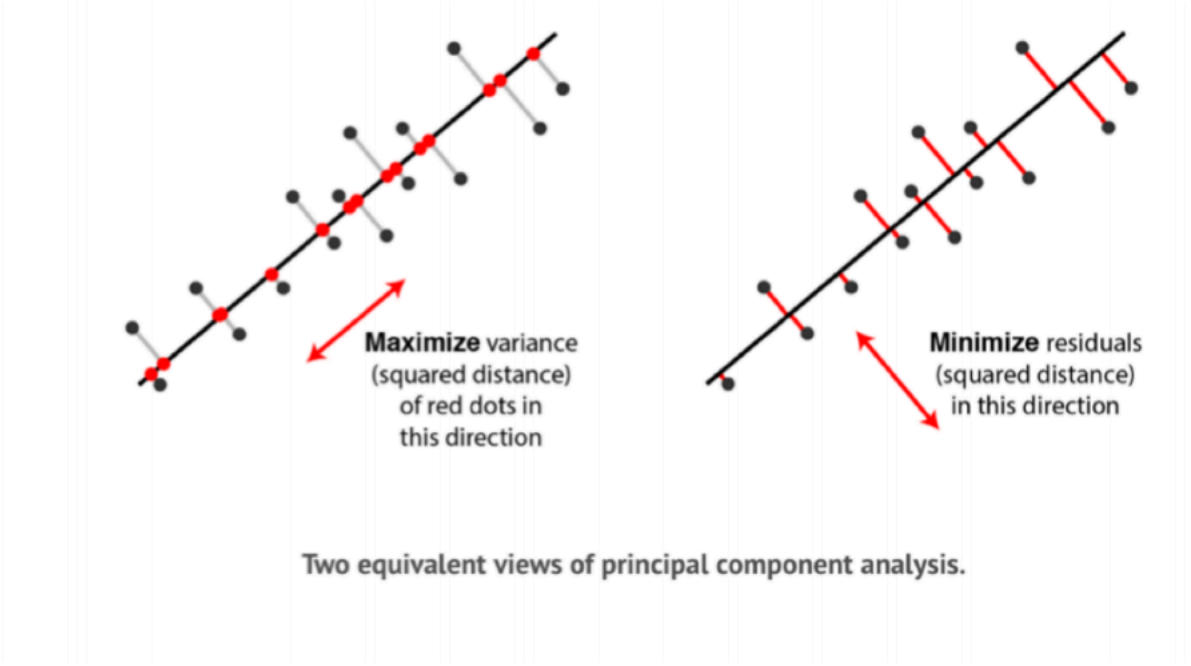
How can I interpret my ML system outputs?

Are my interpretations reasonable?

This course so far

How can one predict some value, find structure from data?



Two equivalent views of principal component analysis.

Let’s consider LASSO

We have come up with some sparse w for predicting y from x

How can we interpret the coefficients? Relative

Consider a linear model

$$\hat{w} = \text{argmin}_w \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2$$

Claim: $\hat{w}_i > \hat{w}_j$ means feature i is more important than feature j

FALSE

What if features have different scales?

- Sqft have a mean of 2500 → w_i is $/sqft
- # baths have a mean of 2 → w_i is $/bath

How to fix?

Normalize features to have mean and unit variance (z-score)

How can we interpret the coefficients? Zeros

Consider a linear model

$$\hat{w} = \text{argmin}_w \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2 + \lambda \|w\|_1$$ with

normalized data

Claim: $\hat{w}_i = 0$ means feature i has no predictive power for y

FALSE

Feature i might be correlated with y but redundant with feature j
e.g. area in square feet and area in square meters

How can we interpret the coefficients? Distributional changes

Consider a linear model

$$\hat{w} = \text{argmin}_w \sum_{i=1}^{n} \left(y_i - w^\top x_i\right)^2$$

Claim: $\hat{w}_i = 90{,}000$ and the ith feature = #fireplaces. If I add 10 more fireplaces, I can expect to sell my house for $900,000 more!

FALSE

(1) Distribution shift: we've never seen a house with 10 fireplaces in the training data, so we don't have accurate predictions for this range. It's OOD: out of distribution

(2) Correlation ≠ causation

(3) Linear model, non-linear relationship

Generalization

Say we've trained a model to interpret medical x-rays using data from a few hospitals. We randomly split the data into train/validation/test splits.

```
# Train on hospitals 1, 2, 3 → Test on hospital 4
```

Claim: The test set performance is always a good indicator of how our model will do if we deploy this model in a new hospital.

FALSE

Distribution shift: new hospital could have different equipment, demographics, staff training, location…

Generalization

Say we've trained a model to interpret medical x-rays using data from a few hospitals. We randomly split the data into train/validation/test splits.

Train on hospitals 1, 2, 3 → Test on hospitals 1, 2, 3

Claim: The test set performance is always a good indicator of how our model will do if we deploy this model in the **same hospitals.**

FALSE

Distribution shift: temporal shift. Conditions can change in our hospital by the time we deploy our system! e.g. new training procedures, new staff, new equipment, etc.

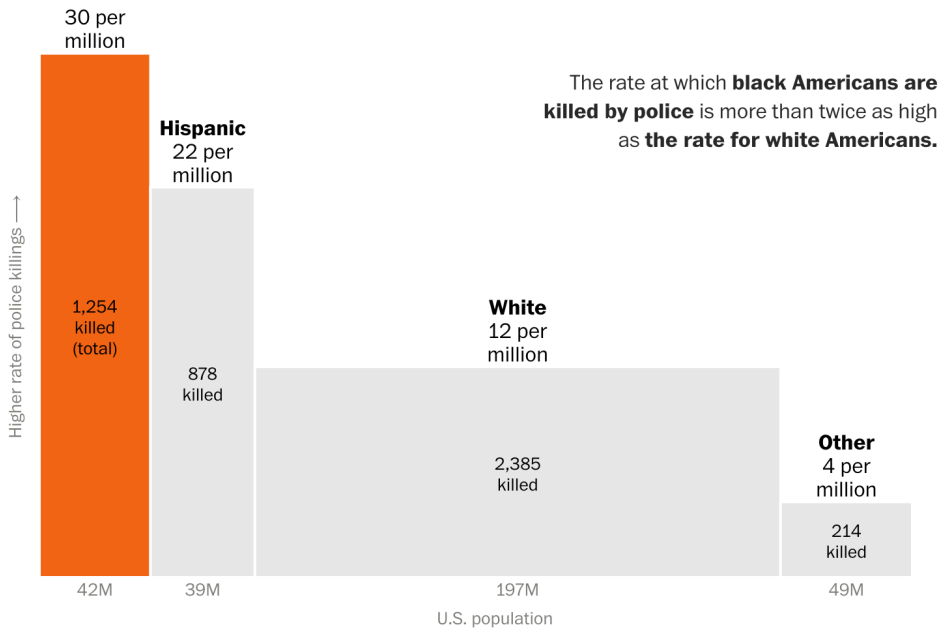
My particular hobby horse: the world is non-stationary!

Now, let’s talk about high-stakes decision making

Some sobering statistics

Many, many people are killed by police every year (10,429 in the past 9 years)

> half of which aren't reported to the FBI (reporting is voluntary)



Source: an excellent Washington Post interactive database

Design choices

Model Class

Features to collect

Loss function

Regularization

Optimization Method

Constraints

…

Design choices affect your model

Model Class

Features to collect

Loss function

Regularization

Optimization Method

Constraints

…

These will all affect what model you find, and how well and when that structure will generalize

… and that can have consequences "beyond" ML

Model Class

Features to collect

Loss function

Regularization

Optimization Method

Constraints

…

These will all affect what model you find, and how well and when that structure will generalize

Different models will have different kinds of errors, predictions, and failure modes

Consequence #1: Good predictions for whom?

©2018 Kevin Jamieson

The US Criminal Justice System

This is a description of how things (largely) are and have been, not how they ought to be.

Any use of ML in the criminal justice system is just a tiny part of the larger CJ ecosystem.

Moreover, “reoffend” or “recidivate” are very, very, very far from precise terms in their use (in this lecture, and in industry).

The US Criminal Justice System

(Lots and lots and lots of ~~possibly~~ biased processes, including racialized choices of where to police, what behaviors are most dangerous, …)

-> A person is charged with a crime

-> A judge determines whether to detain them or release them on bail until their trial

-> (Many, many months or years pass)

-> The trial determines whether the justice system considers a person guilty or innocent of charges

-> A sentencing hearing determines the length of prison time, fines, …

-> With “good behavior”, people in prison can be considered for parole

What does the CJ process have to do with ML?

Predictive policing: determine (statistically) where to send police

-> A person is charged with a crime

Predict probability of reappearing, being charged with another crime if on parole, …

-> (Many, many months or years pass)

-> The trial determines whether the justice system considers a person guilty or innocent of charges

Predict risk of being charged with another crime, use in sentencing

Predict risk of being charged with another crime, use in parole decisions

The US Criminal Justice System

“reoffend” or “recidivate” are very, very, very far from precise terms in their use (in this lecture, and in industry).

Largely, it corresponds to either being:

- **arrested**
- **charged**
- **or being found guilty of a crime**

usually one which is violent.

Can such predictions be fair with respect to race?

Pretrial release risk scale: 1-10
General recidivism scale
Violent recidivism scale

Attempt #1: don't use race as a feature
- Issue: race is strongly correlated with other features

Attempt #2: Remove any feature correlated with race
- Issue: (nearly) all features correlated with race

"Fairness through unawareness"

What exactly is the goal of this?

To accurately predict which people are most likely commit a crime if released, so they can be released

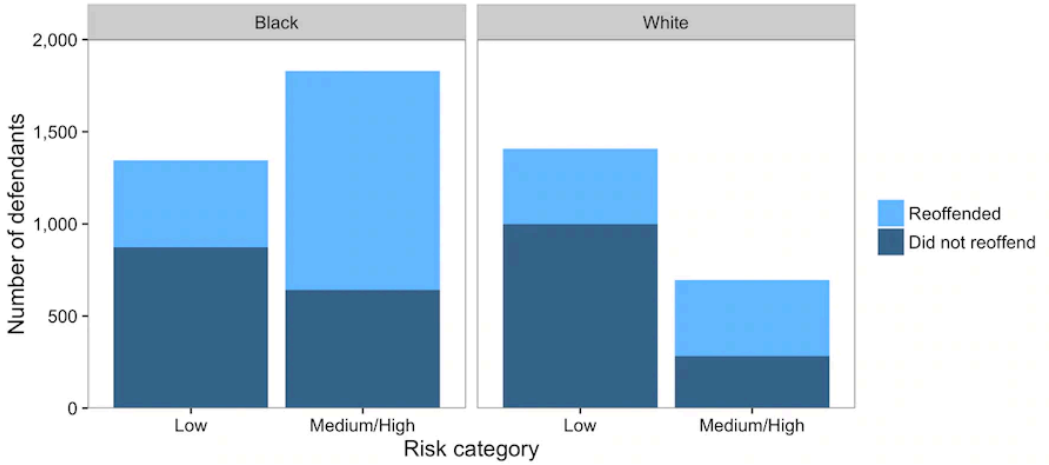
Concern: not all errors are equally costly.

… where these predictions should be accurate? similar? for people of all races.

Can such predictions be fair with respect to race?

NORTHPOINTE

Pretrial release risk scale: 1-10
General recidivism scale
Violent recidivism scale



$$\mathbb{P}[\text{does not reoffend}|\text{predicted low risk, white}] < \mathbb{P}[\text{does not reoffend}|\text{predicted low risk, black}]$$

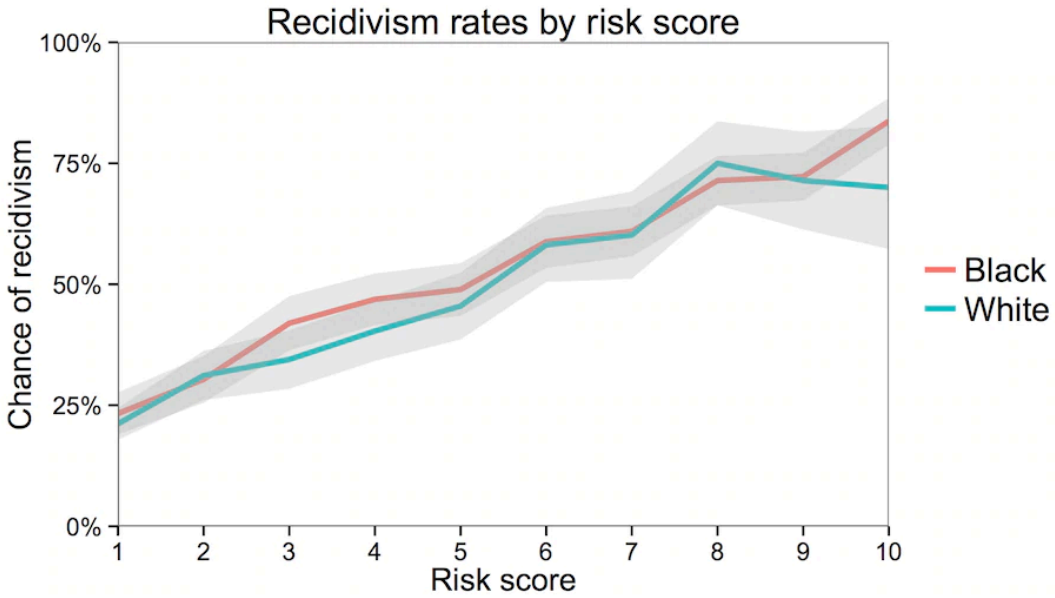
False positive rate for black defendants higher than for white defendants.

Source: Washington Post

Can such predictions be fair with respect to race?

NORTHPOINTE

Pretrial release risk scale: 1-10
General recidivism scale
Violent recidivism scale



$$\mathbb{P}[\text{reoffend}|\text{prediction, white}] \approx \mathbb{P}[\text{reoffend}|\text{prediction, black}]$$

Calibration of scores w.r.t. race

Source: Washington Post

Can any predictions be fair with respect to race?



$$\mathbb{P}[\text{does not reoffend}|\text{predicted low risk, white}] < \mathbb{P}[\text{does not reoffend}|\text{predicted low risk, black}]$$

while

$$\mathbb{P}[\text{reoffend}|\text{prediction, white}] \approx \mathbb{P}[\text{reoffend}|\text{prediction, black}]$$

How is this possible?

Is this avoidable?

Can any predictions be fair with respect to race?

False positives approximately equal and calibration are mutually exclusive, unless we have perfect predictions or the rate of what we are predicting is equal in every racial group.

… and this is true for more settings than just criminal justice.

Lending
Advertising
…

We are making a choice about how we allocate predictions.

Coin flips: accuracy and confidence

For iid coin flips, p is the chance of heads.
$\hat{p}$ is the fraction of heads in n flips.
$P(|\hat{p} - p| > \varepsilon) \leq 2 \exp(-2n\varepsilon^2)$
$n \geq \ln(2/\delta)/(2\varepsilon^2)$ gives confidence $1 - \delta$.
For $\varepsilon = \delta = 0.05$, $n = 738$ suffices.

Choosing a classifier requires control over many candidates.

Binary learning setup

D is a distribution on $X \times \{0,1\}$.
$S = ((x_1,y_1), \ldots, (x_n,y_n)) \sim D^n$ independently.
Choose hypothesis class H before seeing S.
Population risk: $R(h) = P(h(X) \neq Y)$
Training risk: $\hat{R}(h) = (1/n) \Sigma_i 1[h(x_i) \neq y_i]$
δ is failure probability over the sampled dataset.

Realizability and PAC

Realizable: some $h^* \in H$ has $R(h^*) = 0$.
A consistent learner returns $\hat{h}$ with $\hat{R}(\hat{h}) = 0$.
We want $P_s[R(\hat{h}) \leq \varepsilon] \geq 1 - \delta$.

ε controls absolute population risk.
δ controls how often the guarantee can fail.

A bad hypothesis can look perfect

Fix h with population risk $R(h) > \varepsilon$.
Each iid example misses its errors with probability $1 - R(h)$.

$$P_s[\hat{R}(h) = 0] = (1 - R(h))^n$$
$$\leq (1 - \varepsilon)^n \leq \exp(-n\varepsilon)$$

A fixed bad hypothesis rarely survives many examples.

Finite classes: union bound

Let $M = |H|$. A failure requires some bad h to fit S.

$$P_s[\exists\, h \in H : R(h) > \varepsilon \text{ and } \hat{R}(h) = 0] \leq M \exp(-n\varepsilon)$$

If $n \geq [\ln M + \ln(1/\delta)] / \varepsilon$, then with probability $\geq 1 - \delta$ every consistent h has $R(h) \leq \varepsilon$.

Assumptions: iid, finite H, realizability.

Agnostic learning

No perfect classifier is assumed. For finite H, let $M = |H|$.

$P(\sup_h |R(h) - \hat{R}(h)| > \alpha) \leq 2M \exp(-2n\alpha^2)$.

On the event $\sup_h |R(h) - \hat{R}(h)| \leq \alpha$, exact ERM satisfies

$R(\hat{h}) \leq \min_{h \in H} R(h) + 2\alpha$.

Thus $n \geq (2/\varepsilon^2) \ln(2M/\delta)$ ensures

$R(\hat{h}) \leq \min_{h \in H} R(h) + \varepsilon$ with probability $\geq 1 - \delta$.

ε is excess risk relative to the best h in H.

Reading the guarantees

The finite-class bounds just proved:
Realizable, consistent learner: absolute risk $\leq \varepsilon$
Sufficient sample scaling is proportional to $1/\varepsilon$.

Agnostic, exact ERM: excess risk $\leq \varepsilon$ relative to H
Sufficient sample scaling is proportional to $1/\varepsilon^2$.

These are conservative sufficient guarantees under different assumptions, not predicted data needs.

Shattering and VC dimension

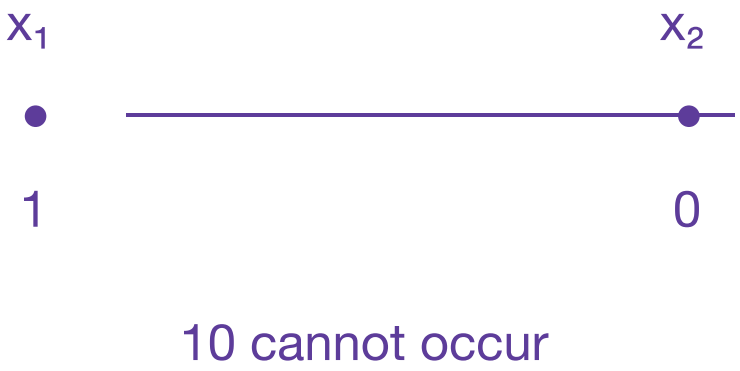
H shatters m points if it realizes all 2^m labelings.
A different h may realize each labeling.

VC(H) is the largest size of a shattered set.
It is infinite if arbitrarily large finite sets are shattered.
To establish VC(H) = v, exhibit one shattered v-set
and show no set of size v + 1 can be shattered.

Thresholds have VC dimension 1

$h_t(x) = 1[x \geq t]$. One point can be 0 or 1.
For $x_1 < x_2$, 00, 01 and 11 are possible.
The labeling 10 is impossible.

A threshold class is infinite, yet its VC dimension is 1.



Intervals have VC dimension 2

h(x) = 1[a ≤ x ≤ b]. Two points admit all four labels:
00, 01, 10, 11.

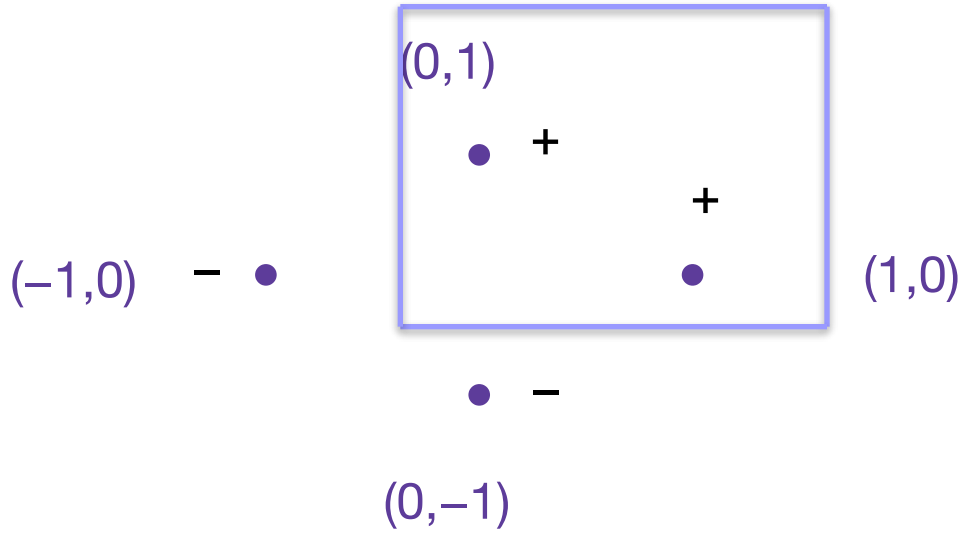
Three ordered points cannot be labeled 101:
an interval containing the outer points contains the middle.



For three points: 1 0 1 is impossible

Rectangles: four points can be shattered

Axis-aligned rectangles in the plane have VC = 4.
Shown: top and right points are positive, others negative.
Choose bounds to include a selected subset and exclude the rest.
One labeling illustrates the idea; shattering requires all 16.



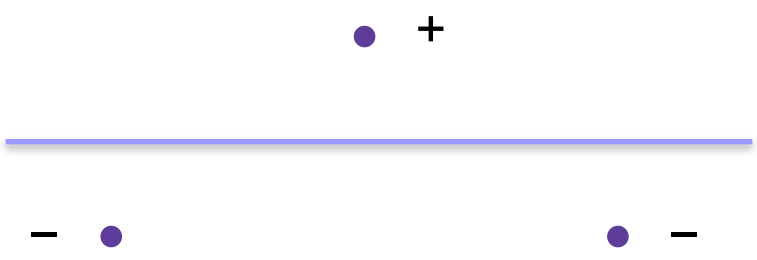
Rectangles: five points cannot be shattered

For any five points, select up to four points achieving
the smallest and largest x- and y-coordinates.
Label the selected extrema 1 and an unselected point 0.

Any axis-aligned rectangle containing all extrema
also contains the unselected point.
Thus no five-point set is shattered, so VC = 4.

Affine halfspaces

$h(x) = 1[w^T x + b \geq 0]$. Three noncollinear points
in the plane can be shattered. One labeling is shown below.
Symmetry and complements cover other nonconstant labelings.
In R^d, affine halfspaces have VC dimension d + 1.
The points 0, e_1, …, e_d form a shattered set.
The upper bound uses affine dependence. Homogeneous VC = d.



A noncollinear triangle

Growth functions

$\Pi_H(n)$: the most labelings H realizes on n points.
For finite v = VC(H), $\Pi_H(n) = 2^n$ for $n \le v$.
Sauer–Shelah: $\Pi_H(n) \le \Sigma_{i=0}^{\min(v,n)} C(n,i)$.
For $n \ge v \ge 1$, $\Pi_H(n) \le (en/v)^v$.

For fixed v, the number of patterns grows polynomially.
A rigorous proof bridge uses a ghost sample/symmetrization.

VC dimension and sample complexity

Binary 0–1 loss, finite $v = VC(H)$, and iid data.
For a universal constant C, these bounds suffice:

Realizable, consistent learner: if
$n \geq C[v \ln(1/\varepsilon) + \ln(1/\delta)]/\varepsilon$, then $R(\hat{h}) \leq \varepsilon$.
Agnostic ERM: if $n \geq C[v + \ln(1/\delta)]/\varepsilon^2$, then
$R(\hat{h}) \leq \inf_{h \in H} R(h) + \varepsilon$.
Each conclusion holds with probability $\geq 1 - \delta$.

What the guarantees assume

Training and future examples follow the same D.
H is fixed before the sample, and the stated learning assumptions must hold.

For binary 0–1 classification, finite VC dimension characterizes distribution-free PAC learnability.
It does not ensure computational efficiency, fairness, or robustness to distribution shift.

Discussion: What changes when the deployment distribution shifts?

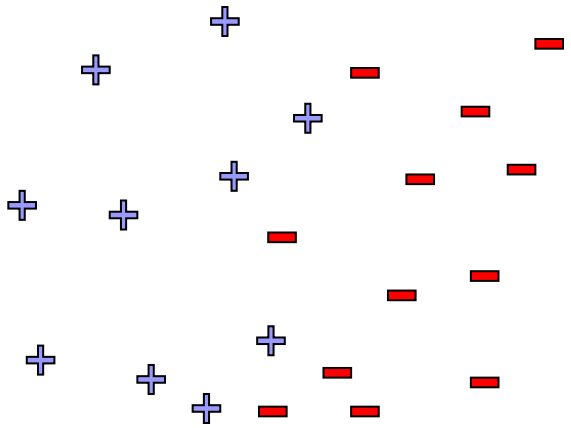
Kernel methods

©2018 Kevin Jamieson



What if the data is not linearly separable?

Use features of features of features…



$$\phi(x) : \mathbb{R}^d \rightarrow \mathbb{R}^p$$

Feature space can get really large really quickly!

©2018 Kevin Jamieson

Dot-product of polynomials

$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) =$ polynomials of degree exactly d

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

©2018 Kevin Jamieson

Dot-product of polynomials

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

©2018 Kevin Jamieson

Dot-product of polynomials

$$\Phi(\mathbf{u}) \cdot \Phi(\mathbf{v}) = \text{polynomials of degree exactly d}$$

$$d = 1 : \phi(u) = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1 v_1 + u_2 v_2$$

$$d = 2 : \phi(u) = \begin{bmatrix} u_1^2 \\ u_2^2 \\ u_1 u_2 \\ u_2 u_1 \end{bmatrix} \qquad \langle \phi(u), \phi(v) \rangle = u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 u_2 v_1 v_2$$

Feature space can get really large really quickly!

General d :

Dimension of $\phi(u)$ is roughly p^d if $u \in \mathbb{R}^p$

©2018 Kevin Jamieson

How do we deal with high-dimensional lifts/data?

A fundamental trick in ML: use kernels

A function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *kernel* for a map ϕ if $K(x, x') = \phi(x) \cdot \phi(x')$ for all x, x'.

So, if we can represent our algorithms/decision rules as dot products
and we can find a kernel for our feature map
then we can avoid explicitly dealing with ϕ(x).

©2018 Kevin Jamieson

Examples of Kernels

- **Polynomials of degree exactly d**

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v})^p$$

- **Polynomials of degree up to d**

$$K(\mathbf{u}, \mathbf{v}) = (\mathbf{u} \cdot \mathbf{v} + 1)^p$$

- **Gaussian (squared exponential) kernel**

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||^2}{2\sigma^2}\right)$$

- **Sigmoid**

$$K(u, v) = \tanh(\gamma \cdot u^T v + r)$$

©2018 Kevin Jamieson

The Kernel Trick

Pick a kernel K

Prove $w = \sum_i \alpha_i x_i$

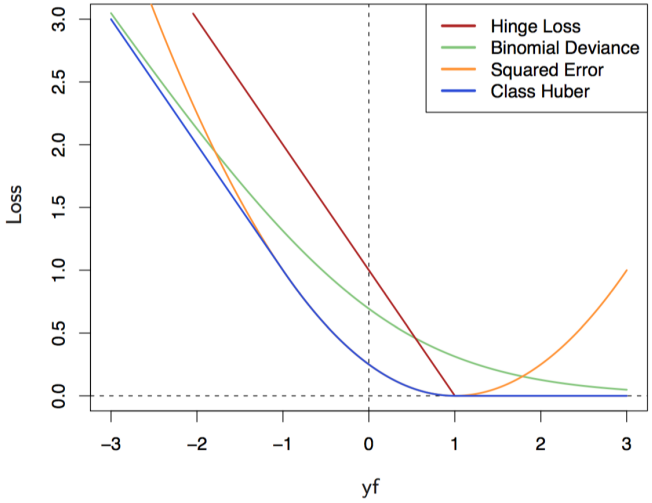
Change loss function/decision rule to only access data through dot products

- **Decision rule is easy: why?**

Substitute $K(x_i, x_j)$ for $x_i^T x_j$

©2018 Kevin Jamieson

Loss Functions



$$\{(x_i, y_i)\}_{i=1}^n \qquad x_i \in \mathbb{R}^d \qquad y_i \in \mathbb{R}$$

- Loss functions: $\sum_{i=1}^{n} \ell_i(w)$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

0/1 loss: $\ell_i(w) = \mathbb{I}[\text{sign}(y_i) \neq \text{sign}(x_i^T w)]$

Hinge Loss: $\ell_i(w) = \max\{0, 1 - y_i x_i^T w\}$

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

There exists an $\alpha \in \mathbb{R}^n$: $\widehat{w} = \sum_{i=1}^{n} \alpha_i x_i$ Why?

©2018 Kevin Jamieson

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

$$\text{There exists an } \alpha \in \mathbb{R}^n \text{: } \widehat{w} = \sum_{i=1}^{n} \alpha_i x_i$$

$$\widehat{\alpha} = \arg\min_{\alpha} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle$$

©2018 Kevin Jamieson

The Kernel Trick for regularized least squares

$$\widehat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^T w)^2 + \lambda ||w||_w^2$$

$$\text{There exists an } \alpha \in \mathbb{R}^n \text{: } \widehat{w} = \sum_{i=1}^{n} \alpha_i x_i$$

$$\begin{aligned}
\widehat{\alpha} &= \arg\min_{\alpha} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j \langle x_j, x_i \rangle)^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle \\
&= \arg\min_{\alpha} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_j K(x_i, x_j))^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) \\
&= \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha
\end{aligned}$$

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

©2018 Kevin Jamieson

Why regularization?

Typically, $\mathbf{K} \succ 0$. What if $\lambda = 0$?

$$\widehat{\alpha} = \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

©2018 Kevin Jamieson

Why regularization?

$$\text{Typically, } \mathbf{K} \succ 0. \qquad \text{What if } \lambda = 0?$$

$$\widehat{\alpha} = \arg\min_{\alpha} ||\mathbf{y} - \mathbf{K}\alpha||_2^2 + \lambda \alpha^T \mathbf{K} \alpha$$

Unregularized kernel least squares can (over) fit **any data**!

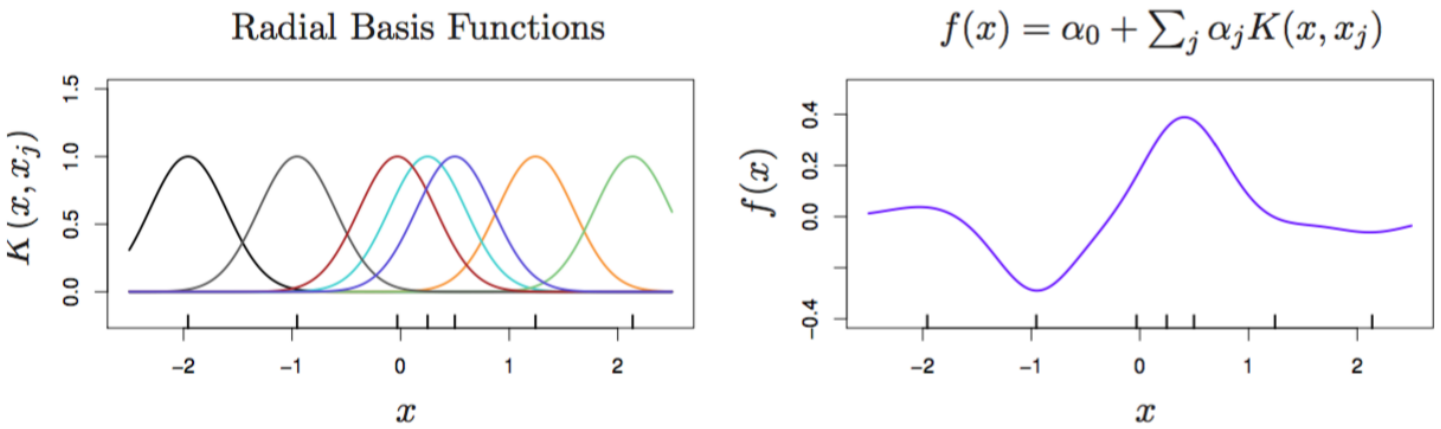
$$\widehat{\alpha} = \mathbf{K}^{-1}\mathbf{y}$$

©2018 Kevin Jamieson

RBF Kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

Note that this is like weighting "bumps" on each point like kernel smoothing but now we learn the weights



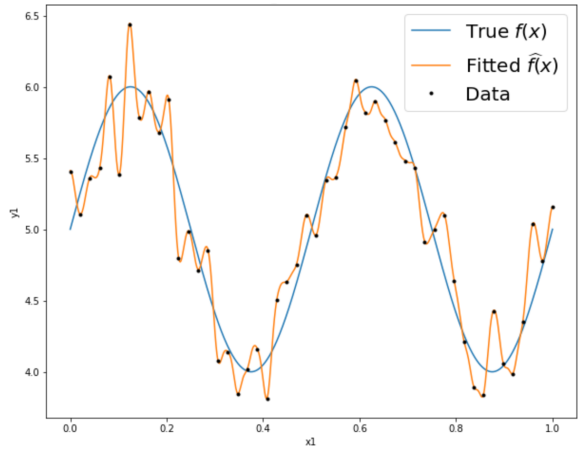
©2018 Kevin Jamieson

RBF Kernel

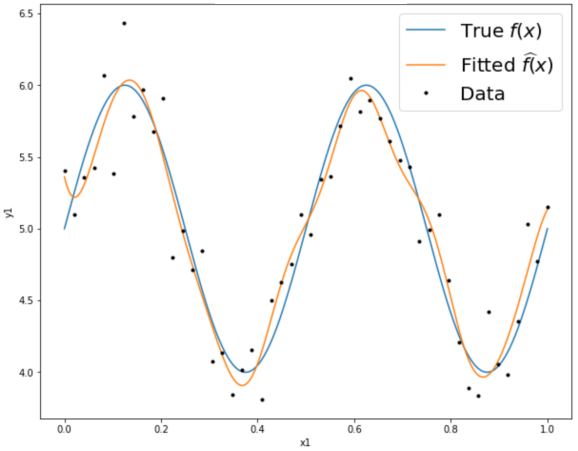
$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

The bandwidth sigma has an enormous effect on fit:

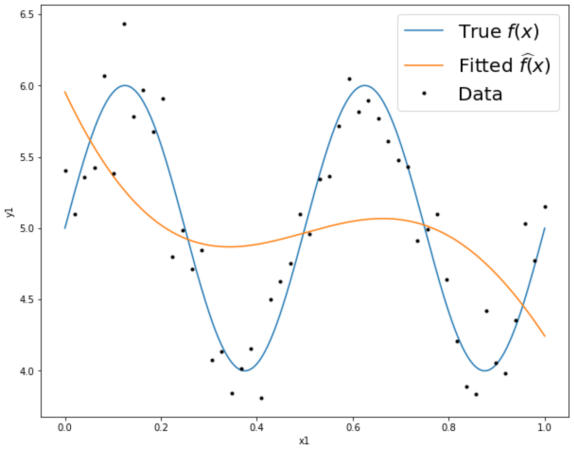
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



$\sigma = 10^{-1}$ $\lambda = 10^{-4}$



$\sigma = 10^{-0}$ $\lambda = 10^{-4}$



- **Note that this is like weighting "bumps" on each point like kernel smoothing but now we learn the weights**

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

©2018 Kevin Jamieson

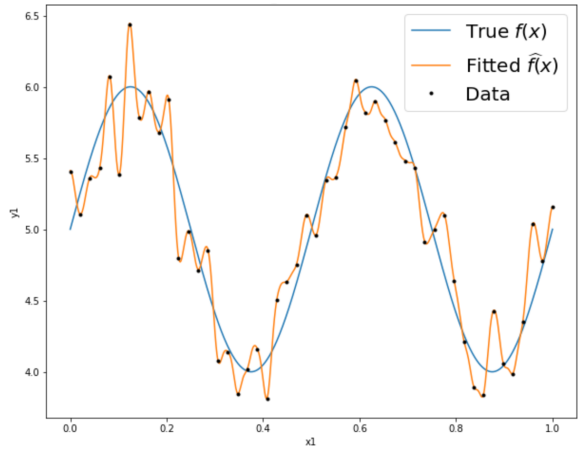
RBF Kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

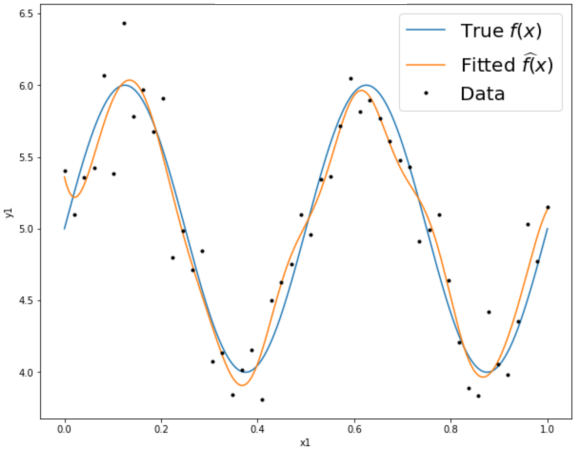
- Note that this is like weighting "bumps" on each point like kernel smoothing but now we learn the weights

The bandwidth sigma has an enormous effect on fit:

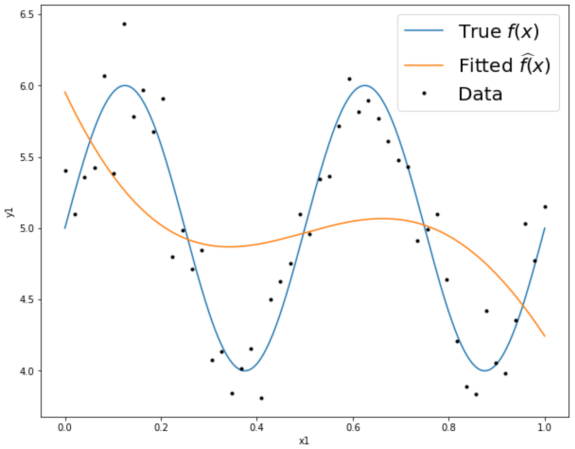
$\sigma = 10^{-2}$ $\lambda = 10^{-4}$



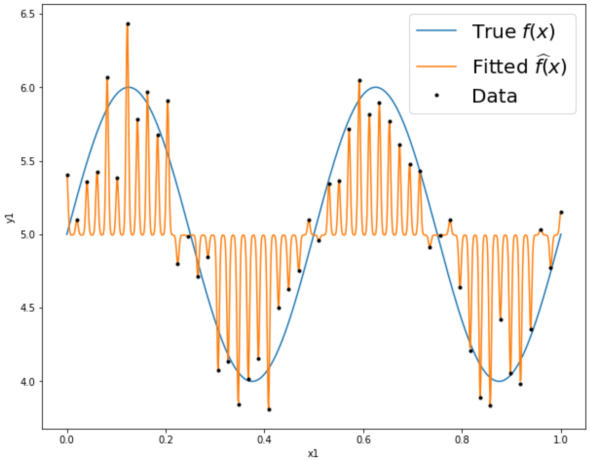
$\sigma = 10^{-1}$ $\lambda = 10^{-4}$



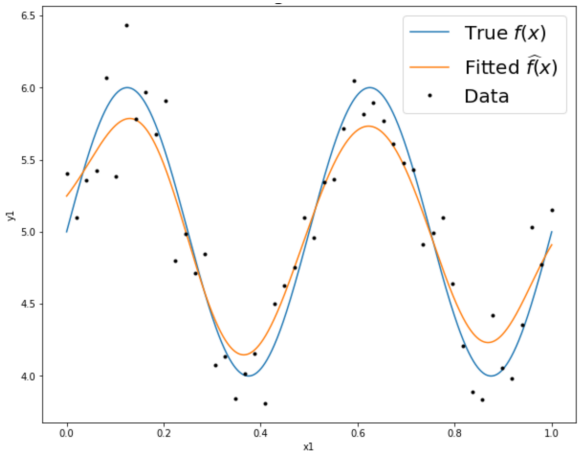
$\sigma = 10^{-0}$ $\lambda = 10^{-4}$



$\sigma = 10^{-3}$ $\lambda = 10^{-4}$



$\sigma = 10^{-1}$ $\lambda = 10^{-0}$



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

©2018 Kevin Jamieson

RBF Kernel

$$K(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{||\mathbf{u} - \mathbf{v}||_2^2}{2\sigma^2}\right)$$

Basis representation in 1d?

$$[\phi(x)]_i = \frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i \quad \text{for } i = 0, 1, \ldots$$

$$\begin{aligned}
\phi(x)^T \phi(x') &= \sum_{i=0}^{\infty} \left(\frac{1}{\sqrt{i!}} e^{-\frac{x^2}{2}} x^i\right)\left(\frac{1}{\sqrt{i!}} e^{-\frac{(x')^2}{2}} (x')^i\right) \\
&= e^{-\frac{x^2+(x')^2}{2}} \sum_{i=0}^{\infty} \frac{1}{i!} (xx')^i \\
&= e^{-|x-x'|^2/2}
\end{aligned}$$

> **Note that this is like weighting "bumps" on each point like kernel smoothing but now we learn the weights**

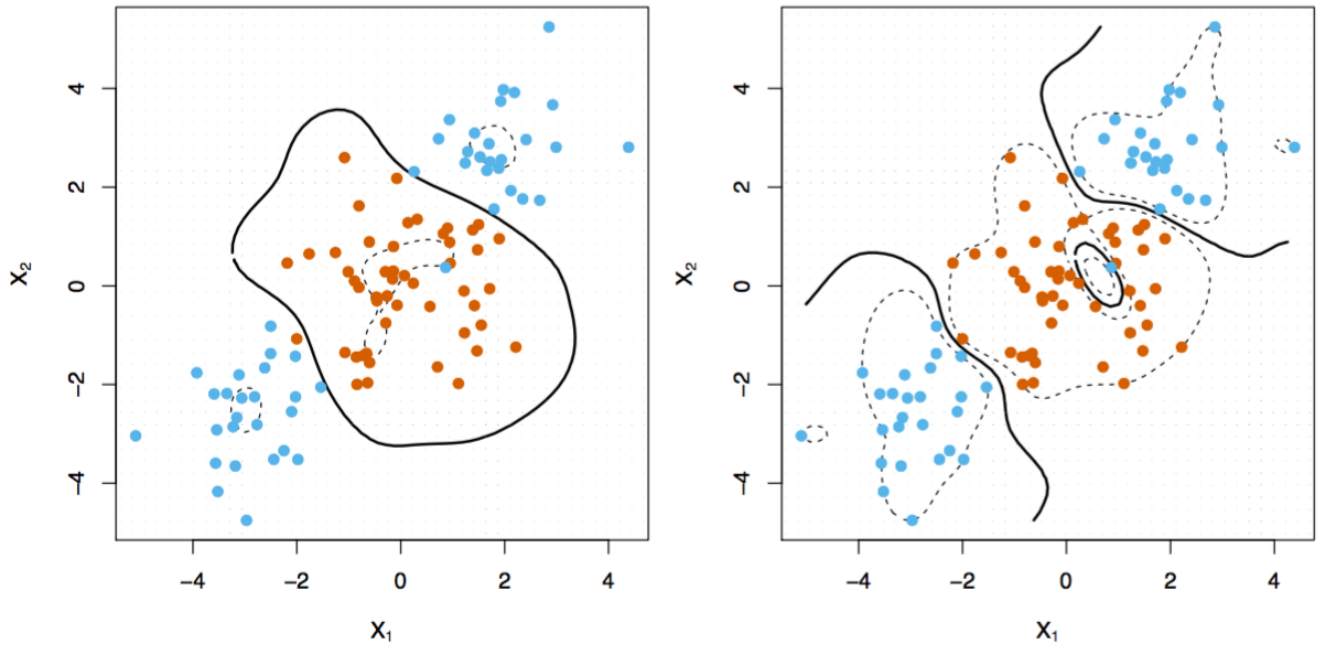
If n is very large, allocating an n-by-n matrix is tough. Can we truncate the above sum to approximate the kernel?

©2018 Kevin Jamieson

RBF Classification

$$\widehat{w} = \sum_{i=1}^{n} \max\{0, 1 - y_i(b + x_i^T w)\} + \lambda ||w||_2^2$$

$$\min_{\alpha,b} \sum_{i=1}^{n} \max\{0, 1 - y_i(b + \sum_{j=1}^{n} \alpha_j \langle x_i, x_j \rangle)\} + \lambda \sum_{i,j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle$$



©2018 Kevin Jamieson