MATH (160)

Mathematics for Data Analytics and Decision Making

Jesus De Loera

UC Davis, Mathematics

Monday, March 28, 2011

Syllabus, Class Schedule, Office Hours, Textbook, etc.

See course website.

`https://www.math.ucdavis.edu/~deloera/TEACHING/MATH160/`

Also your email for announcements and quizzes!

This course is about...Thinking before deciding

MATHEMATICAL MODELS!!!

That use data sets to make intelligent decisions!!

By the end of this course, you should feel ready to...

MULTITRANS is a fictional bus company that is run by a student association in an unnamed college town with partial funding from the city.

(It started out in 1972 with a fleet of historic triple-deck buses from Moscow, but has added more modern buses since.)

During most of the day, though, all scheduled buses are completely empty.

A scientific study was conducted in 2010 to find out why this is the case. The result of the study was:

> *[...] 85% of the representative sample of 1256 potential users of MULTITRANS replied that "I would use it regularly instead of using my bike or car, but the MULTITRANS schedule [is suboptimal]." [...]*

MULTITRANS hires you as a consultant to help them improve the schedule.

What are the next steps you need to take to solve the challenge?

Let's start with a simpler example, though

The Notip Table Company sells two models of its patented five-leg tables. The basic version uses a wood top, requires 0.6 hours to assemble, and sells for a profit of $200. The deluxe model takes 1.5 hours to assemble (because of its glass top), and sells for a profit of $350. Over the next week the company has 300 legs, 50 wood tops, 35 glass tops, and 63 hours of assembly available. Notip wishes to determine a maximum profit production plan assuming that everything produced can be sold.

We want a **mathematical model** for this problem.

We are using two variables to represent the decision on the production quantities:

x_1 – *number of basic tables,* x_2 – *number of deluxe tables*

$$
\begin{array}{lll}
\max & 200x_1 + 350x_2 & \text{(Total profit)} \\
\text{s.t.} & x_1 \leq 50 & \text{(Wood tops available)} \\
& x_2 \leq 35 & \text{(Glass tops available)} \\
& 5x_1 + 5x_2 \leq 300 & \text{(Legs available)} \\
& 0.6x_1 + 1.5x_2 \leq 63 & \text{(Hours of assembly time available)} \\
& x_1, x_2 \geq 0 & \text{(Non-negative quantities to be produced)} \\
& x_1, x_2 \in \mathbf{Z} & \text{(Integer quantities to be produced)}
\end{array}
$$

We study Mathematical Optimization Problems!

Mathematical Optimization Problem, abstract definition

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in F \subseteq X \end{aligned}$$

Here

- "max" means "maximize!"
- f is a function: the *objective function*
- "s.t." stands for "subject to (constraints)" or "such that"
- X is some set, the *space of solutions*
- $F \subseteq X$ is a subset: the set of all *feasible solutions*
- $\mathbf{x}$ is one feasible solution

F can be finite or infinite.

(Which optimization problems are easier to solve?)

Standard Form of Deterministic, Finite-Dimensional Mathematical Optimization Problems

$$
\begin{aligned}
\max(\min) \quad & f(\mathbf{x}) \\
\text{s.t.} \quad & g_1(\mathbf{x}) \leq 0 \\
& \vdots \\
& g_m(\mathbf{x}) \leq 0 \\
& \mathbf{x} = (x_1, \ldots, x_n) \in X
\end{aligned}
$$

where $f, g_1, \ldots, g_m \colon \mathbf{R}^n \to \mathbf{R}$ are functions and $X = \mathbf{R}^n$, or $X = \mathbf{Z}^n$, or $X = \mathbf{R}^{n_1} \times \mathbf{Z}^{n_2}$.

We classify optimization problems according to the properties of the functions f and g_i and the space X.

The problem with Brute Force

To solve a problem in practice on a computer, it is not enough to know that there exists an algorithm, i.e., a finite procedure.

It easily happens that the brute force method is not useful at all because for most interesting problems it takes too long, even on very fast computers.

Large bounds

If we have three integer variables x_1, x_2, x_3 with bounds $1 \leq x_i \leq 1\,000\,000$, we would have to check 10^{18} solutions!
If a computer with 8 processor cores running at $2.5\,\mathrm{GHz}$ could check one solution in one cycle, it would still take $10^{18}/(8 \cdot 2.5 \cdot 10^9) = 50 \cdot 10^6$ seconds (more than 1.5 years). This may be too long!

High dimension

Impressive examples appear when we have many variables, even if they have low bounds.In a combinatorial problem with 100 binary variables ($x_i = 0$ or $x_i = 1$), we would have to check 2^{100} solutions.A current supercomputer, IBM's Blue Gene/P, has 884,736 processors running at 850 MHz. It would take $1.6 \cdot 10^{15}$ years.The age of the universe is only $(13.7 \pm 0.1) \cdot 10^9$ years (Wikipedia).

How to solve this problem, II: Graphing

An important observation is that we can gain insight by graphing the example problem – because it only has 2 variables.

In R^2, we see that that each of the linear inequalities corresponds to a **half space**. Their intersection forms a **convex polygon** (a shape with finitely many corners, connected by straight edges), which we can easily draw.

The role of the objective function is slightly more complicated. For any $\alpha \in \mathbf{R}$, consider the **level set**

$$L_\alpha = \{\mathbf{x} = (x_1, x_2) \in \mathbf{R}^2 : 200x_1 + 350x_2 = \alpha\},$$

that is, the set of points of the plane where the objective function takes the value α. For each α, the level set is a line, and all the level sets are parallel to each other.

We now introduce vector notation. Let $\mathbf{a} = (200, 350)$. Then the equation of the level set reads $\langle \mathbf{a}, \mathbf{x} \rangle = \alpha$, where the left-hand side is the **inner product** (scalar product, dot product) of $\mathbf{a}$ and $\mathbf{x}$. (You may know the notation $\mathbf{a} \cdot \mathbf{x}$, or $\vec{a} \cdot \vec{x}$, or $\mathbf{a}^\top \mathbf{x}$ (assuming $\mathbf{a}$ and $\mathbf{x}$ are written as column vectors).

Geometry of the problem

Now let $\mathbf{x}^0$ be any point in $\mathbf{R}^2$. Set $\alpha = 200x_1^0 + 350x_2^0$, then the point $\mathbf{x}^0$ lies on the level set L_α. Then we can write

$$\begin{aligned} L_\alpha &= \{\, \mathbf{x} = (x_1, x_2) \in \mathbf{R}^2 : \langle \mathbf{a}, \mathbf{x} \rangle = \langle \mathbf{a}, \mathbf{x}^0 \rangle \,\} \\ &= \{\, \mathbf{x} = (x_1, x_2) \in \mathbf{R}^2 : \langle \mathbf{a}, \mathbf{x} - \mathbf{x}^0 \rangle = 0 \,\}. \end{aligned}$$

Both equations will be called the **normal equation** of the line L_α.
Since a zero inner product indicates **orthogonality**, this means:

> *The level set consists of all points* $\mathbf{x}$ *such that the vector* $\mathbf{x} - \mathbf{x}^0$ *is orthogonal (**normal**) to the vector* $\mathbf{a}$.

Geometry of the problem, II

Now the optimization problem, if we **ignore the integrality constraints**, has the following **geometric interpretation**:

> *Among all the parallel lines L_α, find the one with the largest α such that the intersection of L_α with the feasible region (convex polygon) is nonempty.*

In our problem, we find that there is a unique optimal solution: For the optimal α, the intersection consists of a single point. (If we increase α even just a little bit beyond this, we push the line out of the feasible region.)
However, if we change the objective function a bit, it could happen that it is parallel to an edge, and then the intersection would then be the entire edge. Then the optimal solution is not uniquely determined; all solutions on that edge have the same objective function value and are thus optimal.

Geometry of the problem, III

This generalizes to **higher dimensions**:
In $\mathbf{R}^3$, the level set

$$\begin{aligned} L_\alpha &= \{\, \mathbf{x} = (x_1, x_2, x_3) \in \mathbf{R}^3 : \langle \mathbf{a}, \mathbf{x} \rangle = \langle \mathbf{a}, \mathbf{x}^0 \rangle \,\} \\ &= \{\, \mathbf{x} = (x_1, x_2) \in \mathbf{R}^2 : \langle \mathbf{a}, \mathbf{x} - \mathbf{x}^0 \rangle = 0 \,\}. \end{aligned}$$

is a plane; we thus see the normal equations of a plane in $\mathbf{R}^3$.
Then, for higher dimensions, we **define**:

Definition

A **hyperplane** in $\mathbf{R}^n$ is a set that can be described as

$$\{\, \mathbf{x} \in \mathbf{R}^n : \langle \mathbf{a}, \mathbf{x} - \mathbf{x}^0 \rangle = 0 \,\} \quad \text{for some } \mathbf{a} \in \mathbf{R}^n,\ \mathbf{a} \neq \mathbf{0},\ \mathbf{x}^0 \in \mathbf{R}^n,$$

or, equivalently, as

$$\{\, \mathbf{x} \in \mathbf{R}^n : \langle \mathbf{a}, \mathbf{x} \rangle = \alpha \,\} \quad \text{for some } \mathbf{a} \in \mathbf{R}^n,\ \mathbf{a} \neq \mathbf{0},\ \alpha \in \mathbf{R}.$$

Hyperplanes are $(n-1)$-dimensional objects in $\mathbf{R}^n$.

How to solve this problem, III: Using optimization software

We will be using three pieces of software in this lecture. (All of them are free at least for noncommercial and academic use. ZIMPL is free (open source) software.)

SCIP – "Solving Constraint and Integer Programs"

SCIP is a very good solver for **Mixed Integer Linear Optimization Problems**, i.e.,

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0 \qquad \text{for } i = 1, \ldots, m \\ & \mathbf{x} = (x_1, \ldots, x_n) \in X \end{aligned}$$

where f and g_i are linear functions, and $X = \mathbf{R}^n$, or $X = \mathbf{Z}^n$, or $X = \mathbf{R}^{n_1} \times \mathbf{Z}^{n_2}$.

SoPLEX – a solver for linear optimization problems

A super-fast solver for linear optimization problems ($X = \mathbf{R}^n$) is the basic building block of optimization and operations research technology.

ZIMPL – an algebraic modeling language for optimization

ZIMPL provides a convenient way to write down optimization problems in the computer.

SCIP uses both SoPLEX and ZIMPL internally.

The ZIMPL language: Variable definitions

A complete description is found in section 4.4 in ZIMPL User Guide.

Variable type:

`var NAME integer;`	Defines an integer variable
`var NAME binary;`	Defines a binary (0/1) variable
`var NAME real;`	Defines a real variable
`var NAME;`	Also defines a real variable

All variables (except binary) have a lower bound of 0 and no upper bound. If different upper bounds are needed, change the variable definition:

`var NAME integer >= -5 <= 5`	Defines an integer variable that can take values from -5 to 5
`var NAME real >= -infinity <= infinity`	Defines a free variable that can take arbitrary real values

The ZIMPL language: Objective and constraints

The objective function is written as:

```
maximize NAME: TERM;
minimize NAME: TERM;
```

where TERM is an arbitrary (linear) expression.

Each of the constraints is written as:

```
subto NAME: TERM SENSE TERM
```

Here SENSE is one of $<=$, $>=$, $==$.

(The ZIMPL language has a much greater power than this; we'll dive into its complexity gradually.)

The ZIMPL language: Objective and constraints

The objective function is written as:

```
maximize NAME: TERM;
minimize NAME: TERM;
```

where TERM is an arbitrary (linear) expression.

Each of the constraints is written as:

```
subto NAME: TERM SENSE TERM
```

Here SENSE is one of $<=$, $>=$, $==$.

(The ZIMPL language has a much greater power than this; we'll dive into its complexity gradually.)

Our example problem in ZIMPL

ZIMPL input files are plain text files; use a text editor (such as *Emacs* or *gedit*) to create them, not a word processor.

```
var basictables integer;
var deluxetables integer;

maximize profit:
200*basictables + 350*deluxetables;

subto legsconstraint:
5*(basictables + deluxetables) <= 300;

subto woodtopsconstraint:
basictables <= 50;

subto glasstopsconstraint:
deluxetables <= 35;

subto assemblyhoursconstraint:
0.6*basictables + 1.5*deluxetables <= 63;
```

Supervised Learning and Linear Models

- NOW let us use Optimization models for problems in DATA MINING!!
- We have data points $x_1, x_2, \ldots, x_m$ each has n input variables x_{ij}, also called **features**.
  Example: measurements such as living-space area, number of rooms, etc.
- We have y_i to denote the response, training or target variable that we are trying to predict.
  Example: We wish to predict the price of the house.
- A pair (x_i, y_i) is called *a training example*. The dataset that we use to learna list of m training examples. $(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)$ is called a *training set* We will also use X denote the space of input values, and Y the space of output values.
- **GOAL** given a training set, to *learn* a function $h : X \rightarrow Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y.
- When target variables are continuous, we call the supervised learning problem a *regression problem*. When y can take on discrete values (want to predict if a dwelling is a house or an apartment), we call it a *classification problem*.
  To perform supervised learning, we must decide how to represent functions/hypotheses h in a computer!

Regression analysis

- LINEAR CHOICE: Approximate y as a linear function of x:

$$h(x) = \sum_{k}^{n} w_i x_{ik} = w^T x_i$$

Formally we wish to find a function that measures, for each value of the w, how close the $h(x_i)$s are to the corresponding y_is. We define the cost function that minimizes the error.

$$\min \sum_{i}^{m} (h(x_i) - y_i)^2 = \min_{w \in \mathbf{R}^m} ||X^T w - \hat{y}||_2$$

We use the notation X to denote the $m \times n$ matrix whose rows are the data points x_i. Let $\hat{y}$ the row vector with y_i's. This yields the traditional LEAST SQUARES PROBLEM:

- **Proposition** The optimal solution is the solution of a linear system of equations

$$X^T X w = X^T \hat{y}$$

Sometimes a hyperplane does not fit the data well!

- POLYNOMIAL CHOICE: We may add nonlinear relations on the features by considering a polynomial of degree M over n variables as the learning function:

$$h(x) = \sum_{a_1+a_2+\cdots+a_n=M} c_{i_1,i_2,\ldots,i_n} x_{i1}^{a_1} x_{i2} \cdots x_{in}^{a_n}$$

- Because the monomials can be used to represent all such polynomials as a vector space basis (of which dimension??) We still have a least squares problem over a matrix $\Phi(X)$ with m rows and r rows. This is a **polynomial learning model of degree M**

$$\min \sum_{i}^{m} (h(x_i) - y_i)^2 = \min_{w \in \mathbf{R}^m} ||\Phi(X)^T w - \hat{y}||_2$$

- Naively, it might seem that the more features we add, the better. However, there is a danger in adding too many features! We have to be careful with **underfitting** or *overfitting*.

Evaluating the regression result

- How to compare different regression models? How good is prediction?
- **Leave-one-out approach** Instead of using the entire data set to fit our model we al but one point x_j Missing point will be our test
- Solve the smaller Least squares problem, we get a prediction for x_j, $\hat{y}(x_j)$. We can repeat the above process for all points (leave a different one out). Compute the *test set error*:

$$1/m \sum_{j=1}^{m} (\hat{y}(x_j) - y_j)$$

- As the degree M of the polynomial grows the training error decreases, while the test set error typically decreases and then then increases. This is *overfitting*
- Models of higher M complexity fit the training data better, but a model that is too complex (high M) does not behave well on unseen data. We are looking for a compromise.
- **leave-more out**

Regularization

- Suppose we chose a polynomial model of degree M. Still, there are many such polynomials!! Some with very large values of the derivatives!!
- Complexity of the polynomial model depends also on the size of the coefficients!

Theorem

Any univariate polynomial of $p(x) = w_0 + w_1 x + w_2 x^2 + \dots w_M x^M$ *that takes values* x *between* $-1, 1$ *we have*

$$|dp(x)/dx| \leq M^{3/2} ||w||_2$$

Thus if data is bounded in size the can be a way to control the derivatives of the polynomial by controlling the norm of w.

- **Key point:** add penalty term to the original formulation!! **regularized Least Squares problem**

$$\min_{w \in \mathbf{R}^m} ||\Phi(X)^T w - \hat{y}||_2 + \lambda ||w||_2^2$$

Sparse Regression!

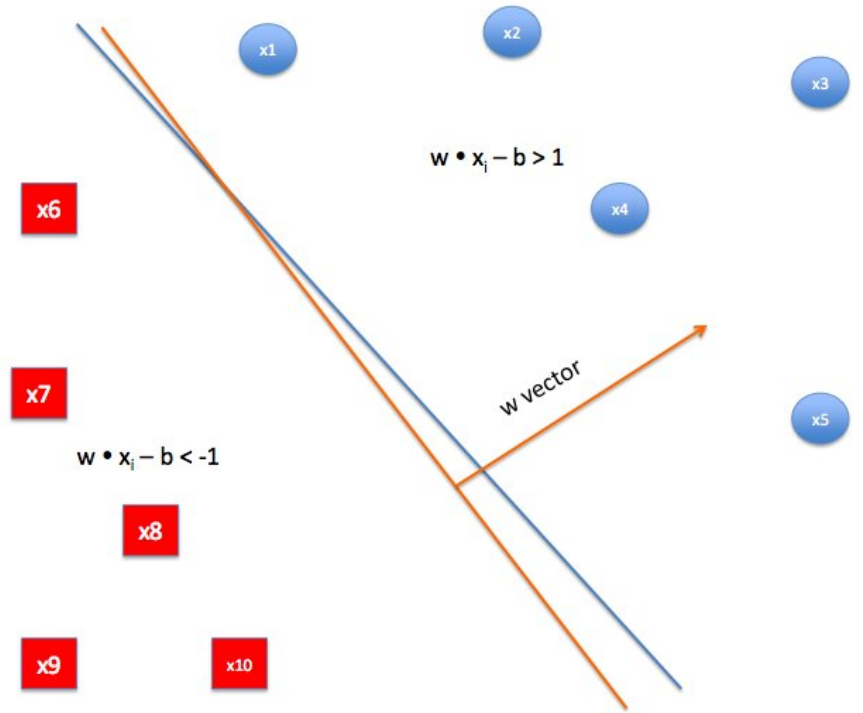
- Instead of taking the 2-norm of the vector w we could have asked for the 1-norm $||w||_1 = \sum |w_i|$
- LASSO problem

$$\min_{w \in \mathbf{R}^m} ||\Phi(X)^T w - \hat{y}||_2 + \lambda ||w||_1$$

- The good news is that the solution of this problem will force a lot of entries to be zero in w. We obtain a sparse solution!
- Why is this interesting? The w_i can be thought of as key reasons explaining the data (e.g. the temperature and humidity are the most essential pieces of data to explain weather!). We want to find the **most significant information from data!**.
- PRACTICAL ISSUES: How to choose λ? How to choose M? How do we solve such problem in practice?

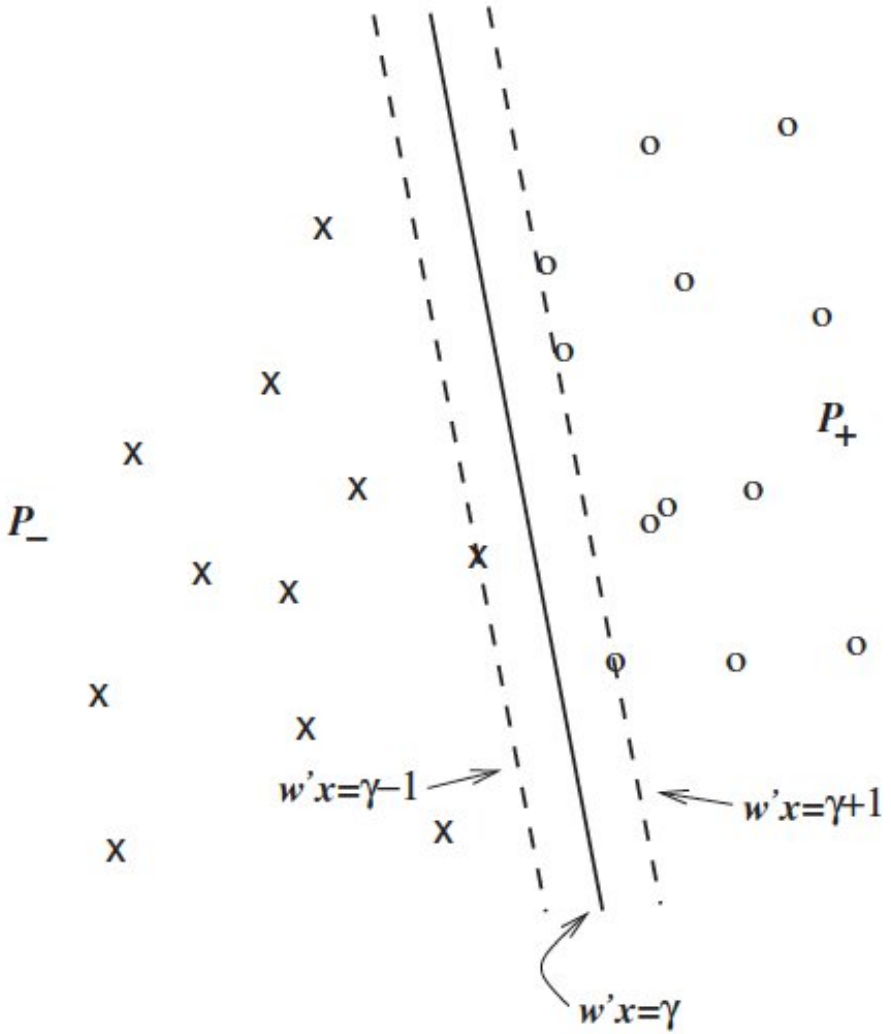
Classification problems

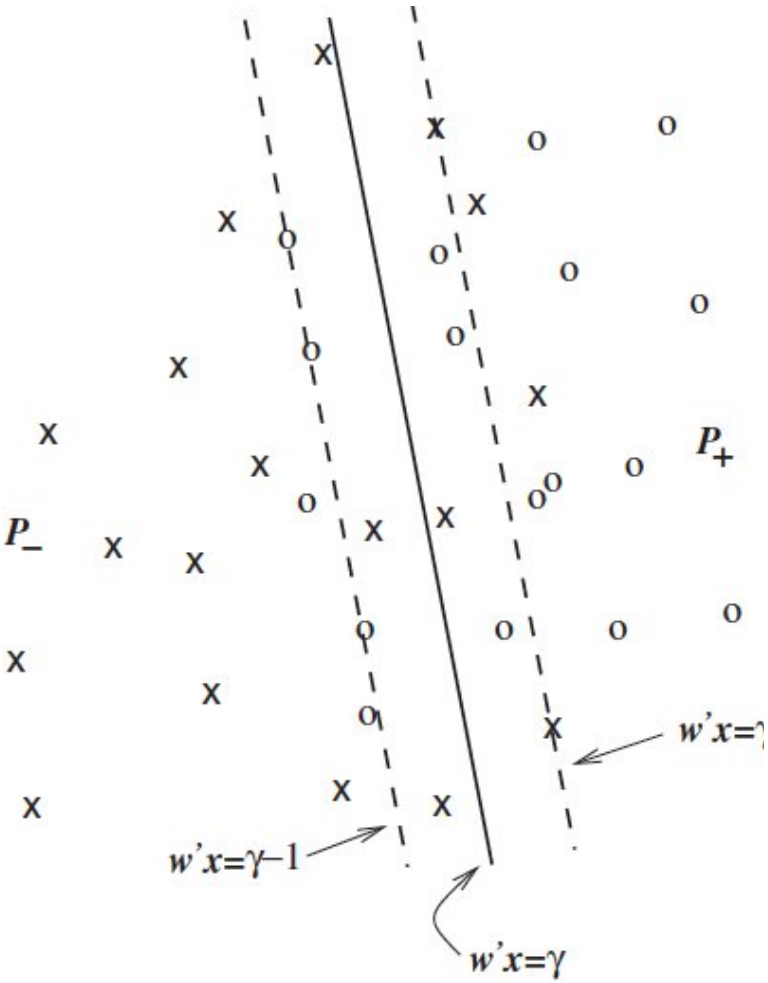
- Like regression, but y takes only a small number of *discrete values*.
- **Binary classification problems:** y_i can take on only two values, 0 or 1. Example: We are trying to build a spam classifier for email, then x_i may be some features of a piece of email, and y_i is 1 if it is a piece of spam mail, and 0 otherwise.



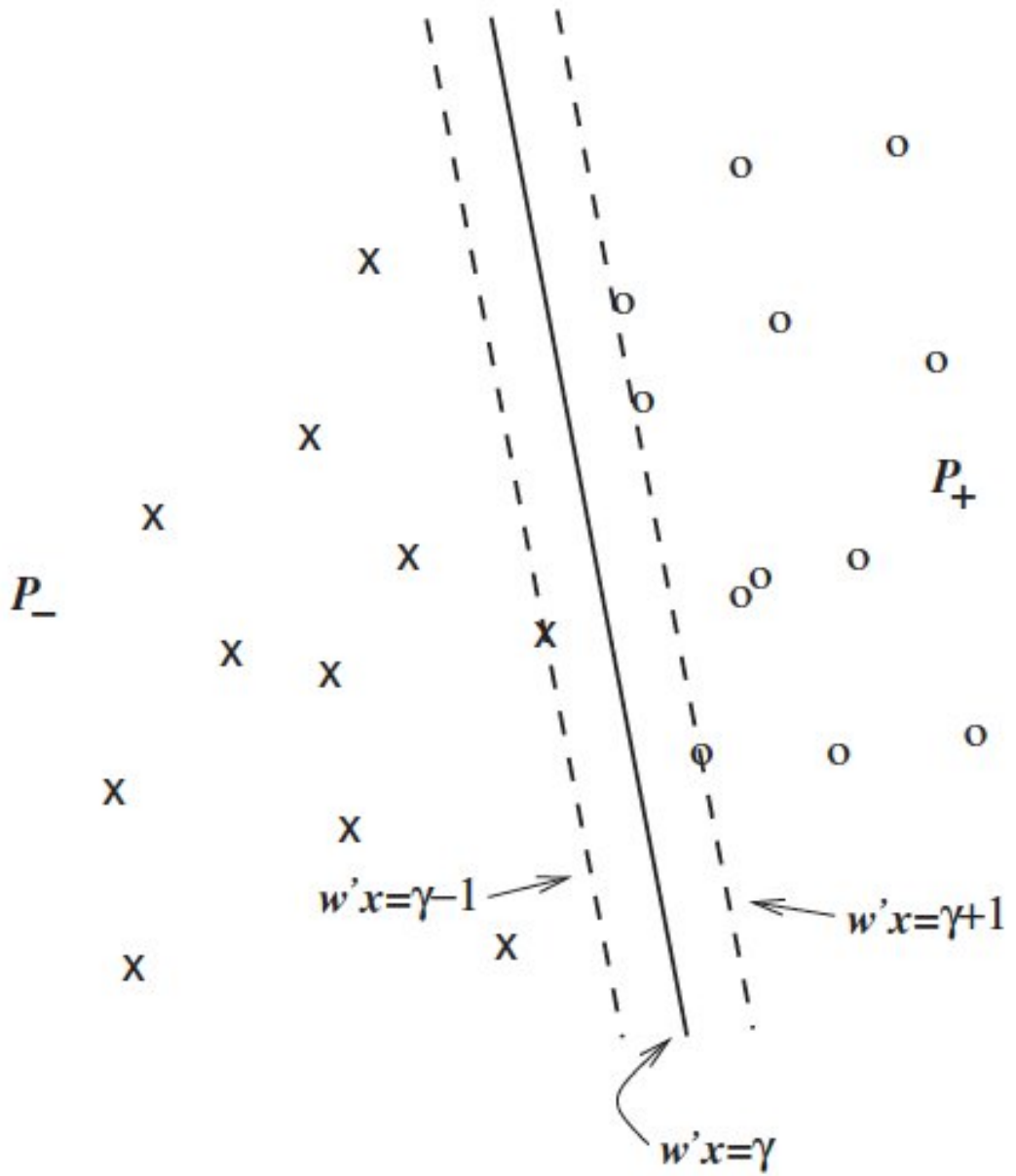
- Geometric intuition: Data points (with features) represent points in space. We wish to separate the points by a hyperplane.

Two practical situations!





SEPARABLE CASE



- We have m points in $\mathbf{R}^n$ divided into two sets of points P_+ and P_-.
- WISH: construct a function $h(x)$ such that $h(x) > 0$ for $x \in P_+$ and $h(x) < 0$ for $x \in P_-$. (But complete separation may not work!!)
- Our classifier $h(x)$ has the form of a LINEAR MAP $w^T x + b$, where $w \in \mathbf{R}^n$ and $b \in \mathbf{R}$.
- If (w, b) exists, then rewrite it as

$$\frac{(w, b)}{\min_{x \in P_+ \cup P_-} |w^T x + b|}$$

- Then, if $x \in P_+$, then $h(x) \geq 1$ and if $x \in P_-$, then $h(x) \leq -1$.
- Express conditions as a LINEAR OPTIMIZATION PROBLEM.
- FIRST ATTEMPT: Find w such that the number of errors in training set is minimized. Define

$$Er(\alpha) = \begin{cases} 1 & \text{if } \alpha < 0 \\ 0 & \text{otherwise} \end{cases}$$

- Classifier w, b produces an error if $y_i(w^T x_i + b)$. WE WISH:

$$\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} Er(y_i(w^T x_i + b)).$$

- But, *what is the problem with this model??* NOT CONVEX!!!

- **FIXING IDEA** We note that $Er(\alpha) \leq max(0, 1-\alpha)$.
- The function is now convex, because $max(0, 1-\alpha)$ convex.
- We now minimize

$$\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} \max(0, 1 - y_i(w^T x_i + b)).$$

- But this can be reformulated as the LINEAR OPTIMIZATION.

$$\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} t_i$$

$$\text{subject to} \quad t_i \geq 1 - y_i(w^T x_i + b), \quad t_i \geq 0$$

- Let us rewrite this in MATRIX NOTATION:
- Let A be the matrix $m \times n$ with rows the points (data!)
- Let D be a $m \times m$ diagonal matrix, with $D_{ii} = 1$ if $x_i \in P_+$, else $D_{ii} = -1$ if $x_i \in P_-$. In other words $D_{ii} = y_i$.
- Let e be a the vector $(1, 1, \ldots, 1)^T$. Also t is the vector with entries t_i.

$$\min ||t||_1$$

$$\text{subject to} \quad D(Aw + eb) + t \geq e, \quad t \geq 0$$

- NOTE: Not a unique solution!!!

- If it is possible to separate the two sets of points, then desirable to **maximize the distance margin** between the bounding hyperplanes!!
- **Lemma** The separation margin is MAXIMIZED by

$$\frac{2}{||w||_2}$$

. This is equivalent to MINIMIZING $||w||_2$.

- Hence, we can solve the **quadratic program**

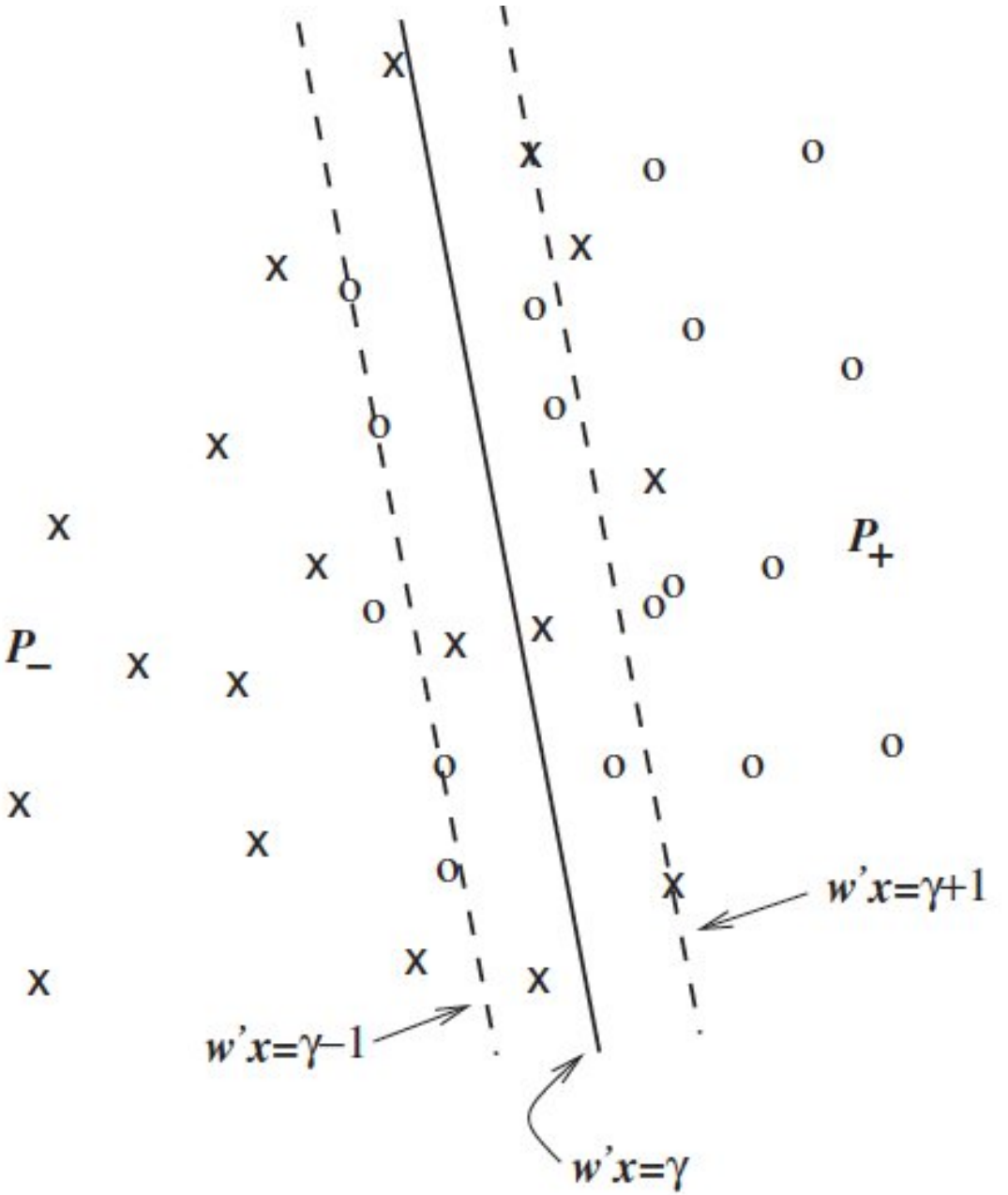
$$\min \lambda(w^T \cdot w) + ||t||_1$$

$$\text{subject to} \quad D(Aw + eb) + t \geq e, \quad t \geq 0$$

to find the best separating hyperplane with maximum margin. **Maximum margin SVM**.

- The **support vectors** are the points x that lie on the bounding hyperplanes and correspond to the active constraints in the solution.

NOT SEPARABLE?



- Often not possible to find a hyperplane that separates the two sets because no such hyperplane exists. The quadratic program is infeasible,
- GOOD NEWS: Recall that we had $t_i \geq 1 - y_i(w^T x_i + b), t_i \geq 0$ rows for linear program.
- Thus t_i components indicate the minimum amount by which the constraints are violated.
- We wish to minimize total violation by summing the components of t, while we decrease the margin. New quadratic optimization problem:

$$\min [\lambda(w^T \cdot w) + (e \cdot t)]$$

$$\text{subject to} \quad D(Aw + eb) + t \geq e, \quad t \geq 0$$

- λ is a fixed value, but we use it to fine tune the answer!!

- We can use the 1-norm instead of 2-norm to improve the quality of w.

$$\min\left[\lambda\left(|w_1|+|w_2|+\cdots+|w_n|\right)+\left(t_1+t_2,\cdots+t_m\right)\right]$$

$$\text{subject to}\quad D\left(Aw+eb\right)+y\geq e,\quad y\geq 0$$

- Set $s_i=|w_i|$. Rewrite without absolute values (we can use MATLAB or SCIP):

$$\min\left[\lambda\left(s_1+s_2+\cdots+s_n\right)+\left(t_1+t_2+\cdots+t_m\right)\right]$$

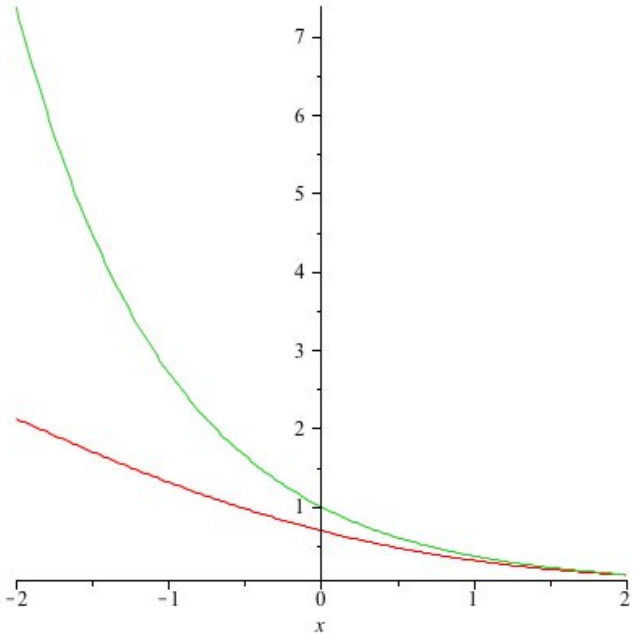
$$\text{subject to}\quad D\left(Aw+eb\right)+t\geq e,\quad t\geq 0,\quad -s_i\leq w_i\leq s_i$$

- λ is a fixed value, but we use it to fine tune the answer!! The vector w is **sparse**.
- **WE HAVE CHOICES!!** We replaced $Er(\alpha)$ by $max(0,1-\alpha)$. But there are other convex functions!!
- For example **logistic function**

$$ln\left(1+e^{-a}\right)$$

can replace $Er(\alpha)$. Or any other convex function.

Probabilistic meaning



Wish to minimize

$$\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} \ln(1 + exp(-y_i(w^T x_i + b)))$$

This corresponds to a probabilistic model: SVM gave only a yes no answer.
The probability of a point x to have label $1, -1$ is given by

$$\frac{1}{1 + exp(-y_i(w^T x_i + b))}$$

Singular Value Decomposition.

- Given an $m \times n$ matrix A, its **singular values** are the positive square roots of the eigenvalues of the matrix $A^T A$ (the **Gram matrix**). The corresponding eigenvectors are called the singular vectors of A.
- By Cholesky's decomposition $A^T A$ is positive semidefinite its eigenvalues are always non-negative and real.
- **Example** Say A is a the square matrix $\begin{bmatrix} 3 & 5 \\ 4 & 0 \end{bmatrix}$ Thus $A^T A = \begin{bmatrix} 25 & 15 \\ 15 & 25 \end{bmatrix}$ This matrix has eigenvalues 40 and 10. and the corresponding eigenvectors are $(1,1)^T$ and $(1,-1)^T$. Thus the singular values are $\sigma_1 = \sqrt{40} = 6.3246$ and $\sigma_2 = \sqrt{10} = 3.1623$. Note singular values are NOT the same as the eigenvalues of A.

- There are a number of properties we will need to understand the importance of the singular values
- **Theorem** Every real symmetric matrix A can be diagonalized by an orthogonal matrix Q, i.e., $A = Q\Lambda Q^T$ where Q is orthogonal and Λ is a diagonal matrix.
- **Proposition** If $A^T = A$ then its singular values are the absolute values of its non-zero eigenvalues and its singular vectors coincide with the associated non-null eigenvectors.

- **Theorem** Any non-zero real $(m \times n)$ matrix A of rank $r > 0$ can be factored

$$A = P\Sigma Q^T$$

  - P is an $m \times r$ matrix with orthonormal columns ($P^T P = I$).
  - Σ is an $r \times r$ diagonal matrix with the singular values of A in the diagonal.
  - Q^T is an $r \times n$ matrix with orthonormal rows ($Q^T Q = I$).
- **Proof** Same as proving $AQ = P\Sigma$. So find orthonormal vectors $q_1, \ldots, q_r$ for Q and $p_1, \ldots, p_r$!!!
- Let $q_1, \ldots, q_r$ be the orthonormal eigenvectors of Gram matrix $A^T A$ corresponding to the **non-zero eigenvalues**.

$$A^T A q_i = \lambda_i q_i = \sigma_i^2 q_i$$

- **Claim 1:** $w_i = Aq_i$ are orthogonal.
- **Claim 2:** $\|w_i\| = \sqrt{w_i^T w_i} = \sigma$
- **Claim 3:** $p_i = \frac{w_i}{\sigma_i} = \frac{Aq_i}{\sigma}$.

- **Theorem** Any non-zero real $m \times n$ A can be factored

$$A = U\Sigma V^T$$

  where
  - U is an $m \times m$ matrix with orthonormal columns ($P^T P = I$).
  - Σ is an $m \times n$ diagonal matrix with the singular values of A in the diagonal.
  - V is an $n \times n$ matrix with orthonormal rows ($Q^T Q = I$).
- Moreover, the columns of U are the eigenvectors of AA^T and the columns of V are the eigenvectors of $A^T A$.
- The r singular values on the diagonal of σ are the square roots of the non-zero eigenvalues of both $A^T A$ and AA^T.

- **Example (continued)** For $A = \begin{bmatrix} 3 & 5 \\ 4 & 0 \end{bmatrix}$, the orthogonal eigenvector basis of $A^T A$ is $q_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T, q_2 = (\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^T$
- Thus, following the method of the algorithm, $Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$ and $P = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix}$
- Thus $A = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{40} & 0 \\ 0 & \sqrt{10} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$

Applications: Numerics, Image Compression, Data Analysis

- The ratio $\sigma_{\mathbf{max}}/\sigma_{\mathbf{min}}$ is the **Condition number**. A matrix with very large condition number is **ill-conditioned**. This is trouble for calculations when the condition number is larger than the reciprocal of the machine precision.
- The singular vectors indicate the **principal components**. One key idea is to discard singular values and vectors that are too small
- The columns of the matrix A represent data vectors (normalized to have mean 0). The matrix A^TA can be thought of as the **covariance matrix**. Its eigenvectors indicate directions of correlation and clustering in the data.
- For Images, instead of sending a 1000×1000 pixels, compute SVD, $A = P\Sigma Q^T = p_1\sigma_1 q_1^T + p_2\sigma_2 q_2^T + \cdots + p_r\sigma_r q_r^T$, we see A is a sum of rank 1 matrices!
  we only keep those pieces with σ_i of "good" size, throw away rest.