***Lecture 1***

***A black and white math symbol

Description automatically generated****we would like to find the classifier with the lowest* ***Expected Error ()*** *out of the learned model class ;( ) naïve – compute , but it’s not possible because is not known. Therefore, we’ll compute the classifier with the lowest* ***Empirical Error(ERM) ():***

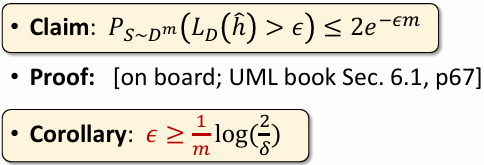
*In conclusion, the goal is but the means are*

*For any separable D there is some with .*

***Example for***

* *– number of intervals = degree of polynomial*

***Lecture 2***

**

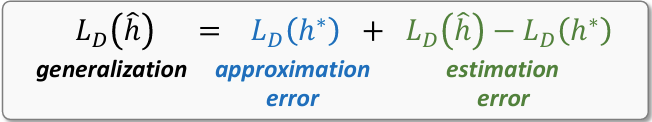


*The Bias-Complexity Tradeoff:*

* *LargerLess Biaslower approx. errorbetter generalization*
* *Smaller worse estimation higher error worse generalization*

*Not all model classes are learnable (polynomian convergence)*

*Proof on page 18*

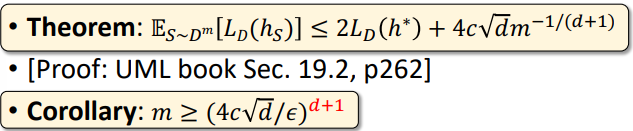
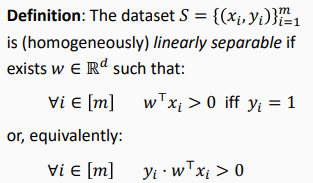
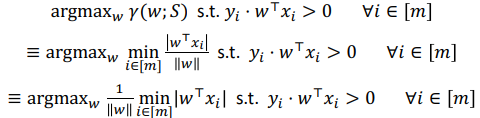
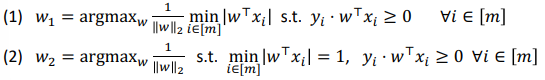
******

*If we restrict model complexity – the ExpectedError and the ERM converge as data size*

***Lecture 3***

*Learning approaches:*

* *Similarity Based(kNN);* 
  + *A black and white text

    Description automatically generatedHighly memory-intensive*
  + *Local empirical estimate of conditional probability:*
  + *Good if changes smoothly in .*
  + *Small k – more noise(large variance); large k – large bias.*
  + **
* *Rule-Based (Decision Trees);* 
  + *Purity: ;*
  + *Entropy:*
  + *largest* ***InformationGain*** *splits*
  + ***What goes inside H? tutorial 3, p13***
  + *Need to cap the depth as to not overfit the data, can also control the minimal node size for each split*
* *Linear Models (SVM)*
  + *= project on ; basically halves the space*
  + *Solving ERM for linear classifiers is NP-hard*
  + *Optimization template:*
  + *****Margin*** *– the distance to* ***w*** *from the closest point in dataset*
  + *hyperplanes are scale-invariant*

**

* + *hard SVM objective:*
  + ***increasing margin ≡ reducing norm***
* ***pages 47-48***
* *the training acc stays the same, if the training acc improves.*
* *looks for max margin classifier*

*****Lecture 4***

***soft SVM:*** *penalize violations linearly*

*Loss – Hinge, logistic, exponential, ramp…; Regularization - … ;*

***Goal****: minimize 0/1 loss - difficult;* ***alternative****: min’ hinge loss – convex & tractable*

*Tractable - the problem can be solved using algorithms that run in polynomial time.*

* *Weakness – outliers*

*– Regularization, penalizes w’s with large norm’s*

* *Low norm w’s provide better generalization*
* *Tuning can control overfitting*
* *Smaller overfittinglower complexity*
* *In regularization, non smooth, very smooth. Shrinks coefficient but doesn’t set them to 0 (unlike )*

***Kernels***

***Feature mapping:***

* *Polynomial:*
* *Radial:*
* *Dimension d – original number of features*
* *Degree k – the maximal degree of the polynomial*
* *meaning that d grows FAST*

*Choose , apply it to all features in dataset, use SVM to learn the modified data – the result is a linear classifier that behaves like a non-linear classifier on the original data.*

*Constrained Optimization* ***– Duality Problem***

* *Problem:*
* *Solve**; ( )*
* ***General Case:***
* *If we derive the general case:*

***The Kernel Trick (useful only if it can be computed efficiently)***

* *–* ***find***  *to validate the kernel or* ***use kernel algebra***
* *Train time:*
* *Test time****:***
* ***Kernel algebra:*** *8 composition rules(p52)*

***Lecture 5***

***PAC***

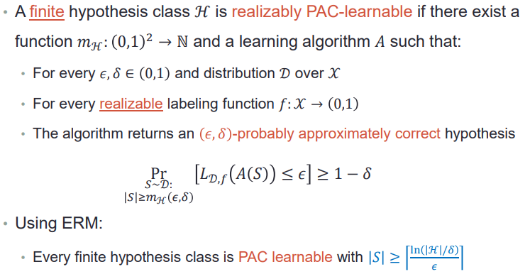
* ***Realizability:***
* *PAC: PAC-learnable*
* *Hoeffding concentration bound:*

***Agnostic PAC learning***

***VC Dimension –*** *the largest set on which is possible(largest set that shatters)*

* *measures the capacity of model classes to express binary patterns*
* *worst case*
* *There exist a set of* ***m*** *points that any labelling of them can be shattered by* 
  + *ℋ shatters 𝐶 ∀𝑦1, … , 𝑦 𝐶 ∈ 𝒴: ∃ℎ ∈ ℋ: ∀𝑥𝑖 ∈ 𝐶 :*
* *For any set of* ***m+1*** *points, there exists labelling that can’t shatter*
* *Rule of thumb: number of parameters = VC Dimension*

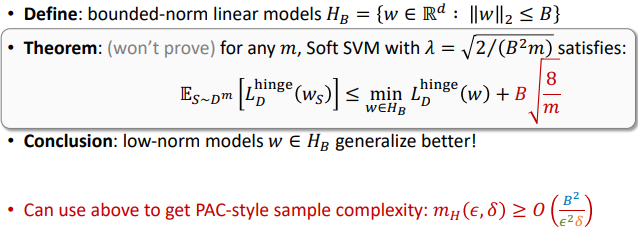
*If < ∞, then is:*

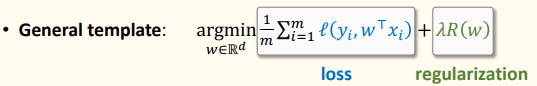
1. *PAC-learnable with*
2. * Agnostic PAC-learnable with*

***Lecture 6***

***Expected Error noise Variance***

* ***Bias –*** *how well the avg model fits the avg data*
* ***Variance*** *– how learned models vary from the average model*
* *More samples lower variance.*
* *Complex models higher variance(avg separator is good) + less bias*
* *Lower complex models higher bias + lower variance*

**

**

***Cross validation*** *– used to ensure that training and tuning are independent, choose a set of , split train-data into k-fold’s % validation and % train, run each fold on the different and avg each lambda from a different fold, when done – choose the avg lambda that has a min error.*

*gen err’ exp’ - val’ val’ – train train*

* + *exp’ - val’: might not be bounded*
  + *val’ – train: if too large - overfits*
  + *train: if too large - underfits*

***Lecture 7***

*Hinge Loss has unnecessary large penalties ->* ***ramp loss***

*Goal:*

***Gradient Descent:*** *, repeat until convergence( )*

* *be a ball of radius around . Then is a local minimum of if:*
* *guaranteed to converge to a local minimum with a small enough learning rate*
* *steps against gradient direction*
* *uses all data*

***Stopping criteria:***

*C is a* ***convex*** *set if*

*is convex func if*

* ***intersection, sum, positive scaling, max*** *of convex is also convex*
* ***linear*** *functions are convex*
* ***composition*** *of* ***linear & convex*** *is* ***convex***
* *twice differentiable is convex iff*
* *-norm squared is convex*
* *is convex*
* ***GD Convergence Rates***
  + ***L-Lipschitz:***
  + ***-Smooth:***
  + ***-Strongly convex:***

***Stochastic Gradient Descent*** *– because regular Gradient Descent is* ***costly***

* *Sample a random mini-batch, compute avg gradient on it, make a SGD step*
* *SGD and GD directions in expectation are* ***equal***
* ***Momentum –*** *increases in dimesions where the grad preservs the direction and decreases in dimensions where the gradient direction varies*

***Lecture 8***

***Feature Scaling***

* ***min-max:***
* ***Standard:***
* *Normalize test using train statistics*

***Puzzle #1-#8***

***Covariate shift (Beyond iid)***

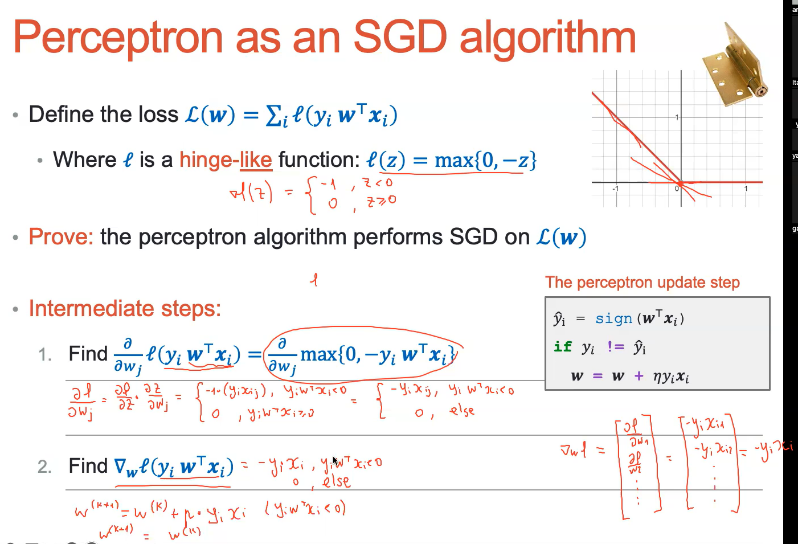
* *– observed distribution*
* *– deploy distribution*
* ***Goal:*** *low expected error on p’ (deploy)*
* ***Problem:*** *have labeled data only from p, not from p’*
* ***Solution:*** *re-weight examples in loss to “mimic” p’*

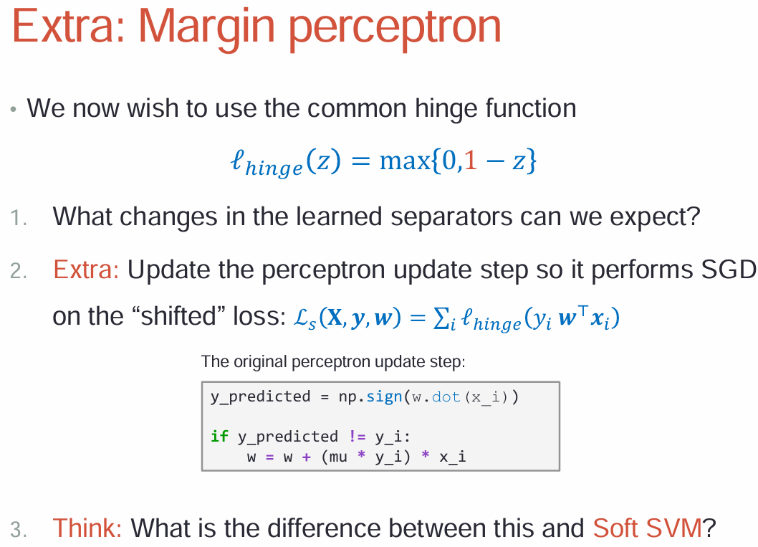
***Perceptron***

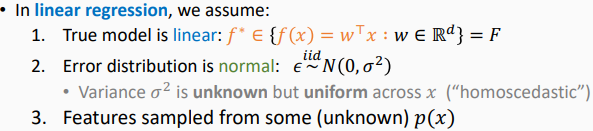
* *If data is not separable, we must limit it in the number of iterations*
* *Can be non-homogeneous(just need to move it into d+1 dimension) : ; b shouldn’t be regularized*

*Update b this way:*

* *(assuming the direction of the normal is classifying positives)*
* *Larger results in a larger degree rotation(capped at 180)*
* *Misclassification of (-1)’s result in the classifier rotating clockwise away from the label.*
* *Misclass’ of 1’s results in the classifier rotating counterclockwise towards the label.*
* *stops when there is a full iteration without misclassifications*
* *will perform the same number of iterations, and will converge to a vector that points to the same direction*
* ***Does not*** *only change the magnitude of the vector in each iteration, also changes the direction depending on the size of .*
* *Converges after making at most mistakes for any*

*Sub-gradient is the slope of* ***any*** *tangent to the function at a given point*

**

*****Lecture 9***

*two ways to interpret*

* *given w, what is the probability of observing y given x?*
* *given (x,y) , what it is the likelihood it was generated by w?*

***Likelihood:***

***MLE:***

***Prediction:***

***Residual*** *-*

***Loss:***

***Regression****:*

* *–* ***convex in w***
* *VC theory does not apply*
* *Min-norm as max-margin does not apply*

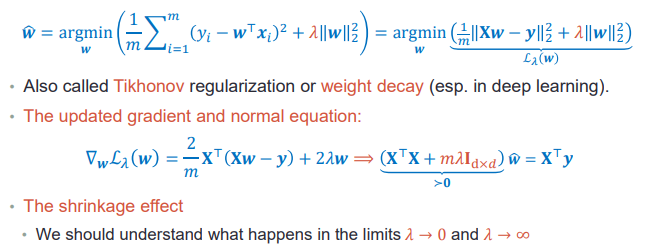
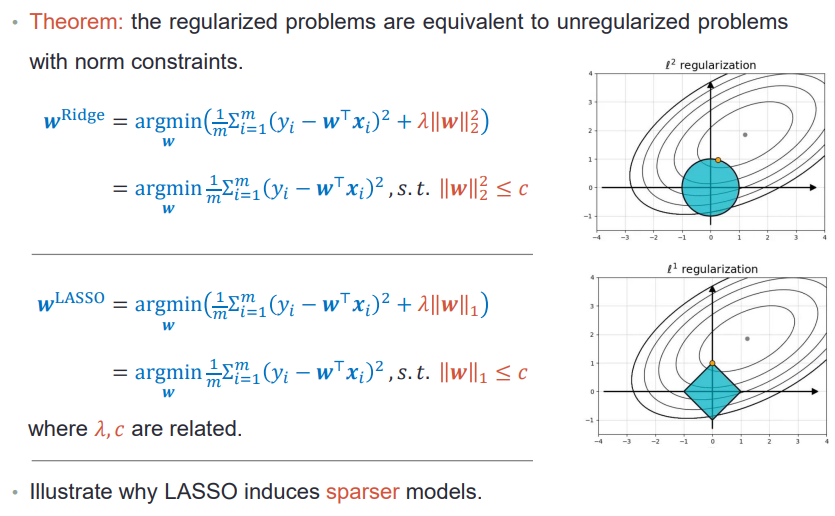
***Vandermonde matrix as a polynomial mapping:***

***Ridge:***

* *Causes weight decay*
* *Loss is convex and differentiable*

***Lasso***

* *Induces sparse solutions(few non-zero entries) – causes “variable selection”*
* *Loss remains convex but no longer differentiable*
* *Could run subgradient descent, but more suitable algorithms exist.*

******

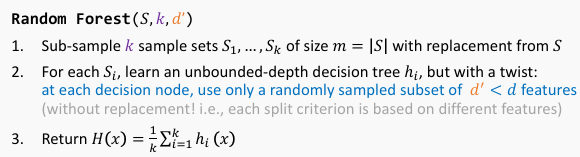
***Lecture 10***

***Ensemble methods –*** *aggregate many simple models into a single powerful model*

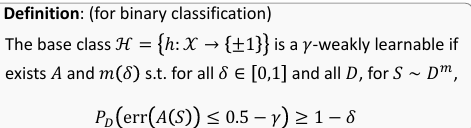
***Bagging = Bootstrap + Aggregating:*** *ensemble approach aimed at reducing variance(without reducing complexity)*

* *Reduces variance by averaging(law or large numbers: stays the same, is divided by*
* *Learn independently*

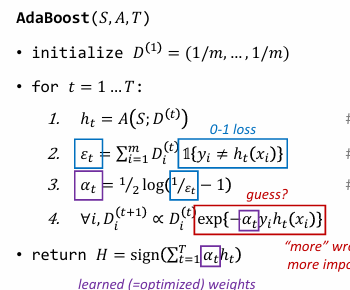
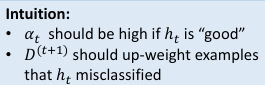
***Bootstrap*** *- sample uniformly with replacement*

* ***0.632 bootstrapping -*** 
  + *******Meaning each includes of the data regardless of*

***Random Forests***

**

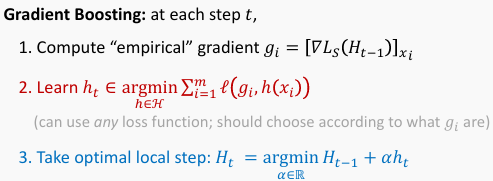
***Boosting –*** *aimed at reducing bias*

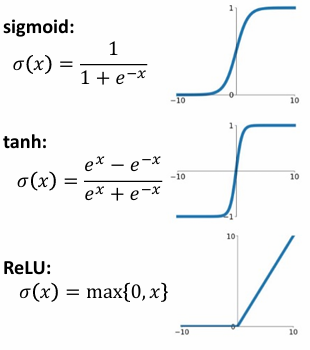
* *Works well for weak base learners(works better than random)*
* *Uses weak lerner to form a strong learner*
* *Trains each on a ‘reweighted’ sample )sequentially)*
* *******Generalizes* ***bagging***

***AdaBoost*** *- Adaptive Boosting*

* *Training error after iterations is bounded by*

* *Greedily optimizes exponential loss.*

***Gradient Boosting***

*****Lecture 11***

***Neural nets***

* *Learn end-to-end*
* ***Back Propagation***

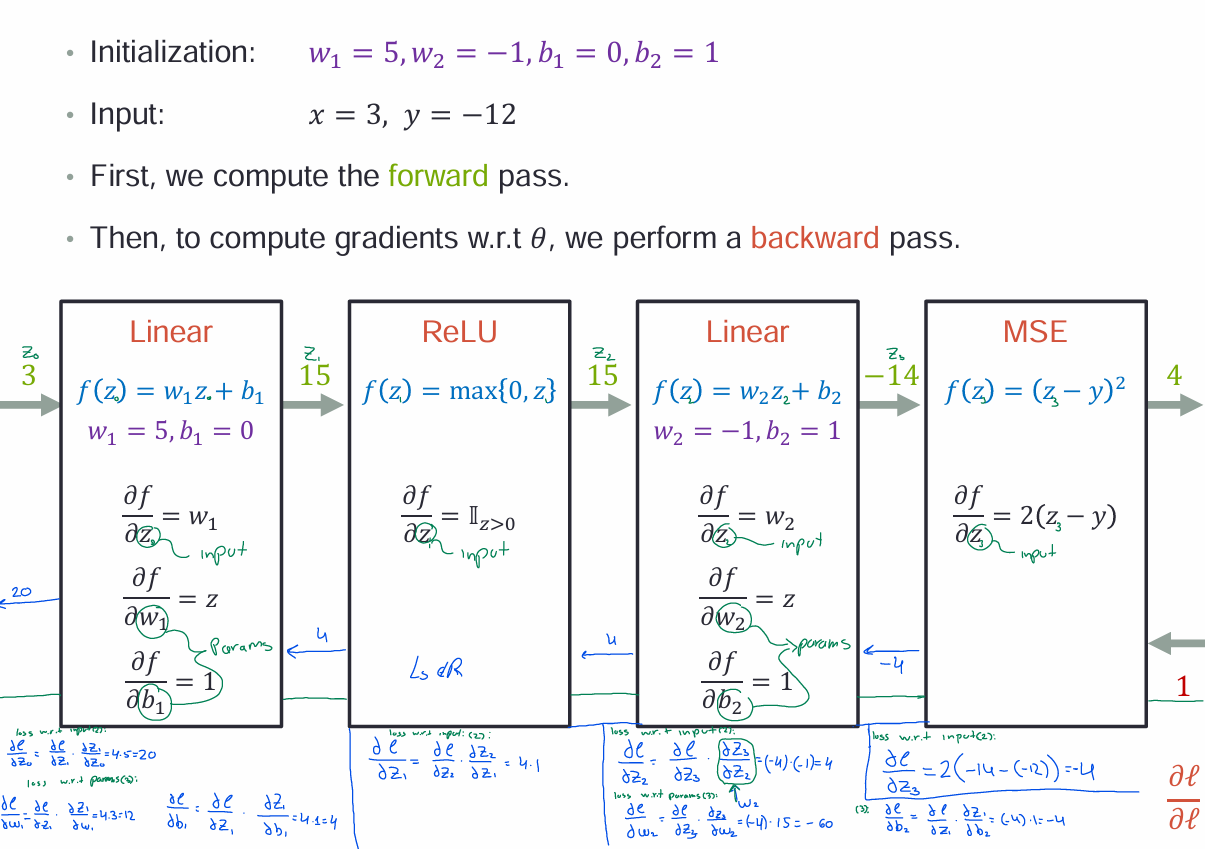
***Gradient descent general scheme:***

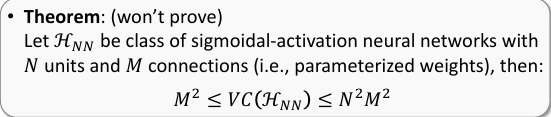
* *Initialize parameters randomly*
* *While model has not converged:* 
  + *Compute gradient*
* *Update weights*

*• Layer specification – each layer should provide three functions:*

*1. The layer output given its input (Forward)*

*2. Loss gradient w.r.t. the input (Backward)*

*3. Loss gradient w.r.t. parameters*

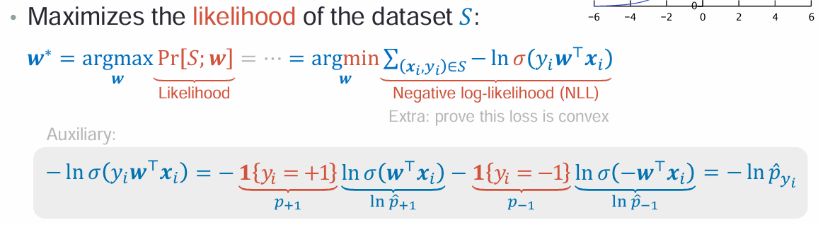
***Lecture 12***

***Multiclass models:***

* ***Goal:*** *classify into many classes ℎ:𝒳 →*
* *Solution: train one multiclass model that predicts a class distribution*

***Cross entropy*** *- measures how one distribution differs from another*

*logistic regression loss function is equivalent to the cross-entropy loss for binary classification.*

*****argMax Likelihood*** *–**the that’ll fit our dataset best (using sigmoid continuous function)*

*This: reads as follows:*

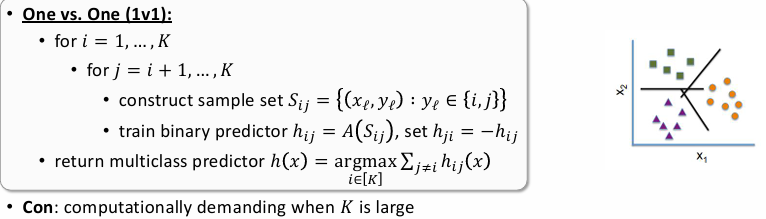
* *the prob that given the input features ​ and the model parameters (y)distributes Binomially*

*Common implicit regularization tricks to control overfitting:*

*1. Early stopping (using held-out validation set; like we saw)*

*2. Drop-out: at each epoch (=SGD step over all data points), temporarily*

*“remove” a random subset of connections (=parameters)*

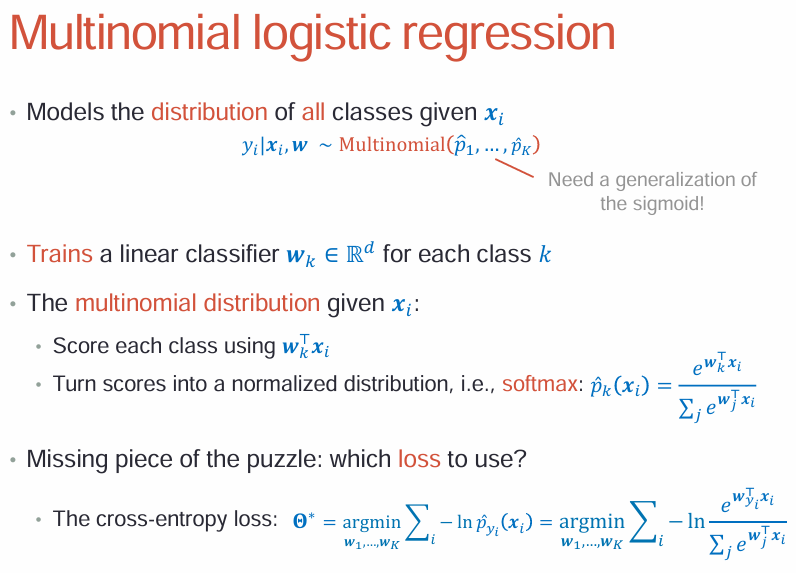
* *Learning setting that extends to other output types:*
  + *multiclass: one object out of many, 𝑦 ∈ 𝒴 = 𝐾*
  + *multilabel: multiple objects, 𝑦 ∈ 𝒴 = 2*
  + *structured: e.g., relations between objects, e.g., 𝑦 ∈*

***OneVsAll*** *reduces a multiclass task into a separate binary task*

***OneVsOne:*** *choose the - model that believes the most that it’s the number(compare all 1v1’s and sum what each believes to score)*

***softmax-***

* *Goal: learn*
* *Problem: isn’t continuous and non-differentiable*
* *Solution: use SoftMax which is continuous and differentiable*
  + *Differentiable, convex for linear and cross-entropy*
  + *Normalized(entries sum to 1)*
  + *When are scaled with “temperature” , SoftMax approaches argmax.*
    - *Larger – tighter approximation(OneHot)*
    - *Smaller – smoother approximation(uniform)*

***Multinomial Logistic Regression***