Language Information Processing, Advanced

Text Classifiers

mcuturi@i.kyoto-u.ac.jp
Today's talk

- Objective: supervised inference on text data.
  - Ex.1 Given a large database of news articles about business, sports, literature, politics etc.
  - Build a system that can classify automatically new articles.

---

**Business**

- **AP reports pipeline operations can restart**
  - KLAS-TV - 8 hours ago
  - The Associated Press reported today that regulators allowed a key pipeline to restart operations. This comes after the line was shut down due to problems with the pipeline's valves.

- **GMail hacking draws FBI interest**
  - Economic Times - 12 hours ago
  - WASHINGTON: The computer hacking campaign that targeted Gmail is reported to have originated in China and was directed at a large number of officials and officials involved in the FBI inquiry that began this week, according to several administration officials.

- **Giant open-pit mine raises questions in Uruguay**
  - AFP - 10 hours ago
  - CERRO CHATO, Uruguay - A plan to build a giant open-pit mine has created a sharp rift between those who think agricultural land should be protected, and those wanting to exploit its wealth. The Aratiri project, owned by Zamin

---

**Sci/Tech**

- **WWDC, iPhone 5 in limelight: What new Android smartphones are lined up?**
  - International Business Times - 2 hours ago
  - By IB Times Staff Reporter | June 6, 2011 7:02 AM EDT All eyes are on Apple Worldwide Developer Conference 2011 on Monday and whether Steve Jobs will unveil Apple iPhone 5. Most observers say Apple will not unveil the next...
Today's talk

- **Ex.2** Given a large set of e-mails in a mailbox, *family, friends, spam, ads, newletters etc.*
  - Build a system that **categorizes** automatically a new email.
Today’s talk

○ **Ex.3** Given a set of requests/messages sent to a retailer: complaints, need for technical support, praise
  ▷ Build a system that **forwards directly** the message to the relevant department.

- Who is interested in this?
  ○ internet companies,
  ○ companies with large customer support receiving requests,
  ○ polling institutions,
  ○ social scientists who want to use text for their studies *etc.*
Text classification & probabilistic framework

• Assume that there is a probability $p_{\text{text}}$ on texts on the internet

  Today will be a rainy day

  In Ecuador tiger-hunters enjoy eating marshmallows

  Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo Buffalo buffalo

• A probability quantifies how likely sentences are to appear

• Any idea on how this likelihood might be measured?
Text classification & probabilistic framework

- This probability takes into account **grammar** and **meaning**.
- Search engines are useful to have an idea about $p_{text}$.

Today will be a rainy day

"today will be a rainy day"
About 288,000 results (0.24 seconds)

In Ecuador tiger-hunters enjoy eating marshmallows

"In Ecuador tiger-hunters enjoy eating marshmallows"

⚠️ No results found for "In Ecuador tiger-hunters enjoy eating marshmallows".

Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo Buffalo buffalo Buffalo buffalo Buffalo buffalo

"Buffalo buffalo Buffalo buffalo buffalo Buffalo buffalo Buffalo buffalo Buffalo buffalo Buffalo buffalo"
About 4,980,000 results (0.29 seconds)
Text classification & probabilistic framework

- We assume that there is something to learn from data (supervised inference)
- We assume our task is to categorize a given text among $C$ given classes
  - agriculture, computer chips, energy, environment, sports, politics, gossip etc.
  - friends, family, spam, advertisements, newsletters etc.

- We also assume there is a probability $p_{\text{cat}}$ on categories.
Text classification & probabilistic framework

- We assume that there is **something to learn from data** (supervised inference)
- We assume our task is to categorize a given text among \( C \) given classes
  - agriculture, computer chips, energy, environment, **sports**, **politics**, **gossip** etc.
  - friends, family, **spam**, **advertisements**, **newsletters** etc.

- Some documents appear more frequently than others.

\[ p_{\text{cat}}(\text{gossip}) > p_{\text{cat}}(\text{philosophy}) \]
Text classification & probabilistic framework

- Our goal will be to understand better the relationship between

\[
\text{TEXT} \leftrightarrow \text{CATEGORY}
\]

- Here, we assume also that there is a joint probability on texts and their category.

\[
P(\text{text}, \text{category})
\]

which quantifies how likely the match between

a text text and a category category is

- For instance,

\[
P(\text{‘I am feeling hungry these days’}, \text{‘poetry’}) \approx 0
\]

\[
P(\text{‘Manchester United’s stock rose after their victory’}, \text{‘business’}) \lor P(\text{‘Manchester United’s stock rose after their victory’}, \text{‘sports’})
\]
Hence, given a sequence of words (including punctuation),

\[ w = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, w_8, \cdots, w_n) \]

assuming we know \( P \), the joint probability between texts and categories,

an easy way to guess the category of \( w \) is by looking at

\[
\text{category-prediction}(w) = \arg\max_{C} P(C|w_1, w_2, \cdots, w_n)
\]
Text classification & probabilistic framework

\[
\begin{align*}
P('\text{poetry}'|'I am feeling hungry these days') &= 0.0037 \\
P('\text{business}'|'I am feeling hungry these days') &= 0.005 \\
P('\text{sports}'|'I am feeling hungry these days') &= 0.003 \\
\quad P('\text{food}'|'I am feeling hungry these days') &= 0.2 \\
P('\text{economy}'|'I am feeling hungry these days') &= 0.04 \\
P('\text{society}'|'I am feeling hungry these days') &= 0.08
\end{align*}
\]
Text classification & probabilistic framework

\[
P('\text{poetry}' | 'I am feeling hungry these days') = 0.0037 \\
P('\text{business}' | 'I am feeling hungry these days') = 0.005 \\
P('\text{sports}' | 'I am feeling hungry these days') = 0.003 \\
\rightarrow P('\text{food}' | 'I am feeling hungry these days') = 0.2 \\
P('\text{economy}' | 'I am feeling hungry these days') = 0.04 \\
P('\text{society}' | 'I am feeling hungry these days') = 0.08
\]
Bayes Rule

- Using Bayes theorem $p(A, B) = p(A|B)p(B)$,

\[ P(C|w_1, w_2, \cdots, w_n) = \frac{P(C, w_1, w_2, \cdots, w_n)}{P(w_1, w_2, \cdots, w_n)} \]

- When looking for the category $C$ that best fits $w$, we only focus on the numerator.

- Bayes theorem also gives that

\[
P(C, w_1, \cdots, w_n) = P(C)P(w_1, w_2, \cdots, w_n|C) \\
= P(C)P(w_1|C)P(w_2, w_3, \cdots, w_n|C, w_1) \\
= P(C)P(w_1|C)P(w_2|C, w_1)P(w_3, w_4, \cdots, w_n|C, w_1, w_2) \\
= \prod_{i=1}^{n} P(w_i|C, w_1, \cdots, w_{i-1})
\]
Examples

• Assume we have the beginning of this news title

\[ w_1, \cdots, w_{12} = \text{‘The weather was so bad that the organizers decided to close the’} \]

• If \( C = \text{business} \), then

\[
P(W_{13} = \text{‘market’} | \text{business}, w_1, \cdots, w_{12})
\]

should be quite high, as well as \text{summit, meeting etc.}.

• On the other hand, if we know \( C = \text{sports} \), the probability for \( w_{13} \) changes significantly...

\[
P(W_{13} = \text{‘game’} | \text{sports}, w_1, \cdots, w_{12})
\]
The Naive Bayes Assumption

• From a factorization

\[ P(C, w_1, \cdots, w_n) = \prod_{i=1}^{n} P(w_i|C, w_1, \cdots, w_{i-1}) \]

which handles all the conditional structures of text,

• we assume that each word appears independently conditionally to \( C \),

\[ P(w_i|C, w_1, \cdots, w_{i-1}) = P(w_i|C, w_1, \cdots, w_{i-1}) \]
\[ = P(w_i|C) \]

• and thus

\[ P(C, w_1, \cdots, w_n) = \prod_{i=1}^{n} P(w_i|C) \]
The Naive Bayes Assumption Leads to Word Counts

• The factorization

\[ P(w_i|C, w_1, \cdots, w_{i-1}) = P(w_i|C) \]

• means that we take for granted that

\[ P(C, \text{‘The weather was bad so the meeting was closed’}) \]

\[ = \]

\[ P(C, \text{‘was The bad the closed meeting weather was so’}) \]
The Naive Bayes Assumption Leads to Word Counts

- Assume we know $P(C, w)$ for all words $w$ in the dictionary and all categories.

  $$P(\text{`business'}, \text{`stock'}) > P(\text{`sports'}, \text{`stock'})$$

- Given a text $T =$ But Federer has been quite a French Open security blanket for Nadal. Their rivalry is one of the greatest in tennis history, yet it has been decidedly short on suspense here. Nadal is now 5-0 against Federer at Roland-Garros. Nadal is the greatest ...

- The only thing the Bayes classifier will consider is the word histogram
The Naive Bayes Assumption Leads to Word Counts

• To each text,
  ◦ count the frequency of each word $w$ in the dictionary $D$, $h_w$. Then

$$P(T|C) = \prod_{w \in D} P(w|C)^{h_w}$$

• In the example below, it seems obvious that the terms

$$P(W = 'Nadal'|\text{tennis}), P(W = 'Federer'|\text{tennis}), \ldots$$

will be quite big.

• The Naive Bayes should easily classify this text as tennis...

  • if the probabilities $P(w|C)$ were known!!!
We need to build an estimate of $P(w|C)$ for all words of $\mathcal{D}$, all categories
We need to build an estimate of $P(w|C)$ for all words of $\mathcal{D}$, all categories.

A typical approach

- Consider a corpus of documents with different categories of text
  $$\{(T_1, c_1), \cdots, (T_N, c_N)\}.$$  
- Build a reduced dictionary $\hat{\mathcal{D}}$
  - using all words appearing in all $T_i$,
  - usually removing non-informative words such as articles, prepositions etc.
- Compute histograms $h^i_w$ for each $T_i$ which only track words in $\hat{\mathcal{D}}$.
- Compute an estimate $\hat{p}(w|c)$ for each word $w \in \hat{\mathcal{D}}$ and estimates $\hat{p}(c)$.  

Term Frequencies

- Use these elements, $\hat{p}$, $\hat{D}$ to classify a new text $T$ using his representation $h^T_w$

\[
\text{category-prediction}(T) = \arg\max_c \left( \hat{p}(c) \prod_{w \in \hat{D}} \hat{p}(w|c)^{h^T_w} \right)
\]

- of course, if we use the logarithm of the r.h.s., we get the rule

\[
\text{category-prediction}(T) = \arg\max_c \log \hat{p}(c) + \sum_{w \in \hat{D}} h^T_w \log \hat{p}(w|c)
\]

Naive Bayes for text $\Leftrightarrow$ Linear Classifier Using Term Frequencies as Features

- Once this is established... we could imagine any linear classifier using TF.
The Data we have:

- texts $T_i$ translated as histograms of words $h^1, h^2, h^3, \ldots, h^N$.
- Each histogram is a vector of the simplex $\Sigma_d$ where $d = \#D - 1$ and

$$\Sigma_d = \{ x \in \mathbb{R}^{d+1} | x_i \geq 0, \sum_{i=1}^{d+1} x_i = 1 \}.$$

We consider 2 categories only here, for instance “spam” vs “non-spam”.

The corpus consists in a large number of histogram/bit pairs

- For illustration purposes only we will consider the 2 dimensional simplex, that is $\#D = 3$. 
Binary Classification Separation Surfaces for Vectors

What is a classification rule?

Kyoto University - LIP, Adv. - 2012
Binary Classification Separation Surfaces for Vectors

Classification rule = a partition of $\mathbb{R}^d$ into two sets
This partition is usually interpreted as the level set of a function
Binary Classification Separation Surfaces for Vectors

Typically, \( \{ h \in \Sigma_d | f(h) > 0 \} \) and \( \{ h \in \Sigma_d | f(h) \leq 0 \} \)
Classification Separation Surfaces for Vectors

Can be defined by a single surface, \textit{e.g.} a curved line
Classification Separation Surfaces for Vectors

Even more simple: using straight lines and halfspaces.
Linear Classifiers

- **Straight lines** (hyperplanes when $d > 2$) are the simplest type of classifiers.

- A hyperplane $H_{c,b}$ is a set in $\mathbb{R}^p$ defined by
  - a normal vector $c \in \mathbb{R}^p$
  - a constant $b \in \mathbb{R}$ as

\[
H_{c,b} = \{ x \in \mathbb{R}^d \mid c^T x = b \}
\]

- Letting $b$ vary we can “slide” the hyperplane across $\mathbb{R}^p$
Linear Classifiers

- In $\Sigma_d$, things hypersurfaces divide $\mathbb{R}^d$ into two halfspaces,

$$\{ h \in \mathbb{R}^d \mid c^T h < b \} \cup \{ h \in \mathbb{R}^d \mid c^T h \geq b \} = \mathbb{R}^d$$

- Linear classifiers attribute the “yes” and “no” answers given arbitrary $c$ and $b$.

![Diagram showing division of space by a hyperplane](image)

- Assuming we only look at halfspaces for the decision surface...
  
  ...how to choose the “best” $(c^*, b^*)$ given a training sample?
Linear Classifiers

- Training a classifier is mapping a dataset to a $c$ and $b$.

  \[
  \{ (h^i \in \Sigma^d, y_i \in \{0, 1\})_{i=1..N} \} \xrightarrow{???} \text{"best\" } c^*, b^*
  \]

  has different answers.

- **Linear Discriminant Analysis** (or Fisher’s Linear Discriminant);

- **Logistic regression** maximum likelihood estimation;

- **Perceptron**, a one-layer neural network;

- **Support Vector Machine**, the result of a convex program

- etc.
What is special about natural text?

- Remember we have
  - A corpus of \( N \) documents \( \{(T_1, c_1), \cdots, (T_N, c_N)\} \).
  - Build a reduced dictionary \( \hat{D} \) of \( M \) words
  - Compute histograms \( h^i_w \) for each \( T_i \) which only track words in \( \hat{D} \).

- What is difficult about text processing usually?

\[
\text{Usually, } M \text{ is very large, possible bigger than } N
\]

\[
H = \begin{bmatrix}
T_1 & T_2 & T_3 & T_4 & \cdots & T_N \\
\text{eat} & 0 & 3 & 1 & 0 & \cdots & 0 \\
\text{ball} & 4 & 0 & 0 & 0 & \cdots & 1 \\
\text{dinosaur} & 0 & 2 & 0 & 0 & \cdots & 0 \\
\text{genome} & 0 & 0 & 2 & 0 & \cdots & 0 \\
\text{planet} & 0 & 1 & 0 & 0 & \cdots & 0 \\
\text{Clooney} & 0 & 0 & 0 & 2 & \cdots & 0 \\
\text{Guatemala} & 0 & 0 & 0 & 2 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\end{bmatrix}
\]
Sparse Classifiers

sparse (adj. sparser, sparsest)
Occurring, growing, or settled at widely spaced intervals; not thick or dense
The goal when estimating linear classifiers: define \( c \in \mathbb{R}^M \) and \( b \in \mathbb{R} \).

The number of words is \( M \), defining a vector \( c \) means setting a value for:

\[
C = \begin{bmatrix}
C_{\text{eat}} \\
C_{\text{ball}} \\
C_{\text{dinosaur}} \\
C_{\text{genome}} \\
C_{\text{planet}} \\
C_{\text{Clooney}} \\
\vdots
\end{bmatrix}
\]
Sparse and non-sparse

• Without any constraint, defining $c^*$ is simply:

$$\min_{c \in \mathbb{R}^M, b \in \mathbb{R}} \text{error}(c, b)$$

for instance, error can be the logistic error, the hinge loss (SVM) etc...

• With a sparsity constraint, we have

$$\min_{c, b \in \mathbb{R}, \|c\|_0 \leq p} \text{error}(c, b), \text{ where } \|c\|_0 \overset{\text{def}}{=} \sum_{i=1}^{M} 1_{c_i \neq 0}$$
Sparse and non-sparse

- sparse vector:

\[ c = [0, 0, 0, 1.324, 0, 0, -3.21, 0, 0, 0] \]

\[ \|c\|_0 = 2 \]

- dense vector

\[ c = [0.21, -4.65, 3.2, 6.982, 5.43, -9.1, 0.004, -0.37, 12.1, 3.94] \]

\[ \|c\|_0 = 10 \]

- a sparsity constraint enforces the solution to be sparse and not dense

\[ \min_{c, b \in \mathbb{R}, \|c\|_0 \leq p} \text{error}(c, b), \text{ where } \|c\|_0 \overset{\text{def}}{=} \sum_{i=1}^{M} 1_{c_i \neq 0} \]
Why we like sparse

Sparse solutions for $c$ are desirable because

- they are lighter in memory. Computations only grow in $p$, not $M$ anymore.

\[ c = \begin{bmatrix} 0 & 0 & 0 & 1.324 & 0 & 0 & -3.21 & 0 & 0 & 0 \end{bmatrix} \]

\[ c^T x = 1.324 \times x_4 - 3.21 \times x_7 \]

- since only $p$ words matter, these are **keywords** which can be interpreted
  - $c_4 > 0$, *genome* is the important word to predict positively
  - $c_7 > 0$, *Guatemala* is the important word to predict negatively
How we can solve a “sparsified” problem

How can we estimate sparse solutions $c^*$?

- **Direct** approach

  $$\min_{c,b \in \mathbb{R}, \|c\|_0 \leq p} \text{error}(c, b), \text{ where } \|c\|_0 \overset{\text{def}}{=} \sum_{i=1}^{M} 1_{c_i \neq 0}$$

  is computationally intractable.

- **Alternative** approach: penalize with the $l_1$ norm

  $$\min_{c,b \in \mathbb{R}} \text{error}(c, b) + \lambda \|c\|_1, \text{ where } \|c\|_1 \overset{\text{def}}{=} \sum_{i=1}^{M} |c_i|$$

  can prove that we can recover sparse solutions.

- Many algorithms: **LASSO**, **FISTA**... see literature on compressive sensing.

- Example: [http://statnews.org/](http://statnews.org/) website by Laurent El Ghaoui
Support Vector Machine

Check the very nice book on the subject by T. Joachims. It’s a bit old now but contains a lot of fundamental ideas.
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
A criterion to select a linear classifier: the margin?
Largest Margin Linear Classifier?
Support Vectors with Large Margin
Finding the optimal hyperplane is equivalent to finding \((w, b)\) which minimize:

\[ \|w\|^2 \]

under the constraints:

\[ \forall i = 1, \ldots, n, \quad y_i \left( w^T x_i + b \right) - 1 \geq 0. \]

This is a classical quadratic program on \(\mathbb{R}^{d+1}\) with linear constraints - quadratic objective.
Lagrangian

• In order to minimize:
  \[ \frac{1}{2} ||w||^2 \]
  under the constraints:
  \[ \forall i = 1, \ldots, n, \quad y_i (w^T x_i + b) - 1 \geq 0. \]

• introduce one dual variable \( \alpha_i \) for each constraint,

• one constraint for each training point.

• the Lagrangian is, for \( \alpha \geq 0 \) (that is for each \( \alpha_i \geq 0 \))
  \[
  L(w, b, \alpha) = \frac{1}{2} ||w||^2 - \sum_{i=1}^{n} \alpha_i \left( y_i \left( w^T x_i + b \right) - 1 \right).
  \]
The Lagrange dual function

\[ g(\alpha) = \inf_{w \in \mathbb{R}^d, b \in \mathbb{R}} \left\{ \frac{1}{2} \|w\|^2 - \sum_{i=1}^{n} \alpha_i \left( y_i (w^T x_i + b) - 1 \right) \right\} \]

has saddle points when

\[ w = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad ( \text{derivating w.r.t } w) \quad (\star) \]

\[ 0 = \sum_{i=1}^{n} \alpha_i y_i, \quad (\text{derivating w.r.t } b) \quad (\star\star) \]

substituting (\star) in \( g \), and using (\star\star) as a constraint, get the dual function \( g(\alpha) \).

- To solve the dual problem, \textbf{maximize} \( g \) w.r.t. \( \alpha \).
- Strong duality holds. KKT gives us \( \alpha_i (y_i (w^T x_i + b) - 1) = 0 \),
  ...\textit{hence}, either \( \alpha_i = 0 \) or \( y_i (w^T x_i + b) = 1 \).
- \( \alpha_i \neq 0 \ \text{only} \) for points on the support hyperplanes \( \{(x, y) | y_i (w^T x_i + b) = 1\} \).
The dual problem is thus

\[
\text{maximize} \quad g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j
\]

such that \( \alpha \succeq 0, \sum_{i=1}^{n} \alpha_i y_i = 0. \)

This is a \textbf{quadratic program} in \( \mathbb{R}^n \), with \textit{box constraints}. \\
\( \alpha^* \) can be computed using optimization software \\
(e.g. built-in \textit{matlab} function)
 Recovering the optimal hyperplane

- With $\alpha^*$, we recover $(w^T, b^*)$ corresponding to the optimal hyperplane.
- $w^T$ is given by $w^T = \sum_{i=1}^{n} y_i \alpha_i x_i^T$,
- $b^*$ is given by the conditions on the support vectors $\alpha_i > 0$, $y_i(w^T x_i + b) = 1$,

$$b^* = -\frac{1}{2} \left( \min_{y_i=1, \alpha_i > 0} (w^T x_i) + \max_{y_i=-1, \alpha_i > 0} (w^T x_i) \right)$$

- the decision function is therefore:

$$f^*(x) = w^T x + b^*$$

$$= \sum_{i=1}^{n} y_i \alpha_i x_i^T x + b^*.$$

- Here the dual solution gives us directly the primal solution.
Interpretation: support vectors

\[ \alpha = 0 \]

\[ \alpha > 0 \]
Another interpretation: Convex Hulls

go back to 2 sets of points that are linearly separable
Another interpretation: Convex Hulls

Linearly separable = convex hulls do not intersect
Another interpretation: Convex Hulls

Find two closest points, one in each convex hull
Another interpretation: Convex Hulls

The SVM = bisection of that segment
Another interpretation: Convex Hulls

support vectors = extreme points of the faces on which the two points lie
Kernel trick for SVM’s

- use a mapping $\phi$ from $\mathcal{X}$ to a feature space,
- which corresponds to the kernel $k$:

$$\forall x, x' \in \mathcal{X}, \quad k(x, x') = \langle \phi(x), \phi(x') \rangle$$

- Example: if $\phi(x) = \phi \left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1^2 \\ x_2^2 \end{bmatrix}$, then

$$k(x, x') = \langle \phi(x), \phi(x') \rangle = (x_1)^2(x_1')^2 + (x_2)^2(x_2')^2.$$
Training a SVM in the feature space

Replace each $x^T x'$ in the SVM algorithm by $\langle \phi(x), \phi(x') \rangle = k(x, x')$

- **Reminder**: the dual problem is to maximize

$$g(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i, x_j),$$

under the constraints:

$$\begin{cases} 
0 \leq \alpha_i \leq C, & \text{for } i = 1, \ldots, n \\
\sum_{i=1}^{n} \alpha_i y_i = 0. 
\end{cases}$$

- The **decision function** becomes:

$$f(x) = \langle w, \phi(x) \rangle + b^*$$

$$= \sum_{i=1}^{n} y_i \alpha_i k(x_i, x) + b^*.$$  \hspace{1cm} (1)
The Kernel Trick?

The explicit computation of $\phi(x)$ is not necessary. The kernel $k(x, x')$ is enough.

- The SVM optimization for $\alpha$ works **implicitly** in the feature space.
- The SVM is a kernel algorithm: only need to input $K$ and $y$:
  \[
  \text{maximize} \quad g(\alpha) = \alpha^T \mathbf{1} - \frac{1}{2} \alpha^T (K \odot yy^T) \alpha \\
  \text{such that} \quad 0 \leq \alpha_i \leq C, \quad \text{for } i = 1, \ldots, n \\
  \sum_{i=1}^{n} \alpha_i y_i = 0.
  \]
- $K$’s positive definite $\iff$ problem has a unique optimum
- The decision function is $f(\cdot) = \sum_{i=1}^{n} \alpha_i k(x_i, \cdot) + b$. 
Kernel example: polynomial kernel

- For $\mathbf{x} = (x_1, x_2)^\top \in \mathbb{R}^2$, let $\phi(\mathbf{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2) \in \mathbb{R}^3$:

$$
\begin{align*}
K(\mathbf{x}, \mathbf{x}') &= x_1^2 x_1'^2 + 2 x_1 x_2 x_1' x_2' + x_2^2 x_2'^2 \\
&= \{x_1 x_1' + x_2 x_2'\}^2 \\
&= \{\mathbf{x}^\top \mathbf{x}'\}^2.
\end{align*}
$$
Kernels are Trojan Horses onto Linear Models

- With kernels, complex structures can enter the realm of linear models
Kernels For Histograms

- An abridged bestiary of **negative definite distances** on the probability simplex:

\[
\psi_{JD}(\theta, \theta') = h \left( \frac{\theta + \theta'}{2} \right) - \frac{h(\theta) + h(\theta')}{2},
\]

\[
\psi_{\chi^2}(\theta, \theta') = \sum_i \frac{(\theta_i - \theta'_i)^2}{\theta_i + \theta'_i}, \quad \psi_{TV}(\theta, \theta') = \sum_i |\theta_i - \theta'_i|,
\]

\[
\psi_{H_2}(\theta, \theta') = \sum_i |\sqrt{\theta_i} - \sqrt{\theta'_i}|^2, \quad \psi_{H_1}(\theta, \theta') = \sum_i |\sqrt{\theta_i} - \sqrt{\theta'_i}|.
\]

- Recover kernels through

\[
k(\theta, \theta') = e^{-t\psi}, \quad t > 0
\]