machine learning

chandan singh

press esc to navigate slides

announcements

- hw2 grades out
- review room permanent

midterm

- next thurs., class time, different room
- not on test: TLS, CCA
- cheat sheet allowed, not necessary (handwritten)

today

- optimization
- neural nets

reference

http://www.eecs189.org/

https://tinyurl.com/cs189feedback

pre-reqs

- linear algebra
- matrix calculus
- probability
- numpy/matplotlib

introduction

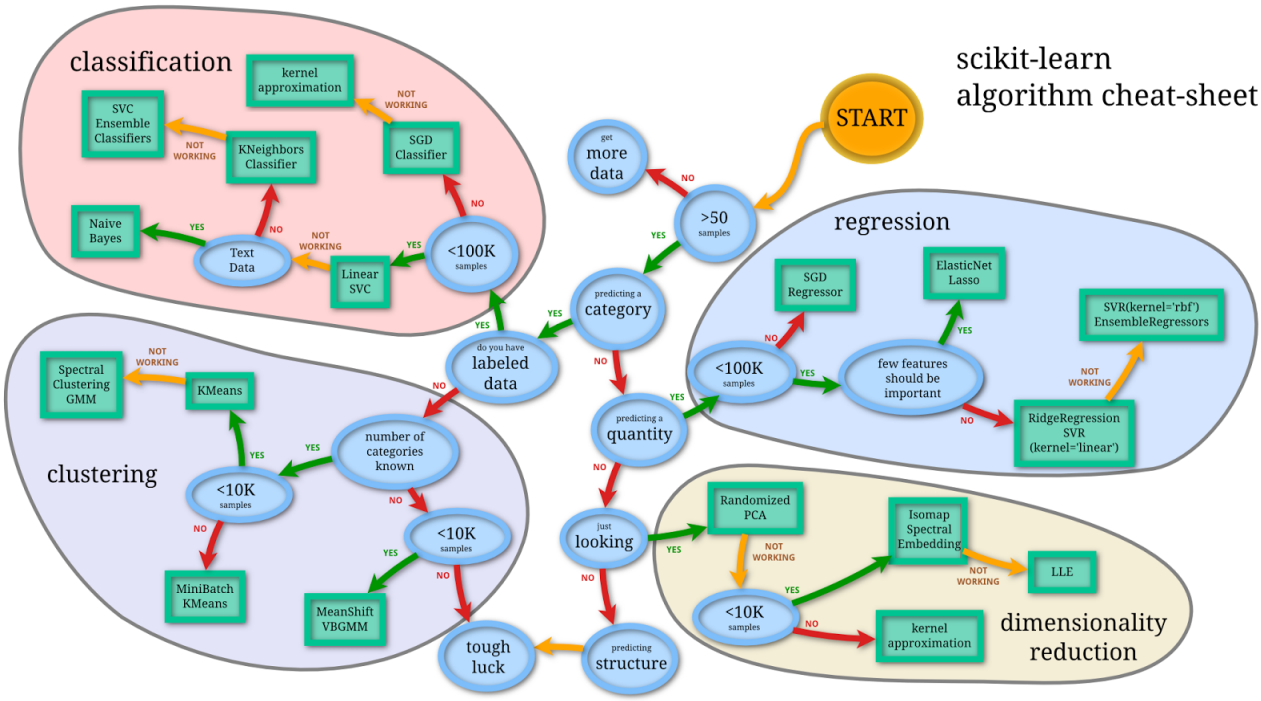
what's in an ml problem?

1. applications + data
2. model
3. loss function
4. optimization algorithm

types of ml problems

- regression
- classification
- density estimation
- dimensionality reduction

visual roadmap



types of learning (by amount of labels)

- supervised
- ~semi-supervised
- reinforcement
- unsupervised

types of models

- sort by problem
- sort by bias/variance
- discriminative vs. generative

parameters and hyperparameters

- result of optimization are a set of parameters
- parameters vs. hyperparameters

errors

- training vs. testing error
- why do we need cross-validation?
- example: train, validate, test

linear algebra review

matrix properties

- *nonsingular* = invertible = nonzero determinant = null space of zero $\implies$ square
    - *rank* = dimension
    - *ill-conditioned matrix* - matrix is close to being singular - very small determinant

vector norms

- $||x||$ is the L_2 norm
- $||x||^2 = x^T x$
- L_p–norms: $||x||_p = (\sum_i \|x_i\|^p)^{1/p}$

matrix norms

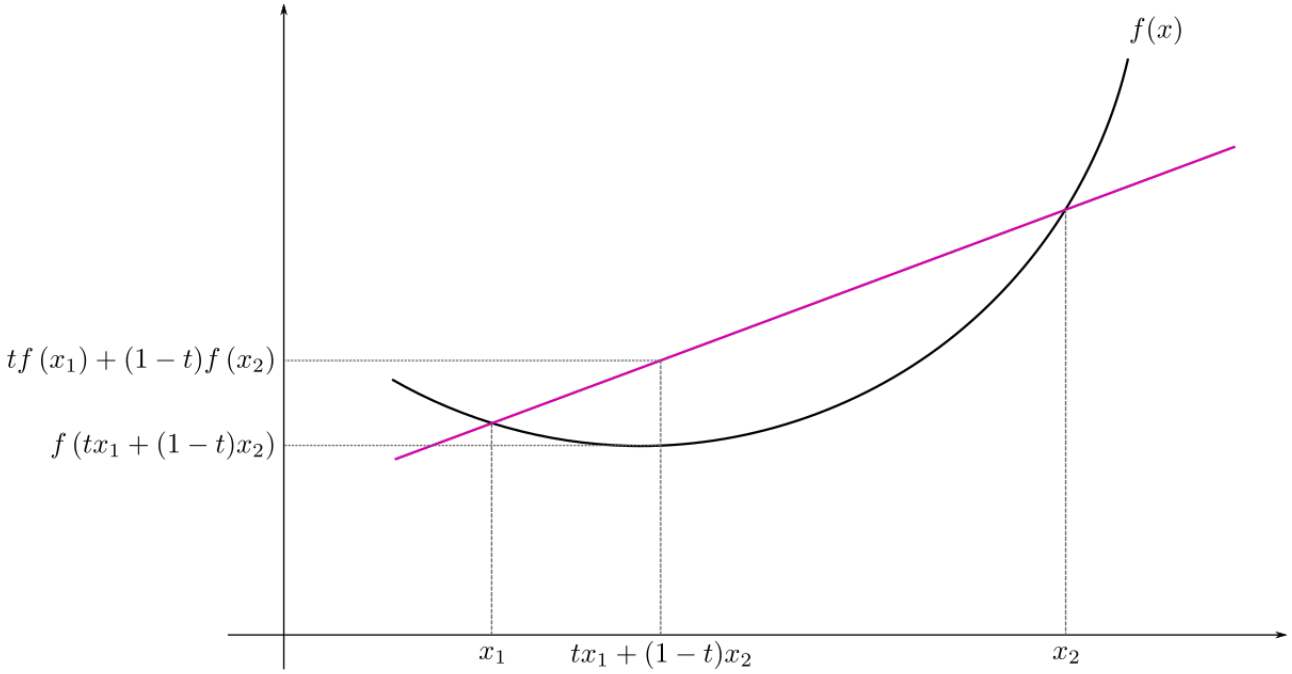
- *frobenius*: like L_2 for matrices
- *spectral norm* = L_2-norm of a matrix
    - $||X||_2 = \underset{max}{\sigma}(X)$

cauchy-shwartz inequality

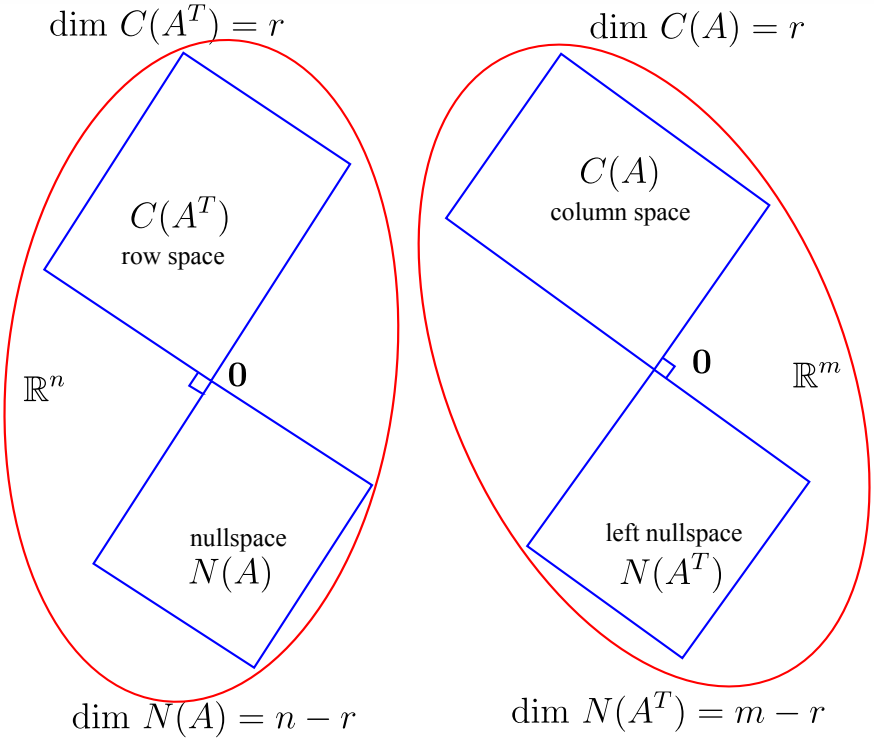
- $|x^T y| \leq ||x||_2 ||y||_2$
- $\implies ||x+y||^2 \leq (||x|| + ||y||)^2$

jensen's inequality

- $f(E[X]) \leq E[f(X)]$ for convex f



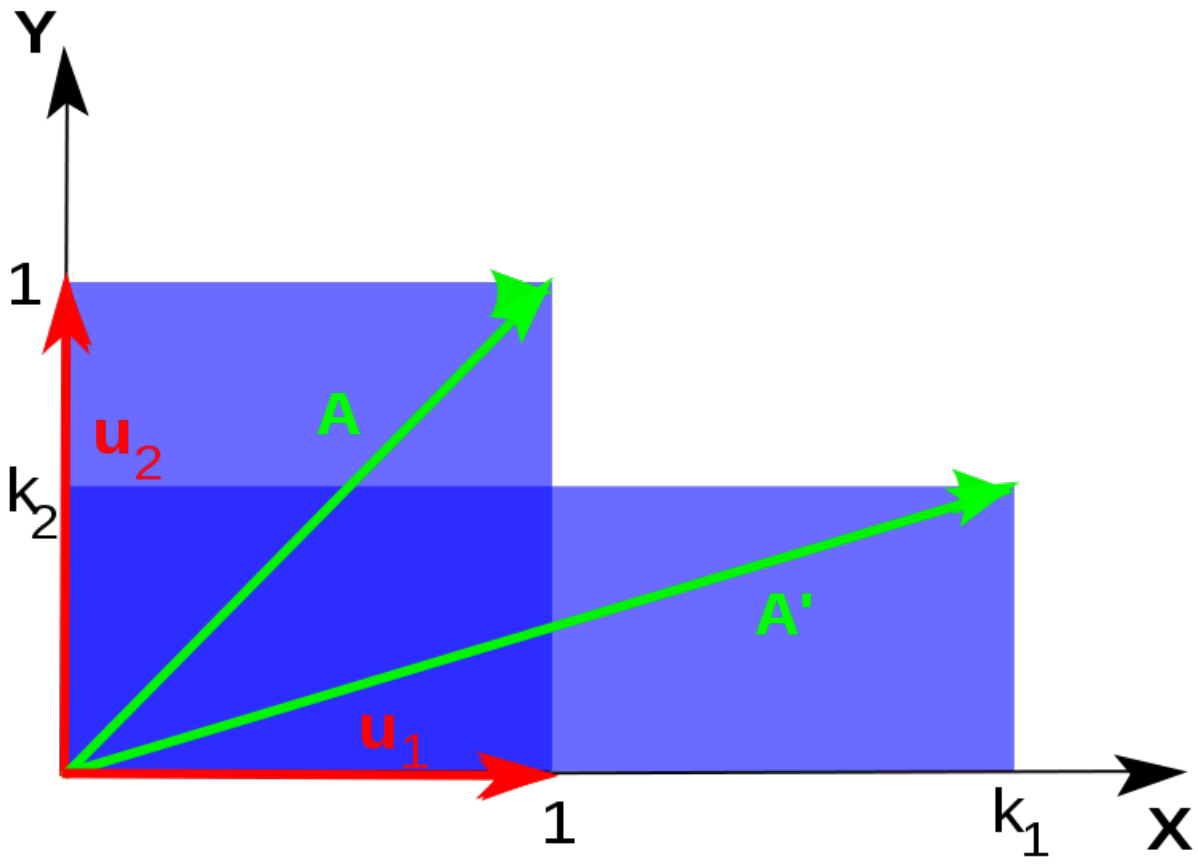
subspaces



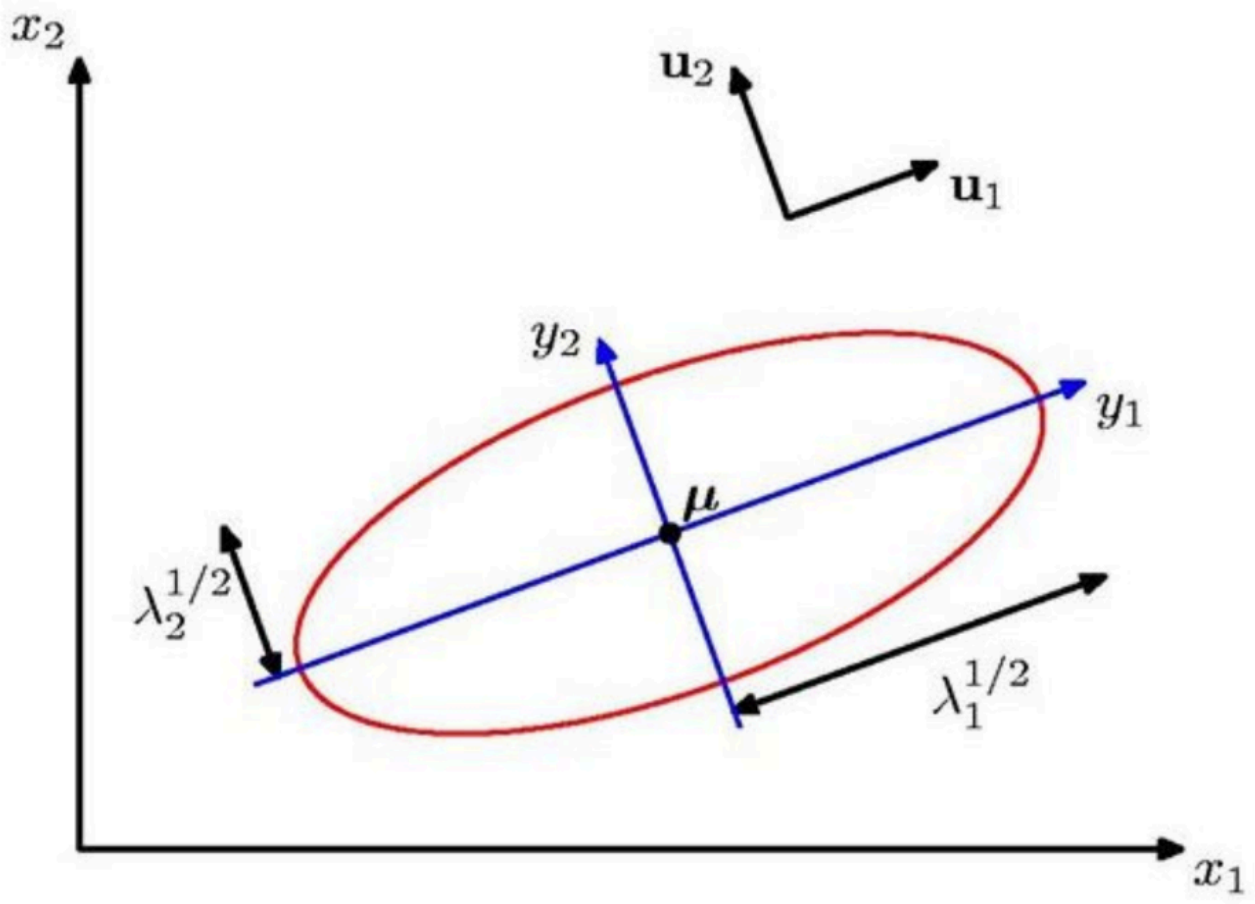
eigenvalues

- *eigenvalue eqn*: $Ax = \lambda x \implies (A - \lambda I)x = 0$
  - $det(A - \lambda I) = 0$ yields *characteristic polynomial*

eigenvectors



eigenvectors in pca



evd

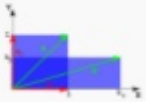
- *diagonalization = eigenvalue decomposition =* *spectral decomposition
- assume A (nxn) is symmetric
  - $A = Q\Lambda Q^T$
  - Q := eigenvectors as columns, Q is orthonormal
  - Λ diagonal

evd extended

- only diagonalizable if n independent eigenvectors
- how does evd relate to invertibility?

Eigenvalues of geometric transformations

The following table presents some example transformations in the plane along with their 2×2 matrices, eigenvalues, and eigenvectors.

	scaling	unequal scaling	rotation	horizontal shear	hyperbolic rotation
illustration					
matrix	$\begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$	$\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$	$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$ $c = \cos\theta$ $s = \sin\theta$	$\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} c & s \\ s & c \end{bmatrix}$ $c = \cosh\varphi$ $s = \sinh\varphi$
characteristic polynomial	$(\lambda - k)^2$	$(\lambda - k_1)(\lambda - k_2)$	$\lambda^2 - 2c\lambda + 1$	$(\lambda - 1)^2$	$\lambda^2 - 2c\lambda + 1$
eigenvalues λ_i	$\lambda_1 = \lambda_2 = k$	$\lambda_1 = k_1$ $\lambda_2 = k_2$	$\lambda_1 = e^{i\theta} = c + s\mathbf{i}$ $\lambda_2 = e^{-i\theta} = c - s\mathbf{i}$	$\lambda_1 = \lambda_2 = 1$	$\lambda_1 = e^{\varphi}$ $\lambda_2 = e^{-\varphi}$.
algebraic multipl. $\mu_i = \mu(\lambda_i)$	$\mu_1 = 2$	$\mu_1 = 1$ $\mu_2 = 1$	$\mu_1 = 1$ $\mu_2 = 1$	$\mu_1 = 2$	$\mu_1 = 1$ $\mu_2 = 1$
geometric multipl. $\gamma_i = \gamma(\lambda_i)$	$\gamma_1 = 2$	$\gamma_1 = 1$ $\gamma_2 = 1$	$\gamma_1 = 1$ $\gamma_2 = 1$	$\gamma_1 = 1$	$\gamma_1 = 1$ $\gamma_2 = 1$
eigenvectors	All non-zero vectors	$u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$u_1 = \begin{bmatrix} 1 \\ -\mathbf{i} \end{bmatrix}$ $u_2 = \begin{bmatrix} 1 \\ +\mathbf{i} \end{bmatrix}$	$u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$u_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $u_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.

svd

nxp matrix: $X = U\Sigma V^T$

- U (nxn) are eigenvectors of XX^T
- columns of V (pxp) are eigenvectors of X^TX
- r singular values on diagonal of Σ (nxp)
    - square roots of nonzero eigenvalues of both XX^T and X^TX

svd vs evd

- evd
    - not always orthogonal columns
    - complex eigenvalues
    - only square, not always possible
- svd
    - orthonormal columns

    - real/nonnegative eigenvalues
- symmetric matrices: eigenvalues real, eigenvectors orthogonal

eigen stuff

- expressions when $A \in \mathbb{S}$
    - $det(A) = \prod_i \lambda_i$
    - $tr(A) = \sum_i \lambda_i$
    - $|||A|||_2 = max|\lambda_i|$
    - $|||A|||_F = \sqrt{\sum \lambda_i^2}$
    - $\underset{max}{\lambda}(A) = sup_{x \neq 0} \frac{x^T Ax}{x^T x}$
    - $\underset{min}{\lambda}(A) = inf_{x \neq 0} \frac{x^T Ax}{x^T x}$

positive semi-definite notation

- $x \preceq y$ - these are vectors and x is less than y elementwise
- $X \preceq Y$ - matrices, $Y - X$ is PSD
    - $v^T Xv \leq v^T Yv \ \forall v$

psd

- all eigenvalues are nonnegative
- if $\forall x \in R^n, x^T Ax \geq 0$ then A is positive semi definite
    - like it curves up

matrix calculus

- *gradient* vector $\nabla_x f(Ax)$- partial derivatives with respect to each element of vector

jacobian

function f: $\mathbb{R}^n \rightarrow \mathbb{R}^m$

$$\textit{Jacobian matrix: } \mathbf{J} = \begin{bmatrix} \dfrac{\partial \mathbf{f}}{\partial x_1} & \cdots & \dfrac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix}$$

hessian

function f: $\mathbb{R}^n \to \mathbb{R}$

$$\mathbf{H} = \nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\, \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\, \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\, \partial x_1} & \dfrac{\partial^2 f}{\partial x_n\, \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

linear regression

regression

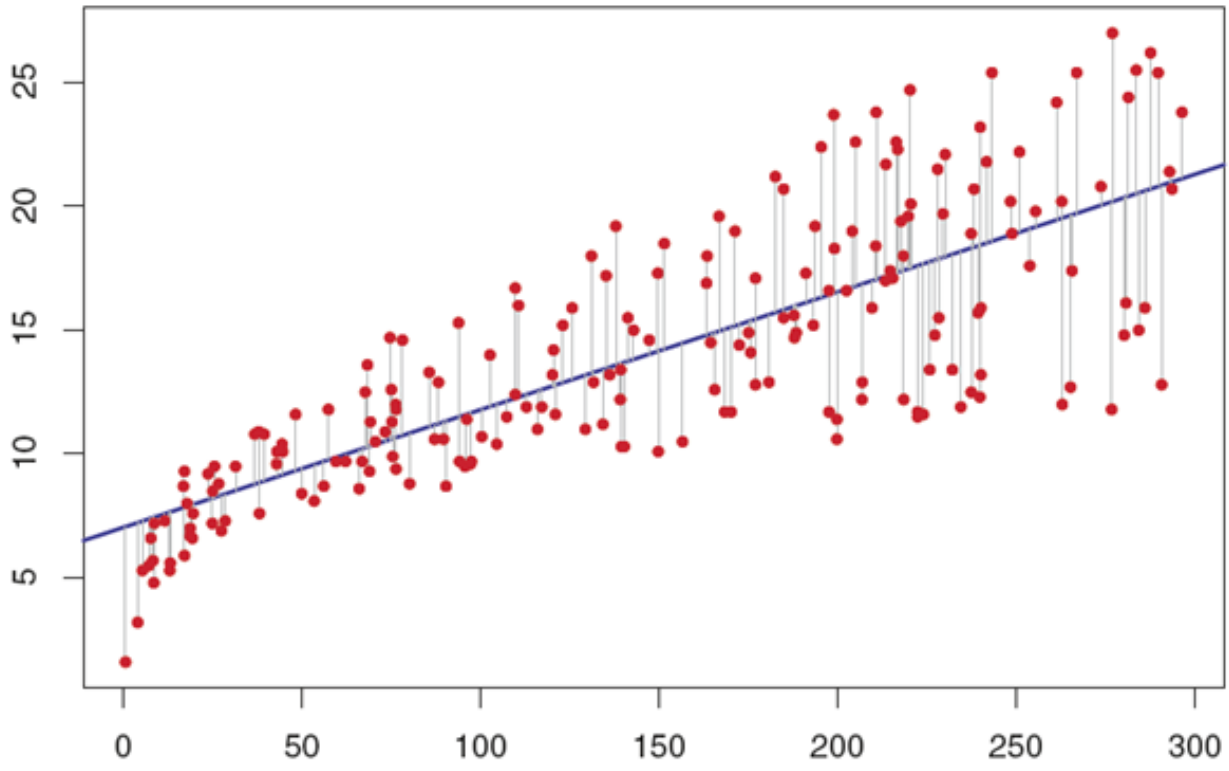
- what is regression?
- how does regression fit into the ml framework?

feature engineering

- what is x?
- what is y?

- $\phi(x)$ can be treated like x

lin. regression intuition 1



lin. regression intuition 2

- here, $\mathbf{y}$ is n-dimensional

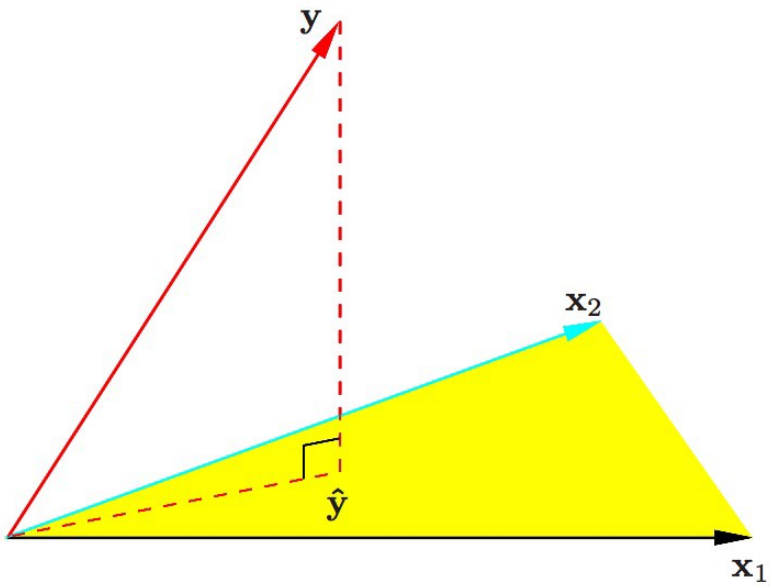


FIGURE 3.2. *The N-dimensional geometry of least squares regression with two predictors. The outcome vector* $\mathbf{y}$ *is orthogonally projected onto the hyperplane spanned by the input vectors* $\mathbf{x}_1$ *and* $\mathbf{x}_2$*. The projection* $\hat{\mathbf{y}}$ *represents the vector of the least squares predictions*

linear regression types

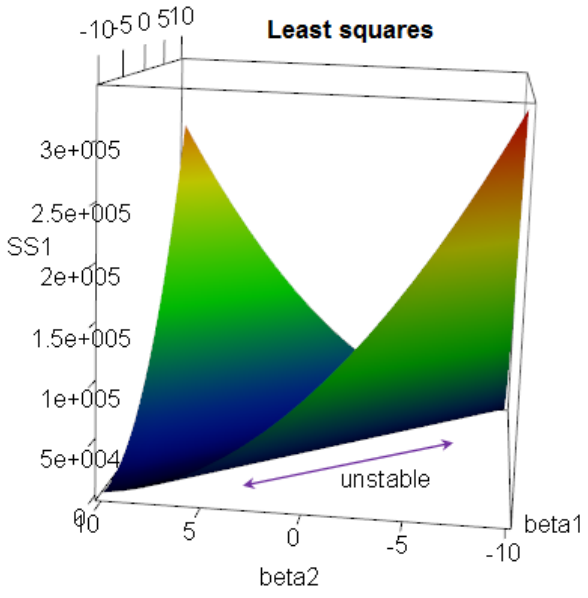
- assume data can be modeled well by a linear combination of the features: $\hat{y} = \mathbf{w}^T \mathbf{x}$
- OLS: $\sum_i (y_i - \hat{y}_i)^2$
- ridge: $\sum_i (y_i - \hat{y}_i)^2 + \lambda ||w||_2^2$
- lasso (will be covered later): $\sum_i (y_i - \hat{y}_i)^2 + \lambda ||w||_1$
- elastic net (will be covered later): $\sum_i (y_i - \hat{y}_i)^2 + \lambda_1 ||w||_1 + \lambda_2 ||w||_2^2$

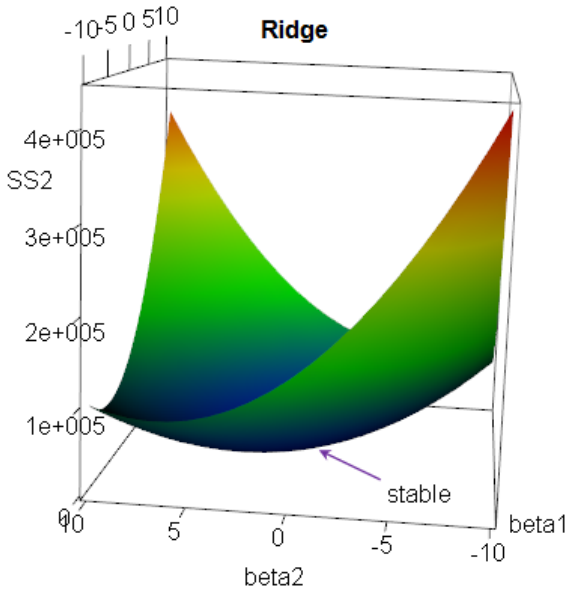
ols solution

- $w^*_{OLS} = (X^T X)^{-1} X^T y$
- 2 derivations: least squares, orthogonal projection

ridge regression intuition

- $w^*_{RIDGE} = (X^T X + \lambda I)^{-1} X^T y$

- 



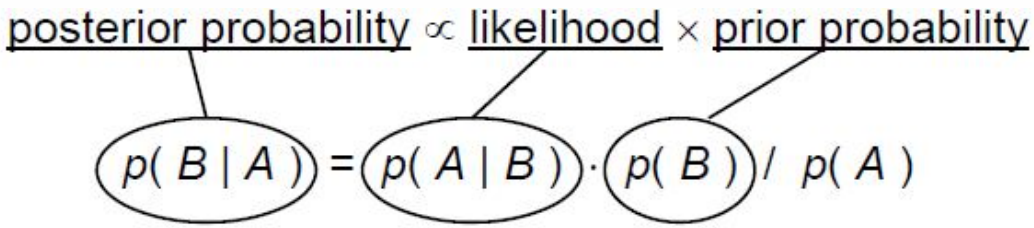
parameter estimation

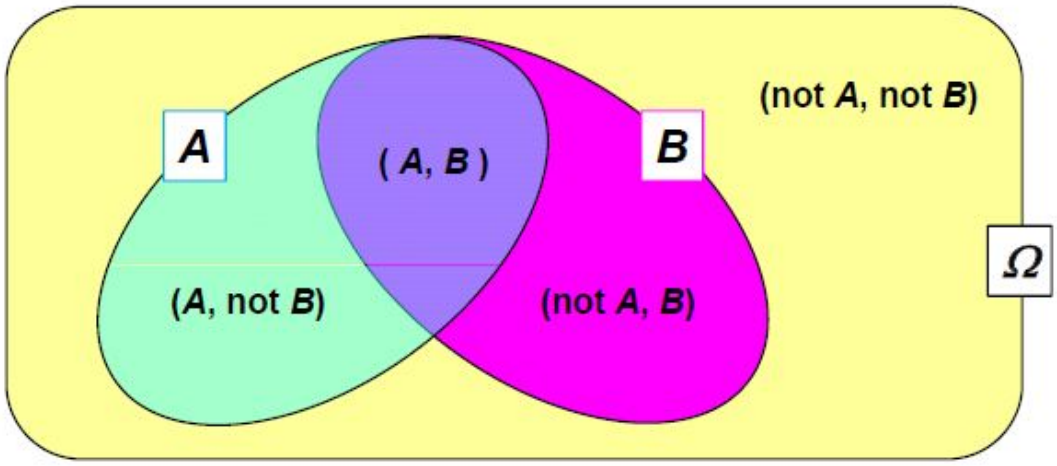
probabilistic model

- assume a **true underlying model**
- ex. $Y_i \sim \mathcal{N}(\theta^T X_i, \sigma^2)$
- this is equivalent to $P(Y_i|X_i;\theta) = \mathcal{N}(\theta^T X_i, \sigma^2)$

bayes rule

- $\overbrace{p(\theta|x)}^{\text{posterior}} = \frac{\overbrace{p(x|\theta)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{p(x)}$



- 

likelihood

$\mathcal{L} = p(data|\theta)$~ product over all n examples

- $p(x|\theta)$?
- $p(y|x;\theta)$?
- $p(x,y|\theta)$?
- depends on the problem

mle - maximum likelihood estimation

- $\hat{\theta}_{MLE} = \underset{\theta}{argmax}\ \mathcal{L}$
- associated with *frequentist* school of thought

how to do mle problems

- write likelihood (product of probabilities)
- usually take log to turn product into a sum
- take derivative and set to zero to maximize (assuming convexity)

map - maximum a posteriori

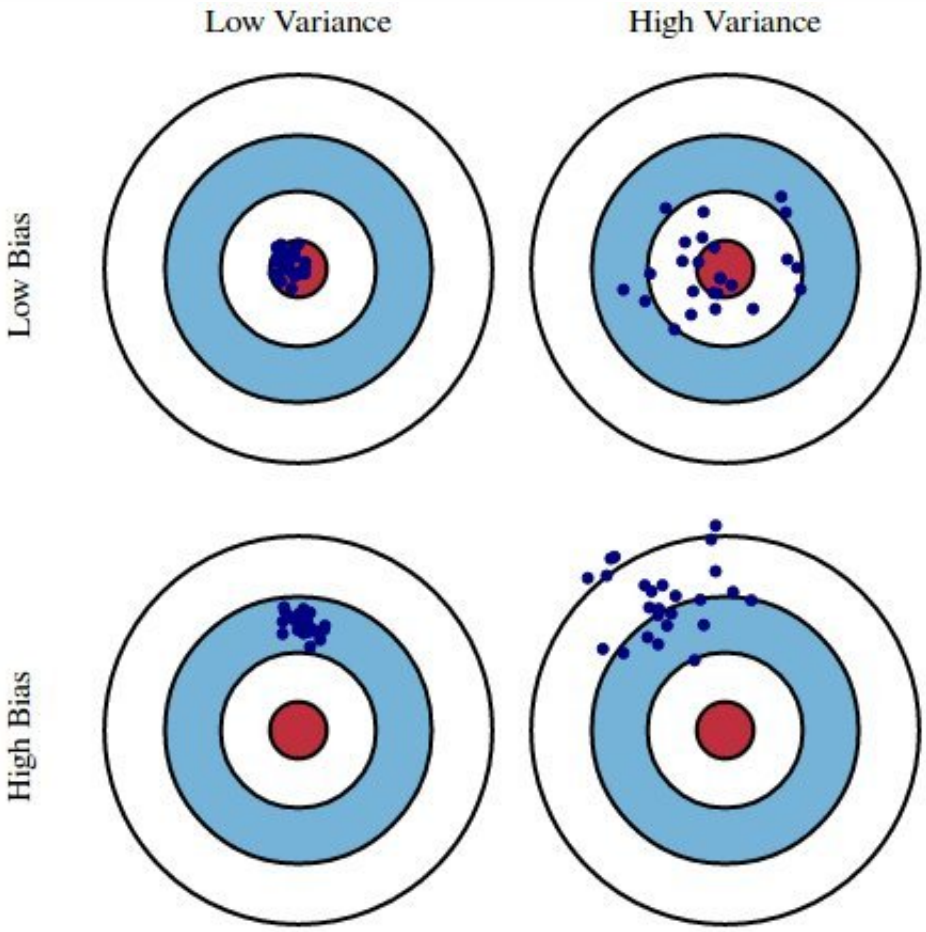
- $\hat{\theta}_{MAP} = \underset{\theta}{argmax}\, p(\theta|x) = \underset{\theta}{argmax}\, p(x|\theta)p(\theta)$
  $= \underset{\theta}{argmax}\, [log\, p(x|\theta) + log\, p(\theta)]$
    - $p(x)$ disappears because it doesn't depend on θ
- associated with *bayesian* school of thought

mle vs. map

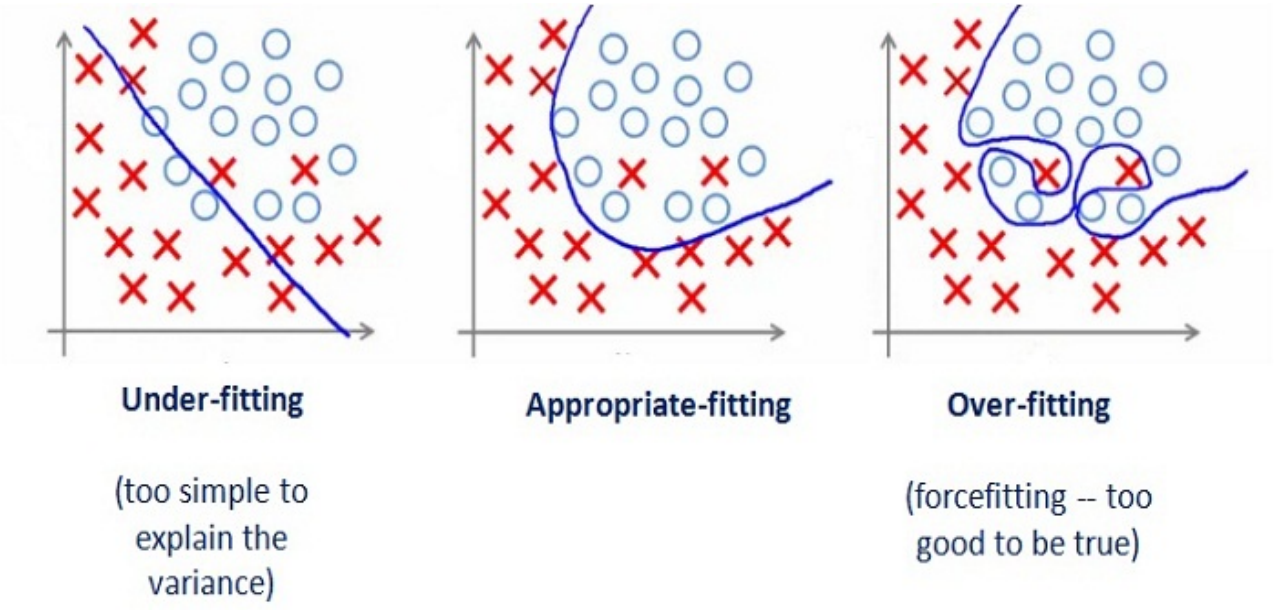
- $\hat{\theta}_{MLE} = \underset{\theta}{argmax}\, \overbrace{p(x|\theta)}^{\text{likelihood}}$
- $\hat{\theta}_{MAP} = \underset{\theta}{argmax}\, \overbrace{p(\theta|x)}^{\text{posterior}} = \underset{\theta}{argmax}\, p(x|\theta)\overbrace{p(\theta)}^{\text{prior}}$
    - `$\hat{\theta}_{\text{Bayes}} = E_\theta \: p(\theta|x) $`

bias-variance tradeoff

intuition 1



intuition 2



bias

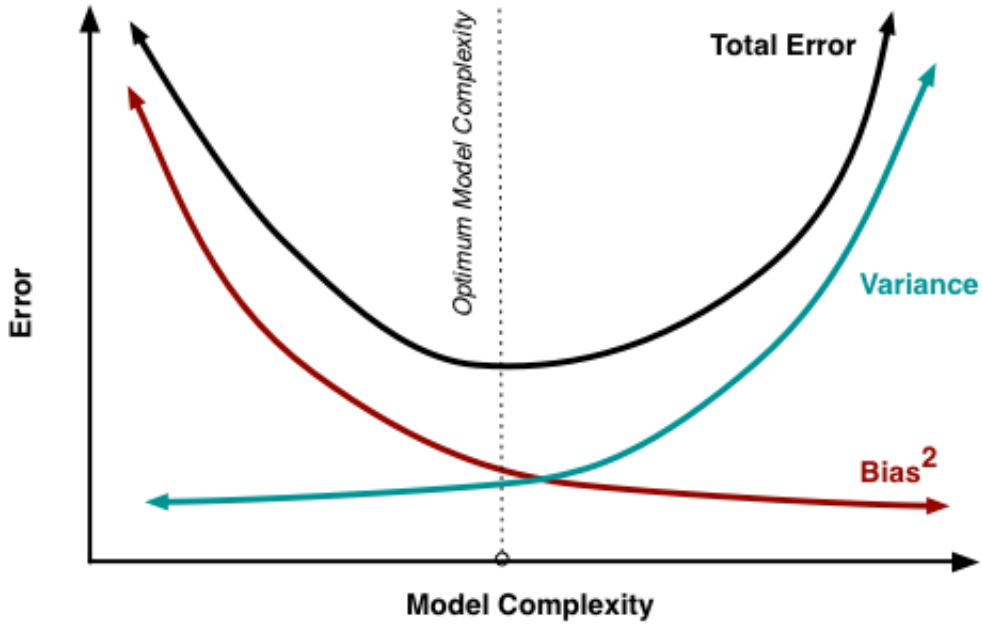
- bias of a method: $E[\hat{f}(x) - f(x)]$
    - averaging over training sets with fixed x
- could also have bias of a point estimate: $E[\hat{\theta} - \theta]$

variance

- "estimation error"
- variance of a method: $V[\hat{f}(x)] = E[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2]$
    - averaging over training sets with fixed x

bias-variance trade-off

- mean-squared error of method: $E[(\hat{f}(x) - f(x))^2]$
    - = bias^2 + variance
    - = $E[\hat{f}(x) - f(x)]^2$ + $E[(f(\hat{x}) - E[f(\hat{x})])^2]$

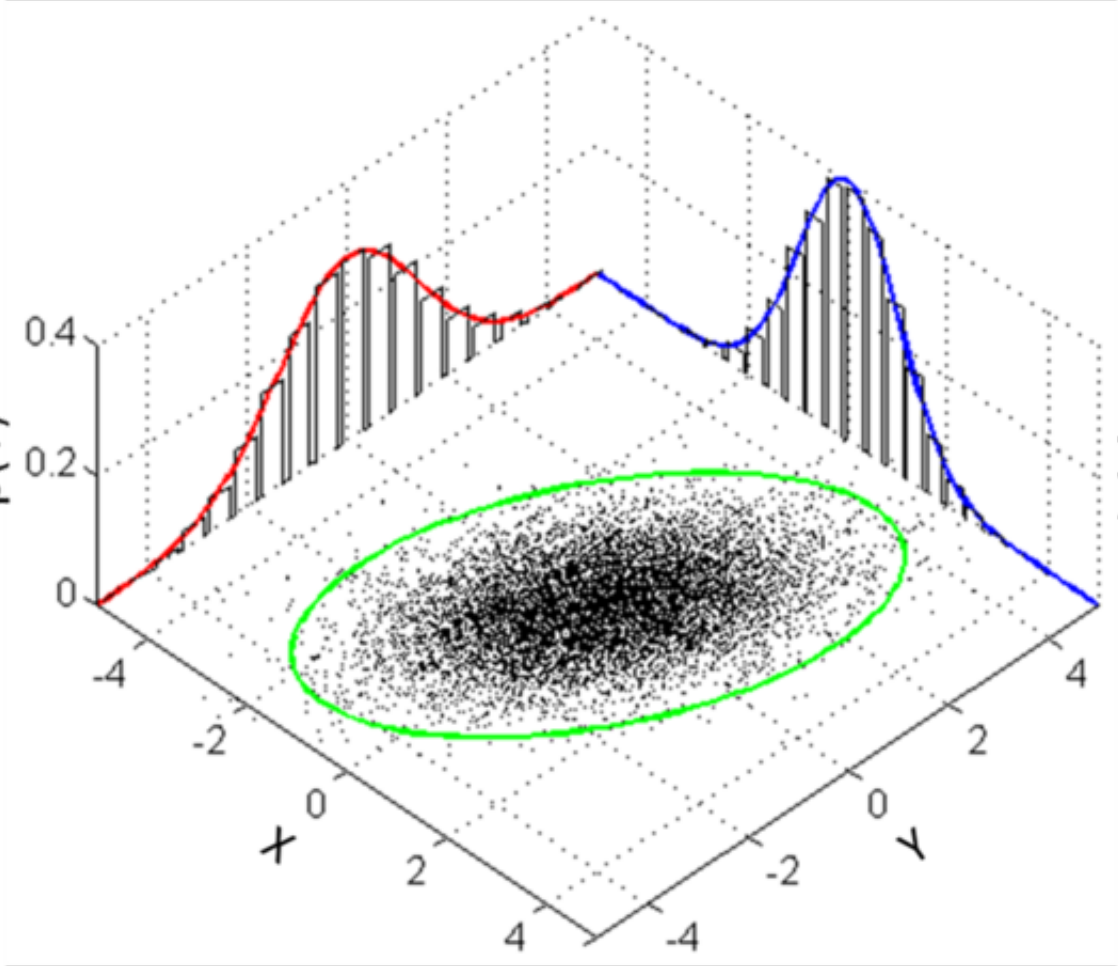


multivariate gaussian

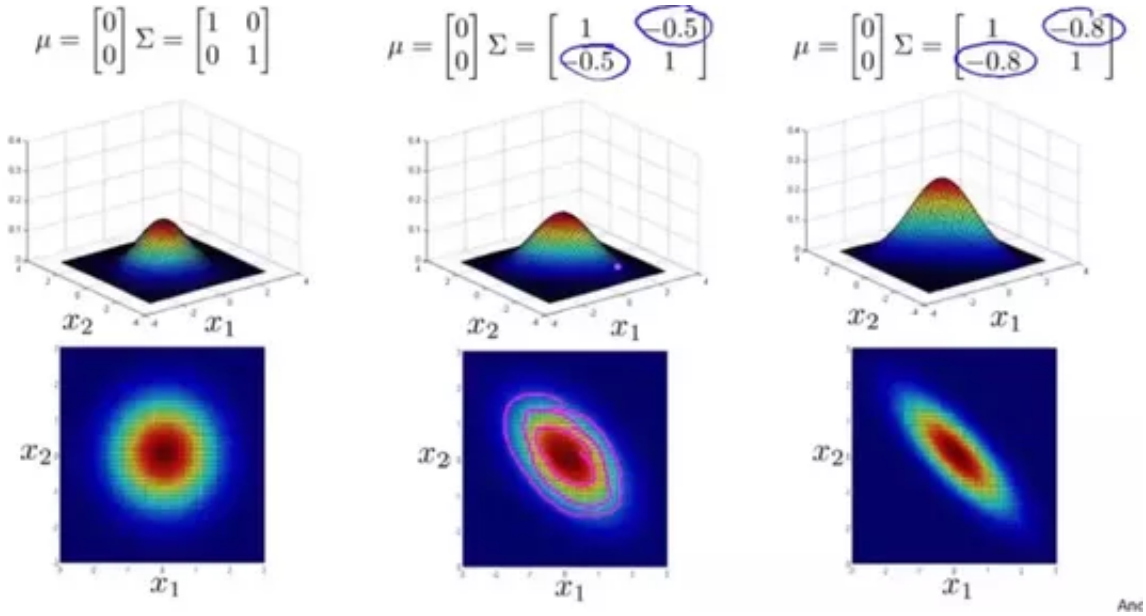
definitions

- $p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$

- μ is mean vector
- Σ is covariance matrix



understanding Σ



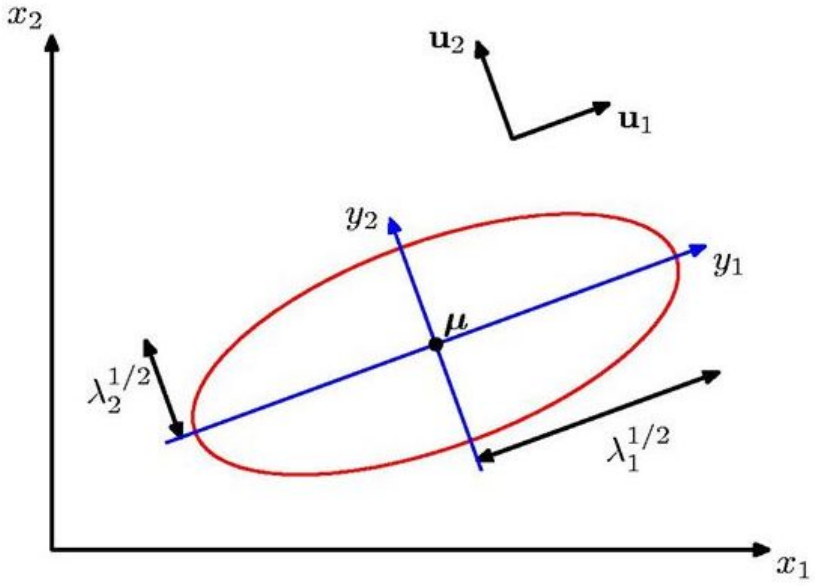
understanding Σ^{-1}

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$$

$$\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}$$

$$\Delta^2 = \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i}$$

$$y_i = \mathbf{u}_i^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu})$$



mle gaussian estimation

- $\hat{\mu}, \hat{\Sigma} = argmax\ P(x_1, \ldots, x_n | \mu, \Sigma)$
- $\hat{\mu} = \frac{1}{n} \sum x_i$
- $\hat{\Sigma} = \frac{1}{n} \sum (x_i - \hat{\mu})(x_i - \hat{\mu})^T$

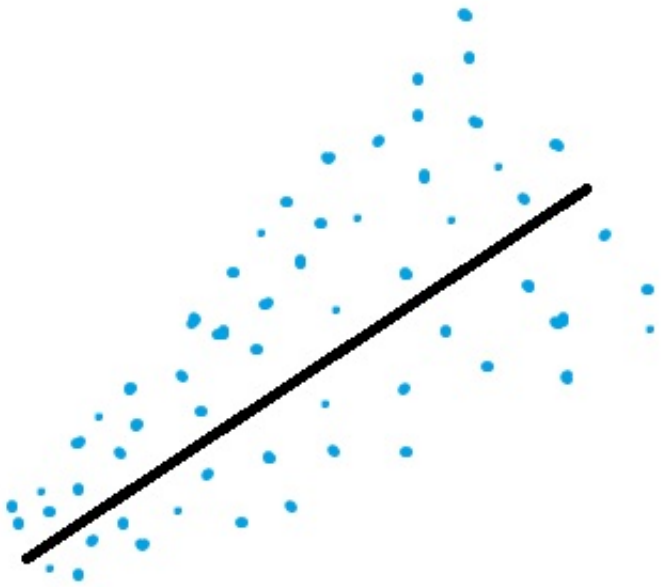
advanced linear least squares

weighted least squares

- weight certain points more ω_i
- $\hat{w}_{\text{wls}} = argmin \left(\sum \omega_i (y_i - x_i^T w)^2 \right)$
- $= (X^T \Omega X)^{-1} X^T \Omega y$

generalized least squares

- noise variables are not independent

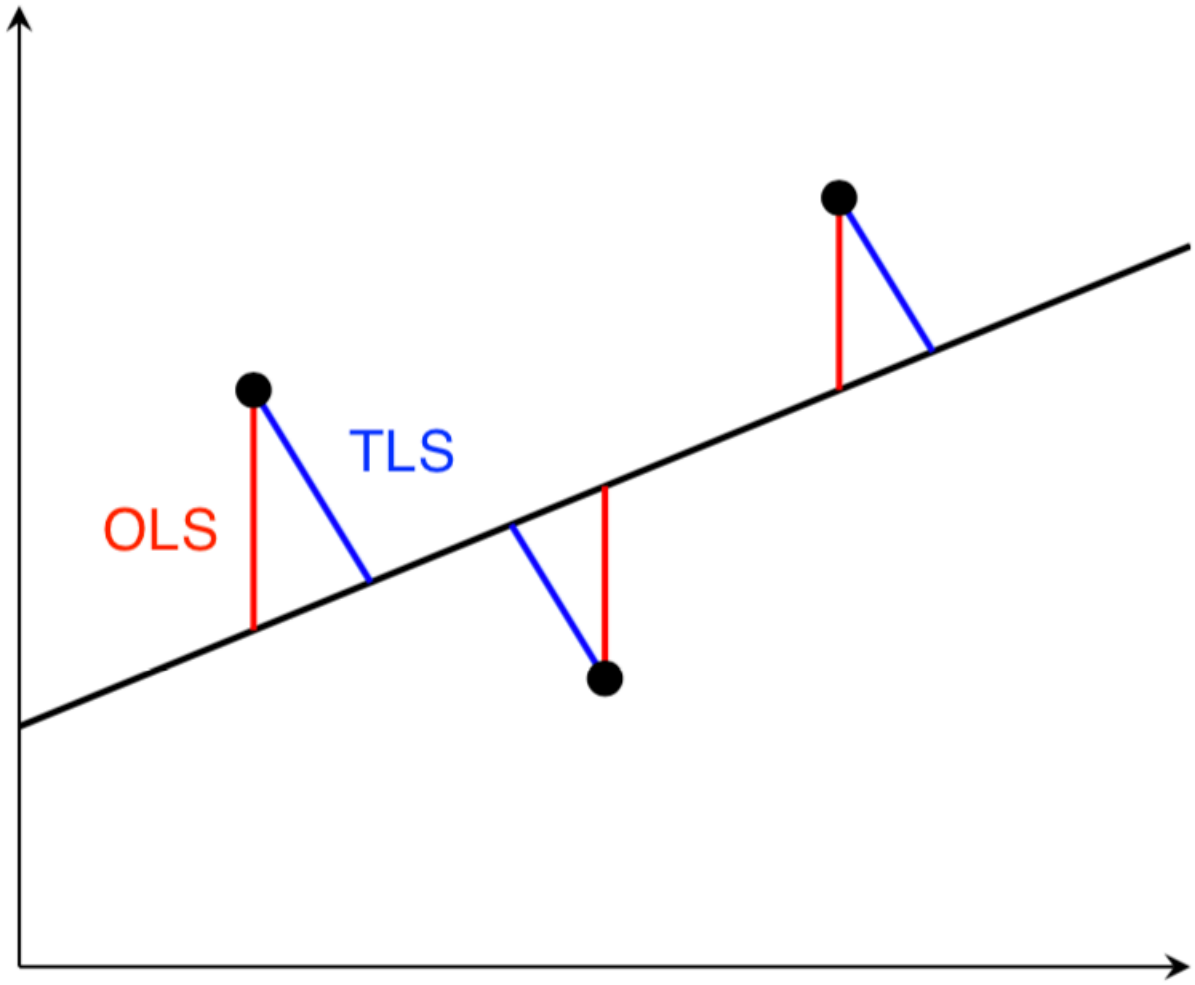


overview

W \ Z	$\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\mathbf{0}, \mathbf{\Sigma_Z})$
No prior	$\hat{\mathbf{w}}_{\text{OLS}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$	$\hat{\mathbf{w}}_{\text{GLS}} = (\mathbf{X}^\top\mathbf{\Sigma_Z}^{-1}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{\Sigma_Z}^{-1}\mathbf{y}$
$\mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$	$\hat{\mathbf{w}}_{\text{RIDGE}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$ Tikhonov	$(\mathbf{X}^\top\mathbf{\Sigma_Z}^{-1}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{\Sigma_Z}^{-1}\mathbf{y}$
$\mathcal{N}(\boldsymbol{\mu}_\mathbf{W}, \mathbf{\Sigma_W})$	$\boldsymbol{\mu}_\mathbf{W} + (\mathbf{X}^\top\mathbf{X} + \mathbf{\Sigma_W}^{-1})^{-1}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\mathbf{W})$	$\boldsymbol{\mu}_\mathbf{W} + (\mathbf{X}^\top\mathbf{\Sigma_Z}^{-1}\mathbf{X} + \mathbf{\Sigma_W}^{-1})^{-1}\mathbf{X}^\top\mathbf{\Sigma_Z}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\mu}_\mathbf{W})$

total LS intuition

- add i.i.d. gaussian noise in x and y - regularization



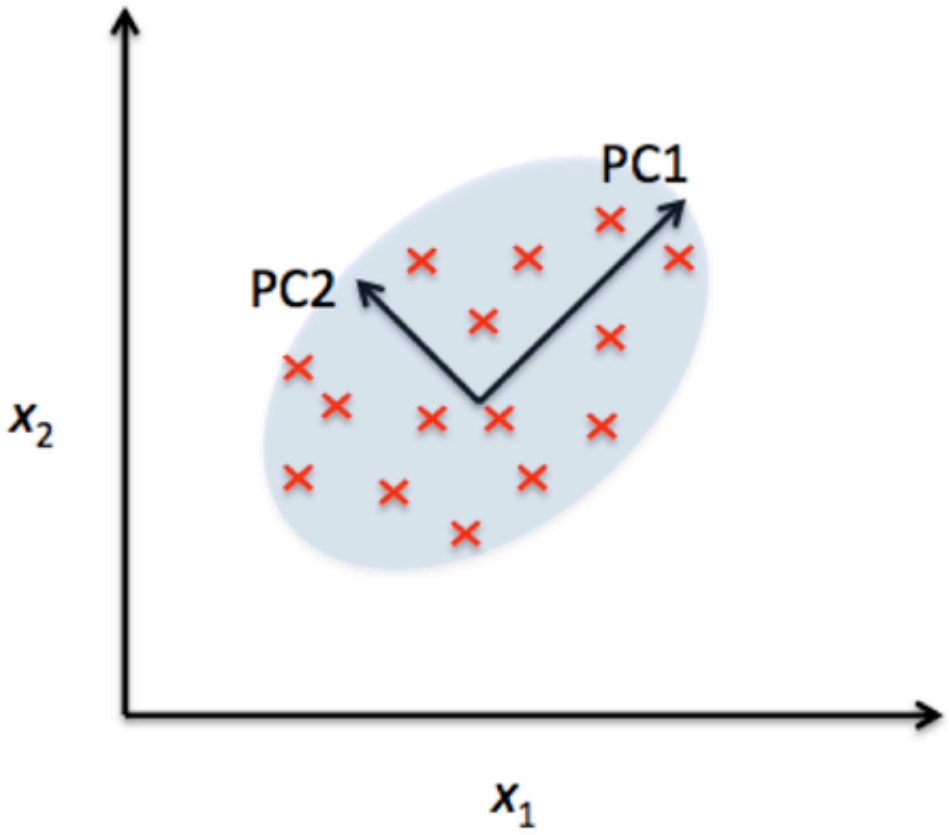
total LS solution

- $\hat{w}_{TLS} = (X^T X - \sigma^2 I)^{-1} X^T y$
  - here, σ is last singular value of $[X\ y]$

dimensionality reduction

pca intuition

- orthogonal dimensions that maximize variance of X



pca in python

```
X -= np.mean(X, axis = 0) #zero-center data (nxd)
cov = np.dot(X.T, X) / X.shape[0] #get cov. matrix (dxd)
U, D, V = np.linalg.svd(cov) #compute svd, (all dxd)
Xrot_reduced = np.dot(X, U[:, :2]) #project in 2d (nx2)
```

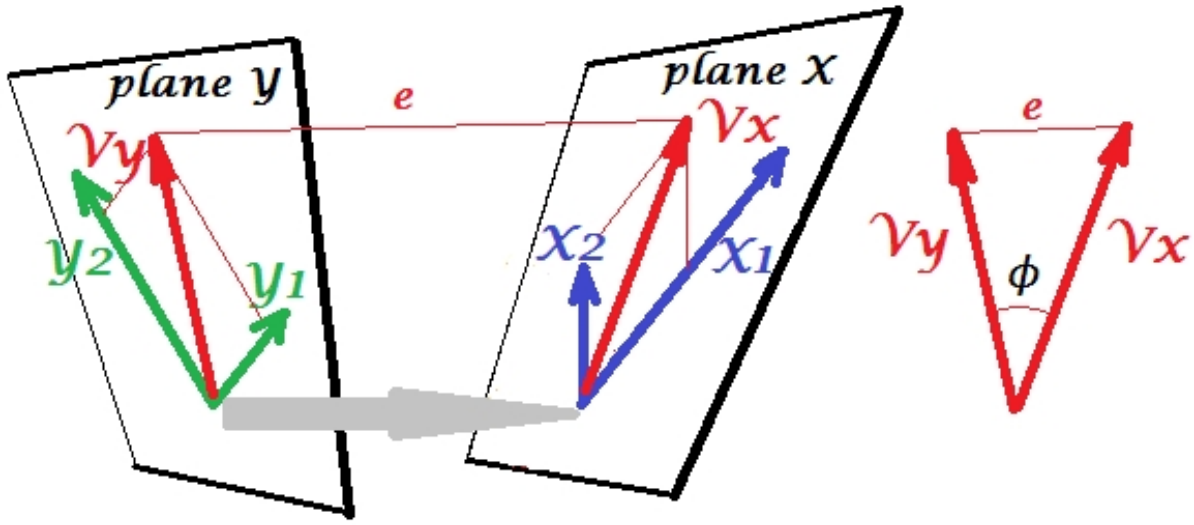
pca in practice

- eigenvalue represents prop. of explained variance:
  $\sum \lambda_i = tr(\Sigma) = \sum Var(X_i)$
- use svd
- adaptive PCA is faster (sequential)

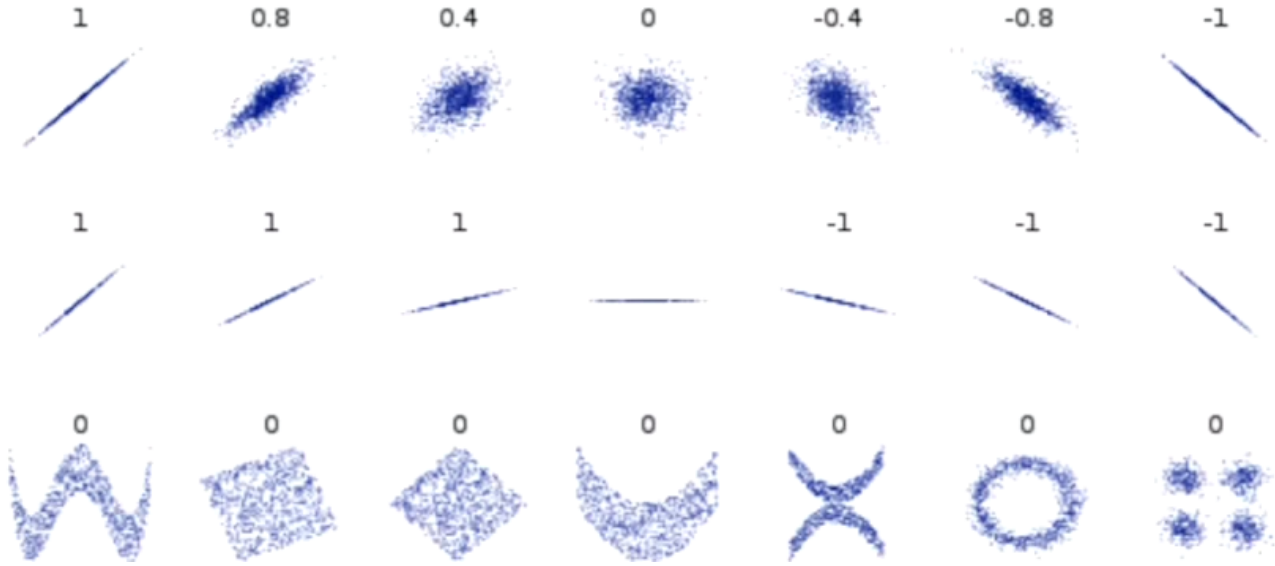
cca

- linearly independent dimensions that maximize correlation between X, Y

- invariant to scalings / affine transformations of X, Y

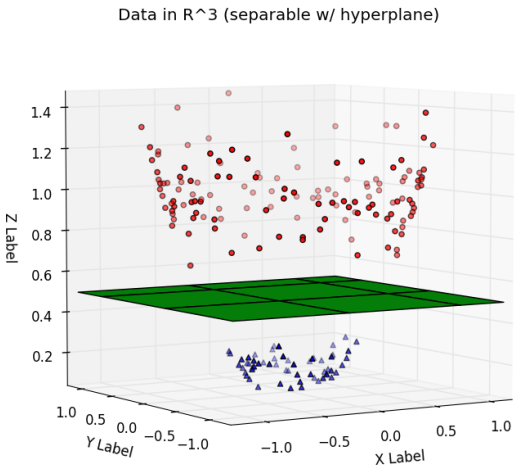


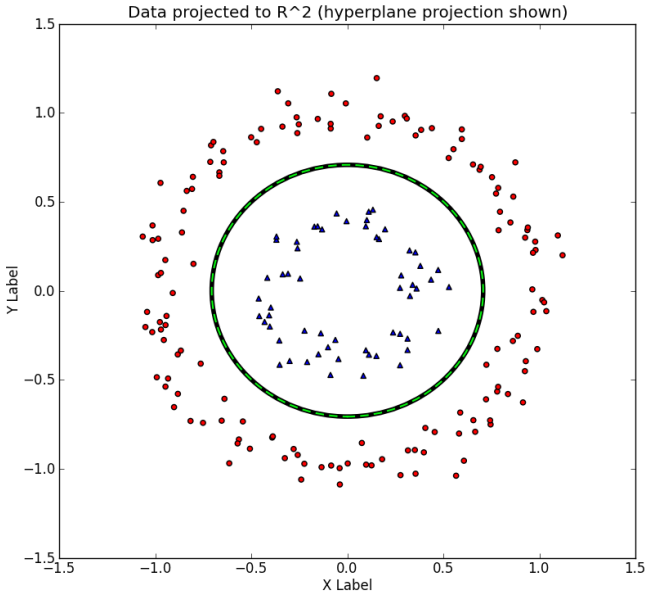
correlations



kernels

why are kernels useful?





ex. ridge regression

- reformulate the problem to be computationally efficient + nonlinear
  - matrix inversion is ~$O(n^3)$
- $\hat{w} = (\underbrace{X^T X}_{dxd} + \lambda I)^{-1} X^T y$ ~ faster when $d << n$
- $\hat{w} = X^T (\underbrace{X X^T}_{nxn} + \lambda I)^{-1} y$ ~ faster when $n << d$

kernels

$$\mathbf{\Phi}\mathbf{\Phi}^\top = \begin{pmatrix} — & \boldsymbol{\phi}_1^\top & — \\ — & \boldsymbol{\phi}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{\phi}_n^\top & — \end{pmatrix} \begin{pmatrix} | & | & & | \\ \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} \boldsymbol{\phi}_1^\top \boldsymbol{\phi}_1 & \boldsymbol{\phi}_1^\top \boldsymbol{\phi}_2 & \cdots \\ \boldsymbol{\phi}_2^\top \boldsymbol{\phi}_1 & \ddots & \\ \vdots & & \boldsymbol{\phi}_n^\top \boldsymbol{\phi}_n \end{pmatrix}$$

- $\phi_i^T \phi_j = \phi(x_i)^T \phi(x_j)$

kernel trick ex.

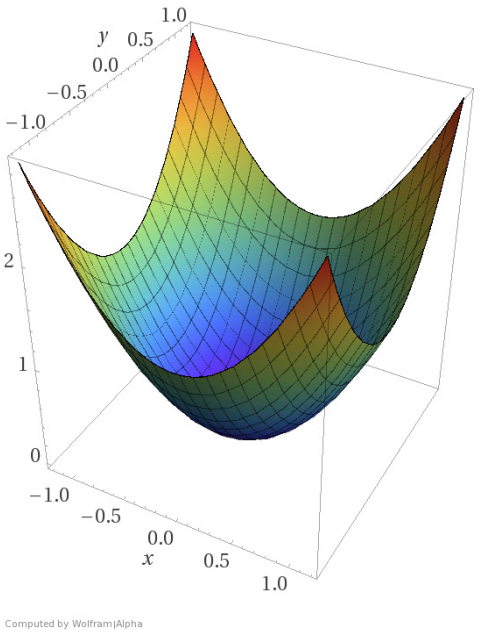
- ex. with degree 2, 2 variables:
  $\phi(\mathbf{x}) = \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1 x_2 & \sqrt{2}x_1 & \sqrt{2}x_2 & 1 \end{bmatrix}^T$
- $\mathbf{x_i} = [x_1, x_2]$

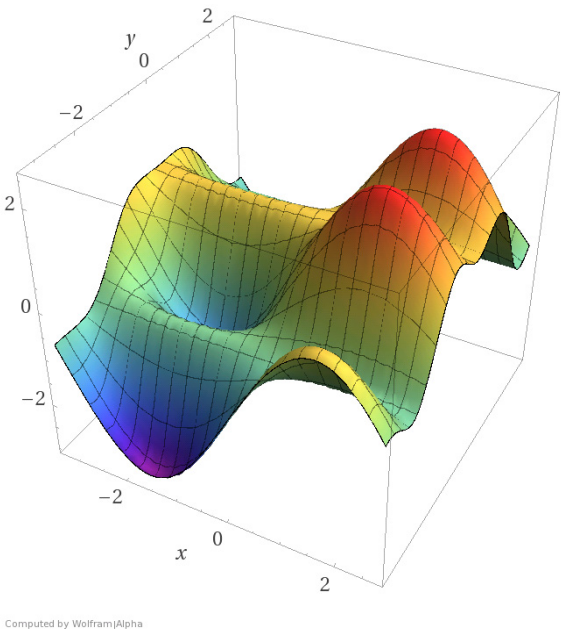
- $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^2$
    - left is O(augmented feature space)
    - right is O(original feature space + log(polynomial degree))
- also works for other kernels...

optimization problems

overview

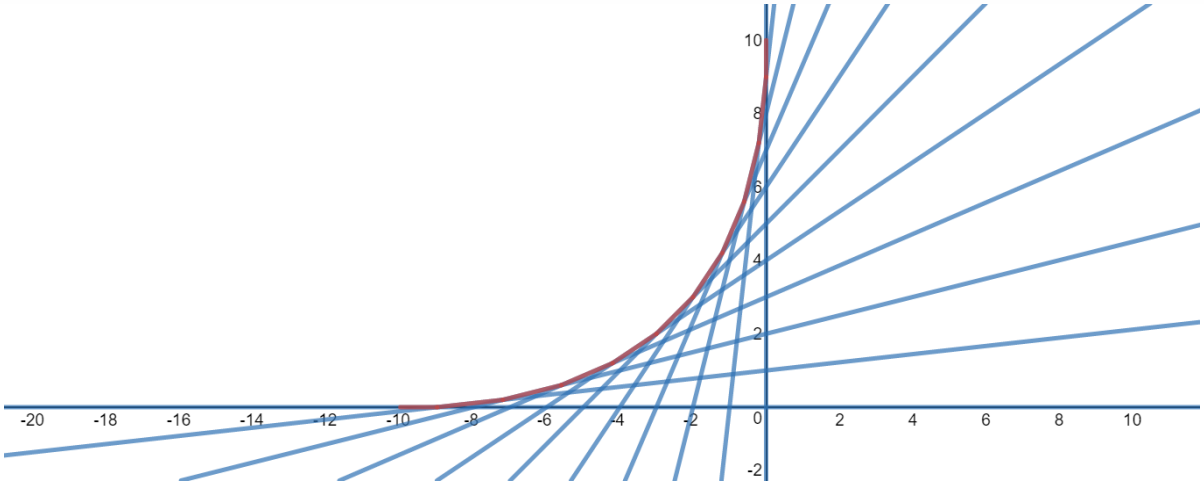
- minimizing things
- ex. $\underset{\theta}{argmin} \sum \left(y_i - f(x_i;\theta)\right)^2$





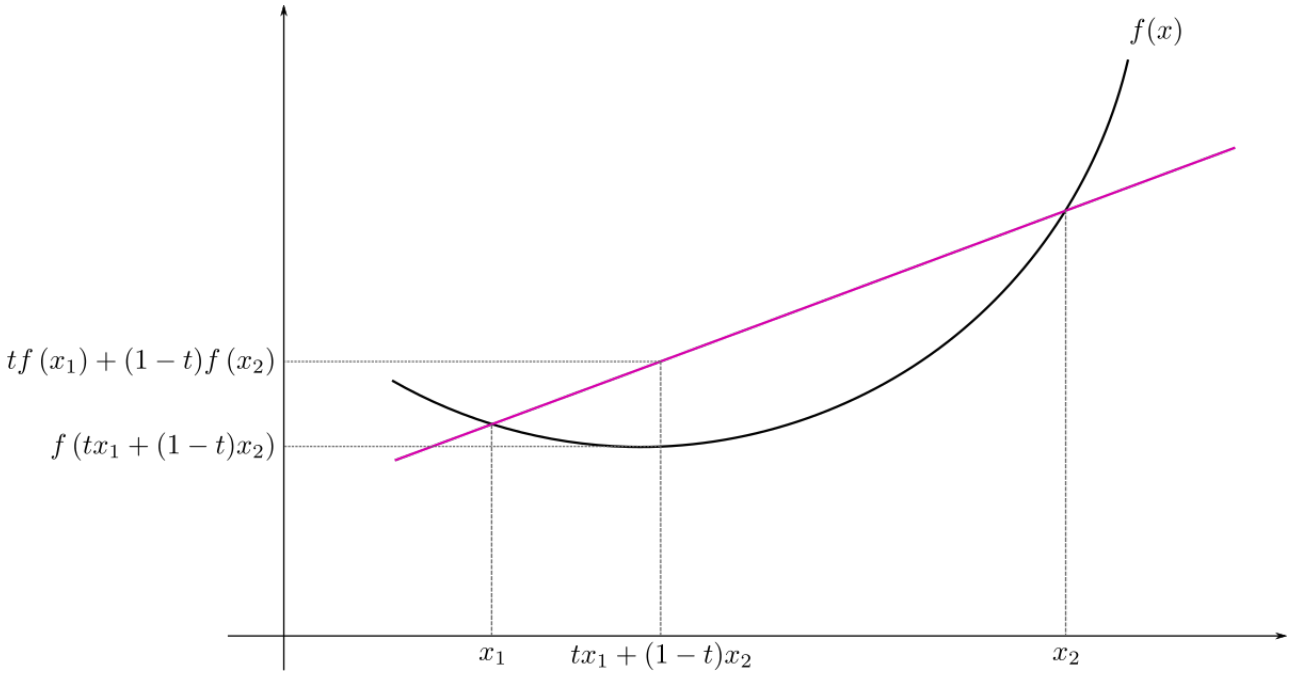
convexity

- Hessian $\nabla^2 f(x) \succeq 0 \, \forall x$
- $f(x_2) \geq f(x_1) + \nabla f(x_1)(x_2 - x_1)$



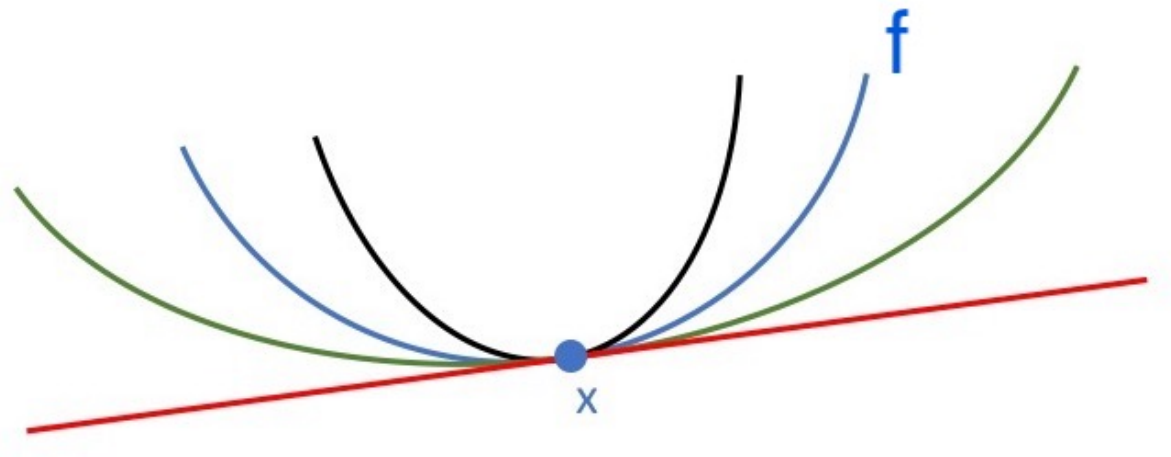
convexity continued

- $tf(x_1) + (1-t)f(x_2) \geq f(tx_1 + (1-t)x_2)$



strong convexity + smoothness

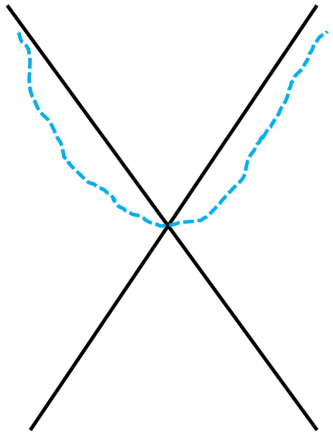
$$0 \preceq \underbrace{mI}_{\text{strong convexity}} \preceq \nabla^2 f(x) \preceq \underbrace{MI}_{\text{smoothness}}$$



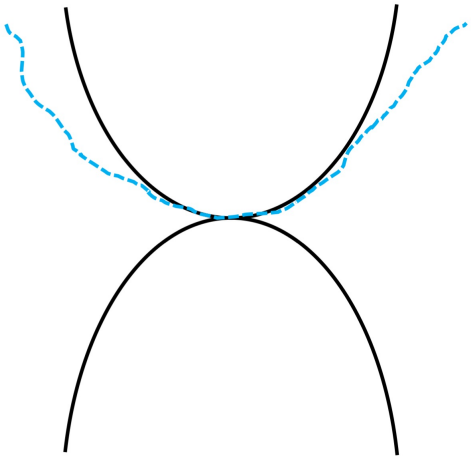
smoothness

- M-smooth = Lipschitz continuous gradient:
  $||\nabla f(x_2) - \nabla f(x_1)|| \leq M||x_2 - x_1|| \quad \forall x_1, x_2$

Lipschitz continuous function f

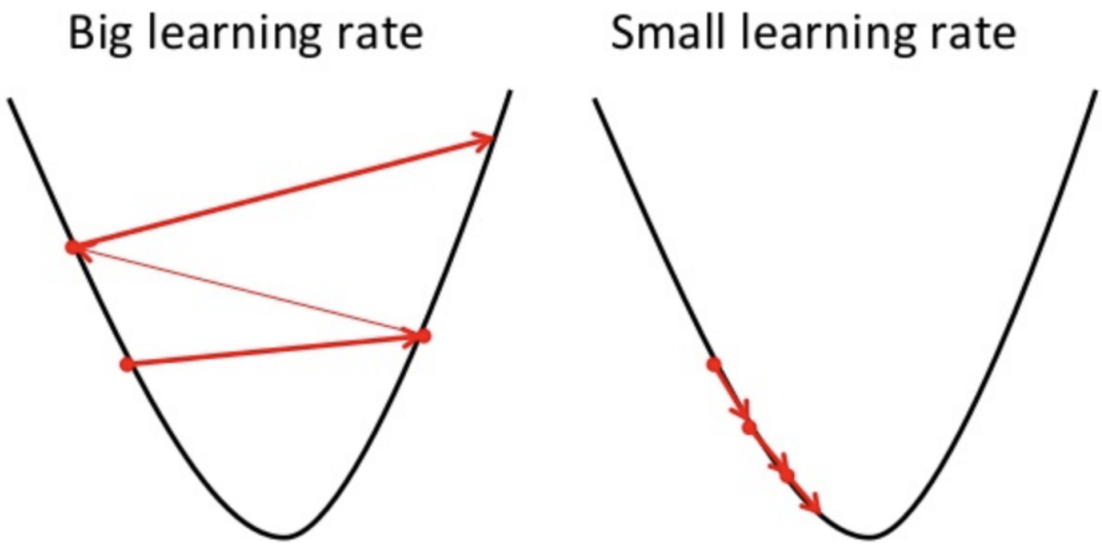


Lipschitz continuous gradient f



optimization algorithms

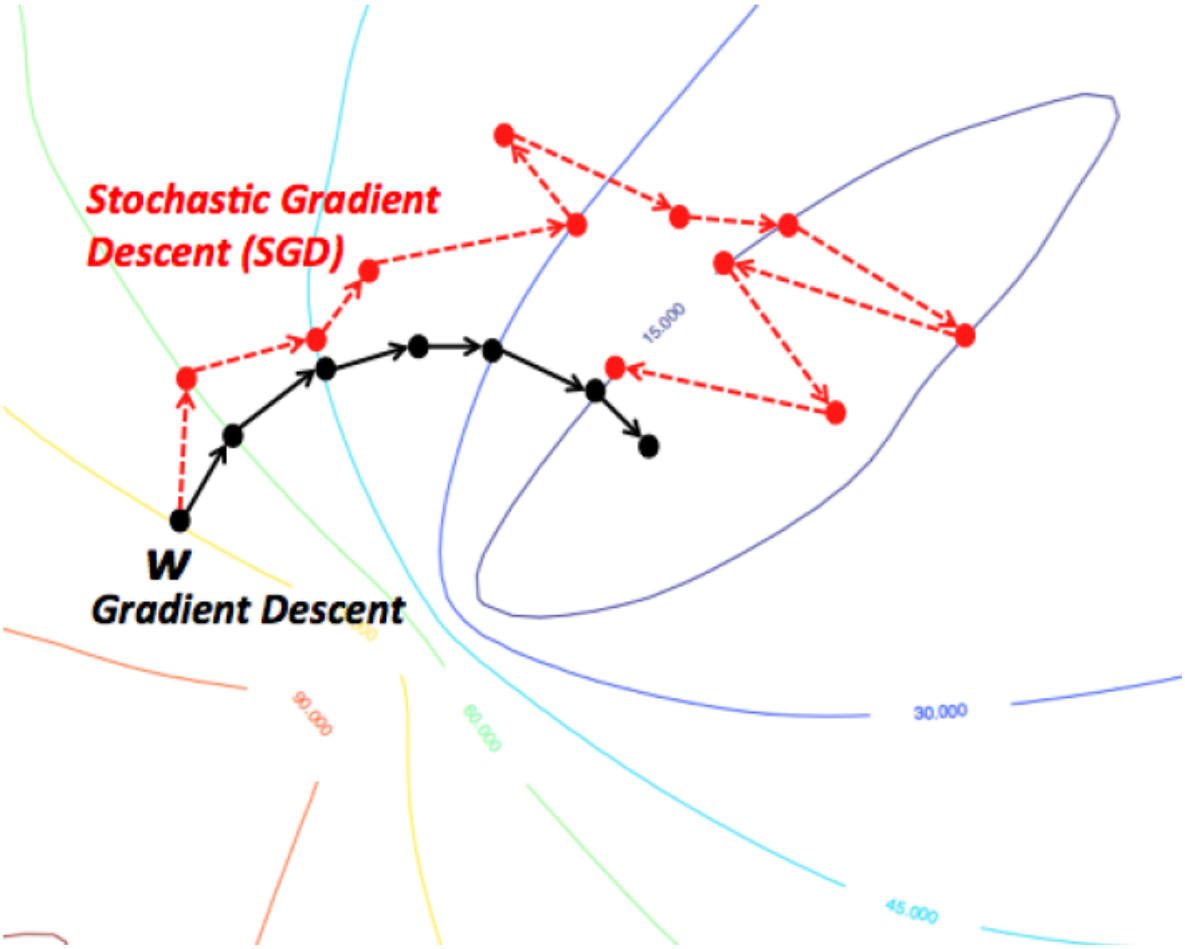
gradient descent



when do we stop?

- validation error stops changing
- changes become small enough

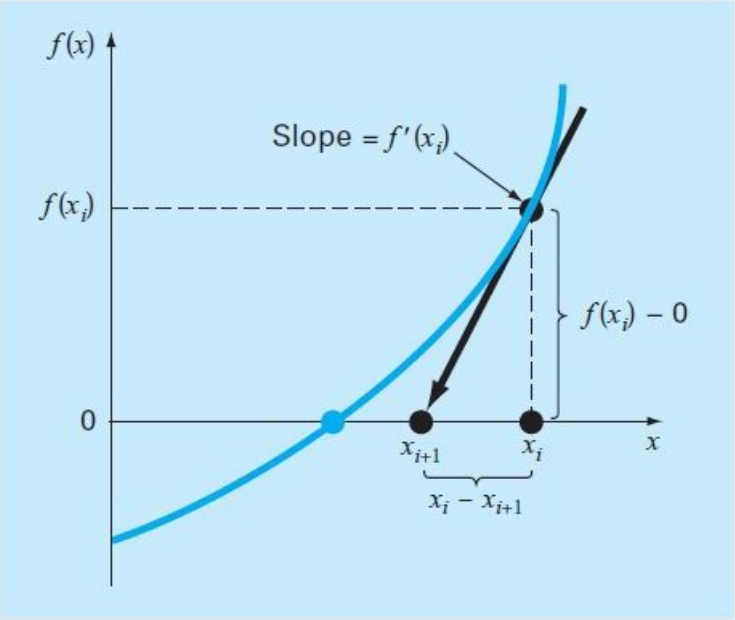
stochastic gradient descent



momentum demo

- $\theta^{(t+1)} = \theta^{(t)} - \alpha_t \nabla f(\theta^{(t)}) + \underbrace{\beta_t (f(\theta^{(t)}) - f(\theta^{(t-1)}))}_{\text{momentum}}$

newton-raphson



$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

- apply to find roots of **f'(x)**: $\theta^{(t+1)} = \theta^{(t)} - \nabla^2 f(\theta^{(t)})^{-1} \nabla f(\theta^{(t)})$

gauss-newton

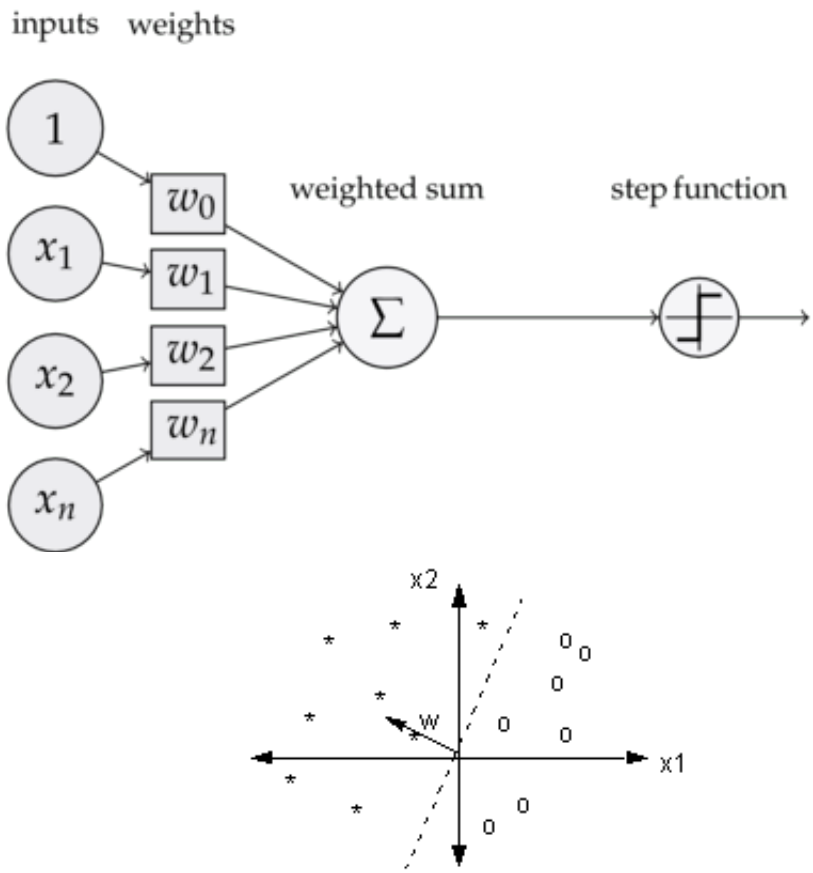
- modify newton's method assuming we are minimizing nonlinear least squares
- $\theta^{(t+1)} = \theta^{(t)} - \nabla^2 f(\theta^{(t)})^{-1} \nabla f(\theta^{(t)})$
- $\theta^{(t+1)} = \theta^{(t)} + (J^T J)^{-1} J^T \Delta y$ J is the Jacobian

neural nets

so much hype

- it predicts: vision, audio, text, ~rl
- it's easy: feature engineering
- it's magical: huge, but doesn't overfit

perceptron

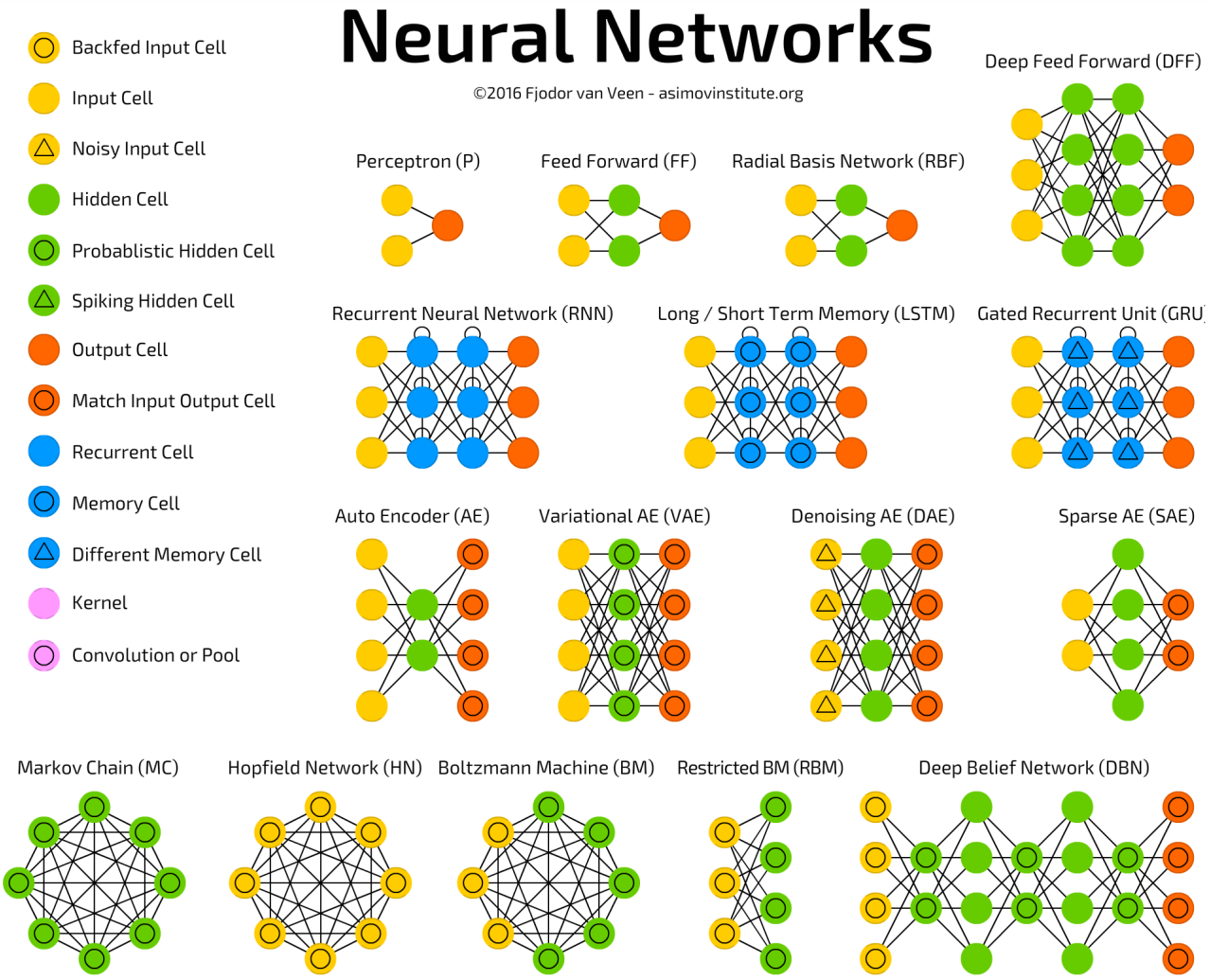


training a perceptron

- loss function: $L(x, y; w) = (\hat{y} - y)^2$
- goal: $\frac{\partial L}{\partial w_i}$ for all weights
- calculate efficiently with backprop

[backprop demo]

going deeper



nn demo

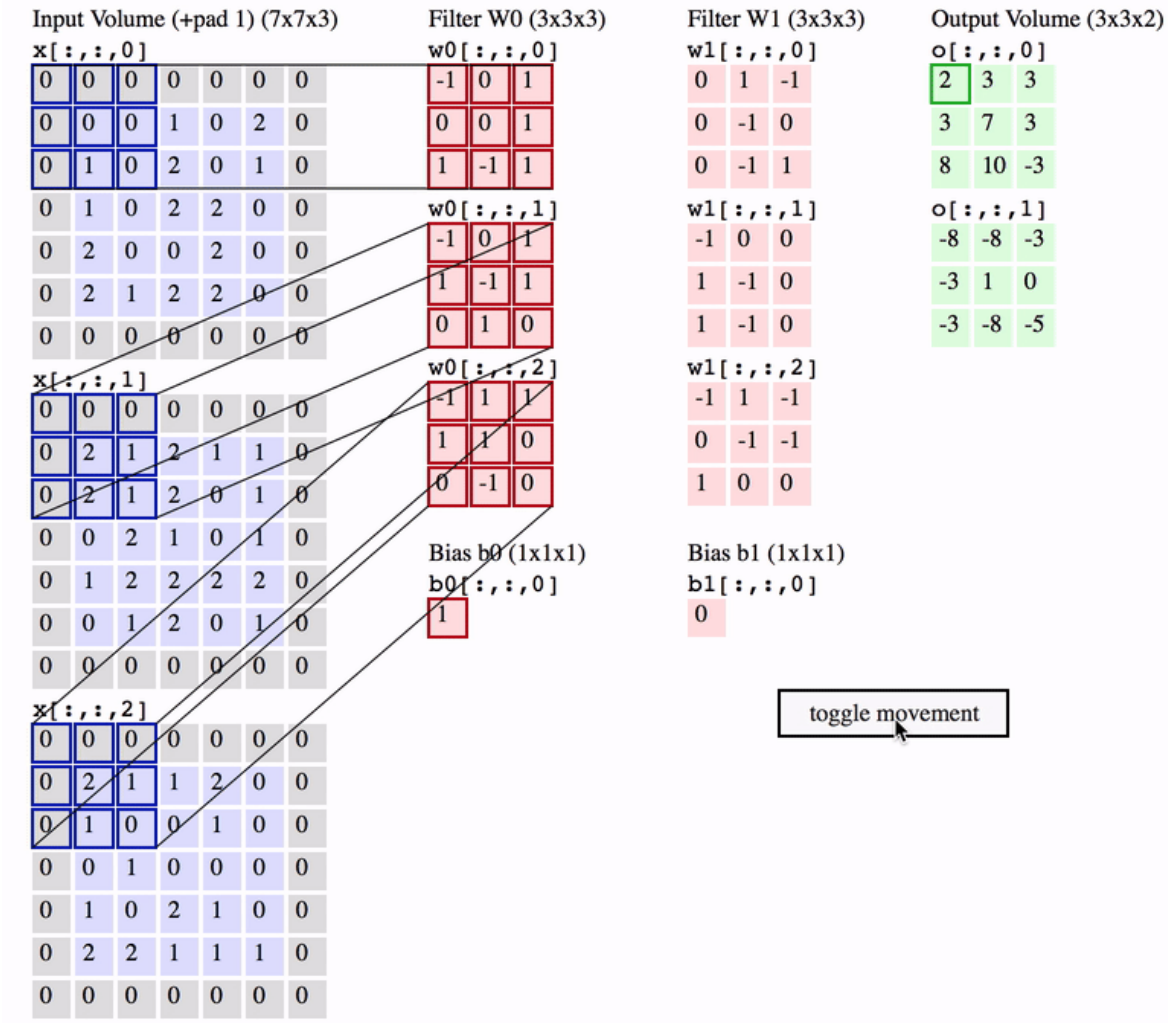
coding DNNs in numpy

```
from numpy import exp, array, random, dot
training_set_inputs = array([[0, 0, 1], [1, 1, 1], [1, 0, 1], [0, 1,
1]])
training_set_outputs = array([[0, 1, 1, 0]]).T
random.seed(1)
synaptic_weights = 2 * random.random((3, 1)) - 1
for iteration in xrange(10000):
    output = 1 / (1 + exp(-(dot(training_set_inputs,
synaptic_weights))))
    synaptic_weights += dot(training_set_inputs.T,
(training_set_outputs - output) * output * (1 - output))
print 1 / (1 + exp(-(dot(array([1, 0, 0]), synaptic_weights))))
```

coding DNNs in advanced numpy

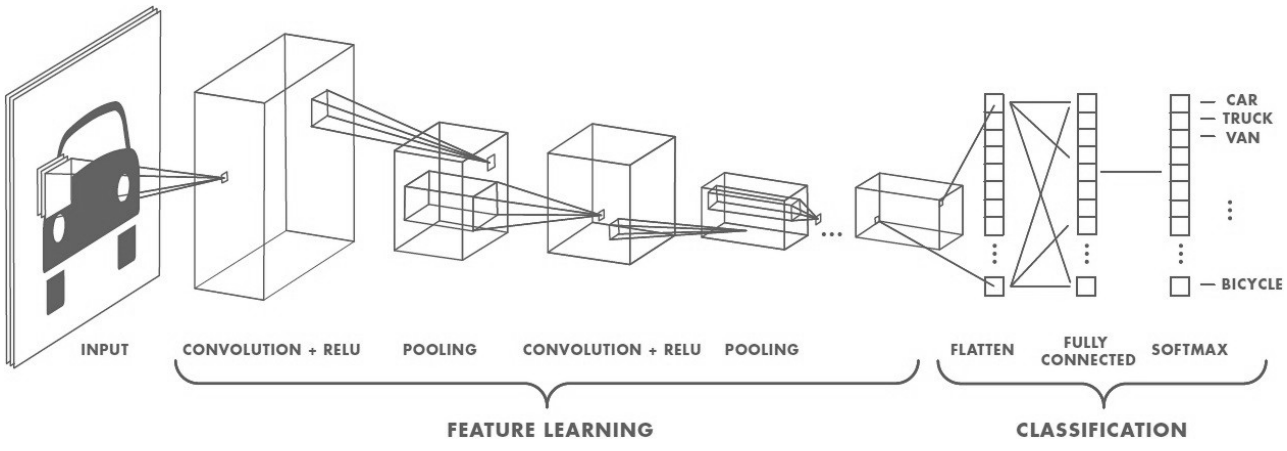
```
import tensorflow as tf
import torch
```

cnns

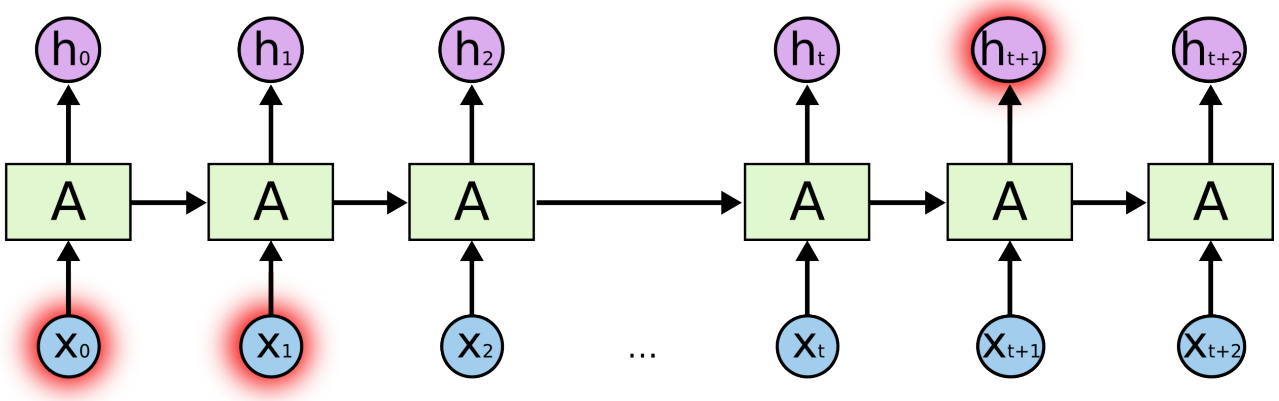

Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 1 0 2 0 1 0
0 1 0 2 2 0 0
0 2 0 0 2 0 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 2 1 1 0
0 2 1 2 0 1 0
0 0 2 1 0 1 0
0 1 2 2 2 2 0
0 0 1 2 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 1 2 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 1 0 2 1 0 0
0 2 2 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 1
0 0 1
1 -1 1
w0[:,:,1]
-1 0 1
1 -1 1
0 1 0
w0[:,:,2]
-1 1 1
1 1 0
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 1 -1
0 -1 0
0 -1 1
w1[:,:,1]
-1 0 0
1 -1 0
1 -1 0
w1[:,:,2]
-1 1 -1
0 -1 -1
1 0 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 3 3
3 7 3
8 10 -3
o[:,:,1]
-8 -8 -3
-3 1 0
-3 -8 -5
toggle movement

---

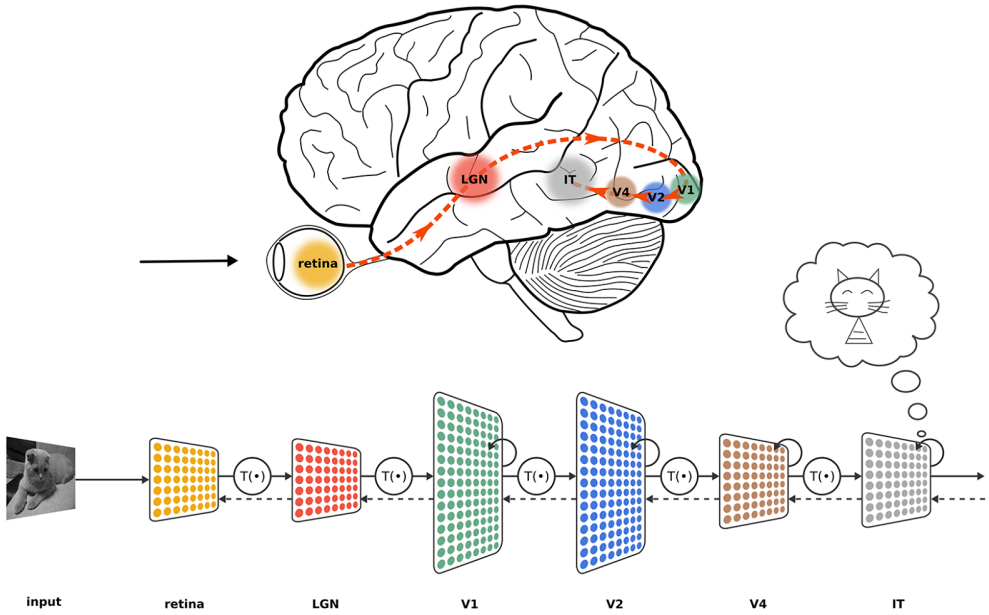
cnns 2


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

rnns

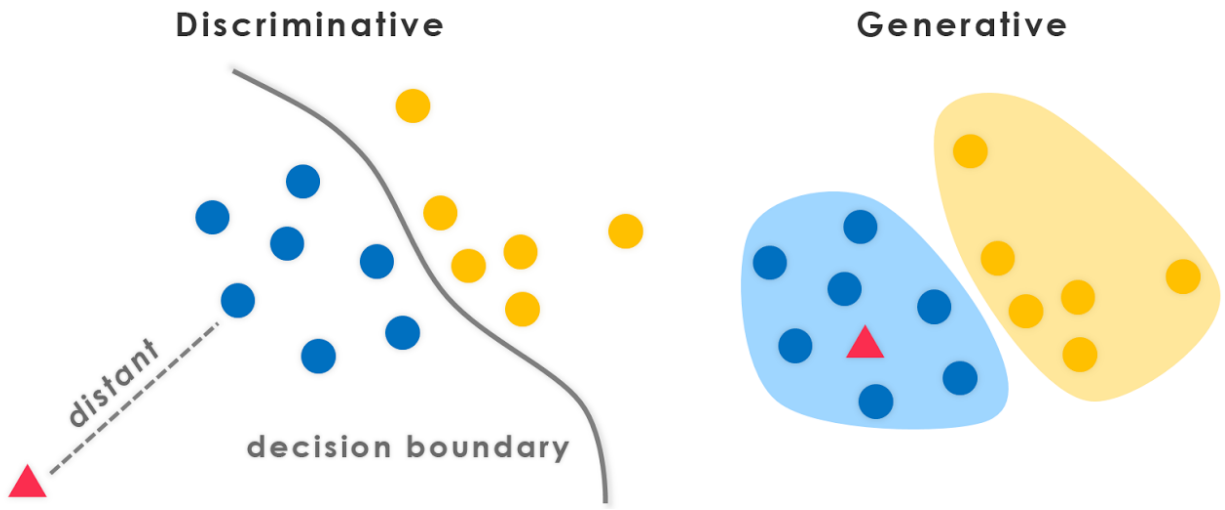


connection to the brain?



discriminative vs. generative

definitions



- discriminative: $p(y|x)$
- generative: $p(y,x) = p(y|x)p(x)$

sorting models

generative

bayes classifier

hmms

gaussian discriminant analysis

bayes classifier

- risk:

discriminative

linear regression

svms

nearest neighbor

decision trees / random forests

$$E_{(x,y)}[L(f(x),y)] = \sum_x p(x) \sum_y L(f(x),y)p(y|x)$$

- bayes classifier:
  $f^*(x) = \underset{y}{argmin} \sum_y L(y,y')p(y'|x)$
  - given x, pick y that minimizes loss
- with 0-1 error: $f^*(x) = \underset{y}{argmax}\, p(y|x)$

bayes classifier example

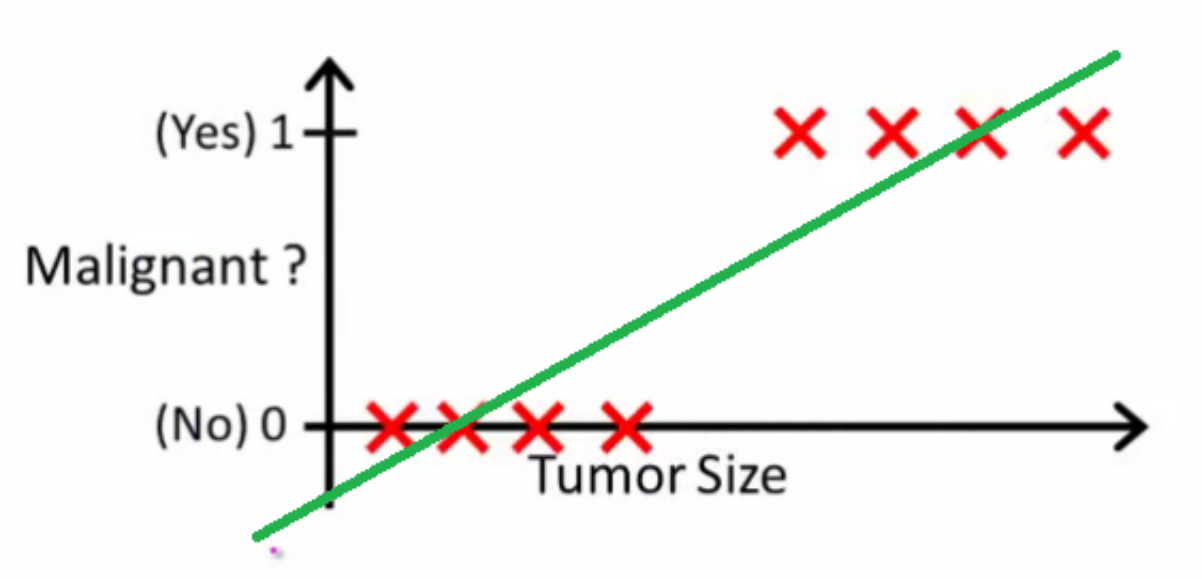
- with 0-1 error: $f^*(x) = \underset{y}{argmax}\, p(y|x) = \underset{y}{argmax}\, p(x|y) \cdot p(y)$
    - let y be sentiment (positive or negative)
    - let x be words

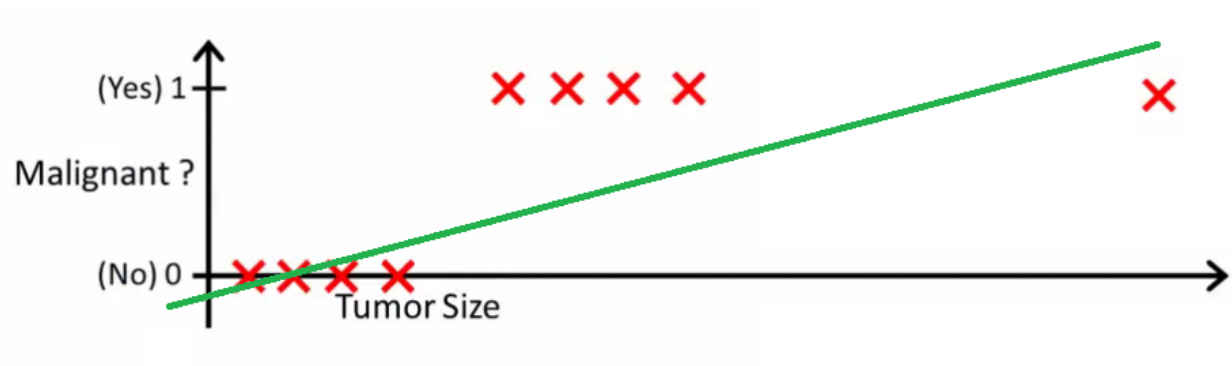
logistic regression

definitions

- is this really regression?
- $\sigma(z) = \frac{1}{1+e^{-z}}$
- $p(Y=1) \begin{cases} 1 & \sigma(w^T x) \\ 0 & \text{otherwise} \end{cases}$

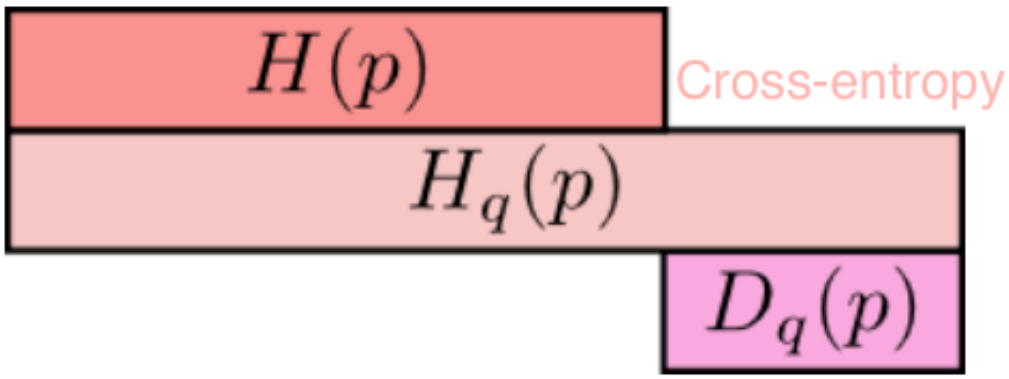
comparison with OLS





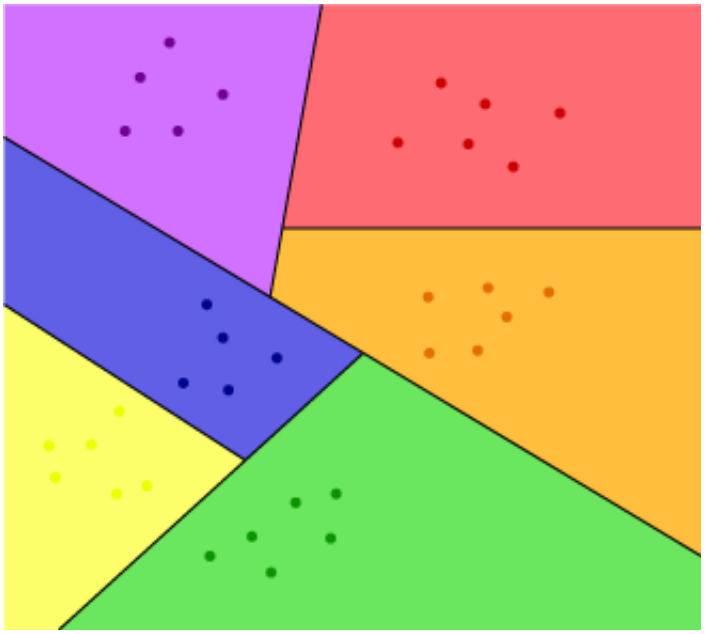
loss functions

- log-loss = cross-entropy: $-\sum_x p(x)\ log\ q(x)$
  - $p(x)$ true y
  - $q(x)$ predicted probability of y
- corresponds to MLE for Bernoulli



multiclass

- one-hot encoding: $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$
- softmax function: $\sigma(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp z_j}$
- loss function still cross-entropy



training

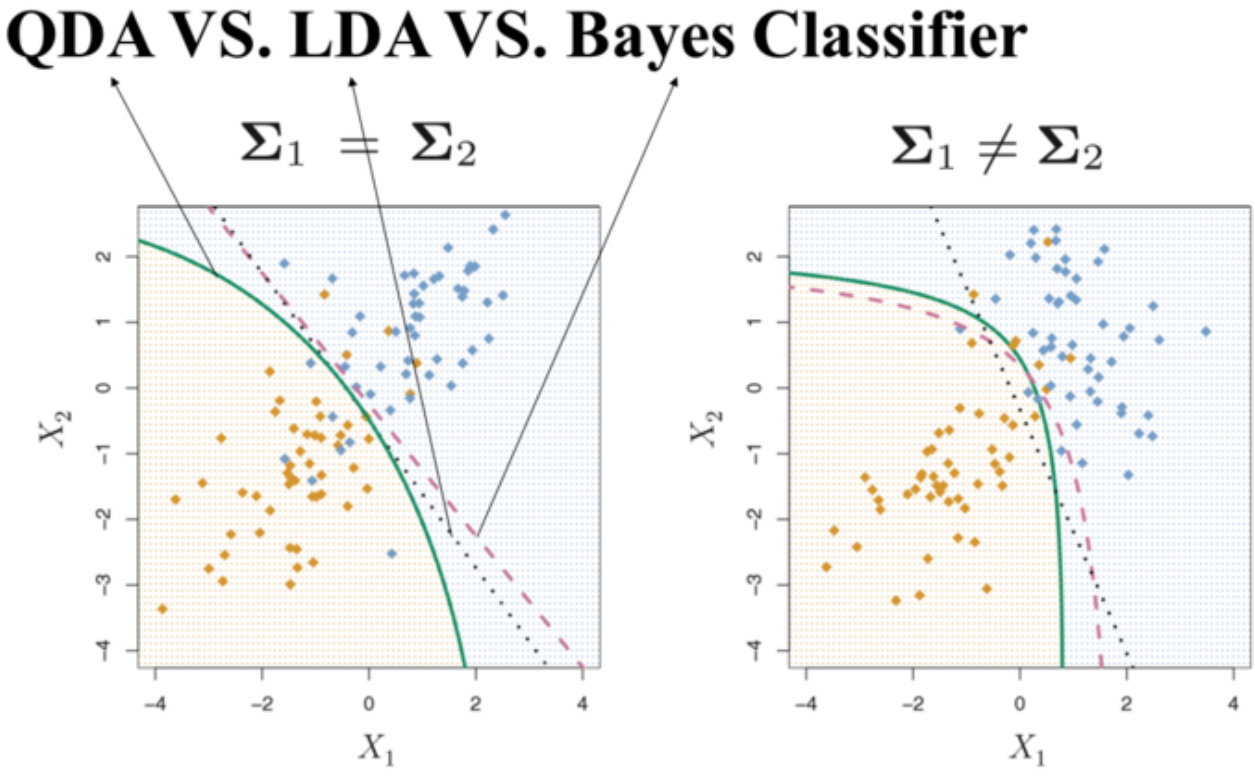
- no closed form, but convex loss $\implies$ convex optimization!

gaussian discriminant analysis

assumptions

- $\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|\mathbf{x})$
  - $P(\mathbf{X}|Y = y) \sim \mathcal{N}(\mu_y, \Sigma_y)$: there are |Y| of these
  - $p(y) = \frac{n_y}{n}$: 1 of these

generative model



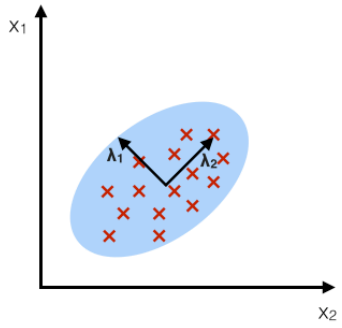
lda vs. log. regression

- differences
  - generative (models p(x|y))
  - treats each class independently

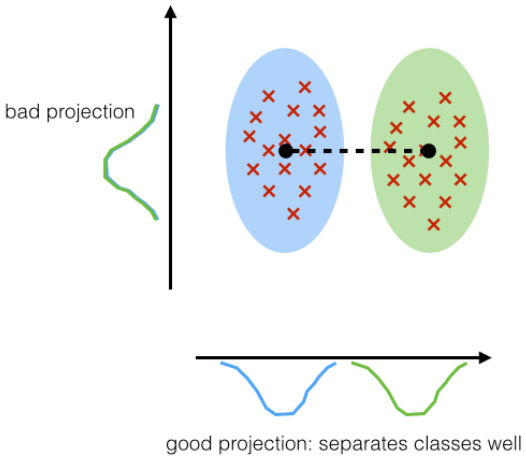
- same
    - same form for posterior (sigmoid)

dimensionality reduction

PCA:
component axes that maximize the variance

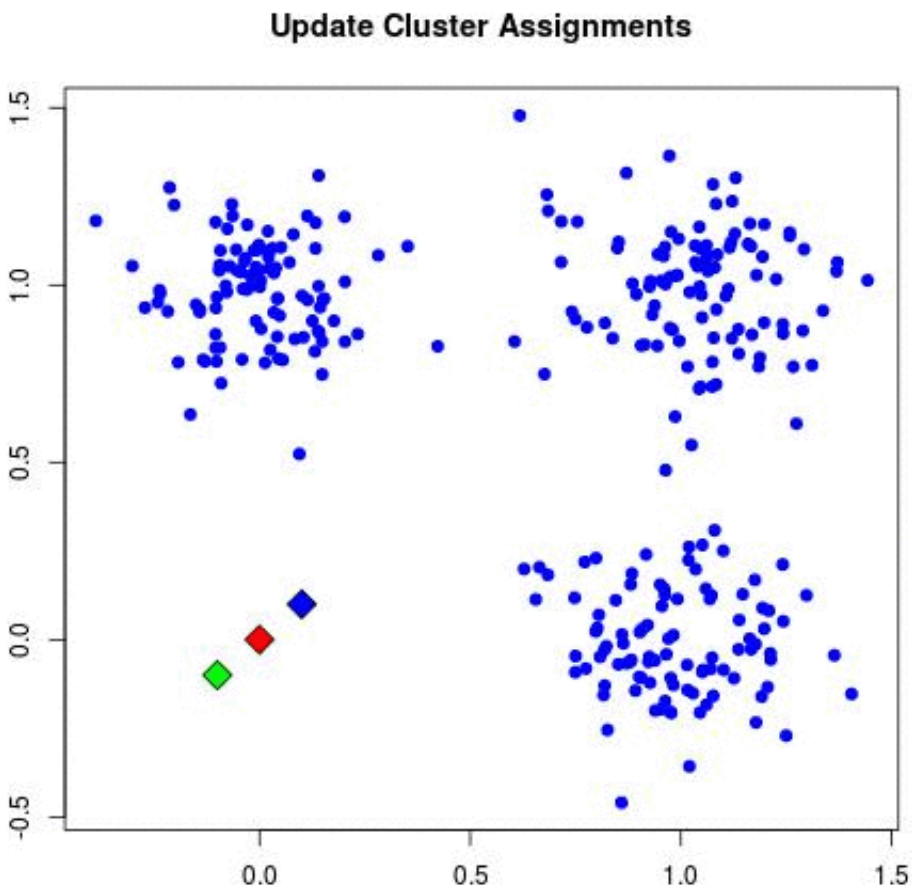


LDA:
maximizing the component axes for class-separation

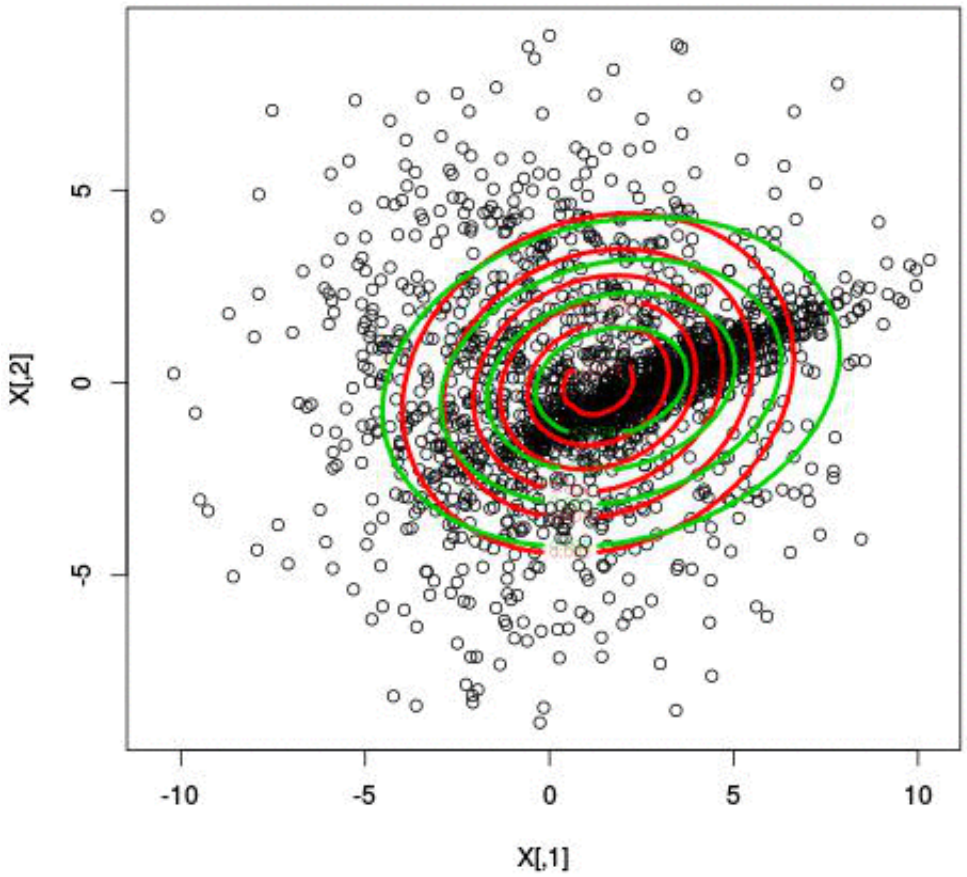


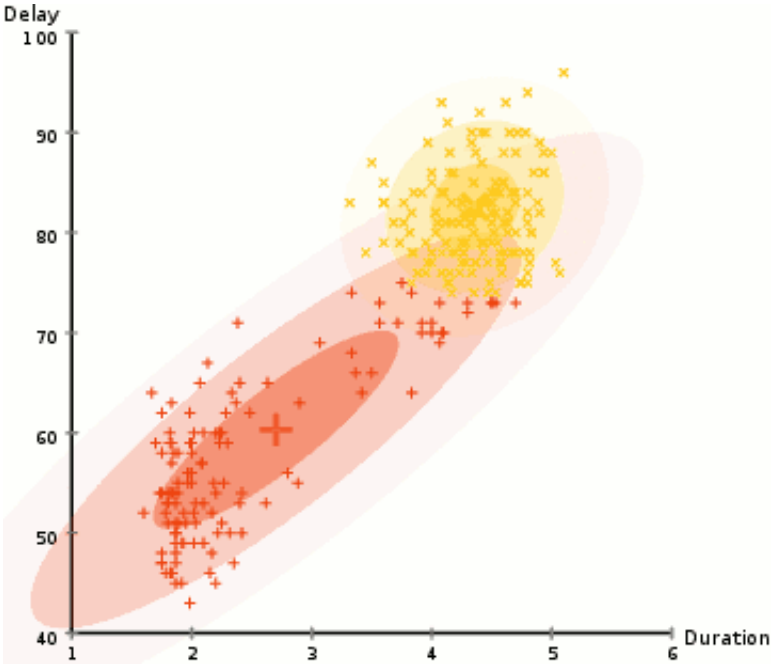
em

k-means intuition

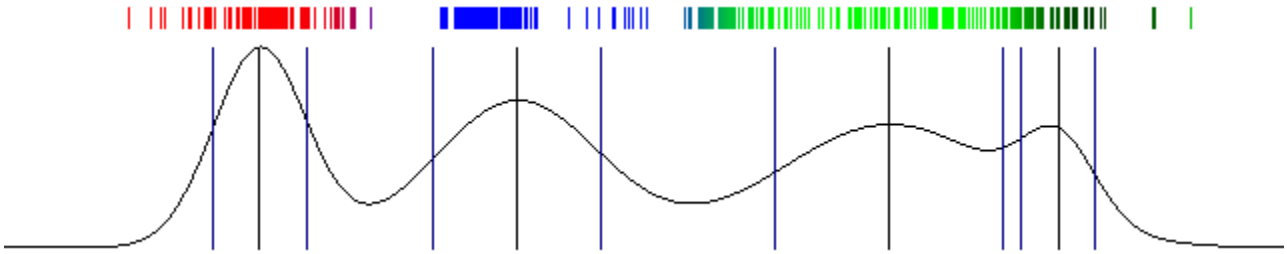


mixture of gaussians intuition





mixture of gaussians (1d)



EM

- want to maximize *complete log-likelihood* $l(\theta; x, z) = log\, p(x, z||\theta)$ but don't know z
    - *expectation step* - values of unobserved latent variables are filled in
    - *maximization step* - parameters are adjusted based on filled-in variables

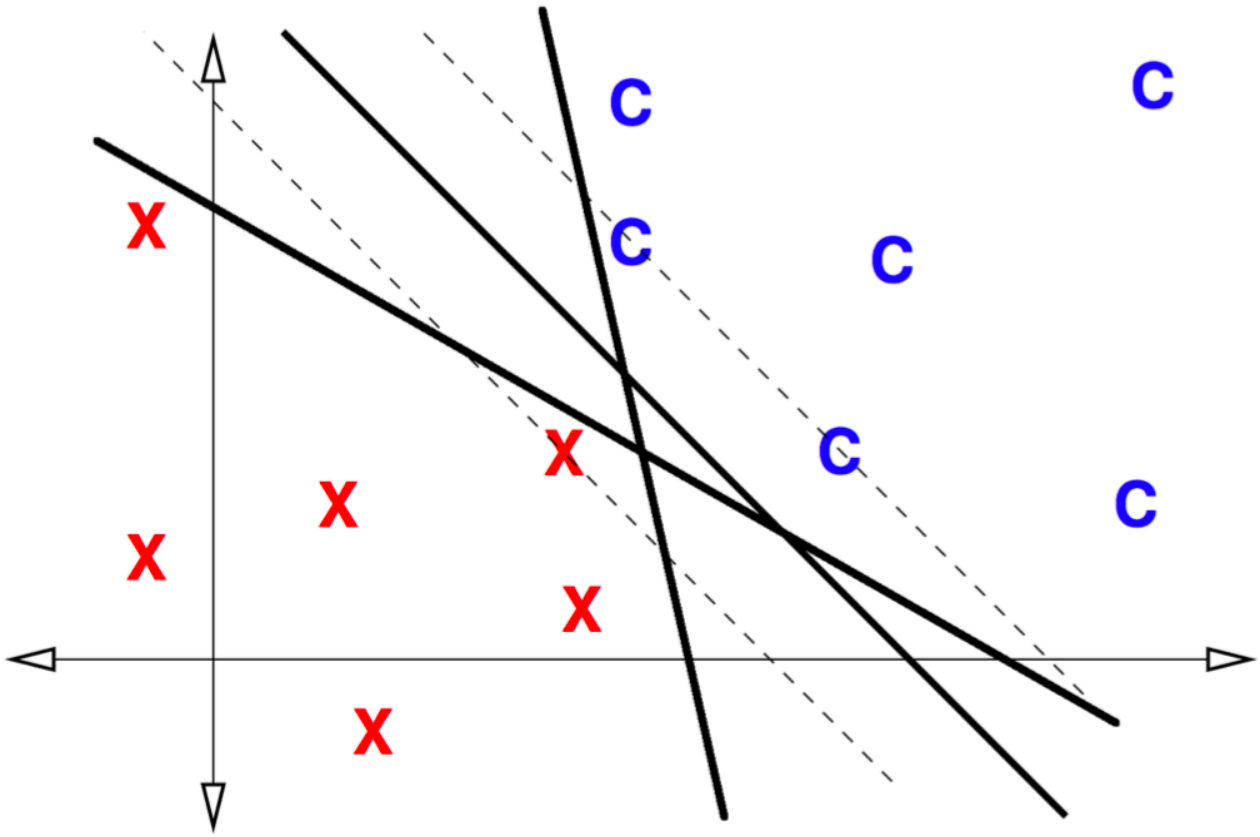
simplifying the math

- x: observed vars, z: latent vars, q: assignments to z
- E: $q^{(t+1)}(z|x) = \underset{q}{argmin}\ D(q||\theta^{(t)})$
    - lower bound on complete log-likelihood (pf: Jensen's inequality)
- M: $\theta^{(t+1)} = \underset{\theta}{argmin}\ D(q^{(t+1)}||\theta)$

svms

note 20 is great

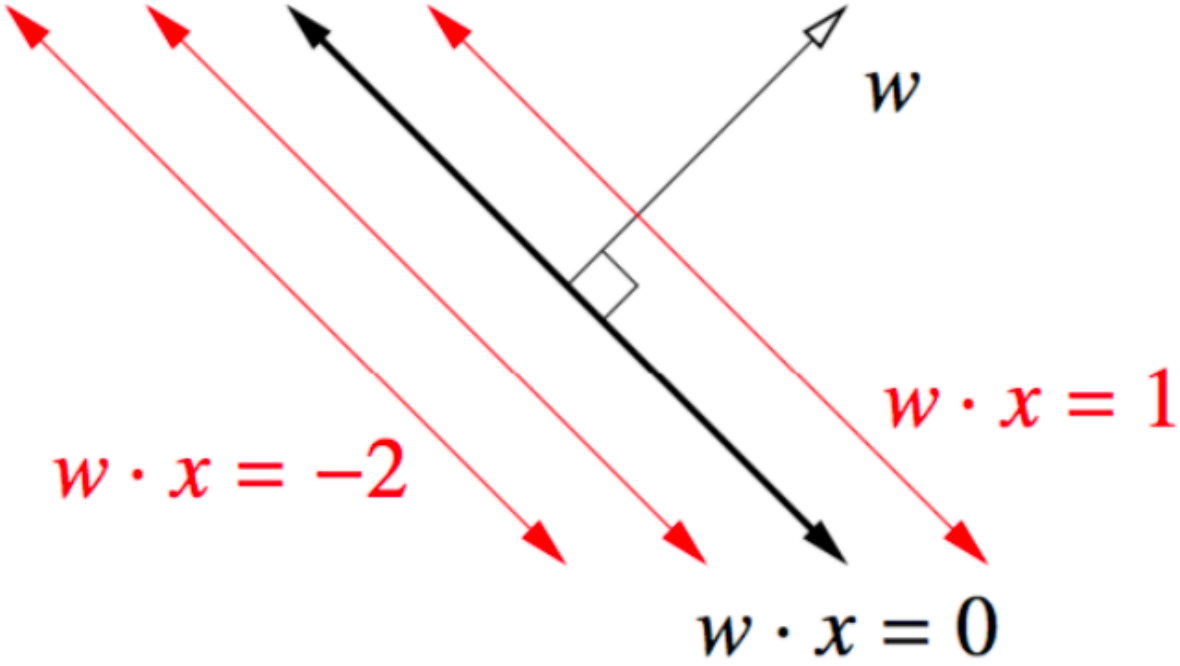
perceptron problems



- doesn't find best solution
- unstable when data not linearly separable

what's w?

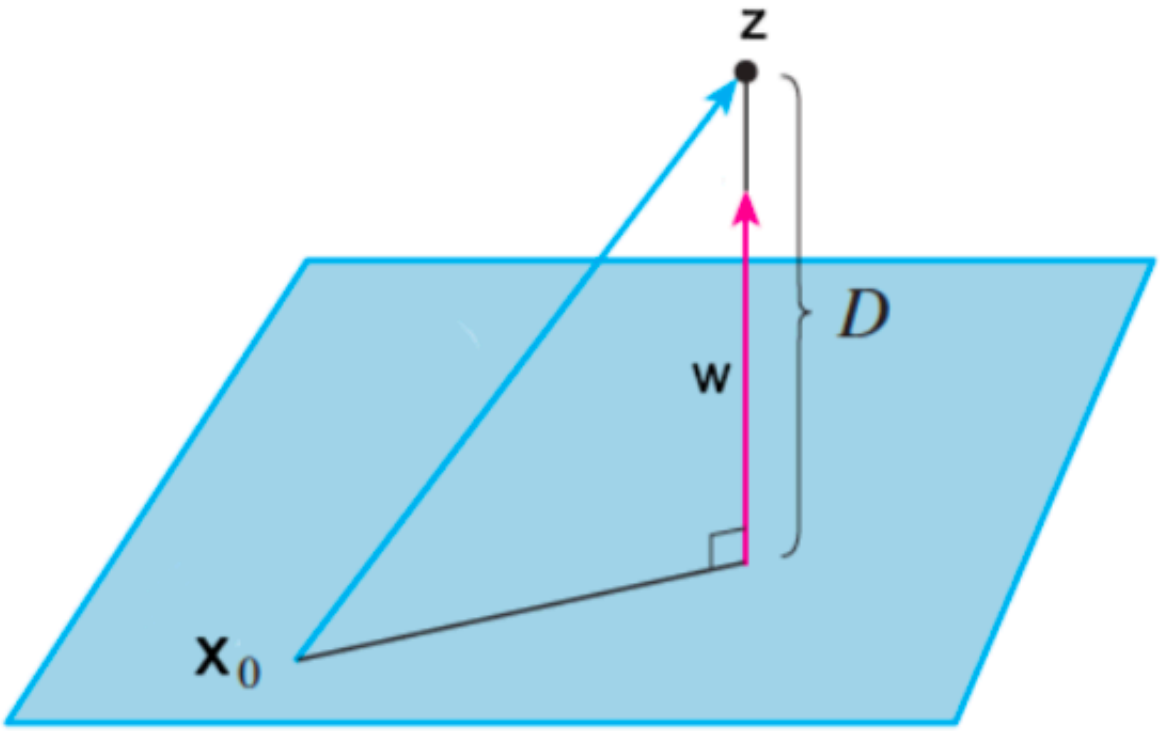
```
$\hat{y} =\begin{cases} 1 &\text{if } w^Tx +b \geq 0 \\ -1
&\text{otherwise}\end{cases}$
```



how far are points?

decision boundary: $\{x : w^T x - b = 0\}$

$D = \frac{|w^T(z - x_0)|}{||w||_2} = \frac{|w^T z - b|}{||w||_2}$



hard margin intuition

```
$\begin{align} \underset{m, w, b}{max} \quad &m \\ s.t. \quad &y_i \frac{(w^Tx_i-b)}{||w||_2} \geq m \: \forall i\\ &m \geq 0\end{align}$
```

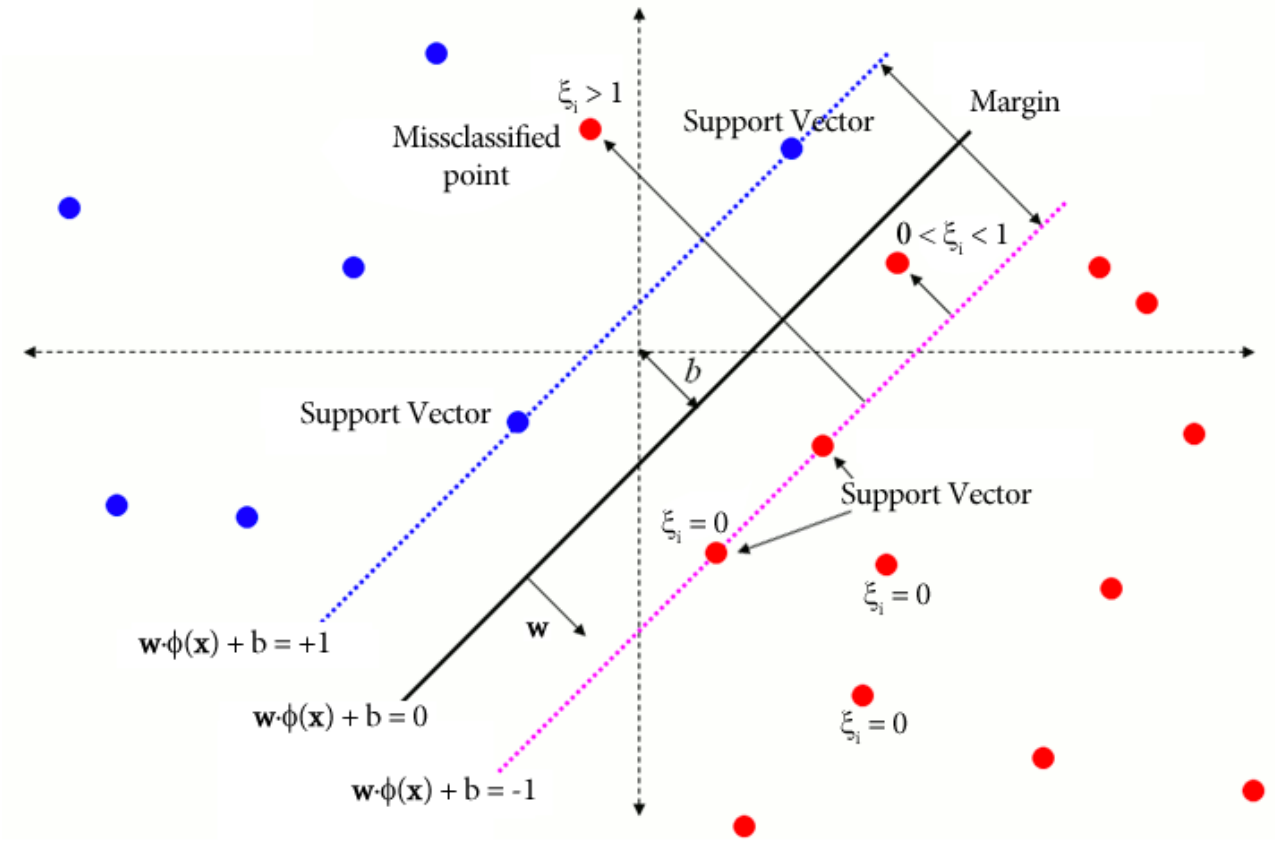
hard margin formulation

- let $m = 1/||w||_2 \implies$ unique soln

```
$\underset{w, b}{min}\quad \frac{1}{2} ||w||_2^2 \\s.t. \quad y_i (w^Tx_i - b) \geq 1\: \forall i$
```

soft margin

```
$\begin{align}\underset{w, b, \color{cadetblue}\xi}{min}\quad &\frac{1}{2} ||w||_2^2 \color{cadetblue}{+C\sum_i \xi_i}\\s.t. \quad &y_i (w^Tx_i - b) \geq 1 \color{cadetblue}{-\xi_i}\: \forall i\\ &\color{cadetblue}{\xi_i \geq 0 \: \forall i}\end{align}$
```

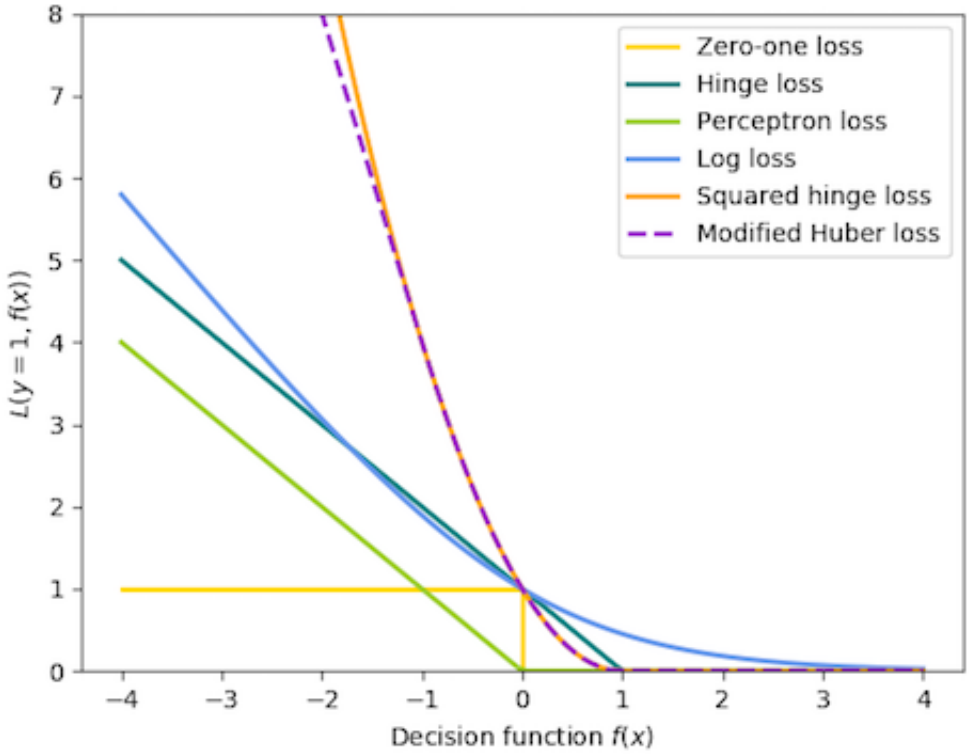


binary classification

- can rewrite by absorbing ξ constraints

$$\min_{w,b} \quad C\sum_i max(1 - y_i(w^T x_i - b), 0) + \frac{1}{2}||w||^2$$

summarizing models



- svm: hinge loss + regularization
- log. regression: log loss
- perceptron: perceptron loss

duality

problem

- consider `$min \: f_0 (x) \\ s.t. \: f_i(x) \leq 0 \\ h_i(x) = 0$`

dual setup

- *lagrangian* $L(x, \lambda, \nu) = f_0(x) + \sum \lambda_i f_i(x) + \sum \nu_i h_i(x)$
- *dual function* $g(\lambda, \nu) = \underset{x \in D}{inf} \, L(x, \lambda, \nu)$ ~ g always concave
    - $\lambda \succeq 0 \implies g(\lambda, \nu) \leq p^*$
- (λ, ν) *dual feasible* if
    1. $\lambda \succeq 0$

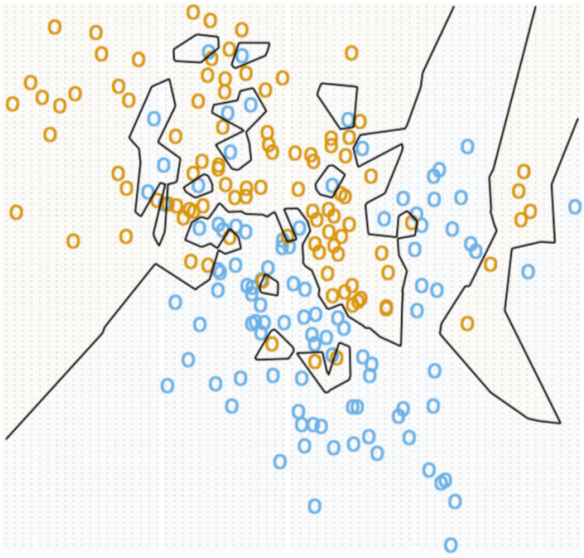
2. $(\lambda, \nu) \in dom\ g$
- $p^* = -\infty \implies$, dual infeasible, $d^* = \infty \implies$, primal infeasible

dual problem

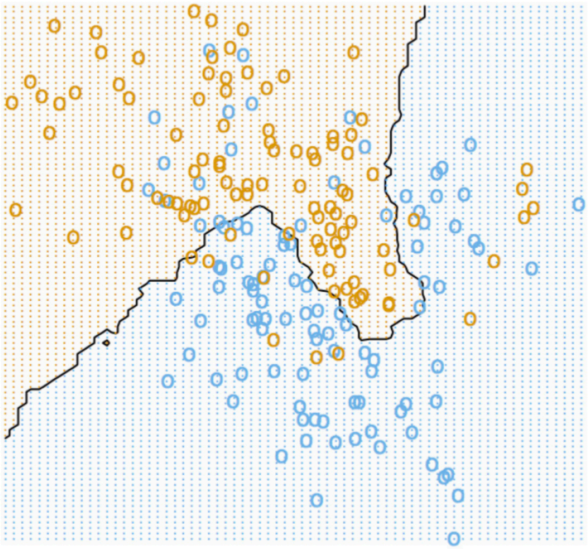
- dual related to to conjugate func
  - ex. min f(x) s.t. $x = 0 \implies g(\nu) = -f^*(-\nu)$
- *lagrange dual problem*: `$max \: g(\lambda, \nu)\\s.t. \: \lambda \succeq 0$`
- *weak duality*: $d^* \leq p^*$
  - *optimal duality gap*: $p^* - d^*$
- *strong duality*: $d^* = p^*$ ~ requires more than convexity

nearest neighbor

intuition



(a) $k = 1$



(b) $k = 15$

comments

- no training, slow testing
- nonparametric: huge memory
- how to pick distance?
- poor in high-dimensions

- theoretical error rate

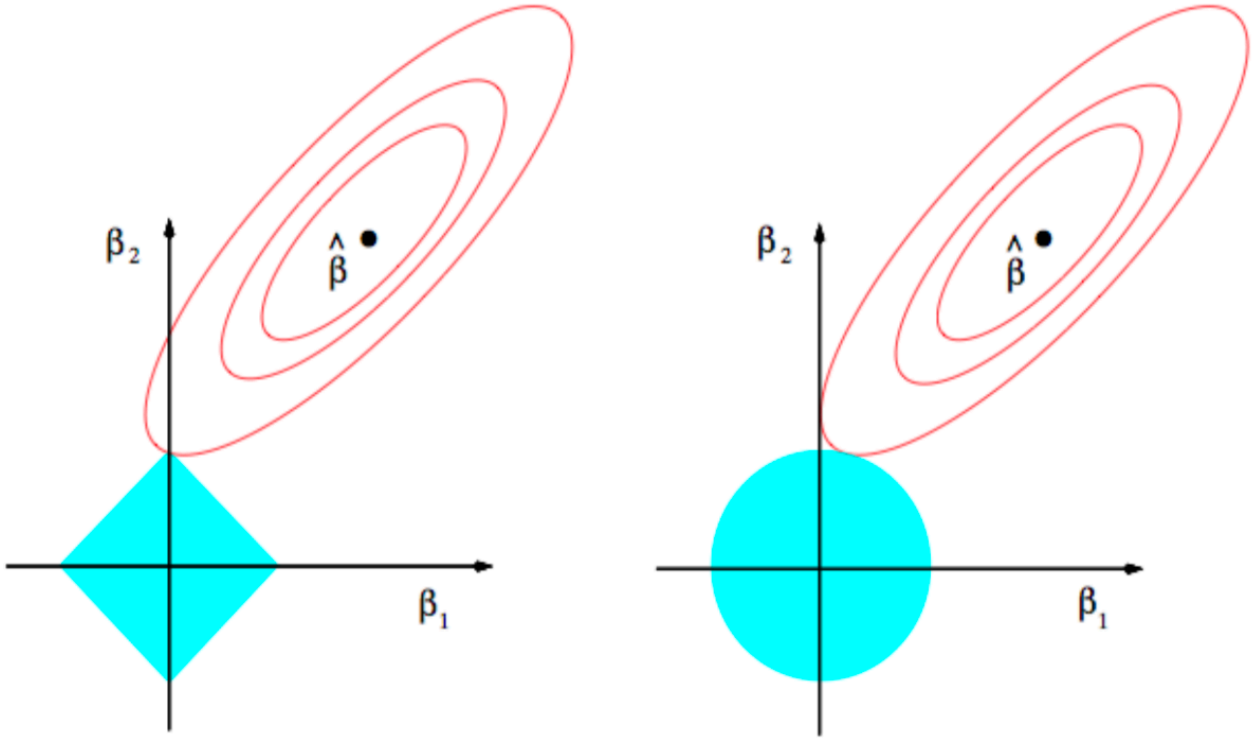
sparsity

$\underset{w}{min} \quad ||Xw-y||_2^2\\s.t.\quad||w||_0 \leq k$

constrained form

lasso: $\underset{w}{min} \quad ||Xw-y||_2^2\\s.t.\quad\color{cadetblue}{||w||_1} \leq k$

ridge: $\underset{w}{min} \quad ||Xw-y||_2^2\\s.t.\quad\color{cadetblue}{||w||_2} \leq k$



dual form

lasso: $\underset{w}{min} \quad ||Xw-y||_2^2+\lambda||w||_1$

- $\Delta = \lambda$

ridge: $\underset{w}{min} \quad ||Xw-y||_2^2+\lambda||w||_2^2$

- $\Delta = 2\lambda w$

lasso optimization

- coordinate descent: requires jointly convex
    - closed form for each w_i
    - iterate, might re-update w_i

matching pursuit

- start with all 0s
- iteratively choose/update w_i to minimize $||y - Xw||^2$

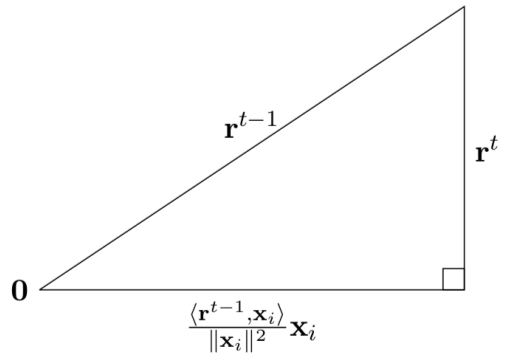


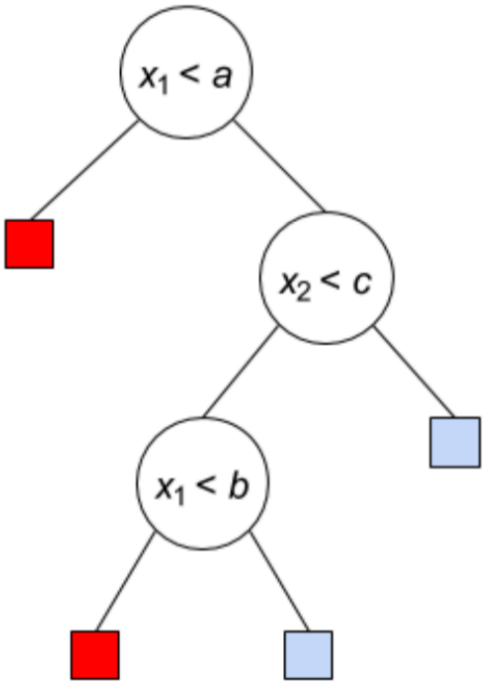
Figure 2: The updated residual $\mathbf{r}^t$, current residual $\mathbf{r}^{t-1}$, and scaled feature $\mathbf{x}_i$ form a right triangle.

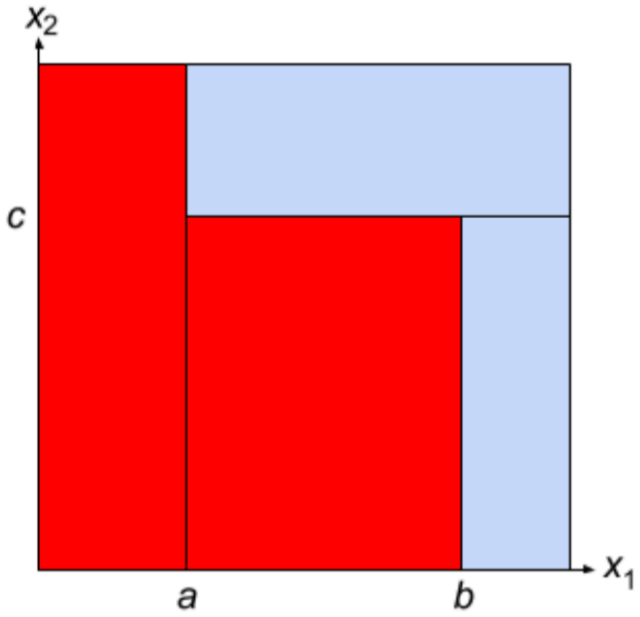
orthogonal matching pursuit

- at each step, update all nonzero weights

decision trees / random forests

decision tree intuition





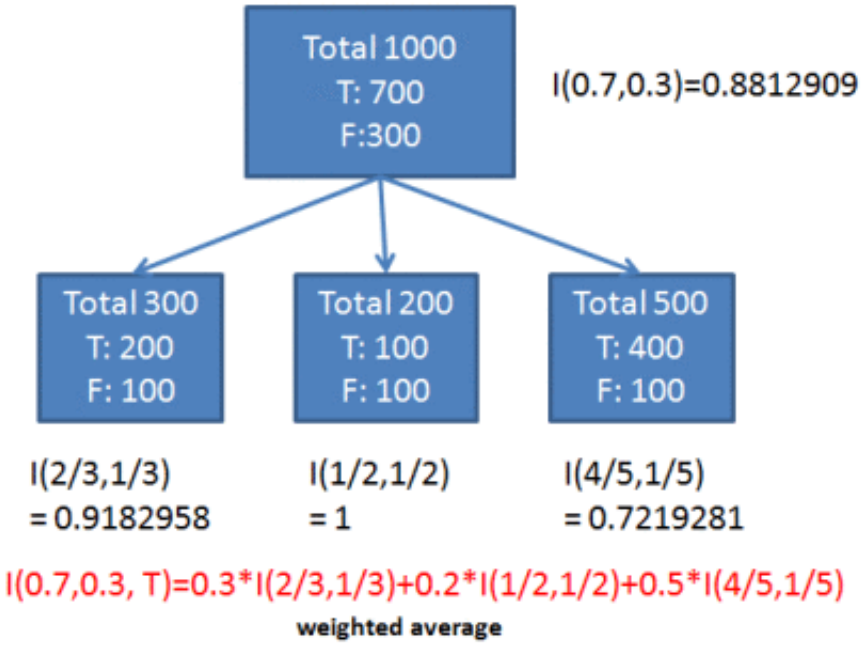
training

- greedy - use metric to pick attribute
    - split on this attribute then repeat
    - high variance

information gain

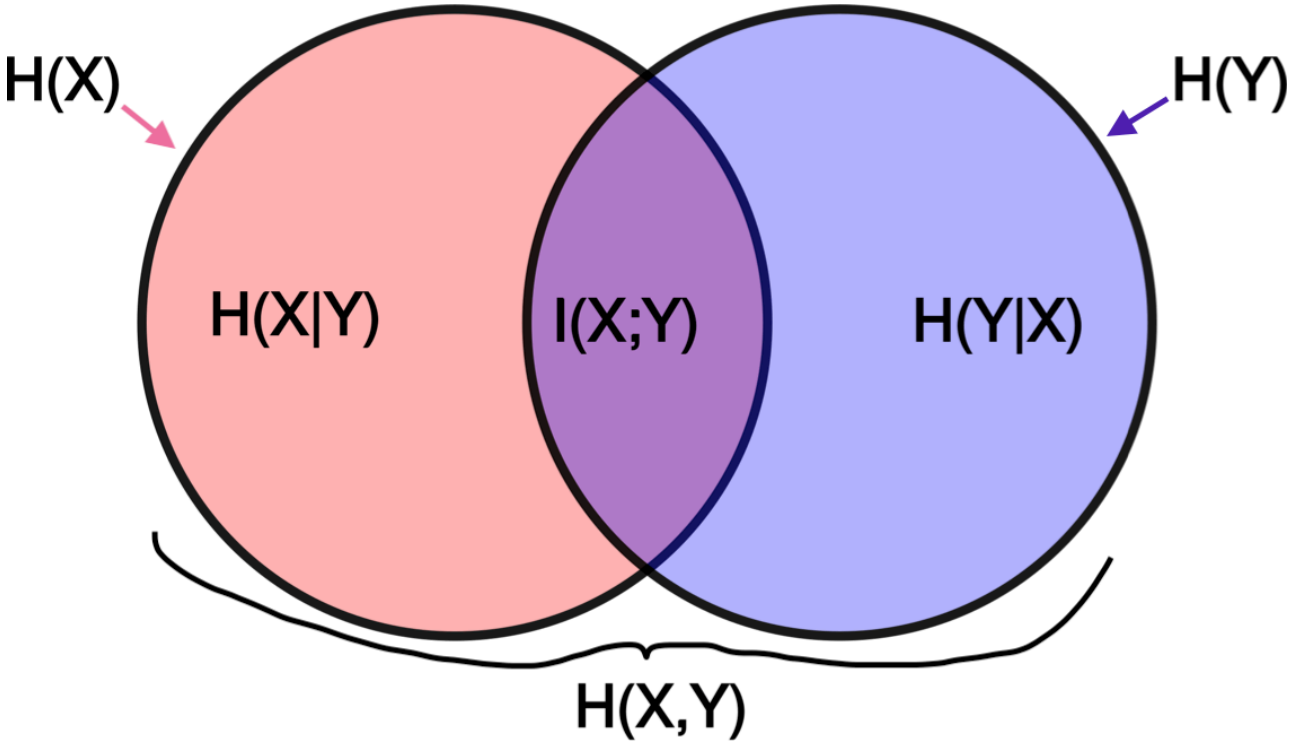
maximize: entropy(parent) - [weighted average] * entropy(children)

- often picks too many attributes



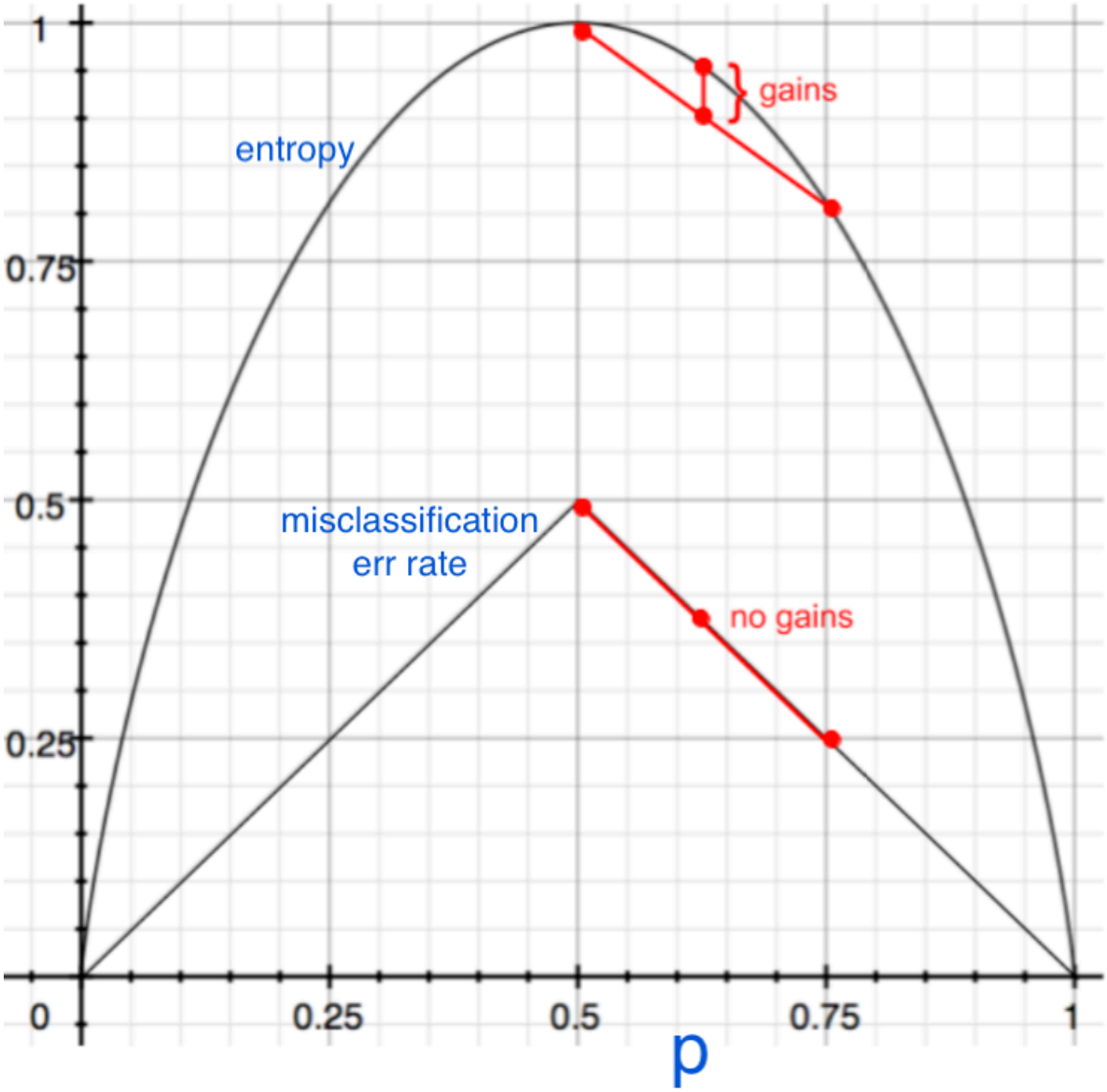
info theory

- maximize $I(X;Y) \equiv$ minimize $H(Y|X)$



split functions

- info gain
- gini impurity - very similar but faster to compute
- misclassification rate - doesn't always pick best splits
    - ex. real distr. ~ (40-40); two possible splits: (30-10, 10-30), (20-40, 20-0)



stopping

- depth
- metric
- node proportion
- pruning

random forests

- multiple classifiers
- *bagging = bootstrap aggregating*: each classifier uses subset of datapoints
- *feature randomization*: each split uses subset of features

random forest voting

- consensus
- average
- *adaboost*

regression tree

- stop splitting at some point and apply linear regression

other values

- missing values - fill in with most common val / probabilistically
- continuous values - split on thresholds

boosting

- train more *weak learners* to fix current errors

adaboost

- initialize weights to 1/n
- iterate
    - classify weighted points
    - re-weight points based on errors
- finally, output error-weighted some of weak learners

adaboost comments

- derived using exponential loss risk minimization
- test error can keep decreasing once training error is at 0

gradient boosting

- want to subtract gradient of loss with respect to current total model
- models need not be differentiable
- for squared loss, just the residual