CS771

Grading Scheme

Quizzes 20%, Assgn 30%., MidSem 20%., EndSem 30%
(24th Aug, 19th Oct)

Quizzes → Short quizzes (~2, 10% + 10%)

Assignments → ~3 A, 2-3 weeks, Groups of 5

☆ Machine Learning is the art and science of solving ill-understood tasks

→ Well-understood tasks → Rajasthan {RJ 23 PA 2262, RJ 23 SM 2824}; Uttar Pradesh {UP 67 AB 3601}; Delhi {DL 7C M0750}

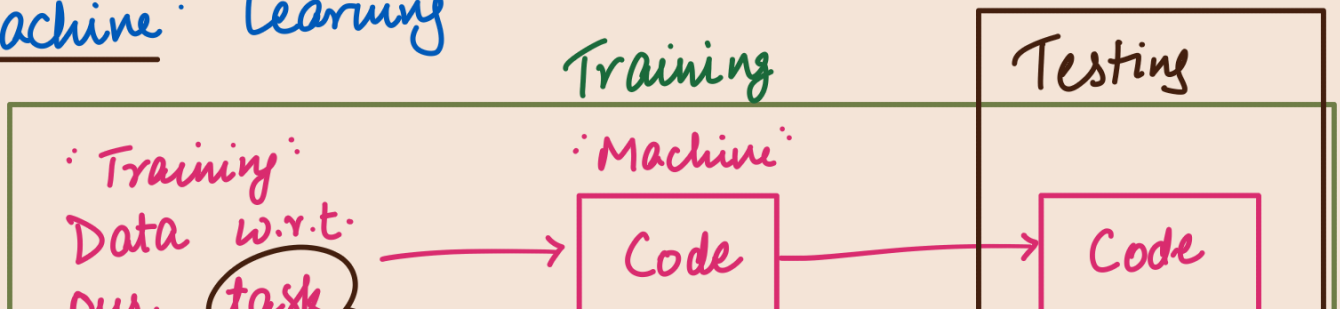
→ Ill-understood

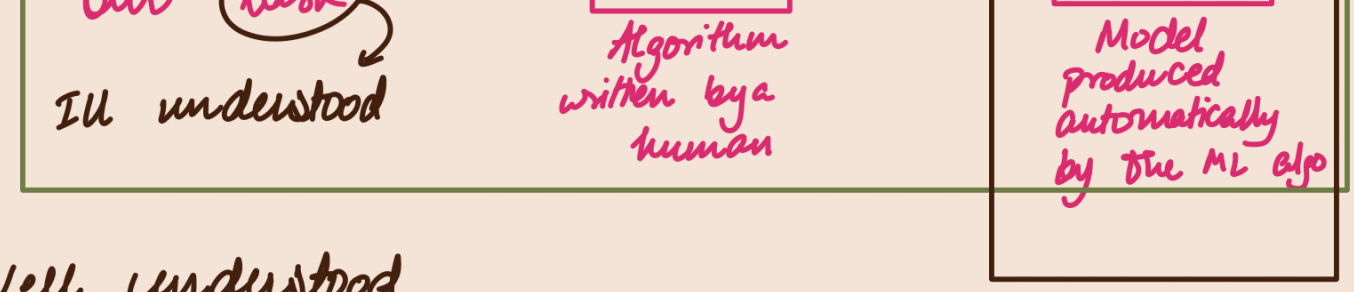
R R → R D D D D → D

Difficult to formally explain the rules of classification.

· Moreover, humans can mostly do only those tasks which they are already familiar with.

Why is it called Machine learning





Well understood task → Code → Solution

Code written by a coder

* Nature is governed by laws. Humans are sometimes able to discover these with a lot of time and effort.

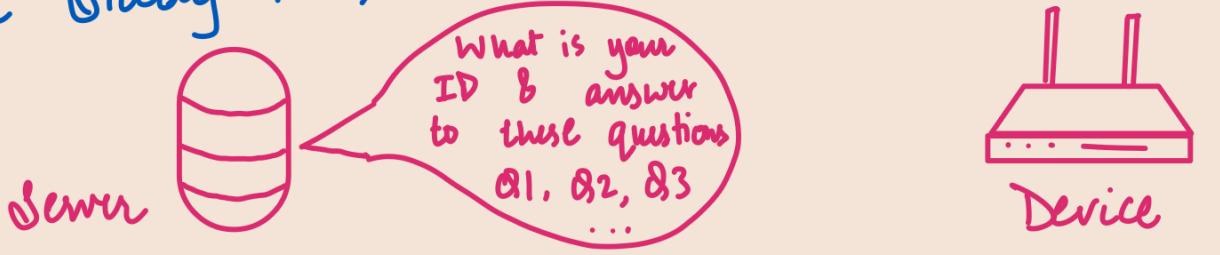
ML works by discovering laws too.

$P = \square \simeq \square + \square$

ML works by analyzing data to identify laws or patterns that seem to explain the data well.

The laws learnt by ML may be too complex to be interpreted by humans.

· Authentication using secret questions (Case Study # 1)

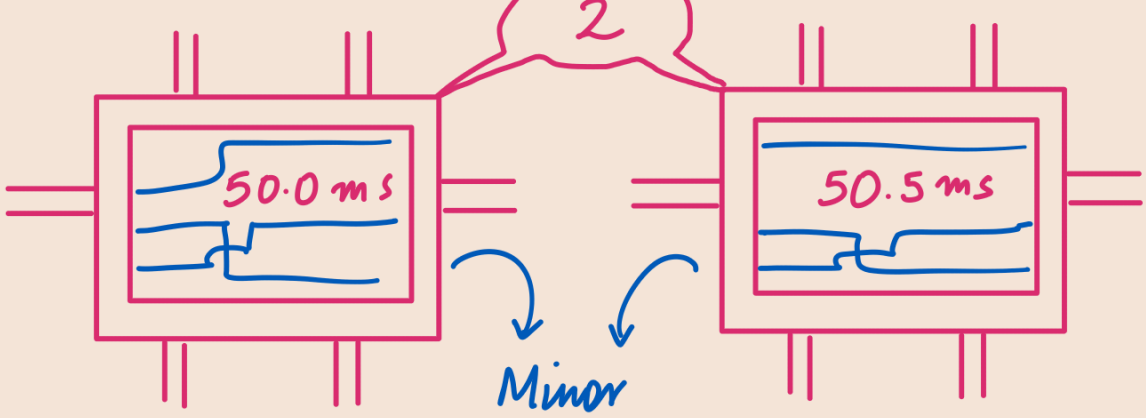


Questions → should be unique w.r.t. Device

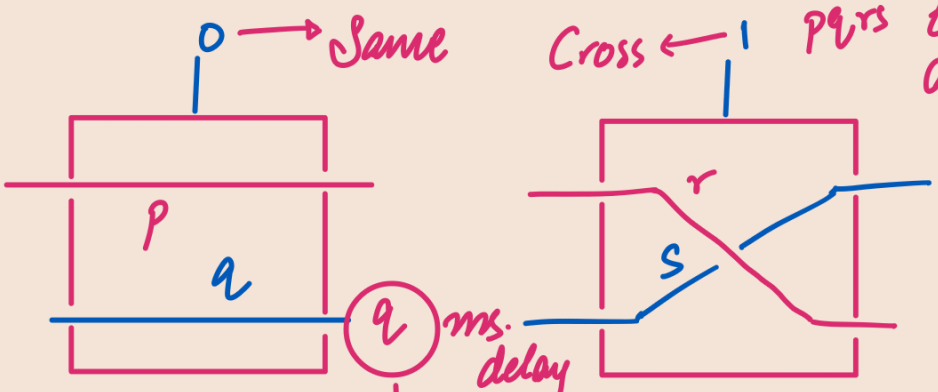
PUFs

Physically Unclonable Functions

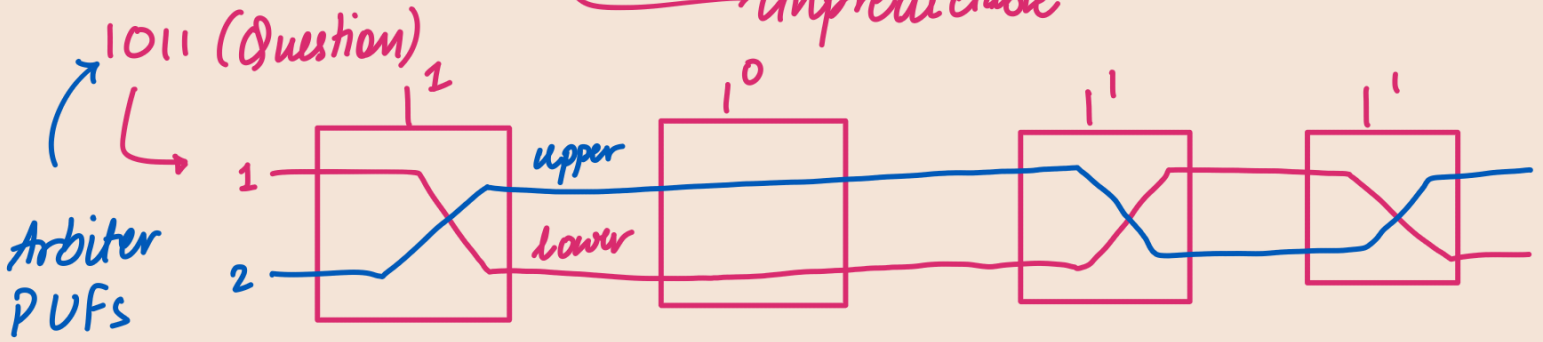
1+1 = ?



* Multiplexer → Switches



pairs causes different amounts of delay even in identical multiplexers



→ If upper signal arrives faster then assign result 0 otherwise 1.

- Questions — Challenges ; Answers — Responses
- Multiple Multiplexers → Circuits more complex ⇒ Difficult to predict
  - → Also ↑es the number of questions

Usually (64) multiplexers → 2^{64} questions

- ∴ Knowing the answer to a few questions

$\not\Rightarrow$ Knowing / Ability to predict answers to other questions : WRONG

An attacker can see responses on a few challenges & use ML to predict responses on all other challenges. No matter if # bits = 32/64

. Assume 64 bits {0 ... 63 labels}

☆ $t_i^u \longrightarrow$ Time taken for upper signal to reach out in the i^{th} ($i \in \{0, 1, \ldots, 63\}$) multiplexer

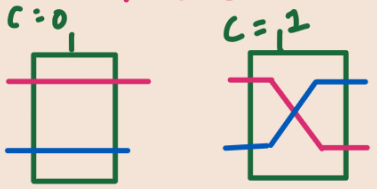
$t_i^l \longrightarrow$ ... lower signal ...

. Observe that answer is 0 if $t_{63}^u < t_{63}^l$, 1 otherwise

Also, note that t_1^u and t_1^l depend on t_0^u, t_0^l, p_1, q_1, r_1, s_1 and c_1.

c_1 dictates if t_0^l / t_0^u (previous delay) will get carried forward in which branch & p_1, q_1, r_1, s_1 give us the delay introduced by the i^{th} mux itself.

More concretely,



$$t_1^u = (1-c_1)\cdot(t_0^u + p_1) + c_1 \cdot (t_0^l + s_1)$$
$$t_1^l = (1-c_1)\cdot(t_0^l + q_1) + c_1 \cdot (t_0^u + r_1)$$

. Let $\Delta_i := t_i^u - t_i^l$ denote the lag

All that matters is whether the upper signal reaches first or not $\Rightarrow \Delta_{63} < 0$ or not

$\alpha_i := (p_i - q_i + r_i - s_i)/2$, $\beta_i := (p_i - q_i - r_i + s_i)/2$,

$\Delta_{-1} = 0$, $d_i := 1 - 2c_i$, $d_i \in \{-1, 1\}$

$$\Delta_i = d_i \cdot \Delta_{i-1} + \alpha_i \cdot d_i + \beta_i$$

$$\Delta_0 = \alpha_0 \cdot d_0 + \beta_0$$

$$\Delta_1 = \Delta_0 d_1 + \alpha_1 d_1 + \beta_1$$
$$= \alpha_0 d_1 d_0 + (\alpha_1 + \beta_0) d_1 + \beta_1$$

$$\Delta_2 = \Delta_1 d_2 + \alpha_2 d_2 + \beta_2$$
$$= \alpha_0 d_2 d_1 d_0 + (\alpha_1 + \beta_0) d_2 d_1 + (\alpha_2 + \beta_1) d_2 + \beta_2$$

$$\ldots \Rightarrow \Delta_{63} = w_0 \cdot x_0 + w_1 \cdot x_1 + \ldots + w_{63} \cdot x_{63} + \beta_{63}$$

$$\boxed{\Delta_{63} = w^T x + \beta} \quad \text{where} \quad \begin{cases} x_i = d_i d_{i+1} \cdots d_{63} \\ w_0 = \alpha_0 \\ w_i = \alpha_i + \beta_{i-1} \ (i > 0) \end{cases}$$

$\Delta_{63} < 0 \Rightarrow$ upper signal wins $\Rightarrow$ answer = 0

$\Delta_{63} > 0 \Rightarrow t^u_{63} > t^l_{63} \Rightarrow$ lower wins $\Rightarrow$ answer = 1

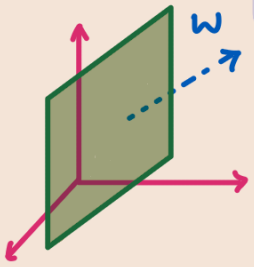
$\Rightarrow$ Answer is simply $\dfrac{\text{sign}(w^T x + \beta) + 1}{2}$

Nothing but a **linear classifier**

↳ **Model** $\equiv \{w, b\}$ → single vector (d-dim.) features are too; scalar term (bias)

↳ · Predict x by checking if $w^T x + b > 0$

· **Decision boundary**: **hyperplane** ($w^T x + b = 0$)



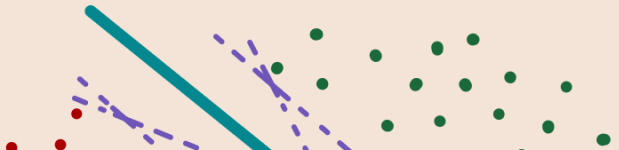
· **w**: $\perp$ / normal vector to the hyperplane

· Let x, y be vectors on hyperplane then
$w^T x + b = 0 = w^T y + b \Rightarrow w^T (x - y) = 0$
$\Rightarrow x - y \perp$ to w & $\parallel$ to the hyperplane

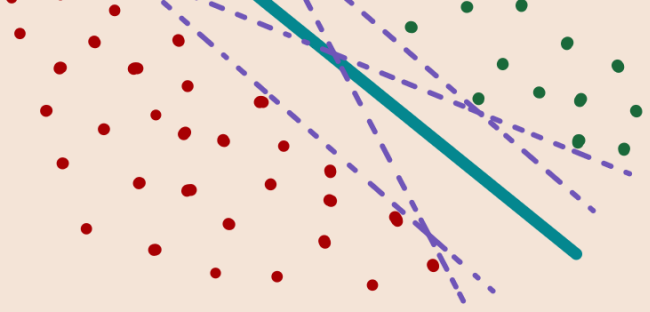
· **bias term** **b** shifts the plane { can be thought of as a threshold that $w^T x$ has to pass to make the decision 1 }

∴ Best: Linear Classifier

· Better not to select



a model whose decision boundary passes very close to a training data point



Fact: Dist. of origin from hyperplane $w^T x + b = 0$ is $|b| / \|w\|_2$ → $\{\|v\|_p = (\sum |v_i|^p)^{1/p} \Rightarrow \|w\|_2 \equiv$ length of $w\}$

Fact: Dist. of a point p from " is $|w^T p + b| / \|w\|_2$

- Given train data $\{(x^i, y^i)\}_{i=1}^n$ for a binary classification problem where $x^i \in \mathbb{R}^d$ and $y^i \in \{-1, 1\}$, we want two things from the classifier:

Demand 1: Classify every point correctly:

→ $\forall\ i \in [n]$ $\text{sign}(w^T x^i + b) = y^i$

→ $\forall\ i \in [n]$ $y^i \cdot (w^T x^i + b) \geq 0$

Demand 2: Not let any data point come close to the boundary

→ $\min_{i \in [n]} |w^T x^i + b| / \|w\|_2$ be as large as possible

Support Vector Machine (S.V.M.)

∴ Please find me a linear classifier that {satisfies D1 & D2} ∴

Mathematical way → $\max_{w,b} \left\{ \min_{i \in [n]} \{|w^T x^i + b| / \|w\|_2\} \right\}$

such that $y^i \cdot (w^T x^i + b) \geq 0 \quad \forall\ i \in [n]$

Known as an Optimization Problem w/ an ∴ objective ∴ and lots of ∴ constraints ∴

- Set of all points satisfying all ∴ constraints ∴ is

called the feasible set of the optimization problem

- Now, assume ∃ params that perfectly classify all train data. Consider one such (w, b):

$$|w^T x^i + b| / \|w\|_2 = \frac{|y_i \cdot (w^T x^i + b)|}{\|w\|_2} \quad \{\because y_i \in \{-1, 1\}$$

& $$\min_{i \in [n]} \frac{|y_i \cdot (w^T x^i + b)|}{\|w\|_2} \equiv \min_{i \in [n]} \frac{y^i \cdot (w^T x^i + b)}{\|w\|_2}$$

↓ Model has perfect classification

- Will remove the assumption that train data is linearly separable. Ex: ∴

- Let i_0 be the data point that comes closest:

$$\Rightarrow \min_{i \in [n]} \frac{y^i \cdot (w^T x^i + b)}{\|w\|_2} = \frac{y^{i_0} \cdot (w^T x^{i_0} + b)}{\|w\|_2}$$

{ Assuming (w, b) is a perfect classifier

Let $\epsilon = y^{i_0} \cdot (w^T x^{i_0} + b)$ & consider $\tilde{w} = w/\epsilon$, $\tilde{b} = b/\epsilon$

$$\Rightarrow y^i \cdot (\tilde{w} x^i + \tilde{b}) \geq 1 \quad \forall \; i \in [n]$$

&

$$\min_{i \in [n]} \frac{y^i \cdot (\tilde{w} x^i + \tilde{b})}{\|\tilde{w}\|_2} = \frac{1}{\|\tilde{w}\|_2} \quad \{\because y^{i_0} \cdot (\tilde{w}^T x^{i_0} + \tilde{b}) = 1$$

⇒ Thus, instead of searching for (w, b), easier to search for $(\tilde{w}, \tilde{b})$

$$\max_{\tilde{w}, \tilde{b}} \frac{1}{\|\tilde{w}\|_2} \rightarrow \min_{\tilde{w}, \tilde{b}} \|\tilde{w}\|_2 \rightarrow \min_{\tilde{w}, \tilde{b}} \|\tilde{w}\|_2^2 \rightarrow \min \frac{\|\tilde{w}\|_2^2}{2}$$

$$\text{s.t.} \quad y^i (\tilde{w}^T x^i + \tilde{b}) \geq 1 \quad \forall \; i \in [n]$$

- If a linear classifier cannot perfectly classify

data, then,

C-SVM

$$\min_{\tilde{w}, \tilde{b}, \{\xi_i\}} \; \tfrac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} \xi_i$$

$$\text{s.t.} \quad y^i \cdot (\tilde{w}^T x^i + \tilde{b}) \geq 1 - \xi_i \quad \forall\, i \in [n]$$

as well as $\xi_i \geq 0 \quad \forall\, i \in [n]$

Slack variables

Allow some data points to come close to the hyperplane or be misclassified altogether; Having $\xi_i \geq 0$ prevents us from artificially inflating the margin from 1.

What if slack variables allow the model to misclassify each data point? C prevents it. If C is set to be ↑, then it penalizes the models that slack too much.

C is a **hyper-parameter**

Now, ξ_i basically allow us to have

$y^i \cdot (\tilde{w}^T x^i + \tilde{b}) < 1$ (even < 0).

No, slack ξ_i is just $\xi_i = 1 - y^i \cdot (\tilde{w}^T x^i + \tilde{b})$

But, $\xi_i \geq 0$ too

∴ If we have $y^i \cdot (\tilde{w}^T x^i + \tilde{b}) \geq 1$,

then we don't need any slack, i.e., $\xi_i = 0$.

Thus,

$$\xi_i = [1 - y^i \cdot (\tilde{w}^T x^i + \tilde{b})]_+$$

where $[x]_+ = \max\{x, 0\}$

Hinge-Loss Function

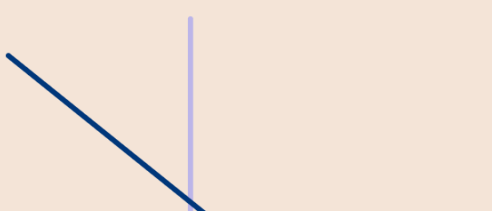
Suppose on (x, y), $y \in \{-1, +1\}$, model (w, b) predicts s $\{s = w^T x + b\}$,

we want $s \cdot y \geq 0$ for correct classification

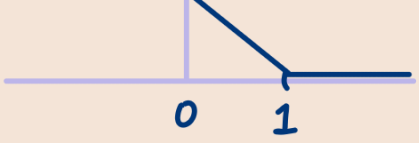
but we also want $s \cdot y \gg 0$ for large margin — hinge loss captures both:

$$\ell_{\text{hinge}}(s, y) = [1 - s \cdot y]_+$$

$$= \begin{cases} 0 & s \cdot y \geq 1 \\ \end{cases}$$



$\begin{cases} \cdot\,, & s \cdot y \geq 1 \\ 1 - s \cdot y\,, & s \cdot y \leq 1 \end{cases}$



∴ Replacing ξ_i by ℓ_{hinge}, we get,

C-SVM: $\min_{\tilde{w},\, \tilde{b}} \; \frac{1}{2} \|w\|_2^2 + C \sum_{i=1}^{n} \ell_{hinge}(\tilde{w}^T x^i + \tilde{b}, y^i)$

Optimization problems generally consist of three components:

1) **Objective** 2) **Constraints** 3) **Regularizer**

Sometimes, regularizers are taken inside the objective

☆ Using **Calculus** for Optimization

Method 1: **First Order** Optimality Condition

- Exploits the fact that gradient must vanish at a local optimum.
- For convex functions, local minima are global
  { only for simple functions w/ no constraints }

$f' = 0$ & $f'' \geq 0$

- Not very practical for checking stability

M2: **(Sub) gradient** Descent

- Direction opposite to gradient offers steepest descent. Choose a subgradient g^t & learning rate / step length η_t & update $w^{t+1} \leftarrow w^t - \eta_t g^t$
- Helpful in extending gradient descent to "not-nice" / non-differentiable functions

Some **Concerns**:

- Might get stuck at local minima / saddle points

⇒ Difficult to find good initialization conditions

- Step length: ↑ overshoot, ↓ slow convergence
- How to define and decide convergence?

M3: Creating a dual problem (constrained optimization)

We wish to solve:

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{s.t. } g(x) \le 0$$

sneak constraints into optimization → Use Barrier (indicator) function

$$\min_{x \in \mathbb{R}^d} f(x) + r(x)$$

Karush-Kuhn-Tucker

$$r(x) := \begin{cases} 0, & g(x) \le 0 \\ \infty, & \text{otherwise} \end{cases}$$

$$r(x) = \max_{\alpha \ge 0} \alpha \cdot g(x)$$

$$\min_{x \in \mathbb{R}^d} \left\{ f(x) + \max_{\alpha \ge 0} \alpha \cdot g(x) \right\}$$

$$\min_{x \in \mathbb{R}^d} \left\{ \max_{\alpha \ge 0} \left\{ f(x) + \alpha \cdot g(x) \right\} \right\}$$

Some steps for handling multiple + equality constraints:

S1: $\max f \longrightarrow \min -f$

S2: $g \ge 0 \longrightarrow -g \le 0$

S3: $s = 0 \longrightarrow s \le 0$ & $-s \le 0$

Only want min. & $\le$

S4: C constraints $\longrightarrow$ C variables

dual variables / Lagrange multipliers ↔ Original variables called primal variables

The Lagrangian

$$\min_{x \in \mathbb{R}^d} f(x) \quad \text{s.t. } g_1 \le 0, g_2 \le 0 \dots g_C \le 0$$

$$\min_{x \in \mathbb{R}^d} \left\{ \max_{\substack{\alpha \in \mathbb{R}^C \\ \alpha_c \ge 0}} \left\{ f(x) + \sum_{c=1}^{C} \alpha_c \cdot g_c(x) \right\} \right\}$$

$$\mathcal{L}(x, \alpha) = f(x) + \sum_{c=1}^{C} \alpha_c \cdot g_c(x)$$

$$\max_{\substack{\alpha \in \mathbb{R}^C \\ \alpha_c \ge 0}} \{ \mathcal{L}(x, \alpha) \} = \begin{cases} \infty, & x \text{ violates even one constraint} \\ f(x), & x \text{ satisfies every single constraint} \end{cases}$$

$$\min_{x \in \mathbb{R}^d} \left\{ \max_{\substack{\alpha \in \mathbb{R}^c \\ \alpha_c \geq 0}} \left\{ f(x) + \sum_{c=1}^{C} \alpha_c \cdot g_c(x) \right\} \right\}$$ **Primal Problem**

↳ Solution : $(\hat{x}^P, \hat{\alpha}^P)$

↳ Solution : $(\hat{x}^D, \hat{\alpha}^D)$

If original problem is convex and "nice"

$$\max_{\substack{\alpha \in \mathbb{R}^c \\ \alpha_c \geq 0}} \left\{ \min_{x \in \mathbb{R}^d} \left\{ f(x) + \sum_{c=1}^{C} \alpha_c \cdot g_c(x) \right\} \right\}$$ **Dual Problem**

Strong duality : $\hat{x}^P = \hat{x}^D$ if original problem is convex and "nice"

Complementary Slackness : $\hat{\alpha}_c^D \cdot g_c(\hat{x}^D) = 0 \quad \forall$ constraints c

Hard SVM (w/o a bias)

$$\min_{w \in \mathbb{R}^d} \frac{1}{2} \|w\|_2^2 \quad \text{s.t.} \quad 1 - y^i w^T x^i \leq 0 \quad \forall \, i \in [n]$$

n constraints $\Rightarrow$ n dual vars.

↳ Lagrangian : $\mathcal{L}(w, \alpha) = \frac{1}{2} \|w\|_2^2 + \sum_{i \in [n]} \alpha_i (1 - y^i \cdot w^T x^i)$

↳ Primal : $\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \underset{\alpha \geq 0}{\operatorname{argmax}} \left\{ \mathcal{L}(w, \alpha) \right\} \right\}$

Dual : $\underset{\alpha \geq 0}{\operatorname{argmax}} \left\{ \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \mathcal{L}(w, \alpha) \right\} \right\}$

simplifying ←

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \frac{1}{2} \|w\|_2^2 + \sum_{i \in [n]} \alpha_i (1 - y^i w^T x^i) \right\}$$

Unconstrained problem w/ a convex & diff. objective

↳ Apply first order optimality to solve it

$\Rightarrow \nabla = 0 \Rightarrow$ $w = \sum_{i \in [n]} \alpha_i y^i x^i$

$\Rightarrow$ Simplifies to

This is the problem several solvers (libsvm, sklearn) solve

$$\underset{\alpha \geq 0}{\operatorname{argmax}} \left\{ \sum_{i \in [n]} \alpha_i - \frac{1}{2} \sum_{i \in [n]} \sum_{j \in [n]} \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle \right\}$$

→ Once we get α_i → we can get w

Support Vectors

- We have α_i for every data point
- After solving the data points for which $\alpha_i \neq 0$: **Support Vectors** (usually $<< n$ in number)
- From complementary slackness, we have: A data point can be SV only if $y^i w^T x^i = 1$
- If we have bias b as a slack variable, then the dual looks like:

$$\underset{\alpha \in \mathbb{R}^n}{\text{argmax}} \left\{ \sum_{i \in [n]} \alpha_i - \frac{1}{2} \sum_{i \in [n]} \sum_{j \in [n]} \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle \right\}$$

s.t. $\alpha_i \in [0, C]$ & $\sum_{i \in [n]} \alpha_i y^i = 0$

Force on hyperplane = 0

$w = \sum_{i \in [n]} \alpha_i y^i x^i$ } Torque on hyperplane = 0

- $\sum \alpha_i y^i = 0$ difficult to solve because all α_i related → Need: Sequential Minimal Optimization (SMO) ← by John Platt
- If we omit bias (hide in model vector), dual is

$$\underset{\alpha \in \mathbb{R}^n}{\text{argmax}} \left\{ \sum_{i \in [n]} \alpha_i - \frac{1}{2} \sum_{i \in [n]} \sum_{j \in [n]} \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle \right\} \text{ s.t. } \alpha_i \in [0, C]$$

- We can solve SVM (no bias) by solving:

primal: $\underset{w \in \mathbb{R}^d}{\text{argmin}} \left\{ \frac{1}{2} \|w\|_2^2 + C \cdot \sum_{i \in [n]} [1 - y^i w^T x^i]_+ \right\}$

OR

dual: $\underset{\alpha \in \mathbb{R}^n}{\text{argmax}} \left\{ \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y^i y^j \langle x^i, x^j \rangle \right\}$

$\max_{\alpha \in \mathbb{R}^n} \min_{i \in [n]} \max_{j \in [n]}$ s.t. $\alpha_i \in [0, c]$

↳ Approaches like sub/gradient ascent/descent, coordinate ascent / descent can be used.

Will use coordinate maximization for dual

↳ optimization var. is α → Take one co-ordinate, ie, one data point at each step.

⇒ Concentrate only on the terms involving α_i, we get,

$$\text{argmax}_{\alpha_i \in \mathbb{R} \text{ s.t. } \alpha_i \in [0, C]} \left\{ \alpha_i - \tfrac{1}{2} \alpha_i^2 \|x^i\|_2^2 - \alpha_i y^i \sum_{j \neq i} \alpha_j y^j \langle x^i, x^j \rangle \right\}$$

↳ Just a constrained quadratic in α_i

$\tilde{\alpha} \in [0, C] \Rightarrow \alpha_i = \tilde{\alpha}$; $\tilde{\alpha} \leq 0 \Rightarrow \alpha_i = 0$

↳ $\frac{1}{2} q \alpha^2 - \alpha(1-p) = 0$ $\quad \tilde{\alpha} \geq C \Rightarrow \alpha_i = C$

$\tilde{\alpha} = (1-p)/q$, $q = \|x^i\|_2$, $p = y^i \sum_{j \neq i} \alpha_j y^j \langle x^i, x^j \rangle$

q: $O(d)$; p: $O(n \cdot d)$ $O(d)$

But, $w = \sum \alpha_i y^i x^i$ ⇝ $p = w^T x^i - \alpha_i y^i \|x^i\|_2^2$

⇒ We can update w, & update α_i in $O(d)$

Gradient

Primal Gradient Descent : $O(nd)$ time / update

Dual Gradient Ascent : $O(nd)$ time / update

Stochastic Gradient Methods

Stochastic PGD : $O(d)$ time / update

Stochastic DGA : $O(n)$ time / update

Coordinate Methods : $O(nd)$ if done naively

Stochastic Primal Coordinate Descent : $O(d)$

Stochastic Dual Coordinate Maximization : $O(d)$

Case 1: $n \gg d$: SDCM or SPGD

Case 2: $d \gg n$: SDGA or SPCD

☆ Co-ordinate Ascent: Small steps to ↑ obj a bit
" Maximization: Completely maximize obj.

ML primitives

1) What sort of action we wish to take on test data
2) In what form is training data available
3) What sort of model we learn & how we learn it

Primitives by type of action

☆ Simplest primitive → Yes / No question
Ex: Spam / Non-Spam ⇝ 2 classes
Often called **binary classification**

☆ What if more than 2 classes?
⇒ **Multi-class classification**
↳ Can be seen as sending an email to one bin
What if we want to send it to more bins?
↳ Useful to call it a labeling / tagging problem than a classification one.
multi-label classification

→ We **assume** that # & the **tags** remain **consistent** throughout training and testing
{ Can **remove** **assumption** w/ **zero-shot learning** }

☆ Assign a value / score to each data point
↳ **Regression**
Ex: 0 (Non-Spam) —— Real valued regression —— 5 (Definitely Spam)

Definitely → Can change scaled range too

But, we need to modify train/test data accordingly

- A few examples:

1) User v/s fruit recommendation

Technique-1: via multi-label classification
fruits: label; likes fruit: tag them;
user: data point
→ No. of labels might ↑↑

T-2: via regression
scale (0-5): how much liked by user;
(user, fruit): data point (👤, 🔺) = 4.8

2) Machine Translation
Ex: My name is Melbo → मेरा नाम Melbo है।

T-1: Multi-Class classification

English sentence: data point
(My name is melbo) → data point
↳ (मेरा name is melbo) → Repeat till done
Classes ↑ but this is the method used

Primitives by form of data

☆ Training data → ML algo → Model
↳ Most popular & simplest setting of learning → supervised learning

☆ Semi-supervised learning → Some training data left unlabelled

☆ Online learning → Model given to us trained
↓
User comes & model predicts
← Model learns & user added to train data afterwards

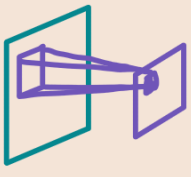
- Promotes continuous learning
- Train data not given to us in one go
- No distinction b/w train / test data

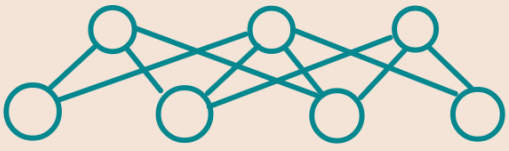
☆ Other variants: Active learning / Reinforcement learning

Primitives by type of model

☆ Geometric models:
Support vector Machins (use hyperplanes), Ridge regression

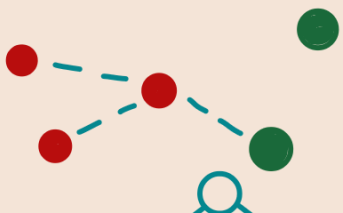
☆ Neural models:
Multi-layer perceptron, convolutional networks, transformers





☆ Probabilistic Models:
Logistic regression, Bayesian models

$P[\text{spam} \mid \boxtimes] < 0.05$



☆ Models with memory:

learning with prototypes, k-nearest neighbours, decision trees

Vectors: An ordered list of numbers
Matrices: An ordered rectangular array of numbers

Interpreting Model Outputs (Regression)

→ Predict temperature tomorrow
↳ Want model to predict b/w [-10, 50]
↳ Ask for scores b/w [0, 1] → Scale → Shift
↳ Use **sigmoid**: $\sigma(z) = \frac{1}{1+e^{-z}}$

Example of an **activation** / **squashing** function

→ If we want sigmoid to be sharper, use $z \to \lambda z$, $\lambda > 1$

Classification

→ **Binary**: Ask **1/0** {or +1/-1 in some cases} & then manually map them to labels
→ **Multi-class**: Ask **one-hot vectors** (<0,1,0,0>) w/ dimensions equal to # classes (Can use **softmax** activation)
→ **Multi-label**: Ask **multi-hot vectors**

Features of Data points

- Numerical features — As is ; age, temp, etc.
- Categorical features — 1-hot vector ; gender, blood group etc.
- Relational features — Many-hot vectors

Derived features

- Bagged / Binned features

Ex: Bag of words; {ESC101 (A), 207 (B), MSO (C), 771 (A), 340 (A), 220 (C)} → A=3, B=1, C=2

- Pooled / aggregated features

Ex: A has friends B, C, F. → # A friends : 3
not friends w/ D, E, G, H
OR take avg & stdev to represent data

Q Find my salary

→ Conjecture some law:
base salary b ; ↑ by d / year of age;
Δ = p/q/r/s depending on education;
M/F/O → Δ = k/l/m

→ $\hat{y} = b + d x_1 + p x_2 + q x_3 + r x_4 + s x_5 + k x_6 + l x_7 + m x_8$ ⇝ $\hat{y} = b + w^T x$

$x = \langle x_i \rangle_{i=1}^{i=8}$, $b = b$, $w^T = \langle d\ p\ q\ r\ s\ k\ l\ m \rangle$

→ $\hat{y} = f(x; w, b) = b + w^T x$

function (f) takes data point (x) and gives output ($\hat{y}$) w/ some unknowns called the parameters (vector w, scalar b) of the model

↪ Goal of ML algos is to identify parameters

Model: form of the function f, could have been $g = (w^T x)^b$

↓ $w^T x + b$ ⇝ Linear Model

Needs to find which params are good :-

If predictions ✓, then params good
↳ predictions match true outputs

Need to make this more mathematical
↳ Common to use **loss functions**:
→ Let feature vector of i^{th} user: x^i, true salary y^i, predicted salary $\hat{y}^i$
→ mismatch can be measured as
$l_i = |\hat{y}^i - y^i|$ OR $(\hat{y}^i - y^i)^2$
Absolute loss ↳ Least squares loss

Can't take just one data point
↳ Usually average of loss (across all data points in training data) is used

A **duality**: Data points and params operate on each other

ML model: Gives prediction $\hat{y}^i$ on data point x^i using param values θ
$\hat{y}^i = f(x^i; \theta)$

Loss function: Gives loss value l_i for params θ using data point (x^i, y^i)
$l_i = l(\theta; x^i, y^i)$

Often loss value computed using predicted and true outputs $l_i = l(\hat{y}^i, y^i) = l(f(x^i; \theta), y^i)$
Almost always, **average loss value** over entire training set is used $L(\theta) := \frac{1}{n} \sum_{i=1}^{n} l(\theta; x^i, y^i)$

- Use this loss as feedback to train the model

! **Train + test data** should be **similar**
!! **NEVER look** at **test** data **during training**
↳ Online learning obeys this because

data is first tested upon and then trained upon, NOT the other way around

☆ Models: take data points as input & use parameters to make predictions
ML algos: use loss functions to search for good parameter values

---

Geometry Basics

- Coordinates of a vector ≡ Point represented by vector
- Stretch: $|\lambda| > 1$
  Shrink: $0 < |\lambda| < 1$
  Flip: $\lambda < 0$
  } **Scalar Multiplication**
  ↳ Element-wise mult.
- **Adding** two vectors $u + v \longrightarrow$ " " addition
  Subtracting " " $u - v \equiv u + (-1)\cdot(v)$
  ↳ Can also use ∥gm law
- **Length** of a vector
  → **Euclidean** Length
  $u = \langle x, y \rangle \rightsquigarrow \sqrt{x^2 + y^2}$
  → Taxicab / **Manhattan** length
  $u = \langle x, y \rangle \rightsquigarrow |x| + |y|$
  → **Norms**; Entire family of so-called **L_p** norms defined as:

$$\|x\|_p := (|x_1|^p + |x_2|^p + \ldots + |x_d|^p)^{1/p}$$

* Notice that $L_2 \equiv$ Euclidean, $L_1 \equiv$ Manhattan

- Lengths can be used for **distances**
  ↳ $d_{Euclid}(u, v) := \|u - v\|_2$, $d_{Man}(u, v) := \|u - v\|_1$.
  These notions of distances are also called metrics. Can use any of the L_p norms to define L_p metric

L_p norms to define L_p metric

as $d_p(u, v) := \|u - v\|_p$

- **Metrics** satisfy 3 nice properties:

$$
\begin{aligned}
d_p(u, v) &= d_p(v, u) && \{\text{Symmetry}\} \\
d_p(u, u) &= 0 && \{\text{Identity}\} \\
d_p(u, v) &\leq d_p(u, w) + d_p(w, v) && \{\Delta \text{ inequality}\}
\end{aligned}
$$

- Measuring **angles**:

Dot product $\rightsquigarrow$ $\langle u, v \rangle := u^T v = \sum_{i=1}^{d} u_i \cdot v_i$

Also, $\langle u, v \rangle = \|u\|_2 \|v\|_2 \cos\theta \Rightarrow \theta = \cos^{-1}\left(\frac{\langle u, v \rangle}{\|u\|_2 \|v\|_2}\right)$

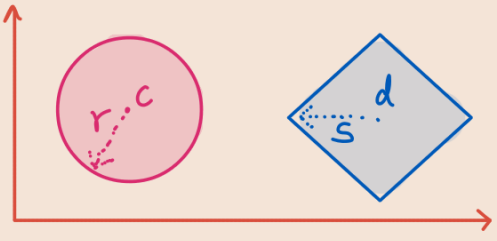
Obtuse angle $\Leftrightarrow$ dot product < 0

Acute angle $\Leftrightarrow$ dot product > 0

Perpendicular $\Leftrightarrow$ dot product $= 0$

Application of Norms

- $B_2(c, r) := \{u : \|u - c\|_2 < r\}$
- $B_1(d, s) := \{v : \|v - d\|_1 < s\}$

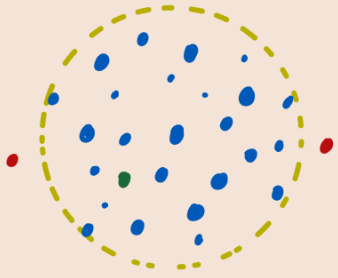


- Anomaly detection:

Use algorithms to find smallest enclosing ball

↳ Safe if pt. inside

↳ Danger if pt. outside



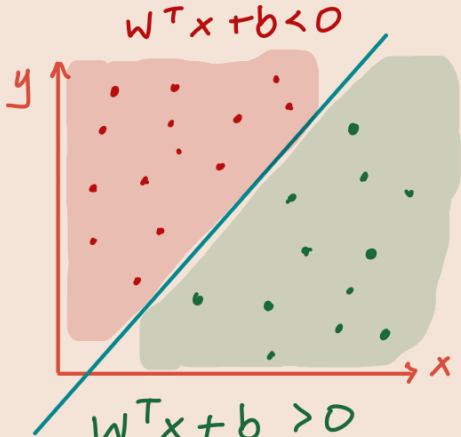
Dot product

- Binary Classification

2D $\rightarrow$ line, $y = mx + b$

Re-write $w^T x + b = 0$

$w = (m, -1)$, $x = (x, y) \in \mathbb{R}^2$



- Both sets called **halfspace**(s), separated by a line / **hyperplane**.
- Same trick works in higher dimensions too ⇝ $w, x \in \mathbb{R}^d$, $b \in \mathbb{R}$
  The hyperplane $\{x \mid w^T x + b = 0\}$ is called the **decision boundary**
- $\vec{w}$ is the **normal** or $\perp$ of hyperplane
  ∵ $x, y \in$ hyperplane $\Rightarrow w^T(x-y) = 0 \Rightarrow w \perp x-y$
  $\Rightarrow w \perp$ to all such difference vectors {which are ∥ to plane}

* **bias** $b \uparrow\downarrow \longrightarrow$ Hyperplane shifts
  $b \uparrow \Rightarrow$ more points classified as green
  $b \downarrow \Rightarrow$ less " " " "
  Changing $w \longrightarrow$ Plane rotates

To b or not to b

often critical for the model to classify better

ML algos simpler if no bias term ↷

↳ Cheat a bit & **hide it**

→ Create another dim in feature vector & fill it w/ 1 ie $\tilde{x} = [x, 1]$

→ Learn a $(d+1)$-dimensional linear model w/o bias term, $\tilde{w} = [w_1, w_2 \dots w_d\ w_{d+1}]$
$\tilde{w} \in \mathbb{R}^{d+1}$. Let $w = [w_1, w_2 \dots w_d] \in \mathbb{R}^d$, then

$$\tilde{w}^T \tilde{x} = w^T x + w_{d+1}$$

Thus, w_{d+1} effectively acts as a bias term

Convex Sets

$C \subseteq \mathbb{R}^d$

$\forall\ x, y \in C$

$\forall \lambda \in [0,1]$

$z = \lambda \cdot x + (1-\lambda) \cdot y$

$\Rightarrow z \in C$

Convex Set Non-Convex Set

Convex Functions

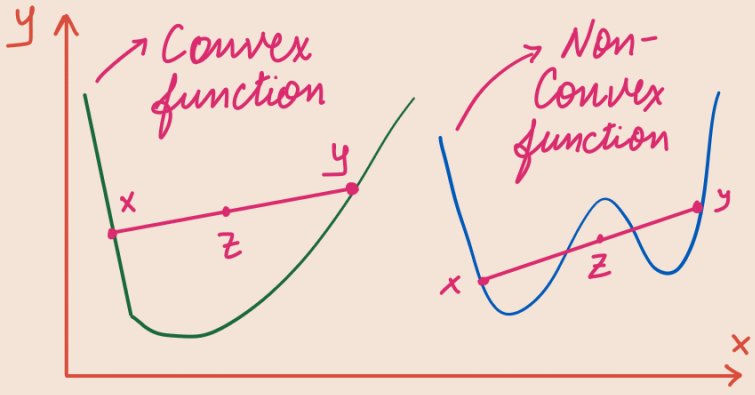
$f: \mathbb{R}^d \longrightarrow \mathbb{R}$

$\forall \; x, y \in C$

$\forall \; \lambda \in [0,1]$

$z = \lambda \cdot x + (1-\lambda) \cdot y$

$f(z) \leq \lambda \cdot f(x) + (1-\lambda) \cdot f(y)$



☆
- All Constant, linear functions are convex
- Sums of convex functions are convex
- Positive multiples " " " "
- $g: \mathbb{R}^d \to \mathbb{R}$ convex ; $f: \mathbb{R} \to \mathbb{R}$ convex & non-dec. then the function $f \circ g: \mathbb{R}^d \to \mathbb{R}$ convex
- $f: \mathbb{R} \to \mathbb{R}$ convex $\Rightarrow h(x) = f(a^T x + b)$ convex

---

Calculus Basics

Derivatives → Help us understand what happens to a function's output if its input is changed/perturbed a little

Taylor's Theorem —(Corollary)→ $f(x_0 + \Delta x) \approx f(x_0) + f'(x_0) \cdot \Delta x$

↳ Holds only if Δx small

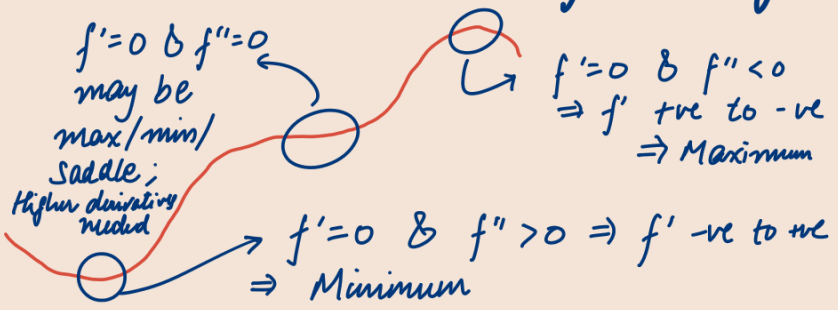
→ Δx sign same as $f'(x) \Rightarrow f(x_0 + \Delta x) > f(x_0)$

" opposite " $\Rightarrow f(x_0 + \Delta x) < f(x_0)$

- Sign of $f'(x_0)$ tells us in which dir^n will function value ↑ : + ≡ right ; − ≡ left
- Magnitude of $f'(x_0)$ tells how much it will change

Stationary Points: places where derivative vanishes
↳ Could be local/global minimum/maximum or a saddle point

- $f' = 0$ tells us that $f \approx$ flat around that point
  Very small scales ←
- Can separate min. w/ max. using 2nd derivative, f''
  ↳ Sign tells in which dir" f' ↑
  Magnitude " how much f' changes
- No general way of telling if a max/min is local or global



- Some rules: $(f+g)' = f' + g'$; $(af)' = a \cdot f'$, $a \in \mathbb{R}$ const.
  $(f(g))' = f'(g) \cdot g'$; $(fg)' = fg' + f'g$; $(f/g)' = (f' \cdot g - f \cdot g')/g^2$
  [$f(x), g(x) : \mathbb{R} \to \mathbb{R}$]

Multivariate Derivatives
a.k.a. Gradients

Consider $f(x, y) : \mathbb{R}^2 \to \mathbb{R}$

Trick: Convert the problem into analyzing $\mathbb{R} \to \mathbb{R}$ functions

→ Fix all variables except one then diff. w.r.t. that

Ex: $\frac{\partial f}{\partial x}$ at (x_0, y_0) : derivative of f w.r.t. x keeping y_0 const

↳ Sign tells us ↑↓ if $\Delta x > 0$ ($y = y_0$ fixed)
↳ Mag. tells how sharply f changes on Δx

Similarly $\partial f / \partial y$

- Vector $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) := \nabla f(x, y)$ ← Gradient of f

Co-ordinate-wise derivatives arranged in the form of vector

- $f: \mathbb{R}^d \to \mathbb{R}$ then $\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \dots \frac{\partial f}{\partial x_d}\right)$
- Stationary points $\longrightarrow$ $\nabla f = 0$

Steepest ascent

$$f(x^0 + t) \approx f(x^0) + t^T \nabla f(x^0)$$

$f: \mathbb{R}^d \to \mathbb{R}$, $t = (t_1, t_2, \dots, t_d) \in \mathbb{R}^d$, t is "small"

$$f(x^0 + t) \approx f(x^0) + \sum_{i=1}^{d} t_i \cdot \frac{\partial f}{\partial x_i}$$

Claim: The dir^n of the gradient vector offers the biggest ↑ in function value out of all directions

Steepest descent

For proof, use $f(x^0+t) \approx f(x^0) + t^T \cdot \nabla f$

$t^T \cdot \nabla f = \|t\|_2 \|\nabla f\|_2 \cos\theta = \langle t, \nabla f \rangle$ & then fix $\|t\|_2 = \epsilon$

Claim: ☆ The dir^n opposite to gradient vector offers the biggest ↓ in function value out of all directions

Some rules:

$f, g: \mathbb{R}^d \to \mathbb{R}$ & $s: \mathbb{R} \to \mathbb{R}$

Sum Rule: $\nabla(f(x)+g(x)) = \nabla f(x) + \nabla g(x)$

Scaling " : $\nabla(a \cdot f(x)) = a \cdot \nabla f(x)$, $a \in \mathbb{R}$ constant w.r.t. x

Product " : $\nabla(f(x) \cdot g(x)) = g(x) \cdot \nabla f(x) + f(x) \cdot \nabla g(x)$

Quotient " : $\nabla(f(x)/g(x)) = [g(x) \cdot \nabla f(x) - f(x) \cdot \nabla g(x)]/[g(x)]^2$

Chain " : $\nabla(s(f(x))) := (s \circ f)'(x) = s'(f(x)) \cdot \nabla f(x)$

Some identities:

- $c \in \mathbb{R}$ constant that doesn't vary w/ x $\quad \nabla(c) = 0$
- $a \in \mathbb{R}^d$ " vector " $\quad \nabla(a^T x) = a$

- $A \in \mathbb{R}^{d \times d}$ " matrix " $\nabla(x^T A x) = (A + A^T)x$
  $(= 2Ax$ if A symmetric$)$
- $a \in \mathbb{R}^d$ " vector " $\nabla(\|x - a\|_2) = \frac{x - a}{\|x - a\|_2}$

Ex: $w^T x + b$ on $\{(x^i, y^i)\}_{i \in [n]}$; $b, y^i \in \mathbb{R}$; $w, x \in \mathbb{R}^d$
w/ $\ell(w, b) = 1/n \sum (\hat{y}^i - y^i)^2 = 1/n \sum ((w^T x^i + b) - y^i)^2$

- Simplify by hiding bias b: $\tilde{x}^i = [x^i \; 1]$, $\tilde{w} = [w, 1]$
  & $\ell(\tilde{w}) := 1/n \sum (\tilde{w}^T \tilde{x}^i - y^i)^2$
- $\ell_i(\tilde{w}) := (\tilde{w}^T \tilde{x}^i - y^i)^2 = \underbrace{(\tilde{w}^T \tilde{x}^i)^2}_{\nabla = \nabla(\tilde{w}^T \tilde{x}^i \tilde{x}^{iT} \tilde{w})} + \underbrace{(y^i)^2}_{\nabla = 0} - \underbrace{2 y^i \cdot \tilde{w}^T \tilde{x}^i}_{\nabla = 2 y^i \tilde{x}^i}$
  $\nabla \ell_i(\tilde{w}) = 2 \tilde{x}^i \tilde{x}^{iT} \tilde{w}$

☆ Let $v = [v_1 \; v_2 \; \dots \; v_d] \in \mathbb{R}^d$ then
$v \cdot v^T = d \times d$ square symmetric matrix
whose $(i,j)^{th}$ entry is $v_i \cdot v_j$
$v^T \cdot v = \|v\|_2^2 \in \mathbb{R}^+$ {squared Euclidean length}

- $\nabla \ell(\tilde{w}) = 2/n \left(\sum \tilde{x}^i \tilde{x}^{iT}\right) \tilde{w} - 2/n \left(\sum y^i \cdot \tilde{x}^i\right)$

As $\ell(\tilde{w})$ diff, minima must be a stationary point where $\Rightarrow \nabla \ell(\tilde{w}) = 0 \Rightarrow (\tilde{X}^T \tilde{X}) w = \tilde{X}^T y$

$$\tilde{X} = \begin{bmatrix} \tilde{x}^1 \\ \tilde{x}^2 \\ \vdots \\ \tilde{x}^n \end{bmatrix} = \begin{bmatrix} x^1 & 1 \\ x^2 & 1 \\ \vdots & \vdots \\ x^n & 1 \end{bmatrix} \in \mathbb{R}^{n \times (d+1)} \quad \& \quad y = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^n \end{bmatrix} \in \mathbb{R}^n$$

- If $(\tilde{X}^T X) \in \mathbb{R}^{(d+1) \times (d+1)}$ invertible, then the solution must be

$$\tilde{w} = (\tilde{X}^T \tilde{X})^{-1} (\tilde{X}^T y)$$

If not invertible, then techniques such as regularization are used

---

Subgradients and Subdifferentials

→ For convex diff funcs, gradient defines a tangent hyperplane at every point & the function must lie above this plane

↳ Subgradients exploit and generalize this property to even points of non-differentiability

* Convex functions lie above all tangents

$(\forall\, x_0)\quad f(x) \geq \nabla f(x_0)(x - x_0) + f(x_0) \quad \forall\, x$

↳ Turn definition around and say that gradient is a vector g so that the hyperplane $0 = g^T(x - x_0) + f(x_0)$ is tangent to f at x_0

Subgradients: Given a (possibly non-diff. but convex) function $f: \mathbb{R}^d \to \mathbb{R}$ & point x°, any vector g that satisfies

$$f(x) \geq g^T(x - x_0) + f(x_0)$$

is called a subgradient of f at x°

Subdifferential: The set of all subgradients of f at a point x° is known as the subdifferential of f at x° & denoted by $\partial f(x^\circ)$

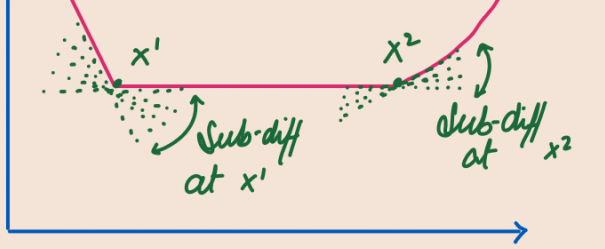
- Function may have multiple subgradients at x° but if it is diff. at x° then the gradient is the only subgradient at x°.

$$\partial f(x^\circ) := \{ g \mid f(x) \geq g^T(x - x^\circ) + f(x^\circ) \}$$

Subgradient Calculus

- Scaling Rule:

$$\partial(c \cdot f(x)) = c \cdot \partial f(x) = \{c \cdot v : v \in \partial f(x)\}$$



- Sum Rule:

$$\partial(f+g)(x) \subseteq \partial f(x) + \partial g(x) = \{u + v : u \in \partial f(x), v \in \partial g(x)\}$$

- Chain Rule:

$$\partial f(a^T x + b) = \{c \cdot a : c \in \partial f(a^T x + b)\}$$

$$\{\nabla f(a^T x + b) = f'(a^T x + b) \cdot a\}$$

- Max Rule: $h(x) = \max\{f(x), g(x)\}$

↳ $f(x^0) > g(x^0) \Rightarrow \partial h(x^0) = \partial f(x^0)$

& $f(x^0) = g(x^0) \Rightarrow h(x^0) = \{\lambda u + (1-\lambda) v : u \in \partial f(x^0), v \in \partial g(x^0), \lambda \in [0,1]\}$

Ex: $l_{hinge}(x) = \max(1-x, 0)$

$= \max(f(x), g(x))$

diff. $\forall x \in \mathbb{R} \setminus \{1\} \Rightarrow \partial l_{hinge}(x) = l'_{hinge}(x)$ if $x \neq 1$

At $x=1 \rightarrow f(x) = 1-x \rightarrow \partial f(x) = f'(x) = -1$

& $g(x) = 0 \Rightarrow \partial g(x) = 0$

$\therefore \partial l_{hinge}(1) = \{\lambda \cdot (-1) + (1-\lambda)(0) = -\lambda : \lambda \in [0,1]\} = [-1, 0]$

- Applying chain rule, we get,

$$l_{hinge}(y^i, \langle w, x^i \rangle) = [1 - y^i \langle w, x^i \rangle]_+$$

Need $v^i \in \partial l_{hinge}(y^i, \langle w, x^i \rangle)$

$$v^i = \begin{cases} 0, & y^i \langle w, x^i \rangle > 1 \\ c \cdot y^i \cdot x^i, & c \in [-1, 0], \quad y^i \langle w, x^i \rangle = 1 \end{cases}; \quad -y^i \cdot x^i, \; y^i \langle w, x^i \rangle < 1;$$

BTS in GD for SVM

$$f(w) = \frac{1}{2}\|w\|_2^2 + C \cdot \sum_{i \in [n]} [1 - y^i \cdot w^T \cdot x^i]_+$$

{ignore bias b}

$$\nabla f(w) = w + C \cdot \sum g^i y^i \cdot x^i, \quad g^i \in \partial l_{hinge}(y^i \cdot w^T x^i)$$

& $w^{new} = w - \eta \cdot \nabla f(w) = (1-\eta) \cdot w - \eta C \cdot \sum g^i y^i \cdot x^i$

Assume $\eta = 1$ for the sake of understanding.

$w^{new} = (1-\eta)\cdot w - \eta C \cdot g' y' \cdot x'$

Small η: $(1-\eta)$ large $\Rightarrow$ don't change w too much

Large η: Change w as much as gradient dictates

↳ w did well on (x', y'), say $y' \cdot w^T x' > 1 \Rightarrow g' = 0$ ← don't change w too much

→ did bad, say $y' \cdot w^T x' < 0$, then $g' = -1$

$\Rightarrow w^{new} = (1-\eta)\cdot w + \eta C \cdot y' \cdot x'$

$\Rightarrow y' \cdot (w^{new})^T \cdot x' = (1-\eta)\, y' \cdot w^T x' + \eta C \cdot \|x\|_2^2$ ← w^{new} gets better margin than w

Stochastic Gradient Method

$\nabla f(w) = w + C \cdot \sum g^i y^i \cdot x^i$, $g^i \in \nabla \ell_{hinge}(y^i \cdot w^T x^i)$

↓ $O(d)$ time only ← Take a random data point instead & update w.r.t. it

Calculating each g^i takes $O(d)$ time $\Rightarrow$ n data points

Total $O(nd)$ time

$\nabla f(w) = w + C \cdot g^{i_t} \cdot y^{i_t} \cdot x^{i_t}$

Might have to do more steps than in GD but each step cheaper, hence faster

Mini-Batch SGD

· If data diverse, the stochastic gradient may vary quite a lot depending on which random data point chosen. ⇝ Called variance

⇒ Choose a bunch (B to be precise) of data points and update w.r.t. them

↳ $O(B \cdot d)$ time per update

$\nabla f(w) \approx w + C \cdot \sum_{b=1}^{B} g^{i_t^b} y^{i_t^b} \cdot x^{i_t^b}$

* If $B = n$, then MBSGD = GD, and if $B = 1$, then MBSGD = SGD

Coordinate Descent

· Similar to GD except only one coordinate is changed in a single step

↳ $\min f(x)$ s.t. $x \in \mathcal{C}$ w/ $\nabla_j f(x) = \frac{\partial f}{\partial}$ ← j^{th} partial derivative

$x \in \mathbb{R}^d$ $\quad \frac{\partial}{\partial x_j}$

- CCD: Choose coordinate cyclically, $j_t = 1, 2 \dots d, 1 2 \dots d, \dots$
- SCD: Choose j_t randomly
- Block CD: Choose a small set of coordinates to update
- Randperm: Permute randomly and choose.
  Once the list is over, choose a new randperm.
- Sometimes, we are able to optimize completely along a given variable, called coordinate minimization (CM)

For $t = 0 \dots$
- Select $j_t \in [d]$
- $x_{j_t}^{t+1} \leftarrow x_{j_t}^{t} - \eta_t \cdot \nabla_{j_t} f(x^t)$
- $x_j^{t+1} \leftarrow x_j^t \quad \forall\ j \neq j_t$
- Repeat until convergence

Issues with GD

→ **Initialization**: Bad initialization → slow convergence unless step lengths nice [convex func]
- [Non-convex func]: Bad init → stuck at saddle points
- Random restarts most common solution to overcome
- For some nice non-convex problems, we know very good ways to provably init close to global optimum

→ **Deciding Convergence**:
- Small dist. of global / local optima
- Not making much progress, $\|w^{t+1} - w^t\| \to 0$
- GD stops on reaching a stationary point
  ↪ Progress stopped w/o a global optimum
- Few heuristics to stop GD:
  - If gradient vector too small
  - If obj. function already acceptably small

→ **Detecting Convergence**:

M1: Tolerance Technique
- Pre-decided tolerance ϵ, if $f(w^t) < \epsilon$, stop

M2: 0^{th} order technique
- If func. value doesn't change much, stop (or tune learning rate)

$|f(w^{t+1}) - f(w^t)| < \tau$ OR $|w^{t+1} - w^t| < \zeta$

M3: 1st order technique

- If gradient too small, $\|\nabla f(w^t)\|_2 < \delta$

M4: Cross validation technique

- Test current model on validation data

Other techniques like primal-duel are usually infeasible for large scale ML problems

→ **Choosing Step Length**

- For nicely behaved convex functions, have formulae for step length
- Set $\eta_t = C/\sqrt{t}$ or $\eta_t = C/t$, C is a hyperparameter. Idea is to choose $\eta_t \to 0$ (diminishing) and $\sum \eta_t \to \infty$ (∞ travel)
- For "nice" convex functions, ϵ convergence in just $O(\log(1/\epsilon))$ steps.

* Powerful but expensive technique:

Newton method: $g^t = (\nabla^2 F(w^t))^{-1} \nabla F(w^t)$

∴ Autotunes the step length so we may use $w^{t+1} \leftarrow w^t - g^t$

- For nice convex func, ϵ convergence in just $O(\log\log(1/\epsilon))$ steps
- But, computation of Hessian $\nabla^2 F(w^t)$ is expensive but can be approximated using a diag or a low-rank matrix
- For not so well-behaved convex & non-convex functions, ∃ several heuristics but no guarantee for them to work
  - Line-search: Find best η_t every time $\eta_t = \arg\min_{\eta \geq 0} f(w^t - \eta \cdot g^t)$
  - Armijo rule:

Start w/ ↑↑ η_t, if it doesn't reduce obj. func. sufficiently, ↓ η_t & repeat.

- Momentum methods like NAG, Adam, which infuses prev. gradients into cur. grad.
- Use diff. step length for each dimension of w (e.g. Adagrad) where η^t replaced by a diag. matrix E^t, $w^{t+1} \leftarrow w^t - E^t \cdot g^t$

Loss Functions

- Used in ML to ask the model to behave a certain way.
- Penalize undesirable behaviour and optimizers like SGD/SDCA then give us a model w/ desirable behaviour
- Ex. Hinge loss function penalizes if it either misclassifies a data point or else classifies it correctly w/ an insufficient margin.
- Might take avg. or sum of all data points. ↳ CSVM
- Ex:
  - Hinge
  - Squared Hinge
  - Logistic

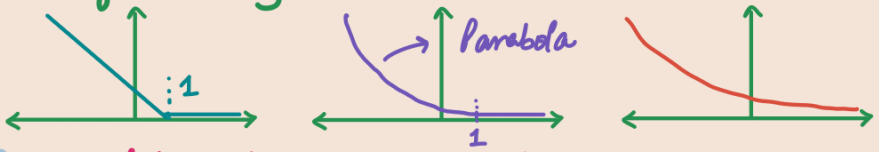
$l_{hinge}(y^i \cdot w^T x^i) = [1 - y^i \cdot w^T \cdot x^i]_+$ ○

$l_{sq\text{-}hinge}(y^i \cdot w^T x^i) = [1 - y^i \cdot w^T \cdot x^i]_+^2$ ○

$l_{logistic}(y^i \cdot w^T x^i) = \ln(1 + e^{-y^i \cdot w^T x^i})$ ○

Differentiable

Related to cross-entropy



- Logistic Regression: Obtaining a model using logistic loss function, i.e., $\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} \ln(1 + e^{-y^i w^T x^i})$
  - → Can be solved in primal using GD/SGD/MB
  - → Can be solved in dual using SDCA
    - ↳ Dual derivation not that straightforward
  - → Doesn't perform "regression"

Regression Problems

- Predict a **real value**
- Training data looks like $\{(x^i, y^i)\}_{i=1}^n$ where $x^i \in \mathbb{R}^d$, $y \in \mathbb{R}$

Loss functions for Regression Problems

- Can use linear models, (w, b), and predict score for test data point x^t as $\hat{y}^t = w^T x^t + b$

- Absolute Loss: $l_{abs}(\hat{y}, y) = |\hat{y} - y|$ → Bad if $\hat{y} \gg y$ & bad if $\hat{y} \ll y$
- Squared Loss: $l_{sq}(\hat{y}, y) = (\hat{y} - y)^2$ → Differentiable
- Vapnik's ϵ-insensitive Loss:

If slightly bad, don't punish, else penalized as a sq. func. Also, diff.

$$l_\epsilon(y, \hat{y}) = \begin{cases} (y - \hat{y} - \epsilon)^2, & \hat{y} < y - \epsilon \\ 0, & \hat{y} - y \in [-\epsilon, \epsilon] \\ (y - \hat{y} + \epsilon)^2, & \hat{y} > y + \epsilon \end{cases}$$

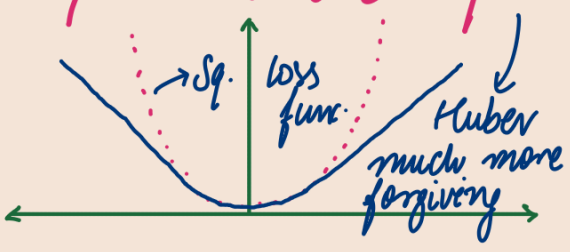
→ Slightly bad: sq. loss, else abs. loss

- Huber Loss: $l_\delta(y, \hat{y}) = \begin{cases} (\hat{y} - y)^2, & |\hat{y} - y| \le \delta \\ \delta \cdot |\hat{y} - y|, & |\hat{y} - y| \ge \delta \end{cases}$

* Loss functions have to be chosen a/c to needs of the problem.

* Huber Loss → Some data points are corrupted

* Hinge / Cross-entropy for binary classification
Squared for regression



Ridge Regression

{Ignore b}

$$\underset{w \in \mathbb{R}^d}{\arg\min} \; \lambda/2 \|w\|_2^2 + \sum_{i=1}^n (w^T x^i - y^i)^2$$

↑ and

Can rewrite with $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$

↳ First order optimality

↳ Gradient: $\lambda \cdot w + 2X^T(Xw - y)$

Must vanish → $w = (2X^TX + \lambda \cdot I)^{-1}(2X^Ty)$

Takes $O(d^3)$ time ← to invert the matrix → May use (S)GD

* Customary in regression to write optimization problem a bit differently than before ($\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{2}\|w\|_2^2 + C \cdot \sum (w^T x^i - y^i)^2$)
  ↳ Can use dual methods here too but not straightforward
  - Can't apply 1st order optimality w/ fancier loss functions (Vapnik's), need to use SGD, SDCA, etc.
  - Ridge Regression uses least squares loss & L_2 regularization

BTS: GD for Ridge Regression

$f(w) = \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^n (w^T x^i - y^i)^2$

$\nabla f(w) = \lambda w + 2 \sum (w^T x^i - y^i) \cdot x^i$

$w^{new} = w - \eta \cdot \nabla f(w) = (1 - \eta\lambda) \cdot w - 2\eta \sum (w^T x^i - y^i) \cdot x^i$

Assume $n = 1$

↳ $\eta \downarrow \Rightarrow (1 - \eta\lambda) \uparrow \Rightarrow$ Don't change w too much

- w does well $\Rightarrow y' = w^T x' \Rightarrow w^{new} = (1 - \eta\lambda) w$
- w does badly {say $w^T x' >> y$} then

{similarly for $w^T x' << y$} $w^{new} = (1 - \eta\lambda) w - 2\eta g \cdot x'$, $g >> 0$

$(w^{new})^T \cdot x' = (1 - \eta\lambda) w^T \cdot x' - 2\eta g \|x'\|_2^2 < w^T \cdot x'$

* GD merely tries to get models that behave "good"

Regularization

- An umbrella term used in ML to describe a

whole family of steps taken to prevent ML algos. from suffering from problems in data
- Help offer stable behaviour
- If data perfectly clean, no need for regularization
- Usually involves its own hyperparameters that need to be tuned using data itself { using validation techniques
- Regularization can be somewhat data independent
- Sometimes, makes the problem well-posed
- How to do regularization often decided w/o looking at data

Ex. $f(w) = \frac{1}{2}\|w\|_2^2 + C \cdot \sum [1 - y^i w^T x^i]_+$

$f(w) = \frac{\lambda}{2}\|w\|_2^2 + \sum [1 - y^i w^T x^i]^2$

$\|w\|_2^2$ → (Squared) L2 regularizer

In regression, ensures uniqueness of solution
- ↳ X^TX non-invertible
- ∞ sol^n ⇐ $\lambda = 0$

In binary classification, encourages large margin
- ↳ $\lambda > 0$ ensures unique solution

☆ A regularizer essentially tells the optimizer to not just blindly return a model that does well on data, but rather one that does well and is simple. [L2 regularizer → simple if small L2 norm
- In bin. class. → simple models also had large margins; Don't over-regularize

Large value of regularization ⇝ Very useless model
- Key is moderation → Use validation

Other considerations: How noisy is the data? How much data? (Rule of Thumb)

Can afford to ⇐ Data ↑ ⇒ ↓ regularization
- Other regularizers:

LASSO: $f(w) = \lambda \|w\|_1 + \sum (w^T x^i - y^i)^2$

L1-reg SVM: $f(w) = \|w\|_1 + C \cdot \sum [1 - y^i w^T x^i]_+$

↳ Prefers sparse vectors
- Faster at test time
  - ↳ Consume less memory
- Often make coordinates ≈ 0, $= 0$ to save space

· Popular in high dimensional problems, $d \approx 1$ million
· ∵ L1 non-diff, need to use subgradient methods
Although, proximal gradient descent does much better

☆ Regularization by Early Stopping
· Before it has solved the optimization problem fully
· Automatically prevents the model from fitting the data too closely (good if noise/outliers exist)
· Often happens implicitly w/ complex optimization problems
· May get an overregularized model that does not fit data at all if stopped too early

☆ Reg. by adding noise
· Add controlled noise so model learns to perform well despite noise
· Most well-known practice → Dropout in DL

* Or use subset of data that seems clean
* Or use corruption-aware loss functions (eg. Huber)
* Or use only informative, non-noisy features out of d_{total}
  ↳ Called sparse recovery
* Or by imposing convex constraints

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \|Xw - y\|_2^2 \quad \text{s.t.} \quad \|w\|_2^2 \leq R^2$$

Convex set {L_2 ball}

* Or by imposing non-convex constraints

$$\underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \|Xw - y\|_2^2 \quad \text{s.t.} \quad \|w\|_0 \leq s$$

Sparse recovery ←
$\|v\|_0$ = # Non-zero entries

Forces $s \ll d$ features ←

Multiclass Classification aka multiclassification

- Can be solved using linear models → Reduce to binary classification

- If there are C classes, train C linear models, each trained to identify one class
- Called the OVA (one v/s all) method
- At test time, ask all C models

OVA

* Create C binary classification datasets
  - For each $c \in [C]$, create the dataset: $y^{i,(c)} = \begin{cases} 1 & : y^i = c \\ -1 & : y^i \neq c \end{cases}$
  - Learn a model to distinguish data points in class c from those not in class c $\hat{w}^c = \arg\min_w \sum_{i=1}^n \ell(y^{i,(c)}, \langle w, x^i \rangle)$
  - At test time, predict the class whose model gives the test point the highest score $\hat{y}^t = \arg\max_{c \in [C]} \langle \hat{w}^c, x^t \rangle$
  - Can introduce margin here as well
  - Demand that if true class of data point x is c^*, $\langle w^{c^*}, x \rangle \geq \langle w^c, x \rangle + 1 \quad \forall\, c \neq c^*$
  - Introducing slack forms an optimization problem $\min_{\{w^c\}, \{\xi_i\}} \frac{1}{2} \sum_{c=1}^C \|w^c\|_2^2 + K \sum_{i=1}^n \xi_i$ $\text{s.t.} \cdot \langle w^{y_i}, x^i \rangle \geq \langle w^k, x^i \rangle + 1 - \xi_i \quad \forall\, k \neq y^i, \xi_i \geq 0$
  - Can rewrite this in terms of the $\eta_c := \langle w^c, x \rangle$ $\min \left(\frac{1}{2} \sum_{c=1}^C \|w^c\|_2^2 + \right.$

Crammer-Singer Loss

$$\{w^c\} \left(k \sum_{i=1}^{n} l_{cs}(y^i, \{\langle w^c, x^i \rangle\}_{c=1}^{C}) \right)$$

where $l_{cs}(y, \{\eta_c\}_{c=1}^{C}) = \left[1 + \max_{c \neq y} \eta_c - \eta_y\right]_+$

- Hinge → (Multiclassification) → Crammer-Singer Loss
  Logistic → (Multiclassification) → Softmax

$\eta_c := \langle w^c, x \rangle$, $l_{SM}(y, \{\eta_c\}_{c=1}^{C}) = -\ln\left(\frac{e^{\eta_y}}{\sum_{c=1}^{C} e^{\eta_c}}\right)$

- Softmax encourages η_y to be the largest of all η_c
  - $l_{SM} \to 0$ if $\eta_y \gg$ all η_c ∵ $\sum e^{\eta_c} \approx e^{\eta_y}$ & $\ln x \to 0$ as $x \to 1$
  - Diff unlike CS loss

☆ Other techniques → Output Codes, Decision Trees (very popular, especially if $C \gg 1$)

- Convert multiclassification into regression problems
- Represent $c \in [C]$ as a k-dim vector $v_c \in \mathbb{R}^k$
- Solve k regression problems ↔ Find v_1 for c_1, Find v_2 for c_2 ...
- At test time, predict k numbers, see which vector out of $v_1, v_2, \ldots$ is it closest to
- k small for speedup but can't have very small k. $k > 1$ — Account for regression mistakes

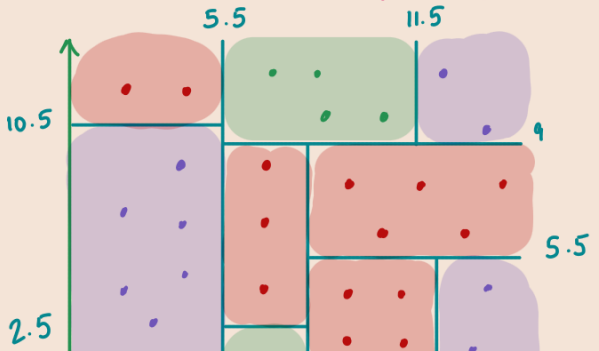
---

Decision Trees

* Can handle non-numeric features easily
* Popular in ML - classification, recsys
* Extremely fast at predicting
* Easy to interpret
* Model size can be large
* Can give good train w/ bad test (overfitting)

Algo: Repeatedly partition feature space $\mathbb{R}^d$

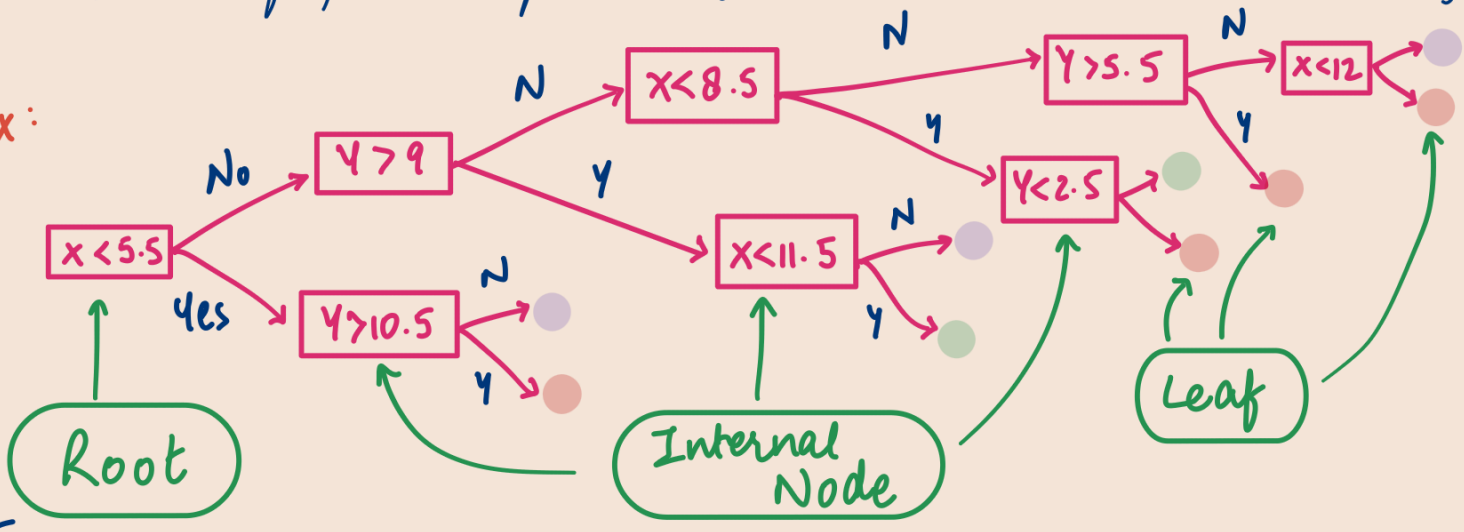
- For test data point, much faster to find out which



partition is it in

- Can consider assigning the label of other points in partition
- Sometimes, nearest neighbours lie in other partitions → DT may be wrong about points at the boundary
  - Rare if partitions fine enough

Ex:



- **Balanced DT**: All leaf nodes at same depth from root
  - Imbalanced → Bad
    - Very poor prediction accuracy
    - Can take as long as kNN to predict
    - Imagine DT: chain of nodes, n data points
      ⇒ O(n) chain ⇒ some predictions take O(n) time
- Balanced DT → O(log n) time
  - May prune the tree to make it shallower

- Regression w/ DT → Can use avg. score of a leaf node to predict scores for test data

can't # of children/node

How many children?

How to learn a DT?

What to do at a leaf?

When to stop splitting & make the node a leaf?

How to send data points to children?

How many trees to train?

- What to do at leaf?

· Can take any (complicated) action at a leaf

Another ML Algo?

For speed, keep it simple → Constant prediction

Cheapest thing: Store majority color at a leaf.

More informative (and expensive) thing: Store how many training points of each color reached that leaf

· Such a DT will encode a piecewise const. prediction function

☆ How to split a node?

· Splitting a node is a classification problem in itself. Binary if 2 children, multiclass otherwise

In principle, can use any ML/classification technique (even a deepnet). But in practice, simple ML algo like linear used to retain speed → Send data point rapidly to a leaf.

One goal of DT is to speed up kNN prediction time. Thus, node splitting algo should be superfast, otherwise, may as well use NN.

· Often just a single feature is chosen and node is split based on that.

Such "simple classifiers" often called decision stumps

How to find this feature? Choose the one that makes the child nodes created as pure as possible

Child node "pure" if it contains training data of only one class

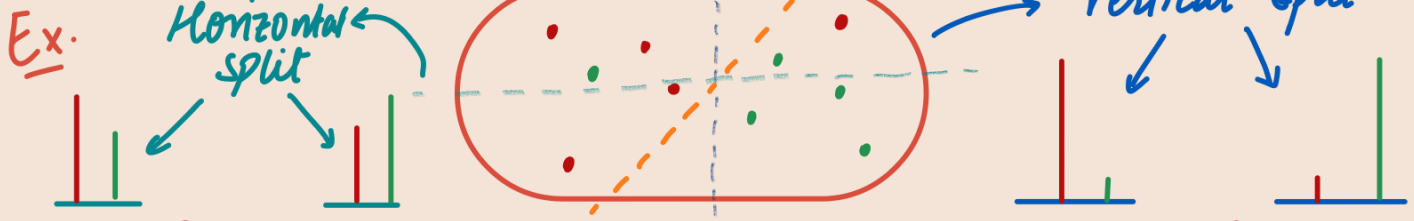
→ Convenient because can be made leaf directly

☆ Notion of purity?

· Various: entropy / Gini index for classification, variance for regression problems

· Making sure that split is balanced is also important, but this is often tricky.

Vertical split



- Purest vertical split is more pure, choose that.

Linear Classifier used to split the node

★ **Pruning Strategies**

- Stop if node is almost / pure
- Stop if all features exhausted, avoid using a feature twice on a path, limits depth to atmost d (# of dimensions)
- Can stop if node is ill-populated (few training points)
- Can also (over) grow a tree and then merge nodes to shrink it
- Merge two leaves and see if it worsens performance on validation set — rinse & repeat
- Always use a validation set to make these decisions, never the test set

---

Probability Basics

- How frequently does something happen
- Way of measuring the amount of uncertainty in a statement, Way of assigning positive scores in a way so that two scores can be easily compared

Sample Space: Denotes an exhaustive enumeration of all possible outcomes that have either happened or could happen (even if extremely unlikely)

→ Often makes the space ∞ in size in real settings

Event: Simply a description of useful facts about an outcome.

- ML can be used to tell us how frequently an event occurs / if one event is more likely than other
- Tell us how confident is the ML algorithm while giving the above replies
- An event may choose to be precise about certain aspects and neglect others.

Random Variable: A way to express useful facts about events as numbers so that we can do math.

- RVs can be categorical or numerical:
  - → Categorical: $X = 1$ if female, 2 male, 3 trans
  - → Numerical (Discrete): Y = age of person in years
  - → Numerical (Continuous): Z = no. of sec. spent on websit
  - → Indicator: $W = 1$ if purchase, 0 if no purchase
- Can arrange many RVs as vectors too
- Features ⟷ R.V.

Probability Distribution

- Given an event, it can tell us how likely it is ⇝ can also compare two events
- Generate a sample outcome

Example

- Sample Space: $\{R, G, B\} \times [6]$

1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

- $\mathbb{P}[R] = 14/24$, $\mathbb{P}[B] = 4/24$.
  $\mathbb{P}[1] = 4/24 = \mathbb{P}[2] = \ldots = \mathbb{P}[6]$. $\mathbb{P}[R \wedge 5] = 3/24$
- Note: $\mathbb{P} \geq 0$ always; $\mathbb{P}$ can be thought of in terms of proportions / frequency
- $\mathbb{P}[\text{A particular ball}] = 1/24$

- R.V.s $X :=$ # on ball $\in [6]$, $Y :=$ Colour on ball $\{R=1, G=2, B=3\}$
- $\mathbb{P}[X=1] = 4/24$, $\mathbb{P}[Y=2] = 6/24$
- Suppose now they are not equally likely, but have prob. assigned → No longer proportions

1/24	1/24	1/96	1/24	1/6	1/96
1/48	1/24	1/12	1/96	1/48	1/24
1/16	1/24	1/48	1/96	1/12	1/96
1/96	1/24	1/24	1/48	1/24	1/12

- $\mathbb{P}[X=1] = 1/24 + 1/48 + 1/16 + 1/96 = 13/96$, $\mathbb{P}[Y=2] = 1/3$

Rules of Probability

Ω: Sample Space, X: Any (discrete) R.V.
S_X: Set of (numerical) values X could take
↳ called the support of the R.V. X
$X(\omega)$: Value of X on outcome $\omega \in \Omega$

Lazy notation

$$\mathbb{P}[x] = \mathbb{P}[X=x] := \sum_{\substack{\omega \in \Omega \\ : X(\omega)=x}} p_\omega = \sum_{\substack{\omega \in \Omega \\ : X(\omega)=x}} \mathbb{P}[\omega] \qquad x \in S_X$$

★ If X is discrete valued, then this must hold:
- $\mathbb{P}[X=x] = 0 \Rightarrow x$ is an impossible value for R.V. X
- $\mathbb{P}[X=x] = 1 \Rightarrow X$ almost surely takes the value x
- $\sum_{x \in S_X} \mathbb{P}[X=x] = 1$ ↳ X cannot remain undefined, must take some valid value.

We may not know this value but $\exists$ some value
$\Rightarrow \forall\, x \in S_X$, $\mathbb{P}[X=x] \leq 1$

Probability Mass Function (PMF)

↳ Tells us prob. of an RV taking a particular value
- For discrete RV X, its PMF $f(\cdot)$ tells us for any $x \in S_X$, $f(x) = \mathbb{P}[X=x]$
- $\mathbb{P}_X[\cdot]$ used to emphasize that this PMF

is for X and not some other Y. $\mathbb{P}[X]$ also used

- Sampling from a PMF: $x \sim \mathbb{P}[X] / \mathbb{P}_X$ or even $X \sim \mathbb{P}[X]$ means that we generated an outcome $\omega \in \Omega$ a/c to prob. dist. and are looking at $X(\omega)$

Joint Probability $\mathbb{P}[X=1, Y=2] = \mathbb{P}[X=1 \wedge Y=2]$
$:=$ prob. that $X=1$ & $Y=2$

$\mathbb{P}[X=2 \wedge Y=3] = 0$

PMF for joint Distribution $\rightarrow \mathbb{P}[X, Y] = \mathbb{P}_{X,Y}[\cdot]$

$\star$ $\mathbb{P}[X=x] = \sum_{y \in S_Y} \mathbb{P}[X=x, Y=y]$

$\star$ $\because \sum_{x \in S_X} \mathbb{P}[X=x] = 1 \Rightarrow \sum_{x \in S_X} \sum_{y \in S_Y} \mathbb{P}[X=x, Y=y] = 1$

- Can sample using PMF for JD too.

Marginal Probability

- When talking about only a subset of R.V.s, the PMF is called marginal prob. dist.
- Given $\mathbb{P}_{X,Y,Z}$, we can obtain

$\mathbb{P}_{X,Z} = \sum_{y \in S_Y} \mathbb{P}_{X,Y,Z}$; $\mathbb{P}_X = \sum_{y \in S_Y} \sum_{z \in S_Z} \mathbb{P}_{X,Y,Z}$

Y has been marginalized out. Y & Z have been marginalized out.

Conditional Probability

$\mathbb{P}[X=1 \mid Y=2] :=$ Prob. of $X=1$ among the samples in which $Y=2$

$\equiv \dfrac{\text{Proportion of samples w/ } Y=2 \text{ \& } X=1}{\text{Proportion of samples w/ } Y=2}$

$\Rightarrow$

$$\mathbb{P}[X=1 \mid Y=2] = \frac{\mathbb{P}[X=1 \wedge Y=2]}{\mathbb{P}[Y=2]}$$

PMF for Conditional Distribution

· We had, for any $x_0 \in S_X$, $\sum_{y \in S_Y} \mathbb{P}[X=x_0, Y=y] = \mathbb{P}[X=x_0]$

$$\Rightarrow \sum_{y \in S_Y} \mathbb{P}[Y=y \mid X=x_0] = \sum_{y \in S_Y} \frac{\mathbb{P}[Y=y \wedge X=x_0]}{\mathbb{P}[X=x_0]} = 1$$

· $\mathbb{P}[Y \mid X] = \mathbb{P}_{Y|X}[\cdot|\cdot]$; May sample $y \sim \mathbb{P}[Y \mid X=x_0]$ too

* For sampling from $\mathbb{P}[Y \mid X=x_0]$, we consider the set of only those $\omega \in \Omega$ where $X(\omega)=x_0$ then sample an outcome ω_0 w/ prob. $\frac{p_{\omega_0}}{\mathbb{P}[X=x_0]}$ & then return $Y(\omega_0)$

Marginal Conditional Probability

$$\mathbb{P}[X=x \mid Z=z_0] = \sum_{y \in S_Y} \mathbb{P}[X=x, Y=y \mid Z=z_0]$$

$$\mathbb{P}[Y=y \mid X=x_0, Z=z_0] = \frac{\mathbb{P}[Y=y \wedge (X=x \wedge Z=z)]}{\mathbb{P}[X=x \wedge Z=z]}$$

$$= \frac{\mathbb{P}[(Y=y \wedge X=x) \wedge Z=z]}{\mathbb{P}[Z=z]} \cdot \frac{\mathbb{P}[Z=z]}{\mathbb{P}[X=x \wedge Z=z]} = \frac{\mathbb{P}[Y=y, X=x \mid Z=z]}{\mathbb{P}[X=x \mid Z=z]} =$$

★ Sum Rule $\mathbb{P}[x] = \sum_{y \in S_Y} \mathbb{P}[x,y]$
(Marginalization Rule) aka Law of Total Probability

★ Product Rule $\mathbb{P}[x,y] = \mathbb{P}[x \mid y] \cdot \mathbb{P}[y]$
(Conditioning Rule) $\mathbb{P}[x] = \sum_{y \in S_Y} \mathbb{P}[x,y] = \sum_{y \in S_Y} \mathbb{P}[x \mid y] \cdot \mathbb{P}[y]$
OR, $\mathbb{P}[X=x, Y=y] = \mathbb{P}[X=x \mid Y=y] \cdot \mathbb{P}[Y=y]$

★ Chain Rule (Iterated Conditioning Rule)

$$\mathbb{P}[x,y,z] = \mathbb{P}[x \mid y,z] \cdot \mathbb{P}[y \mid z] \cdot \mathbb{P}[z]$$

Bayes Theorem

$$\mathbb{P}[X=x \mid Y=y] = \left(\frac{\mathbb{P}[X=x, Y=y]}{\mathbb{P}[Y=y]}\right), \quad \mathbb{P}[Y=y \mid X=x] = \left(\frac{\mathbb{P}[X=x, Y=y]}{\mathbb{P}[X=x]}\right)$$

Equal

$$\Rightarrow \mathbb{P}[X=x \mid Y=y] \cdot \mathbb{P}[Y=y] = \mathbb{P}[Y=y \mid X=x] \cdot \mathbb{P}[X=x]$$

$$\Rightarrow \mathbb{P}[X=x \mid Y=y] = \mathbb{P}[Y=y \mid X=x] \cdot \mathbb{P}[X=x] \,/\, \mathbb{P}[Y=y]$$

- Can create events using R.V.s too
  Ex: $\{X=1\}$, $\{Y=2\}$, $\{X=1 \wedge Y=2\}$, $\{X \leq x_0\}$, $\{X < x_0\}$

Event Calculus → Set of Rules

Let A, B be events,

- $\neg A$ → negation / complement of A (A didn't happen)
- $A \cup B$ → Union (A or B happened)
- $A \cap B$ → Intersection (both A & B happened)

- De-Morgan's Laws (Always hold):
  $\neg(A \cup B) = \neg A \cap \neg B$, $\neg(A \cap B) = \neg A \cup \neg B$
- Complement Rule: $\mathbb{P}[\neg A] = 1 - \mathbb{P}[A]$
- Union Rule: $\mathbb{P}[A \cup B] = \mathbb{P}[A] + \mathbb{P}[B] - \mathbb{P}[A \cap B]$
- Intersection Rule: $\mathbb{P}[A \cap B] = \mathbb{P}[A] + \mathbb{P}[B] + \mathbb{P}[\neg A \cap \neg B] - 1$

The above rules hold even for conditional probability:

- $\mathbb{P}[\neg A \mid C] = 1 - \mathbb{P}[A \mid C]$,
- $\mathbb{P}[A \cup B \mid C] = \mathbb{P}[A \mid C] + \mathbb{P}[B \mid C] - \mathbb{P}[A \cap B \mid C]$
- If $C \Rightarrow A$ then $\mathbb{P}[A \mid C] = 1$, and also
  if $C \Rightarrow \neg B$ then $\mathbb{P}[B \mid C] = 0$
- Bayes' Theorem applies to events just as well.

{ Use indicator variables

* $\mathbb{I}\{E\} = 1$ if E is true else 0

Independence of RVs

- Two RVs X and Y are said to be independent if $\forall\ x \in S_X,\ y \in S_Y$,

$$\mathbb{P}[X=x, Y=y] = \mathbb{P}[X=x] \cdot \mathbb{P}[Y=y]$$

$$\mathbb{P}[X=x \mid Y=y] = \mathbb{P}[X=x]$$

$$\mathbb{P}[Y=y \mid X=x] = \mathbb{P}[Y=y]$$

- If X and Y are independent, then we often write $X \perp\!\!\!\perp Y$. Similarly, $X \not\perp\!\!\!\perp Y$
- Conditional Independence

Conditional Independence

$X \perp\!\!\!\perp Y \mid Z \Leftrightarrow \mathbb{P}[X=x, Y=y \mid Z=z]$
$= \mathbb{P}[X=x \mid Z=z] \cdot \mathbb{P}[Y=y \mid Z=z]$

- ☆ If X, Y are independent, then it is not necessary that they continue to be independent even if conditioned on a 3rd R.V.
- ☆ Even if X, Y are not independent, it's still possible that $\exists$ 3rd R.V. Z s.t. $X \perp\!\!\!\perp Y \mid Z$
  - ↳ commonly found in graphical models

Random Vectors: A collection of R.V.s arranged in an array $X = [X_1, X_2 \dots X_d]^T$

- Can be independent or correlated
- PMF/PDF of $X \equiv$ joint PMF/PDF of $\{X_i\}_{i \in [d]}$
- Can talk about marginal / conditional prob. among $X_1 \dots X_d$.

Expectation of an R.V.

- Mean or the average value that R.V. takes and defined as

$$\mathbb{E}[X] = \sum_{x \in S_X} x \cdot \mathbb{P}[X=x] = \mathbb{E}X$$

NOT most likely value

☆ Linearity of Expectation $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$
↳ Also the Sum rule for expectation

- Scaling Rule: $Y = cX \Rightarrow \mathbb{E}[Y] = c \cdot \mathbb{E}[X]$
  ↳ constant

* $\mathbb{E}[X - \mathbb{E}X] = 0 = \mathbb{E}X - \mathbb{E}X$ $\because \mathbb{E}X$ is constant

* Law of the Unconscious Statistician (LOTUS)

- we have R.V. X w/ PMF P_X & some $g: S_X \to \mathbb{R}$ & $Y := g(X)$

then $\mathbb{E}Y = \mathbb{E}\,g(X) = \sum_{x \in S_X} g(x) \cdot \mathbb{P}[X=x]$

* If R.V.s X and Y are independent, then we have $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$

Sample Mean: Suppose R.V. X sampled again & again, say n times. Values obtained, say $x_1, x_2 \dots x_n$. we can estimate $\mathbb{E}X$ if n sufficiently large

$\hat{\mathbb{E}}X = \bar{x} = 1/n \sum_{i=1}^{n} x_i$

$\hat{\mathbb{E}}X = \arg\min_c \sum_{i=1}^{n} (x_i - c)^2$ → Empirical

&

$\mathbb{E}X = \arg\min_c \mathbb{E}[(X-c)^2]$ → Theoretical

Mode of an R.V.: Value that R.V. takes w/ highest prob.

$\text{mode}(X) := \arg\max_{x \in S_X} \mathbb{P}[X = x]$

- An R.V. may have more than one mode
- Empirical mode:

$$\text{mode}(x_1 \dots x_n) := \arg\max_{x \in S_X} \sum_{i=1}^{n} \mathbb{I}\{x_i = x\} = \arg\max_{x \in \{x_1 \dots x_n\}} \sum_{i=1}^{n} \mathbb{I}\{x_i = x\}$$

Median of an R.V.

- A value m that satisfies $\mathbb{P}[X \geq m] \geq 0.5$ as well as $\mathbb{P}[X \leq m] \geq 0.5$
- Empirically, (# of samples $\geq m$) = (# of samples $\leq m$)
- OR, arrange x_i's $\quad x_1 \leq x_2 \dots \leq x_n$

n is odd $\Rightarrow \quad m = x_{(n+1)/2}$

n is even $\Rightarrow \quad m = (x_{n/2} + x_{1+n/2})/2$

→ There may be infinitely many medians

* $m = \arg\min_c \sum_{i=1}^{n} |x_i - c|$ ← Empirical

$m = \arg\min_c \mathbb{E}[|X-c|]$ → Theoretical

Variance: How "spread out" are the values that an R.V. takes, how far from expectation

does the R.V. often take values

- For R.V. X with $\mathbb{E}X =: \mu$, its variance is,

$$\mathbb{V}[X] = \text{Var}[X] = \sigma^2 = \mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

$$\Rightarrow \mathbb{E}[X^2] \geq (\mathbb{E}[X])^2, \quad \mathbb{V}[X] \geq 0$$

Standard Deviation := $\sqrt{\text{variance}} = \sigma$

Sample Variance: n samples $x_1 \dots x_n$ of R.V. X

$\hat{\mu} = 1/n \sum x_i$, M1: $\hat{\sigma}^2 = 1/n \sum (x_i - \hat{\mu})^2$ → Two passes

M2: $\hat{s} = 1/n \sum x_i^2$, $\hat{\sigma}^2 = \hat{s} - \hat{\mu}^2$

→ Single pass / Running averages

→ Bad if $\hat{s}$ & $\hat{\mu}^2$ both very large & close (due to computers)

Covariance: Tells us how 2 R.V.s behave in tandem

- $\text{Cov}(X, Y) := \mathbb{E}[(X - \mathbb{E}X)(Y - \mathbb{E}Y)] = \mathbb{E}[XY] - \mathbb{E}X \cdot \mathbb{E}Y$
- $\text{Cov}(X, X) = \mathbb{V}[X]$ and $\text{Cov}(X, Y) <=> 0$
- Empirically, $\hat{\mu}_X$, $\hat{\mu}_Y$ precomputation

M1: $\hat{\text{Cov}}(X, Y) = 1/n \sum (x_i - \hat{\mu}_X)(y_i - \hat{\mu}_Y)$

M2: $\hat{c} = 1/n \sum x_i y_i$ and $\hat{\text{Cov}}(X, Y) = \hat{c} - \hat{\mu}_X \hat{\mu}_Y$

→ Same comments as before

Rules of Variance

$b, c \in \mathbb{R}$ constants & X, Y 2 R.V.s

Constant Rule: $\mathbb{V}[c] = 0$

Scaling Rule: $\mathbb{V}[cX] = c^2 \cdot \mathbb{V}[X]$

Shift Rule: $\mathbb{V}[c + X] = \mathbb{V}[X]$

Sum Rule: $\mathbb{V}[X \pm Y] = \mathbb{V}[X] + \mathbb{V}[Y] \pm 2\,\text{Cov}[X, Y]$

Rules of Covariance

Constant Rule: $\text{Cov}[X, c] = 0$

Symmetry Rule: $\text{Cov}[X, Y] = \text{Cov}[Y, X]$

Scaling Rule: $\text{Cov}[bX, cY] = b \cdot c \cdot \text{Cov}[X, Y]$

Shift Rule: $\text{Cov}[X + b, Y + c] = \text{Cov}[X, Y]$

☆ If X, Y independent then $\text{Cov}[X, Y] = 0$

$\Rightarrow \mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y]$

Correlation: Normalized version of covariance

$$\rho_{X,Y} = \text{Corr}[X,Y] = \frac{\text{Cov}[X,Y]}{\sqrt{\mathbb{V}[X]\cdot\mathbb{V}[Y]}} \in [-1,1]$$

- $\rho_{X,Y} = 0 \Rightarrow$ uncorrelated
- Independent R.V.s $\Rightarrow$ Uncorrelated $\not\Leftarrow$
- Empirically, $\hat{\rho}_{X,Y} = \hat{\text{Cov}}(X,Y) / \sqrt{\hat{\sigma}_X^2 \hat{\sigma}_Y^2}$
- $\rho_{X,Y} < 0 \rightarrow$ Typically, X takes larger value than its mean when Y takes smaller value than its mean & vice-versa
  $\rho_{X,Y} > 0 \rightarrow$ ... larger ... larger ...
  $\rho_{X,Y} = 0 \rightarrow$ Typically, even if X takes a value larger than its own mean, Y may take smaller or larger values than its mean

Conditional Formatting

- $[\cdot|\cdot]$ used to express how one quantity behaves when some other quantities are fixed
- "other" quantities could be R.V.s themselves or constants.

Ex: $\mathbb{P}$ of misclassifying $(x,y) \sim \mathcal{D}$ if model w used, i.e., $\mathbb{P}[y \cdot w^T x < 0 \mid w]$

- Conditional Expectation $\mathbb{E}[X \mid Y = y_0] := \sum_{x \in S_X} x \cdot \mathbb{P}[X = x \mid Y = y_0]$
- Conditional Variance $\mathbb{V}[X \mid Y = y_0] := \mathbb{E}[(X-\mu)^2 \mid Y = y_0]$ $\quad \mu = \mathbb{E}[X \mid Y = y_0]$
- Conditional Covariance

$$\text{Cov}[X,Y \mid Z = z_0] = \mathbb{E}[(X-\mu_X)(Y-\mu_Y) \mid Z = z_0]$$
$$= \mathbb{E}[XY \mid Z = z_0] - \mu_X \mu_Y, \quad \mu_X = \mathbb{E}[X \mid Z = z_0]$$

- Conditional Mode $\text{mode}[X \mid Y = y_0] = \underset{x \in S_X}{\arg\max}\ \mathbb{P}[X = x \mid Y = y_0]$

These defs don't require independence.

- Rules of expectation hold if all expressions are systematically conditioned

$$\mathbb{E}[X+Y \mid Z=z_0] = \mathbb{E}[X \mid Z=z_0] + \mathbb{E}[Y \mid Z=z_0],$$
$$\mathbb{E}[c\cdot X \mid Z=z_0] = c\cdot \mathbb{E}[X \mid Z=z_0],$$
$$\mathbb{E}[g(X) \mid Z=z_0] = \sum_{x\in S_X} g(x)\cdot \mathbb{P}[X=x \mid Z=z_0],$$
If $X \perp\!\!\!\perp Y \mid Z$ then $\mathbb{E}[X\cdot Y \mid Z=z_0] = \mathbb{E}[X \mid Z=z_0]\cdot \mathbb{E}[Y \mid Z=z_0]$

- **Expectation** of an **R.V.**

$$\mathbb{E}X = [\mathbb{E}X_1, \mathbb{E}X_2, \dots, \mathbb{E}X_d]^T$$

$\mathbb{E}[X+Y] = \mathbb{E}X + \mathbb{E}Y$, $\mathbb{E}[c\cdot X] = c\cdot \mathbb{E}[X]$, $c\in\mathbb{R}$ constant

$a\in\mathbb{R}^d$ constant, $\mathbb{E}[a^T X] = a^T \mathbb{E}[X]$

$A\in\mathbb{R}^{n\times d}$ constant, $\mathbb{E}[AX] = A\,\mathbb{E}[X]$

- Mode easy to define. Median not so easy:

Definition 1: $\text{med}(X) = [\text{med}(X_1) \dots \text{med}(X_d)]^T$

D2: minimizer of absolute distance (L1 norm)

$$\text{med}(X) = \arg\min_{v\in\mathbb{R}^d} \mathbb{E}[\|X-v\|_1]$$

- $$\text{Cov}(X) = \begin{bmatrix} \mathbb{V}X_1 & \text{Cov}(X_1,X_2) & \cdots & \text{Cov}(X_1,X_d) \\ \text{Cov}(X_2,X_1) & \mathbb{V}X_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \text{Cov}(X_d,X_1) & \cdots & & \mathbb{V}X_d \end{bmatrix} = \mathbb{E}[(X-\mu)(X-\mu)^T] = \mathbb{E}[XX^T] - \mu\mu^T, \quad \mu = \mathbb{E}[X]$$

$\text{Cov}(cX) = c^2\,\text{Cov}(X)$, X is a vector of R.V.s

- If $X\in\mathbb{R}^m$, $Y\in\mathbb{R}^n$ be two R. vectors, then

$\mathbb{R}^{m\times n} \ni \text{Cov}(X,Y) = \mathbb{E}[(X-\mu_X)(Y-\mu_Y)^T] = \mathbb{E}[XY^T] - \mu_X\mu_Y^T$

$a\in\mathbb{R}^d$ constant, then $\mathbb{V}[a^T X] = a^T \text{Cov}[X]\, a$

$A\in\mathbb{R}^{n\times d}$ constant, then $\text{Cov}[AX] = A\,\text{Cov}[X]\,A^T \in \mathbb{R}^{n\times n}$

Continuous R.V.s

- R.V. that can take infinitely many possible values that are not discrete but cts., i.e., support is $\mathbb{R}$ or some subset of $\mathbb{R}$
- Instead of PMF, use **PDF** (prob. dist. func.)
- $f = \text{PDF}: S_X \to \mathbb{R}$, f can take values > 1

as well but must not be -ve

- For any $x \in S_X$, $f_X(x)$ tells us how likely is X to take a value around x. ie, for some $\delta > 0$, we have $\mathbb{P}[X \in [x-\delta, x+\delta]] \approx f_X(x) \cdot 2\delta$

$\int_{x-\delta}^{x+\delta} f_X(t)\,dt$: Exact formula ←

More generally, we have,

$$\mathbb{P}[X \in [a,b] \subseteq S_X] = \int_a^b f_X(t)\,dt$$

- PDF f of an R.V. X satisfies

$f_X(x) \geq 0 \quad \forall\ x \in S_X$ & $\int_{S_X} f_X(t)\,dt = 1$

Expectation : $\mathbb{E}X := \int_{S_X} t \cdot f_X(t)\,dt$

LOTUS : $\mathbb{E}[g(X)] = \int_{S_X} g(t)\, f_X(t)\,dt$

Variance : $\mathbb{V}X := \int_{S_X} (t - \mathbb{E}X)^2 f_X(t)\,dt = \int_{S_X} t^2 \cdot f_X(t)\,dt - (\mathbb{E}X)^2$

Joint PDFs : $f_{X,Y}$, $[a,b] \subseteq S_X$, $[c,d] \subseteq S_Y$

$$\mathbb{P}[X \in [a,b],\ Y \in [c,d]] = \int_a^b \int_c^d f_{X,Y}(s,t)\,dt\,ds$$

↳ Make sense even if one cts. & other discrete

Marginal PDF : $f_X(x) = \int_{S_Y} f_{X,Y}(x,t)\,dt$

- Conditional probabilities also make sense

$\mathbb{P}[X \in [a,b] \mid Y = r]$ can be found using limits or Radon-Nikodym derivative

in general zero ←

- Conditional expectations, (co)variances also defined
- All rules of Probability, Expectation, (Co)Variance continue to hold

Probability Distributions

Bernoulli Distribution

- Support : $\{0,1\}$ ⇝ Useful in binary classification
- PMF of R.V. Y uniquely specified by

$$\mathbb{P}[Y=1] = p \iff \mathbb{P}[Y=0] = 1-p$$

Bias / Success probability

- Mean : p, Mode : 1 if $p > 0.5$, 0 if $p < 0.5$, $\{0,1\}$ if

Variance: $p(1-p)$

Rademacher Distribution

- Support: $\{-1, 1\}$

X distributed over Bern $\Leftrightarrow$ $2X-1$ over Rade

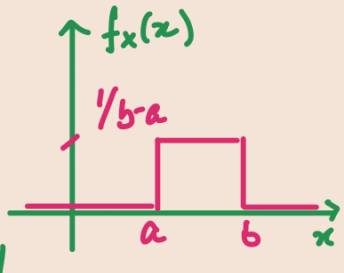
Y dist. over Rade $\Leftrightarrow$ $(Y+1)/2$ over Bern

- $\mathbb{P}[Y=1] = p$, Mean $= 2p-1$, Mode: 1 if $p > 0.5$, -1 if $p < 0.5$, $\{-1,1\}$ $p=0.5$

Variance: $4 \cdot p \cdot (1-p)$

Uniform Distribution

- X cts. R.V. w/ $S_X = [a, b] \subseteq \mathbb{R}$
- PDF $f_X(x) = \begin{cases} 1/b-a & , x \in [a,b] \\ 0 & , \text{otherwise} \end{cases}$



$f_X(x) \geq 0$ if $x \in S_X$ & $\int_{S_X} f(t)dt = 1$

- $\mathbb{E}[X] = (a+b)/2$; $\mathbb{V}[X] = (b-a)^2/12$ ↑ as $(b-a)$ ↑
- UNIF($[a,b]$) used to denote uniform dist.

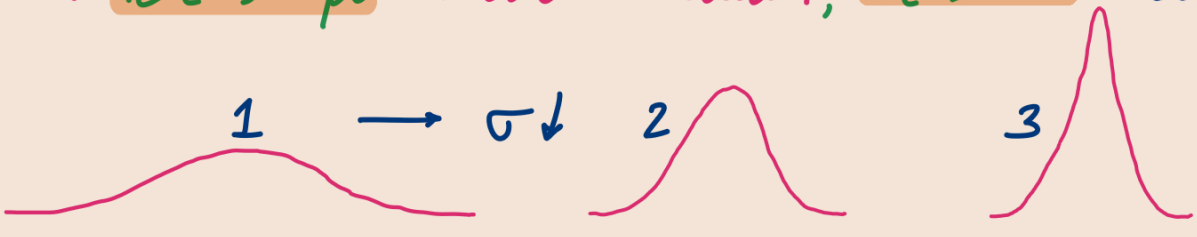
Gaussian (aka Normal) Distributions

- $S_X = \mathbb{R}$

$$f_X[x \mid \mu, \sigma^2] = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

- Takes two numbers to describe
- μ : $\in \mathbb{R}$; σ^2 : non-negative real number
- Often written as $\mathcal{N}_X(x; \mu, \sigma^2)$ or $\mathcal{N}(x; \mu, \sigma^2)$ (Conditioned to constants)

Mean $= \mathbb{E}[X] = \mu$ = Mode = Median, $\mathbb{V}[X] = \sigma^2$



- $Z = cX \longrightarrow \mathcal{N}_Z(\cdot; c \cdot \mu_X, c^2 \cdot \sigma_X^2)$
- $W = X + Y \longrightarrow \mathcal{N}_W(\cdot; \mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$

X, Y Gaussian & independent

$\mathbb{P}[x \geq \mu_x + t \cdot \sigma_x] = \mathbb{P}[x \leq \mu_x - t \cdot \sigma_x] \leq e^{-t^2/2}$

- As $\sigma_x \downarrow$, R.V. gets more concentrated around its mean

☆ **68-95-99.7 rule**

$\mathbb{P}[|x - \mu_x| \leq \sigma_x] \approx 0.68$, $[\leq 2\sigma_x] \approx .95$, $[\leq 3\sigma_x] \approx 0.997$

Gaussian Random Vector

- Need only the mean $\mu \in \mathbb{R}^d$ and the covariance $\Sigma \in \mathbb{R}^{d \times d}$ to be specified $\mathcal{N}(\mu, \Sigma)$

$$\mathbb{P}[x \mid \mu, \Sigma] = \frac{e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}}{\sqrt{(2\pi)^d |\Sigma|}}$$

* Special case $\mu = 0$, $\Sigma = I_d$ called standard Gaussian / Normal dist,

$$\mathbb{P}[x \mid 0, I_d] = \frac{e^{-\|x\|_2^2/2}}{\sqrt{(2\pi)^d}} = \prod_{i=1}^{d} \frac{e^{-x_i^2/2}}{\sqrt{2\pi}}$$

However, $e^{-x_i^2/2}/\sqrt{2\pi}$ is simply $\mathcal{N}(0,1)$, ie,

$\mathbb{P}[x_1 \dots x_d \mid 0, I] = \prod \mathbb{P}[x_i \mid 0, 1]$

$\Rightarrow$ All d co-ordinates of a standard Gaussian R.V. are independent

- Given a Gaussian vector $\mathbb{R}^d \ni x \sim \mathcal{N}(\mu, \Sigma)$:
  - ↳ Every coordinate of x is a Gaussian R.V.
    - ↳ Holds true even if conditioned on all other co-ords.
  - Need not be independent
- Consider $j \in [d]$
  - x_j distributed as $\mathcal{N}(\mu_j, \Sigma_{jj})$
  - Given $x_k = v_k \ \forall \ k \neq j$, x_j still Gaussian, expression complicated
- If $a \in \mathbb{R}^d$ const, $\mathbb{R} \ni a^T x \sim \mathcal{N}(\mu^T a, a^T \Sigma a)$
- If $A \in \mathbb{R}^{n \times d}$ const, $\mathbb{R}^n \ni Ax \sim \mathcal{N}(A\mu, A\Sigma A^T)$

Laplacian Distribution

- Concentrate much more strongly

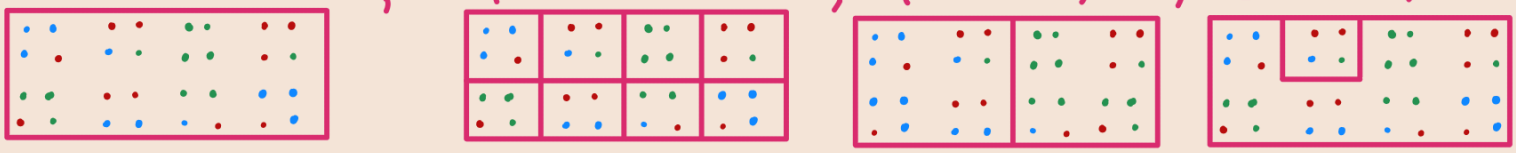
around mean than Gaussian

$$f_x[x \mid \mu, \sigma] = \frac{e^{-|x-\mu|/\sigma}}{2\sigma}$$

- Mean = Mode = Median = μ, Variance = $2\sigma^2$
- $Y = aX + b$ also Laplacian w/ $\mu_y = a \cdot \mu_x + b$, $\sigma_y = a \cdot \sigma_x$

Uncertainty

- Start State, Goal States, Good Ques, Bad Ques.



→ Asking trivially true questions: Useless
Asking rarely true questions: Useless
- Must ask questions that eliminate wrong answers quickly

Entropy (H): If we have a set of n words, then that set has an entropy of $\log_2 n$
- Larger sets have larger entropy; 1 word: 0 entropy
- More generally, if there is a set S of n elements of C types with n_c elements of type c, then its entropy:

$$H(S) := -\sum_{c \in [C]} \frac{n_c}{n} \log_2\left(\frac{n_c}{n}\right) = -\sum_{c \in [C]} p_c \log_2 p_c$$

where p_c is the proportion of elements of type c

* A pure set, ex: $p = [0\,0 \ldots 1 \ldots 0\,0]$ has 0 entropy whereas a set with same number of elements of each class i.e., $p = [1/C \;\; 1/C \ldots 1/C]$: $\log_2 C$ entropy

What is a good question? — Depends on the application

- ID3 (iterative dichotomizer 3)
  - ↳ reduces confusion/entropy the most

- Suppose question splits S into K subsets $S_1, \dots, S_K$,

$$\left\{ \begin{array}{ll} S_i \cap S_j = \phi \ \ \forall\ i \neq j, & \cup_{k \in [K]} S_k = S, \\ n_k := |S_k|, & \sum_{k \in [K]} n_k = n := |S| \end{array} \right\}$$

Then the entropy of the collection of sets is: $\sum_{k \in [K]} n_k/n \cdot H(S_k)$

- Can be interpreted as "average" or "weighted" entropy

Ex: $\{4096 \text{ words}\} \xrightarrow[\text{Split}]{\text{Question}} \{1024, 1024, 1024, 1024\}$

Old entropy: $\log_2 4096 = 12$

New entropy: $1024/4096 \log_2 1024 + \dots = 4 \cdot 1024/4096 \log_2 1024 = 10$

We gained $12 - 10 = 2$ bits of information

* Information gained this way can be added up. Can be proved mathematically that the chosen definition of entropy is the only one satisfying 3 intuitive requirements. Suppose an event occurs with prob. p and we wish to measure the information from that event's occurence, say $I(p)$, s.t:

1) Sure event conveys no information:
2) More common the event, less info it conveys: $p \uparrow \Rightarrow I(p) \downarrow$
3) Info conveyed by two indep. events adds up: $I(p_1 \cdot p_2) = I(p_1) + I(p_2)$

$$\Rightarrow I(p) = -\log_b p \Rightarrow H(S) = \sum_{c \in [C]} p_c \cdot I(p_c)$$

$b = 2$: bits (binary digits), $b = e$: nits (natural digits)

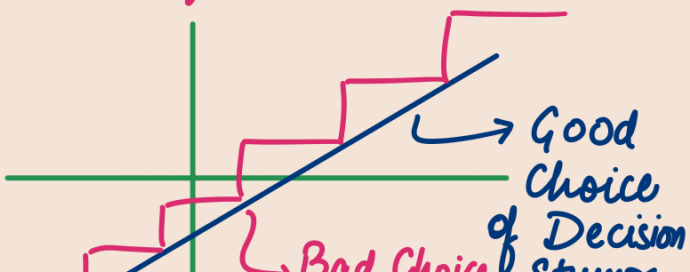
$b = 10$: dits (decimal digits / hartleys)

ID3 Algorithm

- With T as set of all train points, create a root node r, and call train(r, T)
  train(node n, set S)
  - If S is sufficiently pure or small, make n a leaf, decide a simple leaf action (e.g. most popular class, label popularity vector, etc.) and return
  - Else, out of available choices, choose the splitting criteria (eg. a single feature) that causes maximum information gain / reduces entropy the most
  - Split S along that criteria to get $S_1, \ldots, S_k$ partition of S (e.g. if that feature takes k distinct values)
  - Create K child nodes c_k, $k \in [K]$ and call train(c_k, S_k) $\forall$ $k \in [K]$
- ID3 does not ensure a balanced tree but the balance is decent.
- Several augmentations to this algorithm: C4.5, C5.0 allow real-valued features, missing features, boosting, etc.

Careful Use of DTs

- can be tweaked to give very high training accuracies — can badly overfit if tree too large
- Choice of decision stumps is critical:
  PUF: single linear model
  DT will struggle and



eventually overfit

Probabilistic ML techniques, given a data point, do not output a single label, they instead **output** a **distribution** over all possible labels

- For binary classification, output a PMF over $\{-1, 1\}$
  For multiclassification, output a PMF over $\{1, 2, \ldots, C\}$
  For regression, output a PDF over $\mathbb{R}$
- The probability of a label indicates how likely does the ML model think that label is the correct one for that data point.
- The algo. is allowed to output a possibly different PMF / PDF for every data point. However, the support is always the set of all possible labels (even very unlikely labels)

Prob. ML for Classification

- Say we have learnt a PML model w which, for a data point x, gives us a PMF $P[Y \mid x, w]$ over the set of all possible labels, say $\mathcal{Y}$
- $\mathcal{Y} = \{-1, +1\}$ for bin, $\mathcal{Y} = [C]$ for multiclassification
- **May use the mode**
  if single label predicted: $\hat{y} = \arg\max_{y \in \mathcal{Y}} P[Y = y \mid x, w]$
- May use the median / mean as well — Bayesian ML exploits this
- **Use** $P[Y = \hat{y} \mid x, w]$ to find out if the ML model is **confident** about its prediction or confused.
  {**Confidence ≠ Correctness**}
- In many ML applications (eg. active learning), if we find that the model is unsure, we can switch to another model or ask a human to step in

- May use variance of $\mathbb{P}[y|x,w]$ as well
  (low variance: very confident; high-var: less confident)

Prob. Binary Class.

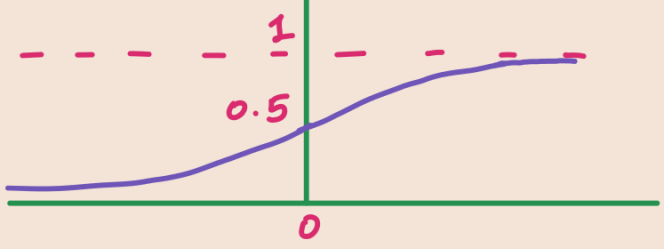
- Find a way to map every data point x to a Rademacher distribution, ie, map each x to a prob $p_x \in [0,1]$
  → Gives us a PMF $[1-p_x, p_x]$ → $\mathbb{P}[-1|x] = 1-p_x$, $\mathbb{P}[+1|x] = p_x$
- If using mode predictor, $\hat{y}$, then this PMF will give us the correct label only if:
  - When true label of x is $+1$, $p_x > 1-p_x \Leftrightarrow p_x > 0.5$
  - " " " " " " -1, $p_x < 1-p_x \Leftrightarrow p_x < 0.5$
  - $p_x = 0.5 \Rightarrow$ model is completely confused
    ↳ x is on the decision boundary
- As usual, we want a healthy margin:
  True label of x is $+1$, we want $p_x \gg 0.5 \Rightarrow p_x \approx 1$
  " " " " " -1, " " $p_x \ll 0.5 \Rightarrow p_x \approx 0$

Mapping feature vectors of x to $p_x \in [0,1]$
could treat it as a regression problem
∵ $p_x \in \mathbb{R}$ ⇝ Will need to modify the training set (change all -1 labels to 0)

- Could use DT
- Using linear models presents a challenge
  - If we learn a linear model w using regression, $w^T x < 0$ / $w^T x > 1$ may happen
- DT won't run into this problem ∵ p_x always in $[0,1]$ {Average of 0s & 1s $\in [0,1]$ too}
- Can use logistic regression in linear models for PML

Sigmoid Function

$$\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} = 1-\sigma(-t)$$



- Learn a linear model w (possibly with a hidden/explicit bias) & map $x \to \sigma(w^T x)$
  Valid PMF ← Always gives a value in $[0,1]$
- $\sigma(t) > 0.5 \iff t > 0$; $\sigma(t) < 0.5 \iff t < 0$;
  $\sigma(t \to \infty) \to 1$; $\sigma(t \to -\infty) \to 0$
  $\Rightarrow$ Sigmoidal map will predict $p_x \approx 1$ if $w^T x \gg 0$ and $p_x \approx 0$ if $w^T x \ll 0$
- Several other such wrapper / quashing / link / activation functions like tanh, ramp, ReLU
- How to learn such a w?

Likelihood

- Suppose we have a model w (assume hidden b)
- Given a data point (x^t, y^t), $x^t \in \mathbb{R}^d$, $y^t \in \{-1, +1\}$, the use of a sigmoidal map gives us a Rademacher PMF $\mathbb{P}[y \mid x^t, w]$
- The probability that this PMF gives to the correct label, ie, $\mathbb{P}[y^t \mid x^t, w]$ is called the 'likelihood' of this model to this data point.
- Easy to show that $\mathbb{P}[y^t \mid x^t, w] = \sigma(y^t \cdot w^T x^t)$
- If we have several data points $(x^1, y^1) \dots (x^n, y^n)$ then we define the likelihood of w w.r.t. entire dataset as $\mathbb{P}[y^1, \dots, y^n \mid x^1, \dots x^n, w]$
  Usually assumed that the data points are independent:
  $\mathbb{P}[y^1, \dots, y^n \mid x^1, \dots, x^n, w] = \prod_{i=1}^{n} \mathbb{P}[y^i \mid x^i, w] = \prod_{i=1}^{n} \sigma(y^i \cdot w^T x^i)$
  → Not always independent
- The expression $\mathbb{P}[y^i \mid x^i, w]$ tells us if the model w thinks the label y^i is a very likely

label given the feature vector x^i

- Similarly $\mathbb{P}[y^1 \dots y^n \mid x^1 \dots x^n, w]$
- If we trust the training data then we should learn a model w that considers training labels to be very likely.

Maximum Likelihood Estimator (MLE)

The model that gives highest likelihood to observe labels

$$\hat{w}_{MLE} = \arg\max_{w \in \mathbb{R}^d} \prod_{i=1}^{n} \mathbb{P}[y^i \mid x^i, w]$$

- If we're learning the model using a sigmoidal map, then product of several such values $([0,1])$ can be extremely small.
  ↳ Take log & exploit $\max_w f(w) = \max_w \ln(f(w))$
- Working w/ products can be numerically unstable
- $\hat{w}_{MLE} = \arg\max_{w \in \mathbb{R}^d} \ln\left(\prod_{i=1}^{n} \sigma(y^i \cdot w^T x^i)\right)$
  $= \arg\min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \underbrace{\ln\left(1 + e^{-y^i \cdot w^T x^i}\right)}_{\text{Also called negative log-likelihood}}$

* Logistic loss function pops out automatically when learning a model that maximizes likelihood

Prob. Multiclassification

- Suppose C classes, then for every data point, we would like to output a PMF over the support $[C]$

→ Assign a +ve score to all classes & normalize
To convert any score to +ve — exponentiate

- Learn C models $w^1, \dots, w^C$. given a point (x^t, y^t). $x^t \in \mathbb{R}^d$, $y^t \in [C]$. assign a +ve score per class $\eta_c = e^{\langle w^c, x^t \rangle}$; then normalize to obtain the PMF $\mathbb{P}[y \mid x^t, \{w^c\}] = \eta_y / \sum \eta_c$ for any $y \in [C]$

- **Likelihood:** $\mathbb{P}[y^t \mid x^t, \{w^c\}] = \eta_{y^t} / \sum \eta_c$
- **Log-likelihood:** $\ln(\eta_{y^t} / \sum \eta_c)$
- Distributions for $C > 2$ called Multinoulli / Categorical distributions

Softmax Regression

- If we now want to learn the MLE, we would have to find

$$\{\hat{w}^1_{MLE} \dots \hat{w}^C_{MLE}\} = \arg\max_{w^1 \dots w^C \in \mathbb{R}^d} \prod_{i=1}^{n} \mathbb{P}[y^i \mid x^i, w]$$

$$= \arg\max_{w^1 \dots w^C \in \mathbb{R}^d} \prod_{i=1}^{n} \eta^i_{y^t} / \sum_{c=1}^{C} \eta^i_c, \quad \eta^i_c = e^{\langle w^c, x^i \rangle}$$

$$= \arg\min_{w^1 \dots w^C \in \mathbb{R}^d} \sum_{i=1}^{n} \underbrace{-\ln(\eta^i_{y^t} / \sum_{c=1}^{C} \eta^i_c)}$$

Cross-entropy / softmax loss function

- But $\exists$ other ways to do prob. multiclass. as well

General Recipe for MLE Algorithms

- Label set $\mathcal{Y}$, find a way to map data features x to PMFs $\mathbb{P}[\cdot \mid x, m]$ w/ support $\mathcal{Y}$ (parameters of the model)
- $\mathbb{P}[\cdot \mid x, m] \longrightarrow$ Likelihood function
- $-\ln \mathbb{P}[\cdot \mid x, m] \longrightarrow$ Log-likelihood function
- Given data $\{(x^i, y^i)\}_{i=1}^n$, find model parameters that maximize likelihood function, ie, think that the training labels are very likely

$$\hat{m}_{MLE} = \arg\min_{m} \sum_{i=1}^{n} -\ln \mathbb{P}[y^i \mid x^i, m]$$