta \cdot Y}{S_C^m(Y|A,X)} + \int E^m(T|T>t,A,X) \left\{\frac{dN_C(t)}{S_C^m(t|A,X)} + I(Y \geq t)\frac{dS_C^m(t|A,X)}{S_C^m(t|A,X)^2}\right\}\right] \frac{I\{A \neq \mathcal{D}(X)\}}{p(A|X)}\right) \tag{7}$$

Theorem (Doubly Robust)

We have $L^m(\mathcal{D}) = L(\mathcal{D})$, if one of the following two condition holds,

1. $S_C^m(t|A,X) = S_C(t|A,X)$,
2. $E^m(T|T>t,A,X) = E(T|T>t,A,X)$.

Original objective function,

$$D_o \quad = \quad \underset{D \in R}{\operatorname{argmin}}\, n^{-1} \sum_{i=1}^{n} \frac{T_i}{p(A_i|X_i)} I\left\{A_i \neq \mathcal{D}(X_i)\right\}.$$

ITR.ABC objective function,

$$\underset{f \in \mathcal{F}}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^{n} \frac{T_i}{\Pr(A_i|X_i)} \ell\{\langle f(x_i), W_{a_i} \rangle\} + \lambda J(f).$$

Definition (ITR.Survival)

$$L_n^m(f) \triangleq \frac{1}{n} \sum_{i=1}^{n} \left(\left[\frac{\Delta_i \cdot Y_i}{S_C^m(Y_i|A_i, X_i)} + \int E^m(T|T > t, A_i, X_i) \right.\right.$$
$$\left.\left. \left\{ \frac{dN_C(t)}{S_C^m(t|A_i, X_i)} + I(Y_i \geq t) \frac{dS_C^m(t|A_i, X_i)}{S_C^m(t|A_i, X_i)^2} \right\} \right] \frac{\ell\{\langle f(x_i), W_{a_i} \rangle\}}{p(A|X)} \right) + \lambda J(f)$$

Fisher's Consistency for ITR.Survival

Theorem (Fisher consistency for ITR.Survival)

A classifier $f^*(\cdot)$ is called Fisher's consistence if it satisfies that, $\forall x$,

$$\underset{\forall j}{\operatorname{argmax}} \langle f^*(x), W_j \rangle \quad = \quad \underset{\forall j}{\operatorname{argmax}}\, E(T|A=j,x)$$

ITR.Survival is Fisher consistency if ℓ is a convex, the derivative ℓ' exists and $\ell'(x) < 0, \forall x$.

Outlines

Classes of Machine Learning Algorithms

- Unsupervised learning: data driven (e.g. clustering).
- Supervised learning: task driven (e.g. classification).

Classes of Machine Learning Algorithms

- Unsupervised learning: data driven (e.g. clustering).
- Supervised learning: task driven (e.g. classification).
- Reinforcement learning.
  - it is close to mammal learning.
  - our agent has a goal to achieve.
  - our agent becomes smarter and smarter from experience interacting with environment.
  - examples include: self driven-car, AlphaGo, a humanoid walking robot.

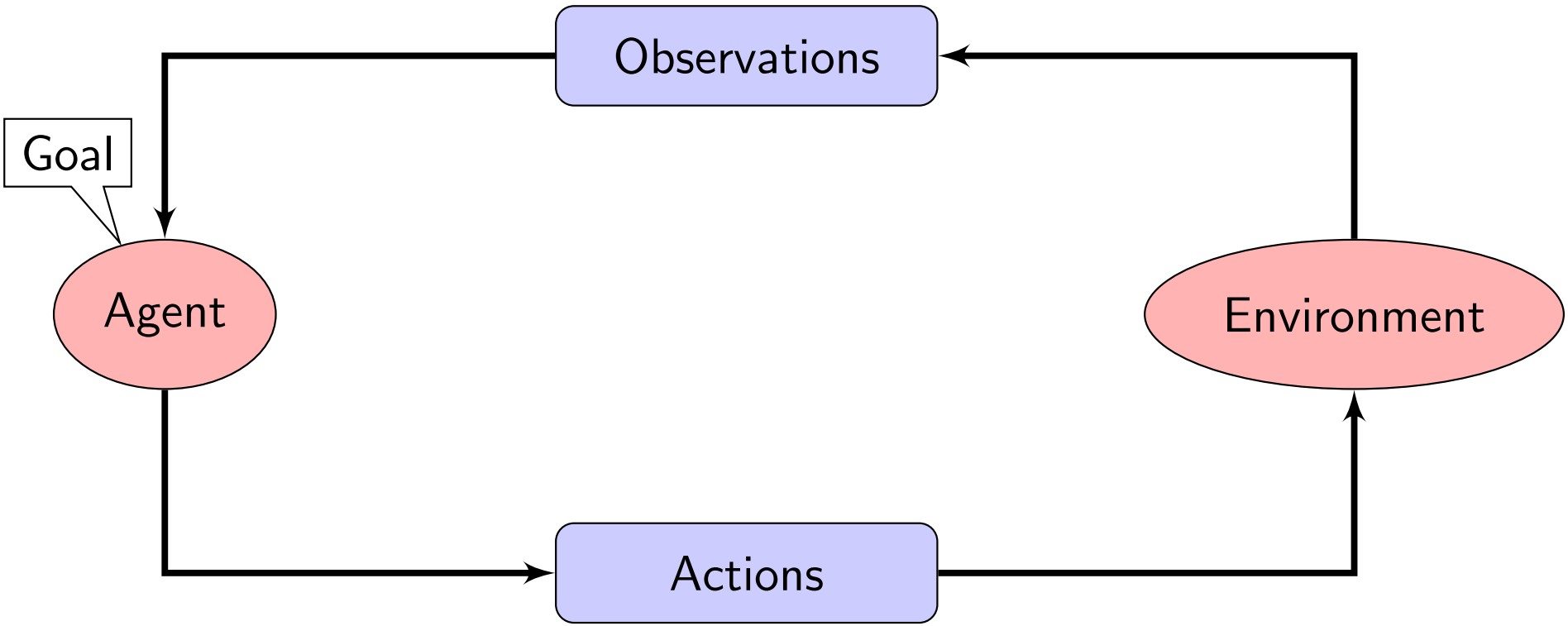


Figure 4: The agent-environment interaction in reinforcement learning.

Example Demo

A Closer Look

A tuple formulation for reinforcement learning: $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$.

- $\mathcal{S}$ is a countable set of states.
- $\mathcal{A}$ is a countable set of actions.
- $\mathcal{P}$ is a state transition probability matrix.
- $\mathcal{R}$ is a reward function.
- γ is a discount factor.

Examples: Atari games, patients' journey, closed loop control system.

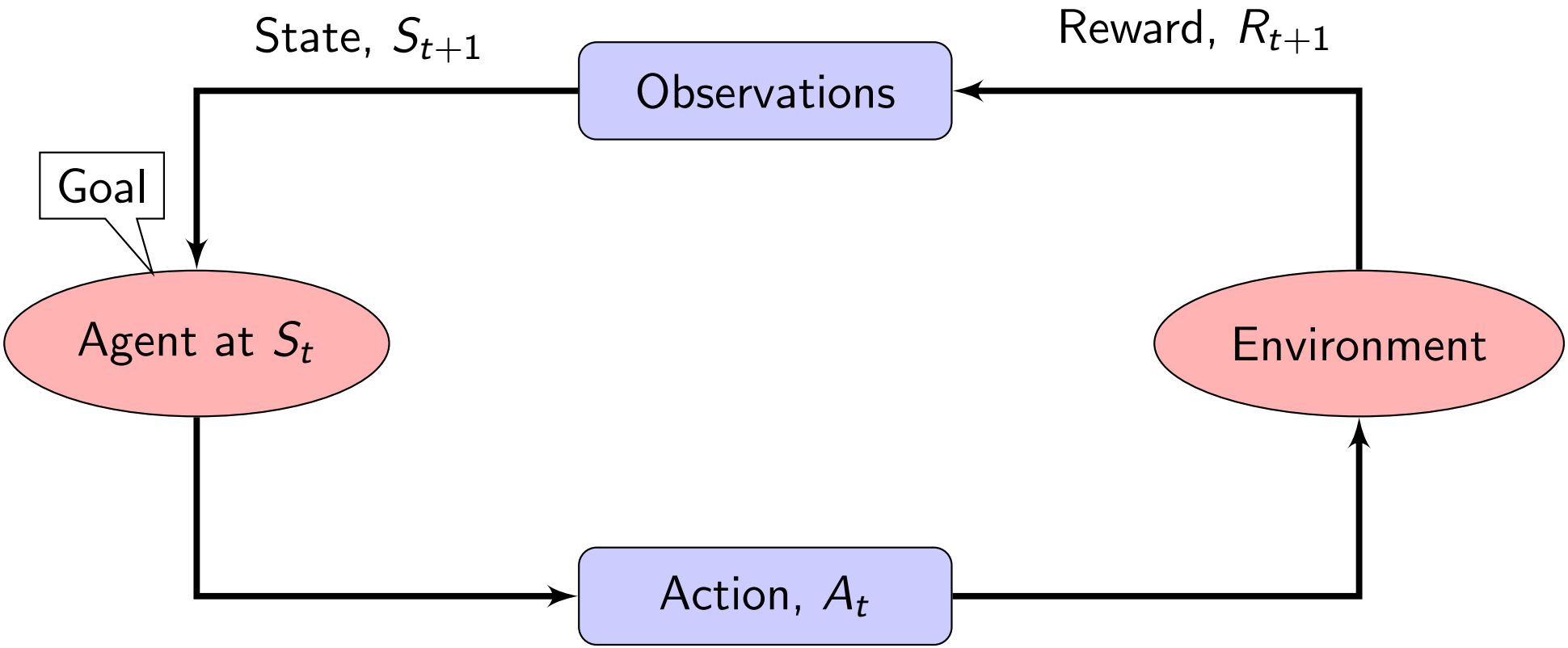


Figure 5: The agent-environment interaction in reinforcement learning.

We Can Solve the Problem for You!

Key Message

If you can formulate your research question into our tuple form, $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, we have tools to solve it for you!

Definition

- A policy π is a mapping from each state $s \in \mathcal{S}$, and action, $a \in \mathcal{A}(s)$ to the probability $\pi(a|s)$ of taking action a when in state s.

Definition

- A policy π is a mapping from each state $s \in \mathcal{S}$, and action, $a \in \mathcal{A}(s)$ to the probability $\pi(a|s)$ of taking action a when in state s.
- Value function $v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s \right]$.

Definition

- A policy π is a mapping from each state $s \in \mathcal{S}$, and action, $a \in \mathcal{A}(s)$ to the probability $\pi(a|s)$ of taking action a when in state s.
- Value function $v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s \right]$.
- Action-value function for policy π, $q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s, A_t = a \right]$.

Definition

- A policy π is a mapping from each state $s \in \mathcal{S}$, and action, $a \in \mathcal{A}(s)$ to the probability $\pi(a|s)$ of taking action a when in state s.
- Value function $v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s \right]$.
- Action-value function for policy π, $q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s, A_t = a \right]$.
- $\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s), \forall s \in \mathcal{S}$.

Definition

- A policy π is a mapping from each state $s \in \mathcal{S}$, and action, $a \in \mathcal{A}(s)$ to the probability $\pi(a|s)$ of taking action a when in state s.
- Value function $v_\pi(s) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s \right]$.
- Action-value function for policy π, $q_\pi(s, a) = E_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \middle| S_t = s, A_t = a \right]$.
- $\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s), \forall s \in \mathcal{S}$.
- Optimal state-value function $v_*(s) = \max_\pi v_\pi(s), \forall s \in \mathcal{S}$, and optimal action-value function $q_*(s, a) = \max_\pi q_\pi(s, a), \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$.

Bellman Equations

Bellman optimality equation,

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_{a \in \mathcal{A}(s)} E_{\pi_*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_t = s, A_t = a \right] \\
&= \max_{a \in \mathcal{A}(s)} E\left[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a \right] \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r | s, a)[r + \gamma v_*(s')].
\end{aligned}
$$

Bellman optimality equation for q_*,

$$
\begin{aligned}
q_*(s,a) &= E\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \middle| S_t = s, A_t = a \right] \\
&= \sum_{s',r} p(s', r | s, a)[r + \gamma \max_{a'} q_*(s', a')].
\end{aligned}
$$

Prediction and Control

- Prediction: evaluate the future.
  - given a policy.
- Control: optimise the future
  - find the best policy.

Planning and Learning

A model predicts what the environment will do next, $\mathcal{P}$ predicts the next state, $\Pr(S_{t+1}|S_t = s, A_t = a)$, and $\mathcal{R}$ predicts the next (immediate) reward, $E(R_{t+1}|S_t = s, A_t = a)$.

- Planning:
  - a model of the environment is known.
  - planning by dynamic programming.
  - prediction: iterative policy evaluation.
  - control: policy iteration and value iteration.
- Learning:
  - the environment is initially unknown.
  - the agent interacts with the environment.
  - model-free prediction: Monte-Carlo learning, TD learning, TD(λ).
  - model-free control: MC, SARSA, Q-learning, SARSA(λ).

On and Off-Policy Learning

- On-policy learning
  - "learn on the job".
  - learn about policy π from experience sampled from π.
  - example: SARSA and SARSA(λ).
- Off-policy learning
  - "look over someone's shoulder".
  - learn about policy π from experience sampled from μ.
  - example: Q-learning.

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$
- While following behavior policy $\mu(a|s)$

$$\{S_1, A_1, R_2, \cdots, S_T\} \quad \sim \quad \mu$$

Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$
- While following behavior policy $\mu(a|s)$

$$\{S_1, A_1, R_2, \cdots, S_T\} \quad \sim \quad \mu$$

- Why is this important?
  - learn from observing humans or other agents.
  - learn about multiple policies while following one policy.
  - learn about optimal policy while following exploratory policy.
  - re-use experience generated from old policies $\pi_1, \pi_2, \cdots, \pi_{t-1}$.

Highlight in this section

Planning by Dynamic Programming: Solve a known MDP.

Model free prediction: estimate the value function of an unknown MDP.

Model free control: optimise the value function of an unknown MDP.

Policy Evaluation

Algorithm 3 Iterative policy evaluation

Input: $\pi \leftarrow$ policy to be evaluated
Initialize: $V \leftarrow$ an arbitrary state-value function
repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \epsilon$ (a small positive number)
Output: $V \approx v_\pi$

Policy Improvement Theorem

Theorem (Policy Improvement Theorem)

Let π and π' be any pair of deterministic policies, we have,

$$q_\pi\{s, \pi'(s)\} \geq v_\pi(s), \forall s \in \mathcal{S} \quad \Rightarrow \quad v_{\pi'}(s) \geq v_\pi(s), \forall s \in \mathcal{S}.$$

Proof:

$$\begin{aligned}
v_\pi(s) &\leq q_\pi\{s, \pi'(s)\} \\
&= E_{\pi'}\{R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\} \\
&\leq E_{\pi'}\left[R_{t+1} + \gamma q_\pi\{S_{t+1}, \pi'(S_{t+1})\} | S_t = s\right] \\
&= E_{\pi'}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s\right\} \\
&\vdots \\
&\leq E_{\pi'}\left\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots | S_t = s\right\} \\
&= v_{\pi'}(s).
\end{aligned}$$

Policy Iteration

Policy Iteration

Let $\xrightarrow{E}$ to denote policy evaluation, and $\xrightarrow{I}$ to denote policy improvement. We have,

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*.$$

Remark: one drawback to policy iteration is that each of its iteration involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set.

Policy Iteration

Algorithm 4 Policy iteration

Initialization: $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
Initialize an array $V(s) = 0, \forall s \in \mathcal{S}$
repeat
  Policy Evaluation
  repeat
    $\Delta \leftarrow 0$
    for each $s \in \mathcal{S}$ **do**
      $v \leftarrow V(s)$
      $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
  until $\Delta < \epsilon$ (a small positive number)
  Policy Improvement
  Policy-stable $\leftarrow$ *true*
  for each $s \in \mathcal{S}$ **do**
    $a \leftarrow \pi(s)$
    $\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    if $a \neq \pi(s)$ **then** *Policy-stable* $\leftarrow$ *false*
    end if
  end for
until *Policy-stable*
Output: V and π

Remark: This algorithm has a subtle bug, in that it may never terminate if the policy continually switches between two or more polices that are equally good. The bug can be fixed by adding additional flags, but it makes the pseudo code so ugly that it is not worth it.

Value Iteration

Value iteration combines the policy improvement and truncated policy evaluation steps:

$$\begin{aligned} v_{k+1}(s) &= \max_a E\{R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a\} \\ &= \max_a \sum_{s',r} p(s', r|s, a)\{r + \gamma v_k(s')\}. \end{aligned}$$

In fact, the policy evaluation step of policy iteration can be truncated in several ways without losing the convergence guarantees of policy iteration. The above is a special case that policy evaluation is stopped just after one sweep (one backup of each state).

Value Iteration

Algorithm 5 Value iteration

Initialize an array $V(s) = 0, \forall s \in \mathcal{S}$
repeat
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\{r + \gamma V(s')\}$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \epsilon$ (a small positive number)
Output: a deterministic policy π, such that,

$$\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\{r + \gamma v_k(s')\}.$$

Monte-Carlo Policy Evaluation for v_π

- Goal: learn v_π from episodes of experience under policy π,

$$S_1, A_1, R_2, \cdots, R_T, S_T \quad \sim \quad \pi.$$

- Recall that the *return* is the total discounted reward:

$$G_t \quad = \quad R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$$

- Recall that the value function is the expected return:

$$v_\pi(s) \quad = \quad E_\pi(G_t | S_t = s).$$

- Monte-Carlo policy evaluation uses empirical mean return instead of expected return

Incremental Mean

To save memory, the mean can be calculated more efficiently. The mean $\mu_1, \mu_2, \cdots$ of a sequence $Y_1, Y_2, \cdots$ can be computed incrementally,

$$\begin{aligned}
\mu_n &= \frac{1}{n}\sum_{i=1}^{n} Y_i \\
&= \frac{1}{n}\left(Y_n + \sum_{i=1}^{n-1} Y_i\right) \\
&= \frac{1}{n}\{Y_n + (n-1)\mu_{n-1}\} \\
&= \mu_{n-1} + \frac{1}{n}(Y_n - \mu_{n-1}).
\end{aligned}$$

Monte-Carlo Method for Evaluation v_π

Algorithm 6 Monte-Carlo method for estimating v_π

Input: $\pi \leftarrow$ policy to be evaluated
Initialize: $V \leftarrow$ an arbitrary state-value function
Generate N (a large number) episodes of experience from π
for each state $s \in \mathcal{S}$ **do**
    for $N(s) \leq N$ **do**
        $N(s) \leftarrow N(s) + 1$
        G_t is calculated for the first (or every) visit of s
        $V(s) \leftarrow V(s) + \frac{1}{N(s)}\{G_t - V(s)\}$
        (Or $V(s) \leftarrow V(s) + \alpha\{G_t - V(s)\}$ i.e. to forget old episodes)
    end for
end for
Output: $v_\pi \approx V$

Temporal-Difference Learning

- MC update: $V(S_t) \leftarrow V(S_t) + \alpha\{G_t - V(S_t)\}$
- TD(0) update: $V(S_t) \leftarrow V(S_t) + \alpha\{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\}$
- $R_{t+1} + \gamma V(S_{t+1})$ is called the TD target
- $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error

Temporal-Difference Learning to Evaluate v_π

Algorithm 7 Tabular TD(0) for evaluating v_π

Input: $\pi \leftarrow$ policy to be evaluated
Initialize: $V \leftarrow$ an arbitrary state-value function
Generate N (a large number) episodes of experience from π
for each episode **do**
    Initialize S
    for each step of episode **do**
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\{R + \gamma V(S') - V(S)\}$
        $S \leftarrow S'$
    end for
end for
Output: $v_\pi \approx V$

TD(λ)

- Consider the following n-step returns:

$$\begin{array}{lll} n=1 & \text{TD(0)} & G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1}) \\ n=2 & & G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2}) \\ \vdots & & \vdots \\ n=\infty & \text{MC} & G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T \end{array}$$

- n-step TD learning

$$V(S_t) \leftarrow V(S_t) + \alpha\{G_t^{(n)} - V(S_t)\}$$

- TD(λ) (forward view),

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$$

$$V(S_t) \leftarrow V(S_t) + \alpha\{G_t^{\lambda} - V(S_t)\}$$

- TD(λ) forward-view is easy to understand but difficult to compute

Eligibility Trace and Backward View of TD(λ)

$E_t(s)$ is called eligibility trace, and it can be used for the update as below,

$$\begin{aligned} E_0(s) &= 0 \\ E_t(s) &= \gamma\lambda E_{t-1}(s) + \mathbb{I}(S_t = s) \\ \delta_t &= R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \\ V(s) &\leftarrow V(s) + \alpha\delta_t E_t(s) \end{aligned}$$

Theorem (Backward-view TD(λ))

The sum of offline updates is identical for forward-view and backward-view TD(λ),

$$\sum_{t=1}^{T} \alpha\delta_t E_t(s) = \sum_{t=1}^{T} \alpha\{G_t^\lambda - V(S_t)\}\mathbb{I}(S_t = s)$$

TD(λ) to Evaluate v_π

Algorithm 8 On-line tabular TD(λ) evaluating v_π

Input: $\pi \leftarrow$ policy to be evaluated
Initialize: $V \leftarrow$ an arbitrary state-value function (set 0 for terminal state)
Generate N (a large number) episodes of experience from π
for each episode **do**
    Initialize $E(s) = 0, \forall s \in \mathcal{S}$
    Initialize S
    for each step of episode **do**
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $\delta \leftarrow R + \gamma V(S') - V(S)$
        $E(S) \leftarrow E(S) + 1$
        for $s \in \mathcal{S}$ **do**
            $V(S) \leftarrow V(S) + \alpha\delta E(s)$
            $E(s) \leftarrow \gamma\lambda E(s)$
        end for
        $S \leftarrow S'$
    end for
end for
Output: $v_\pi \approx V$

Monte Carlo Control

Without a model, state values alone are not sufficient. One must explicitly estimate the value of each action in order for the values to be useful in suggesting a policy. Thus, one of primary goals for Monte Carlo methods is to estimate q_*.

Monte Carlo Policy Iteration

Let $\xrightarrow{E}$ to denote policy evaluation, and $\xrightarrow{I}$ to denote policy improvement. We have,

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} q_*.$$

To maintain exploration, i.e. visit every state-action pairs, we can use

- Exploring starts: every pair has a nonzero probability of being selected as the start.
- The ϵ-greedy policy: $\pi(a|s) = \epsilon/|\mathcal{A}(s)|$ for all non-greedy actions, and $\pi(a|s) = 1 - \epsilon + \epsilon/|\mathcal{A}(s)|$ for the greedy action.

Improvement is guaranteed by policy improvement theorem.

Monte Carlo Control with Exploratory Starts

Algorithm 9 Monte Carlo control with exploratory starts

Initialize: $Q(s,a) \leftarrow$ arbitrary, $\pi(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$.
repeat
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0.
  Generate an episode starting from S_0, A_0, following π.
  for each pair s, a appearing in the episode **do**
    $N(s,a) \leftarrow N(s,a) + 1$
    G_t is calculated for the first (or every) occurrence of s, a
    $Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)}\{G_t - V(s,a)\}$
    or $Q(s,a) \leftarrow Q(s,a) + \alpha\{G_t - Q(s,a)\}$
  end for
  for each s in the episode **do**
    $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$
  end for
until Converge
Output: $q_* \leftarrow Q$

Monte Carlo Control with ϵ-soft Policies

Algorithm 10 Monte Carlo control with ϵ-Soft policies

Initialize: $Q(s,a) \leftarrow$ arbitrary, $\pi(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$.
repeat
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0.
  Generate an episode starting from S_0, A_0, following π.
  for each pair s, a appearing in the episode **do**
    $N(s,a) \leftarrow N(s,a) + 1$
    G_t is calculated for the first (or every) occurrence of s, a
    $Q(s,a) \leftarrow Q(s,a) + \frac{1}{N(s,a)}\{G_t - Q(s,a)\}$
    or $Q(s,a) \leftarrow Q(s,a) + \alpha\{G_t - Q(s,a)\}$
  end for
  for each s in the episode **do**
    $A^* \leftarrow \arg\max_a Q(s,a)$
    for each $a \in \mathcal{A}(s)$ **do**

$$\pi(a|s) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \epsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

    end for
  end for
until Converge
Output: $q_* \leftarrow Q$

SARSA: On-Policy Temporal-Difference Control

Algorithm 11 SARSA: on-policy TD control

Initialize: $Q(s, a) \leftarrow$ arbitrary, $\pi(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ and $Q(s_T, \cdot) = 0$.
repeat(for each episode)
    Initialize S
    Choose A from S using policy derived from Q, e.g. ϵ-greedy
    repeat(for each step of episode)
        Take action A, observe R and S'
        Choose A' from S' using policy derived from Q, e.g. ϵ-greedy
        $Q(S, A) \leftarrow Q(S, A) + \alpha\{R + \gamma Q(S', A') - Q(S, A)\}$
        $S \leftarrow S'$ and $A \leftarrow A'$
    until S is terminal
until Converge
Output: $q_* \leftarrow Q$

SARSA(λ) Algorithm

Algorithm 12 SARSA(λ) control

Initialize: $Q(s,a) \leftarrow$ arbitrary, $\pi(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ and $Q(s_T, \cdot) = 0$.
repeat(for each episode)
  Initialize S
  Choose A from S using policy derived from Q, e.g. ϵ-greedy
  repeat(for each step of episode)
    Take action A, observe R and S'
    Choose A' from S' using policy derived from Q, e.g. ϵ-greedy
    $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
    $E(S, A) \leftarrow E(S, A) + 1$
    for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$ **do**
      $Q(s, a) \leftarrow Q(s, a) + \alpha\delta E(s, a)$
      $E(s, a) \leftarrow \gamma\lambda E(s, a)$
    end for
    $S \leftarrow S'$ and $A \leftarrow A'$
  until S is terminal
until Converge
Output: $q_* \leftarrow Q$

Importance Sampling for Off-Policy Learning

Estimate the expectation of a different distribution,

$$\begin{aligned} E_{X\sim\mathcal{P}}[f(X)] &= \int f(x)d\mathcal{P} \\ &= \int f(x)\frac{d\mathcal{P}}{d\mathcal{Q}}d\mathcal{Q} \\ &= E_{X\sim\mathcal{Q}}\left[f(X)\frac{\mathcal{P}(X)}{\mathcal{Q}(X)}\right]. \end{aligned}$$

Application: a single importance sampling correction,

$$V(S_t) \leftarrow V(S_t) + \alpha\left[\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\{R_{t+1} + \gamma V(S_{t+1})\} - V(S_t)\right],$$

where μ is behavior policy and π is target policy.

Q-Learning

- Now we consider off-policy learning of action-values $Q(s, a)$
- No important sampling is required
- Next action is chosen using behavior policy $A_{t+1} \sim \mu(\cdot|S_t)$
- But we consider alternative successor action $A' \sim \pi(\cdot|S_t)$
- And update $Q(S_t, A_t)$ towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\{R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)\}$$

- A special case is to allow both behavior and target policy to improve
- The target policy π is greedy w.r.t. $Q(s, a)$,

$$\pi(S_{t+1}) = \underset{a'}{\text{argmax}}\, Q(S_{t+1}, a')$$

- The behavior policy μ is e.g. ϵ-greedy w.r.t. $Q(s, a)$

Q-Learning Algorithm for Off-Policy Control

Algorithm 13 Q-Learning

Initialize: $Q(s,a) \leftarrow$ arbitrary, $\pi(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ and $Q(s_T, \cdot) = 0$.
repeat(for each episode)
    Initialize S
    repeat(for each step of episode)
        Choose A from S using policy derived from Q, e.g. ϵ-greedy
        Take action A, observe R and S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha\{R + \gamma \max_a Q(S',a) - Q(S,A)\}$
        $S \leftarrow S'$
    until S is terminal
until Converge
Output: $q_* \leftarrow Q$

Large Scale Reinforcement Learning

Reinforcement learning can be used for solving large scale problem, for example:

- Gomoku (five stones): 10^{50} states.
- Computer Go game (Weiqi): 10^{170} states.
- Helicopter: continuous state space.

How can we scale up the model-free methods for prediction and control?

Value Function Approximation

The objective function is defined as the Mean Squared Value Error, or MSVE:

$$\text{MSVE}(\theta) \doteq \sum_{s \in \mathcal{S}} d(s) \left\{ v_\pi(s) - \widehat{v}(s, \theta) \right\},$$

where $d(s)$ is the fraction of times spent in s under the target police π which is often referred to as the on-policy distribution.
Parameter updates:

$$\theta \quad \leftarrow \quad \theta + \alpha \left\{ v_\pi(S) - \widehat{v}(S, \theta) \right\} \nabla \widehat{v}(S, \theta).$$

Incremental Prediction and Control Algorithms

- Have assumed true value function $v_\pi(s)$ given by supervisor.
- But in RL there is no supervisor, only rewards
- In practice, we substitute a target for $v_\pi(s)$:
  - For MC, the target is the return G_t,

$$\Delta\theta \quad \leftarrow \quad \alpha \{G_t - \widehat{v}(S_t, \theta)\} \nabla \widehat{v}(S_t, \theta).$$

  - for TD(0), the target is the TD target $R_{t+1} + \gamma \widehat{v}(S_{t+1}, \theta)$,

$$\Delta\theta \quad \leftarrow \quad \alpha \{R_{t+1} + \gamma \widehat{v}(S_{t+1}, \theta) - \widehat{v}(S_t, \theta)\} \nabla \widehat{v}(S_t, \theta).$$

  - For TD(λ), the target is the $\lambda-$return G_t^λ,

$$\Delta\theta \quad \leftarrow \quad \alpha \left\{G_t^\lambda - \widehat{v}(S_t, \theta)\right\} \nabla \widehat{v}(S_t, \theta).$$

- For control problems, approximate the action-value value by $\widehat{Q}(S, A, \theta) \approx Q(S, A)$.

Value-Based and Policy-Based RL

- Value Based
  - Learn value function
  - implicity policy (e.g. $\epsilon-$greedy)
- Policy Based
  - No value function
  - Learn policy
- Actor-Critic
  - learn value value function
  - learn policy

Policy Gradient Methods

- Model policy as:

$$\pi(a|s,\theta) \doteq \Pr(A_t = a|S_t = s, \theta_t = \theta).$$

- For some performance measure $\eta(\theta)$ with respect to the policy parameters θ,

$$\theta \leftarrow \theta + \alpha\nabla\widehat{\eta}(\theta).$$

- For example, the performance measurement can be,

$$\eta(\theta) = v_{\pi_\theta}(s).$$

- Policy gradient theorem,

$$\nabla\eta(\theta) = \sum_s d_\pi(s) \sum_a q_\pi(s,a)\nabla_\theta\pi(a|s,\theta).$$

Reducing Variance Using a Critic

- Monte-Carlo policy gradient still has high variance
- We use a critic to estimate the action value function,

$$Q(s, a, \omega) \quad = \quad Q_{\pi_\theta}(s, a).$$

- Actor-critic algorithms maintain two sets of parameters
  - Critic: update action-value function parameter ω
  - Actor: update policy parameters θ, in direction suggested by critic

Summary

1. Precision Medicine
2. Individualized Treatment Recommendation Framework
3. Support Vector Machines, Angel Based Classifiers, and Outcome Weighted Learning in Reproducing Kernel Hilbert Spaces
4. Reinforcement Learning and Multi-Stage Decision Making

Thank You!!