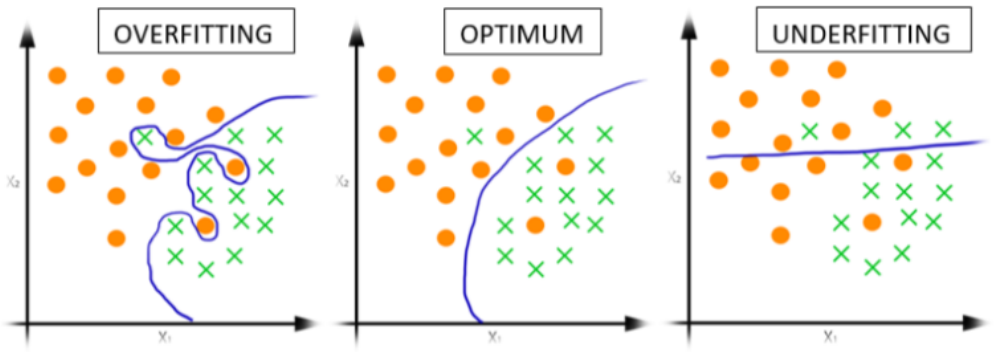
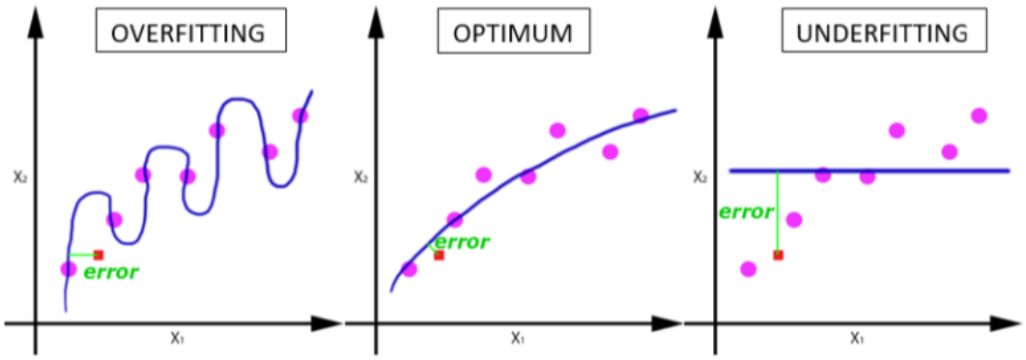
Review

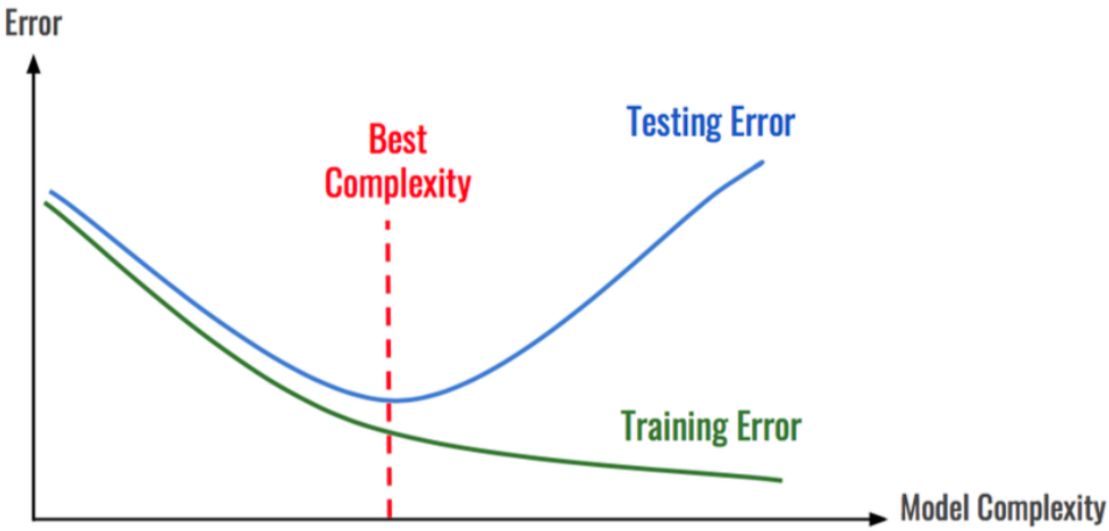
- We have learnt 4 classifiers in depth: KNN, Naive Bayes, LDA and Logistic Regression
    - KNN: Results sensitive to k
    - Logistic Regression: $p(y|x;w)$ sensitive to the magnitude of w

Capacity and Generalization

- Model capacity:
    - **Deterministic classifiers** that can represent **more complex decision boundaries** (in log form) are said to have **higher *capacity*** than classifiers that can only represent simpler boundaries
    - **Probabilistic classifiers** that can represent **more complex sets of conditional** $p(y|x)$ are said to have higher $capacity$ than probabilistic classifiers that can only represent simpler sets of conditionals
    - Rank of the leart classifiers: KNN > Naive Bayes, Logistic Regression $\approx$ LDA
- Reason of not always using the classifier with the highest possible capacity for every problem
    - This would minimize the error on the **training data**. However, what we really care about for prediction problems is *generalization*
- **Generalization**: The ability of a trained classifier to achieve an error rate on *future*, *unseen examples* (the generalization error rate) that is comparable to the training error rate
- **Capacity Control**: To achieve optimal generalization performance for a given traing set, we often need to control model capacity carefully
- **Overfitting vs underfitting**
    - **Overfitting**: The generalization error for a classifier is much worse than the training error. This usually results from choosing a classifier with too much capacity so that it models the noise in the training data
    - **Underfitting**: Occurs when the capacity of the classifier is too low to capture the actual structure in the training data, leading to both high training error and high generalization error
    - Examples(error on the down-left corner graph):



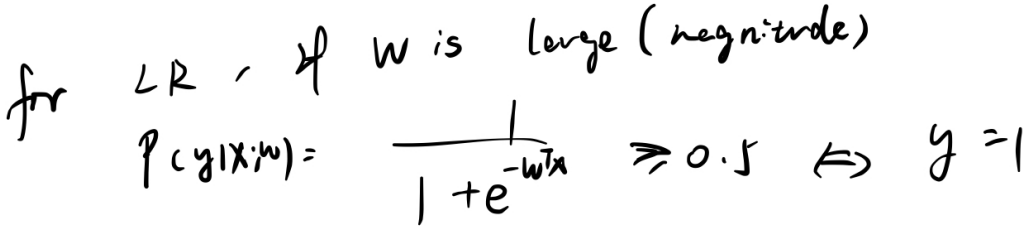


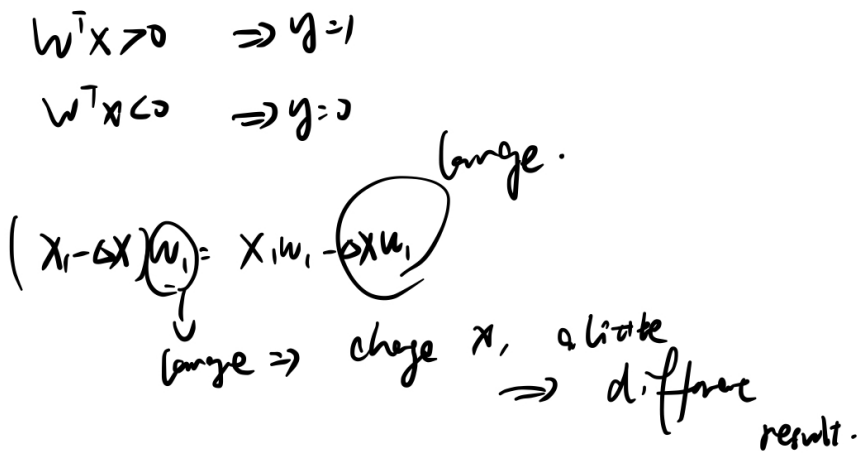


- Bias-Variance Trade-Off (both of them are negative things)
  - Bias: A classifier is said to have low *bias* if the true decision boundary or conditionals $p(y|x)$ can be **approximated closely by the model**
  - Variance: A classifier is said to have low *variance* if the decision boundary or conditionals $p(y|x)$ it constructs are stable with respect to small changes in the training data
  - Bias-Variance Dilemma: To achieve **low generalization error**, we need classifiers that are low-bias and low-variance, but this isn't always possible
  - Bias-Variance and Capacity: On **complex data**, models with **low capacity have low variance**, but **high bias**; while models with **high capacity** have **low bias**, but **high variance**

Hyperparameters

- Why need to control hyperparameters
  - In order to control the capacity of a classifier, it needs to have capacity control parameters
  - Because capacity control parameters can not be chosen based on training error, they are often called hyperparameters
  - For KNN: k, for logistic regression: $\alpha \; and \; \lambda$
- Capacity, Smoothness and Regularization





  - **Capacity and Smoothness**: In the case of probabilistic classifiers like naive Bayes, LDA, and logistic regression, we can think of capacity in terms of the *smoothness* of $p(y|x)$
  - **Regularization**: We can control the smoothness of $p(y|x)$ using a technique called *regularization* that penalizes parameters that result in overly complex $p(y|x)$ during learning
  - **Laplace Smoothing**: In the case of Naive Bayes, we introduced Laplace smoothing for a categorical distribution

$$\theta_{x,j|y} = \frac{\sum_{i=1}^{m} \mathbb{I}[y^{(i)} = y \cap x_j^{(i)} = x] + \alpha}{\sum_{i=1}^{m} \mathbb{I}[y^{(i)} = y] + \alpha|\mathcal{X}|}$$

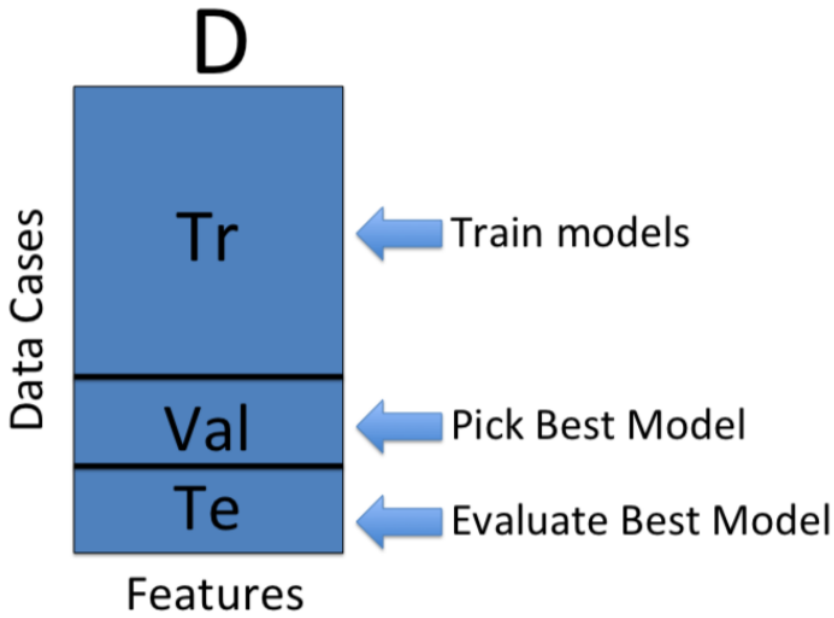
    - As the regularization hyperparameter α increases, our estimate of the distribution smooths out toward being uniform
  - In the case of logistic regression, the smoothness of $p(y|x;\theta)$ is determined by the magnitude of θ
    - Similar in linear regression, we ca control the magnitude of θ by introducing a regularization term into the conditional log likelihood learning criteria that penalizes the l_p-norm of θ

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \; -\sum_{i=1}^{m} \log p(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_p^p$$

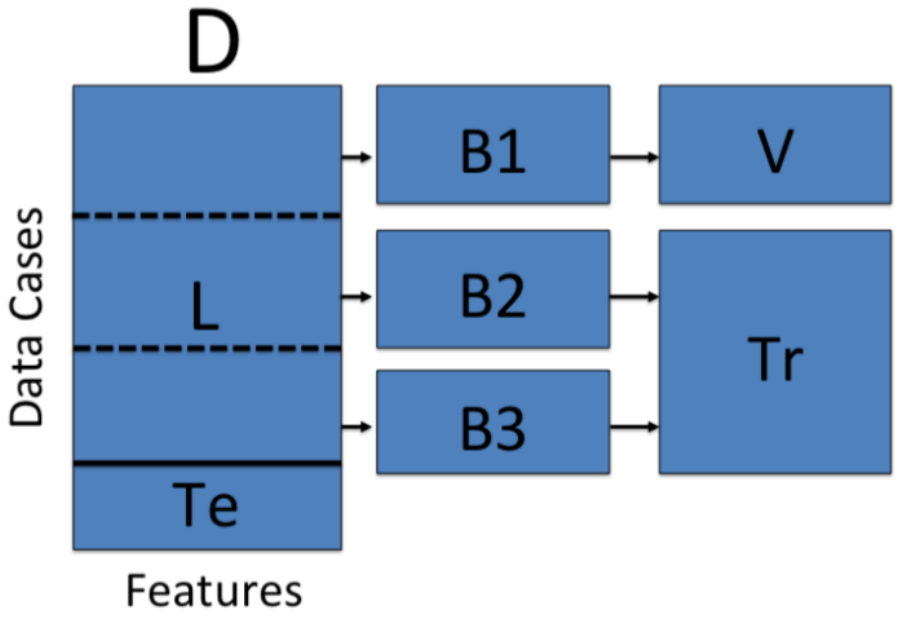
- As the regularization hyperparameter λ increases, the learned $p(y|x;\theta)$ will smooth out

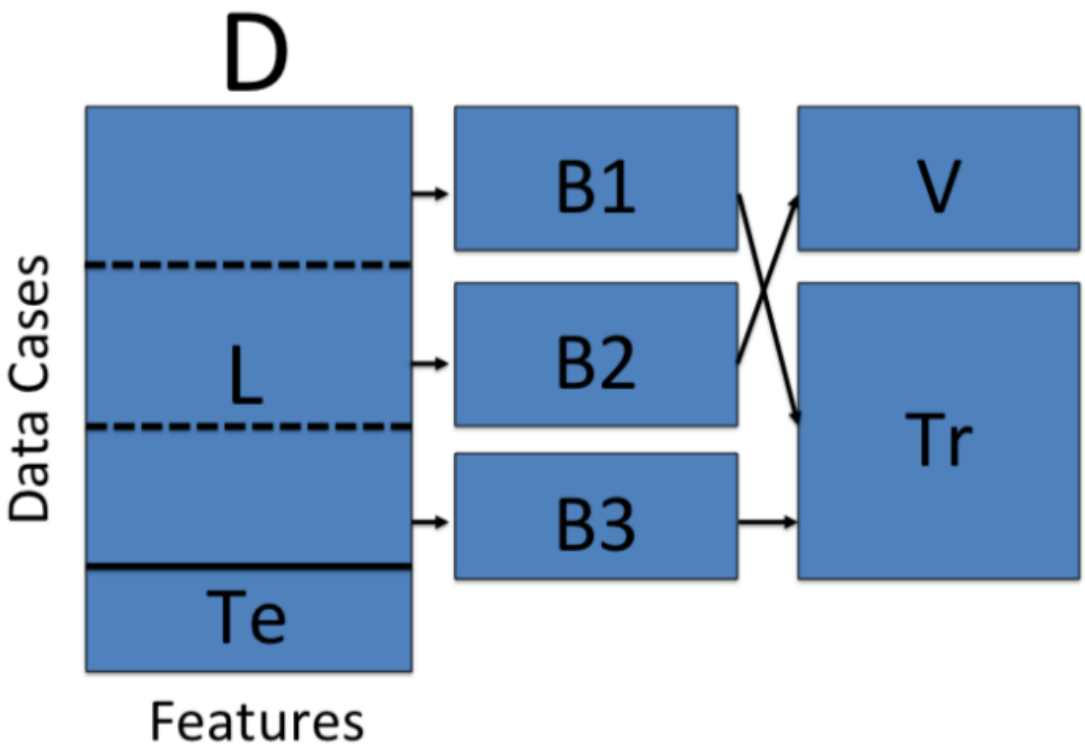
Model Selection and Evaluation

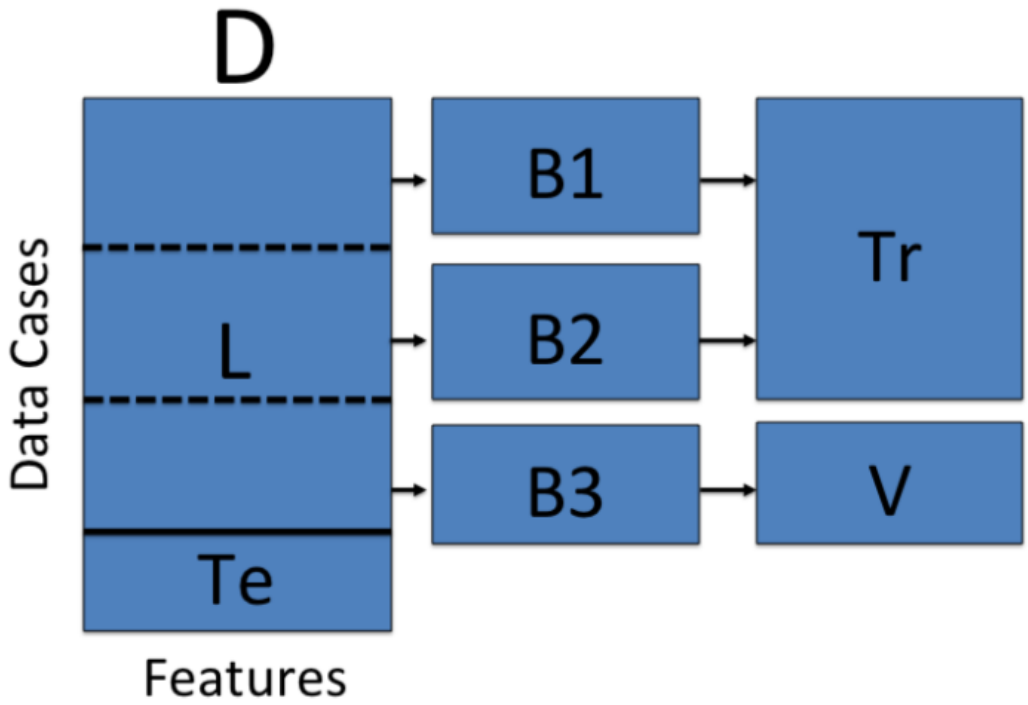
- Principles:
  - We've identified the parameters that **control the capacity** of our models, we need a way to choose optimal values for these parameters
  - In addition, we will want an estimate of the **generalization error** that the selected parameters achieve
  - To obtain valid results, we need to use appropriate methodology and construct learning experiments carefully
  - **Guiding Principle**: Data used to estimate generalization error(**testing set**) can not be used for any other purpose, $i.e.$, model training, hyperparameter selection, feature selection, etc. Otherwise, the results of the evaluation will ve $biased$
- Methods:
  - Recipe 1: Train, Validation, and Test
    - Given a data set D, we randomly partition the data cases into a **training set (Tr)**, a **validation set (Val)**, and a **test set (Te)**. Typical splits are 60/20/20, 90/5/5, etc.
    - Training
      - Models $\{f(i)\}$ are learned on Tr for each **one** choice of hyperparameters $\theta(i)$ (Choose weigth matrix in training set)
    - Validation
      - The validation error $Err_v^{(i)}$ of each model $f^{(i)}$ is evaluated on Val v
      - The hyperparameters $\theta^\star$ with the lowest validation error are selected and the classifier is re-trained using these hyperparameters on **Tr + Val**, yielding a final model $f^\star$ (Choose hyperparemeters in training & validation set)
    - Testing
      - Generalization performance is estimated by evaluating error/accuracy of $f^\star$ on the test data Te (Evaluate the final model)
    - Example: (Notice that the data cases needs to be randomly shuffled before partitioning D)



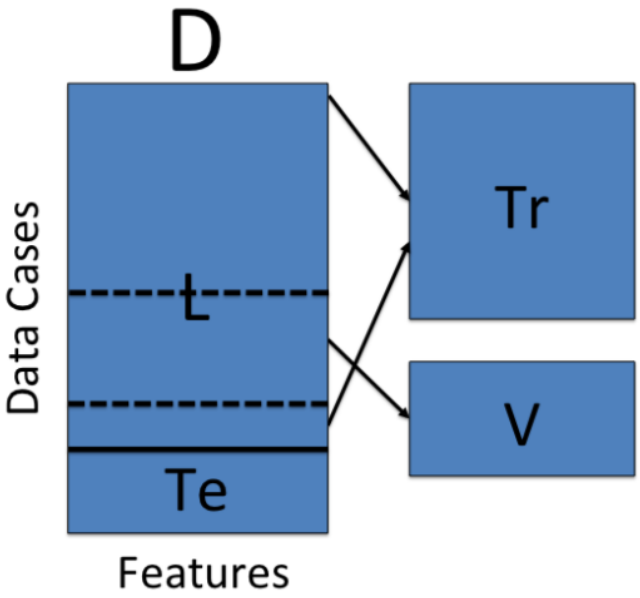
- Recipe 2: Cross-Validation and Test
  - Randomly partition D into a learning set L and a test set Te, typically 50/50, 80/20, etc.
  - We next **randomly partition** L into a set of K blocks $B_1, \ldots, B_K$
  - For each cross-validation fold $k = 1, \ldots, K$:
    - Let $Val = B_k$ and $Tr = L/B_k$ (the remaining $K-1$ blocks)
    - Learn $f^{(i,k)}$ on Tr for each choice of hyperparameters $\theta^{(i,k)}$
    - Compute $Err^{(i,k)}$ of $f^{(i,k)}$ on Val v
  - Select hyperparameters $\theta^\star$ minimizing $\frac{1}{K}\sum_{k=1}^{K} Err_v^{(i,k)}$ and re-train model on L using these hyperparameters, yielding the final model $f^\star$
  - Estimate generalization performance by evaluating error/accuracy of $f^\star$ on Te
  - Example: Three Fold Cross-Validation and Test (Notice that the data cases needs to be randomly shuffled before partitioning D into L and Te)


D
Data Cases
L
Te
Features
B1
B2
B3
V
Tr


D
Data Cases
L
Te
Features
B1
B2
B3
V
Tr



- Recipe 3: Random Resampling Validation and Test (Notice that the data cases needs to be randomly shuffled before partitioning D into L and Te)
  - Randomly partition the data cases into a learning set L and a test set Te, typically 50/50, 80/20, etc.
  - For sample $k = 1, \ldots, K$:
    - Randomly partition L into Tr and Val, again 50/50, 80/20, etc.
    - Learn $f^{(i,k)}$ on Tr for each choice of hyperparameters $\theta^{(i,k)}$
    - Compute $Err^{(i,k)}$ of $f^{(i,k)}$ on Val v
  - Select hyperparameters $\theta^\star$ minimizing $\frac{1}{K}\sum_{k=1}^{K} Err_v^{(i,k)}$ and re-train model on L using these hyperparameters, yielding the final model $f^\star$
  - Estimate generalization performance by evaluating error/accuracy of $f^\star$ on Te
  - Example:



- Trade-Offs:
  - In cases where the data has a benchmark split into a training set and a test set, we can use Recipes 1-3 by preserving the given test set and splitting the given training set into train and validation sets as needed
  - Choosing larger K in cross-validation will reduce bias. Choosing larger K in random re-sampling validation will reduce variance and bias. However, both increase computational costs. K = 3, 5, 10 are common choices for cross-validation. , $K = m$, also known as ***leave-one-out cross validation*** is also popular when feasible

Feature Selection

- Feature Selection: As a special and important case of model selection, feature selection selects a useful subset from all the features
- Reason of feature selection:
  - Some algorithms (e.g. normal equation) scale (computationally) poorly with increased feature dimension
  - Irrelevant features can confuse some algorithms
  - Redundant features may adversely affect regularization
  - Reduces data set and resulting model size
- Methods
  - Wrapper methods (keep the learning algorithm in the loop)
    - Requires repeated runs of the learning algorithm with different sets of features
    - Can be computationally expensive
  - Filter feature selection methods
    - Use some ranking criteria to rank features
    - Select the top ranking features
- Wrapper Methods
  - Forward search

        - Start with no features
        - Greedily include the most relevant feature
        - Stop when selected the desired number of features
        - Steps:
            - Let $F = \{\}$
            - While not selected desired number of features
            - For each unused feature f
                - Estimate model's error on feature set $F \cup f$
            - Add f with lowerst generalization error to F
    - Backward search (Maybe computational expensive)
        - Start with all the features
        - Greedily remove the least relevant feature
        - Stop when selected the desired number of features
        - Steps:
            - Let $F = \{all\ features\}$
            - While not reduced to desired number of features
            - For each unused feature $f \in F$
                - Estimate model's error on feature set $F \setminus f$ (using cross-validation)
            - Remove f with lowerst generalization error from F
    - Inclusion/Removal criteria uses cross-validation (*i.e.*, train model using the current subset of features and estimate the generalization error)
- Filter Feature Selection Methods (outdated => deep learning nowadays)
    - Uses heuristics but are much faster than wrapper methods
    - Correlation criterion: Rank features according to their correlation with the labels:

$$\text{Corr}(\mathbf{x}_j, \mathbf{y}) = \frac{\sum_{i=1}^{m}(x_j^{(i)} - \mu_{\mathbf{x}_j})(y^{(i)} - \mu_{\mathbf{y}})}{\sqrt{\sum_{i=1}^{m}(x_j^{(i)} - \mu_{\mathbf{x}_j})^2}\sqrt{\sum_{i=1}^{m}(y^{(i)} - \mu_{\mathbf{y}})^2}}$$

    - Mutual information criterion:

$$\text{MI}(x_j; y) = \sum_{x_j \in \mathcal{X}_j} \sum_{y \in \mathcal{Y}} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)},$$

        - where the probabilities$p(x_j, y), p(x_j)$ and $p(y)$ can be estimated according to their empirical distributions on the training set
- More on Mutual Information

- Mutual information is a fundamental concept in *information theory*, measuring the mutual dependence between two variables
- It quantifies the "amount of information" obtained about one random variable through observing the other random variable
- As a function of $p(x,y) = p(x)p(y|x)$, $I(x;y)$ is concave in $p(x)$ for fixed $p(y|x)$, and is convex in $p(y|x)$ for fixed $p(x)$
- Maximizing $I(x;y)$ w.r.t. $p(x)$ leads to *noisy-channel coding theorem*, establishing the maximum communication rate through a noisy channel
- Minimizing $I(x;y)$ w.r.t. $p(y|x)$ leads to *rate-distortion theorem*, laying the theoretical foundations for lossy data compression

---

Lecture07 Support Vector Machine

Target: Increase the capacity of linear classifiers so they can produce non-linear classification boundaries

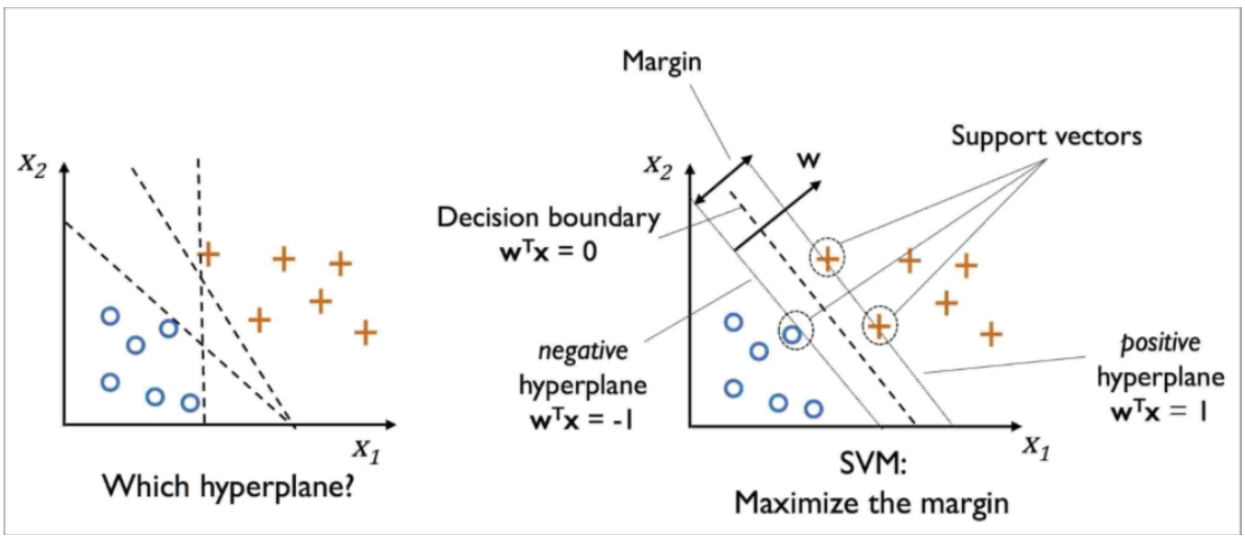
---

Support Vector Machine (SVM)

- Decision Boundary
  - A binary SVM is a discriminative classifier that takes labels in the set $\{-1, 1\}$
  - The decision function has the form:

$$f_{SVM}(x) = sign(\mathbf{w}^T\mathbf{x} + b)$$
$$\mathbf{w}^T\mathbf{x} + b = 0 \ (on\ the\ decision\ boundary)$$

  - It's easy to show that the decision boundary for logistic regression can ve written in exactly the same way
  - Question: If logistic refgression and SVMs have the same form for their decision boundaries, how do they differ
    - Logistic regression minimize the log probability of miss-classification
    - Logistic regression maximize the probability on the correct label (i.e. if the probability is more than 0.5, it can be correctly classified. But it tends to increase the probability during training)
- Maximum Margin Principle (**notice the axis**)

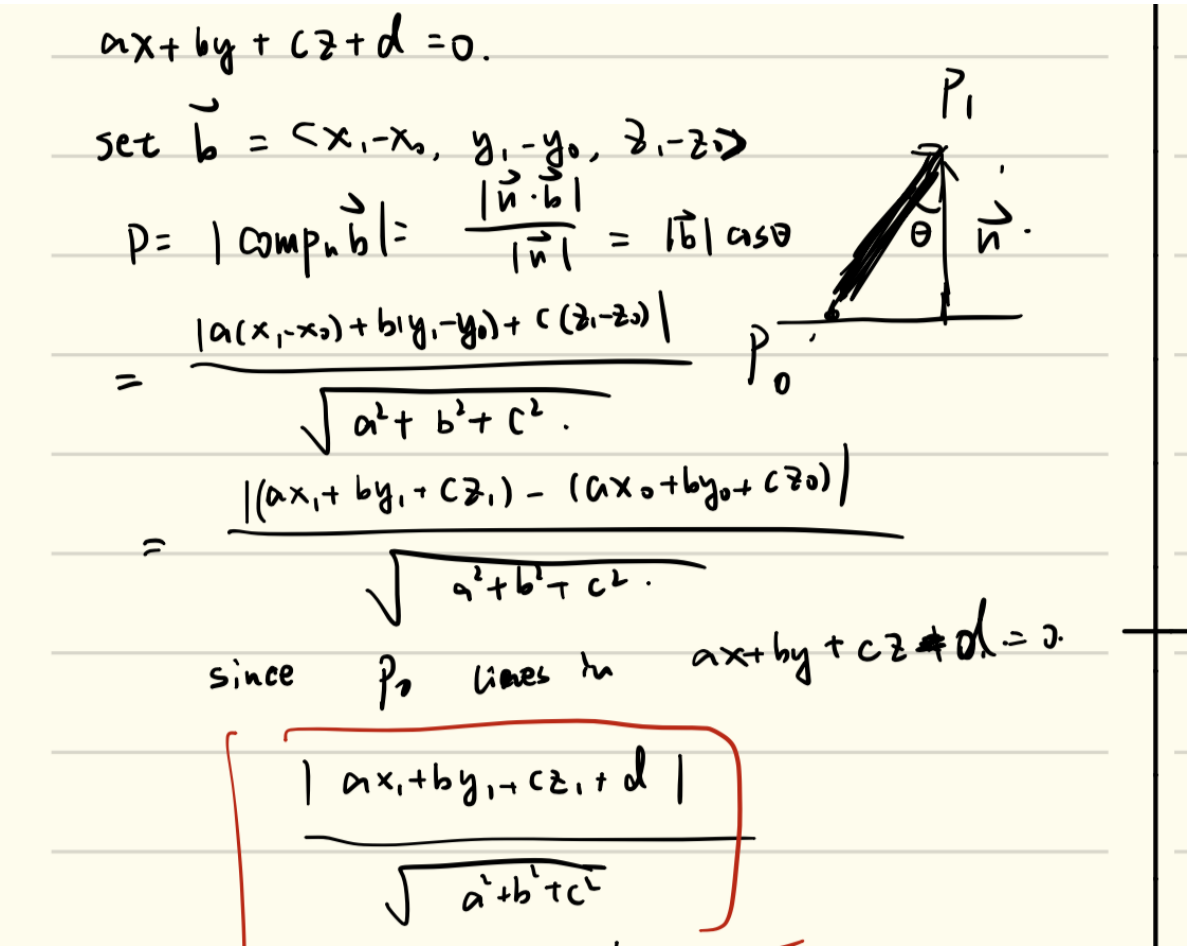


- Define the *margin* of a linear classifier as the width that the **boundary could be increased before hitting a data point**
- The simplest kind of SVM is a linear classifier with the maximum margin
- Reason of MAX margin:
  - If we've made a small error in the location of the decision boundary, this gives us least chance of causing a misclassification
  - There's solid theory (using VC dimension) that provides indirect support for this proposition

- Compute the Margin
  - The distance between a point to a line:

$$d^{(i)} = \frac{|\mathbf{w}^T\mathbf{x} + b|}{||\mathbf{w}||_2} = \frac{y^{(i)}(\mathbf{w}^T\mathbf{x} + b)}{||\mathbf{w}||_2}$$

  - Proof:



  - The $y^{(i)}$ can be added because the value can ve only found in the set $\{-1, 1\}$
  - The margin of $\mathbf{w}^T x + b = 0$ with respect to a training set $D = \{(\mathbf{x}^{(i)}, y^{(i)}), i = 1, \ldots, m\}$, where $||\mathbf{w}||_2 = \sqrt{\sum_{j=1}^{n} w_j^2}$

$$d(margin) = min\ d^{(i)} = min_{(\mathbf{x},y)\in D} \frac{y^{(i)}(\mathbf{w}^T\mathbf{x} + b)}{||\mathbf{w}||_2}$$

- Maximize the Margin
  - The maximum margin solution is found by solving

$$max_{\mathbf{w},b}(\frac{1}{||\mathbf{w}||_2}min_{(\mathbf{x},y)\in D}\ y^{(i)}(\mathbf{w}^T\mathbf{x}+b))$$

- Note that if we make the rescaling $\mathbf{w} \rightarrow \gamma\mathbf{w}$ and $b \rightarrow \gamma b$, then the objective is unchanged
  - We can use this freedom to set: $min_{(\mathbf{x},y)\in D}\ y(\mathbf{w}^T\mathbf{x}+b) = 1$

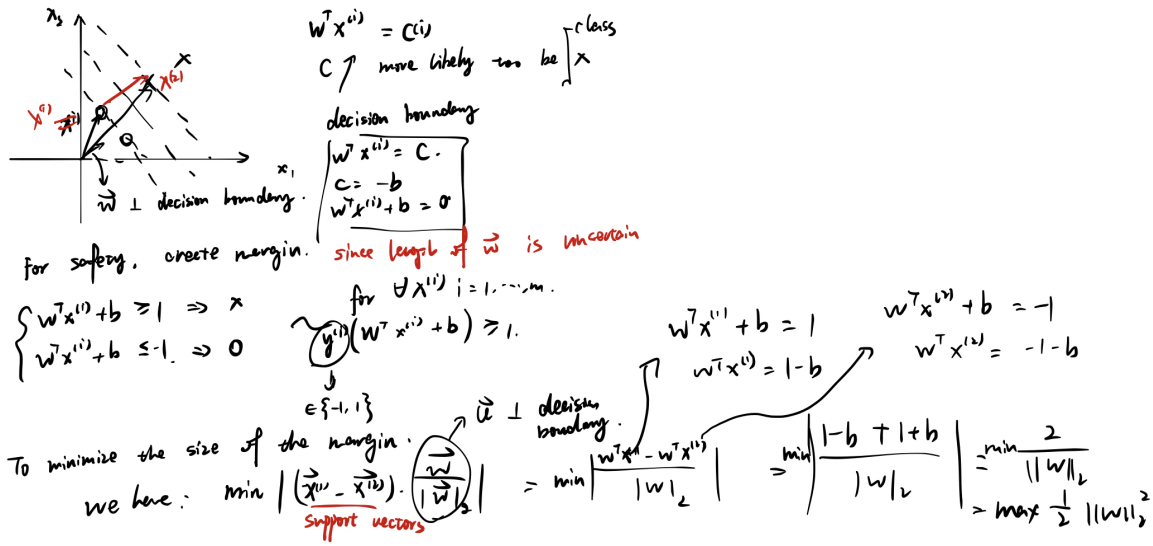
$$Ax + By + C = 0 <=> 2Ax + 2By + 2C = 0$$

  - Or equivalently: $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \forall i = 1, \ldots, m$

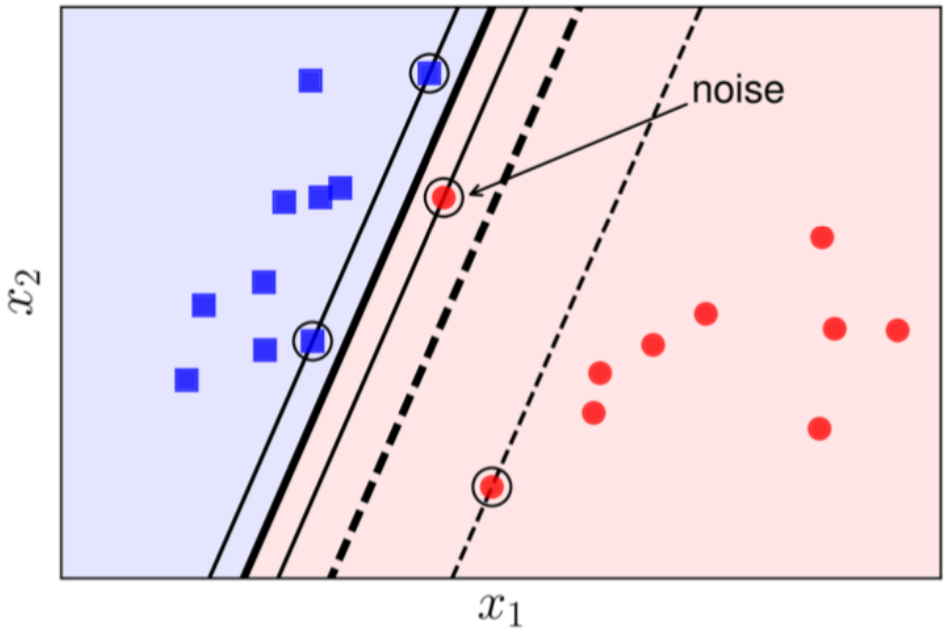
- Primal Form, **Hard-Margin** of SVM

$$max_{\mathbf{w},b}(\frac{1}{||\mathbf{w}||_2}) <=> min_{\mathbf{w},b}\ \frac{1}{2}||\mathbf{w}||_2^2 (for\ convenience)$$
$$subject\ to\ y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1, \forall i = 1, \ldots, m$$

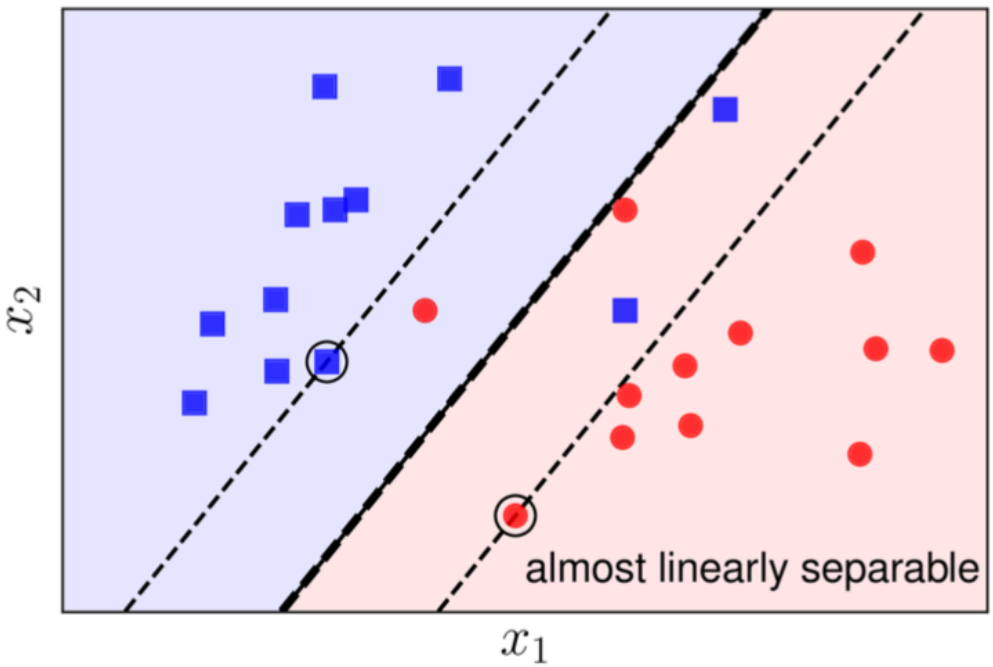
  - Quadratic programming problem with a quadratic objective function and linear inequality contstraint
  - Well studied problem in operations research
  - Another ponit of view: (min and max in the last line should be reversed)



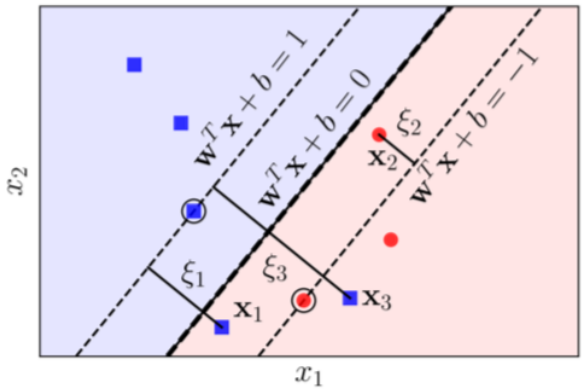
- Primal From, Soft-Margin
  - Hard margin is not enough:
    - When there is a noise point to narrow down the margin



- when the data is not linear-discriminatable: **notice that in this time, margin is nearly zero**



- We need to add some level of tolerances to the model:



$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{m}\xi_i$$

$$\text{subject to} \quad y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i,\ i = 1, \ldots, m$$

$$\xi_i \geq 0,\ i = 1, \ldots, m$$

- In other word, for normal case, $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$, the penalty of misclassification is zero, while on the other hand, there is a penalty with the value $1 - y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)$
- The level of tolerance can be found on the second equation:
  - Originally, $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1$ means that, all pairs of the points should obey this rule, therefore, **the "distance" between two boundaries generated by the support vector should be at least 2**
  - Now, $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i, \xi_i > 0$ means that, for any one of the points, there is a **tailored** ξ_i to make sure that the point always lies in the safe constraint. Once $y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) < 1$, the difference between $1\ and\ y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b)$ will be counted as penalty.
  - This approach makes sure that all points are not forced to have a closest support vector on one or two sides of the decision boundary, but the mistake caused by it will be countered as part of the loss function

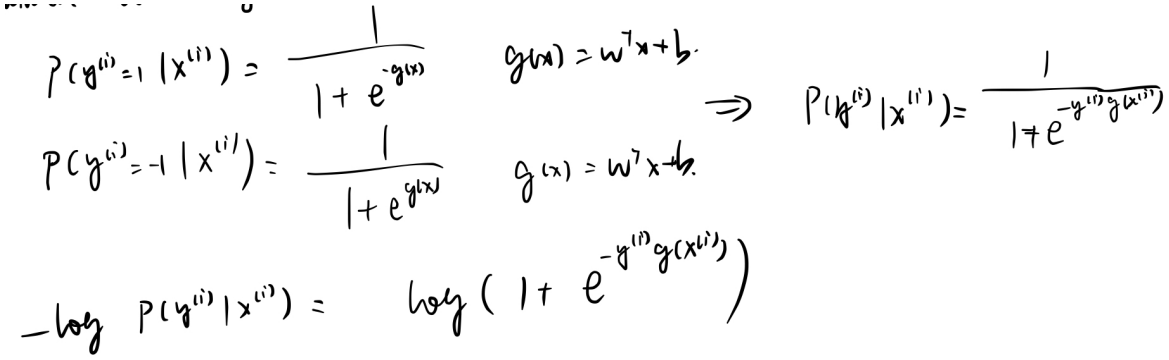
- Hinge Loss
  - In the case of primal-form SVM with soft-margin, it is equivalent to minimizing the function:

$$C\sum_{i=1}^{m} max(0, 1 - y^{(i)}.\, g(\mathbf{x}^{(i)})) + \frac{1}{2}||\mathbf{w}||_2^2$$

    - where $g(x) = \mathbf{w}^T\mathbf{x} + b$ and C is the hyperparameter that controls the amount of regularization
  - The function $l_h(y, g(\mathbf{x})) = max(0, 1 - y.\, g(\mathbf{x}))$ is called the *hinge loss*
- Logistic Loss
  - In the case of logistic regression with l_2 regularization, we select the model parameters by minimizing the function

$$C\sum_{i=1}^{m} -log\, p(y^{(i)}|\mathbf{x}^{(i)}) + \frac{1}{2}||\mathbf{w}||_2^2$$

- Under the assumption that the labels take the values $\{-1, 1\}$:

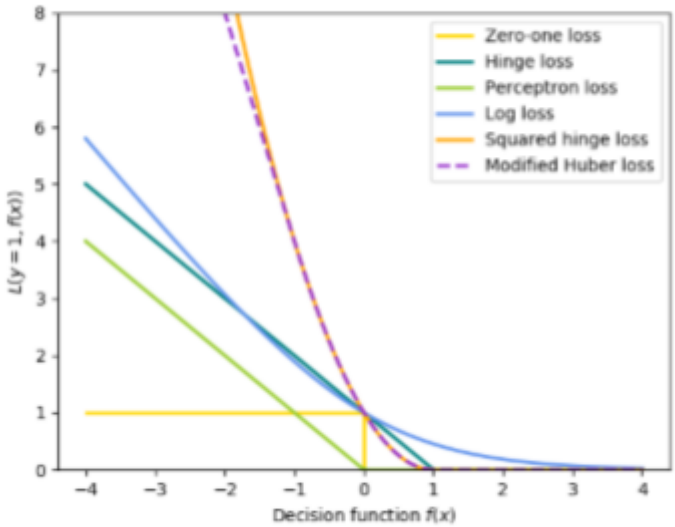


$$C\sum_{i=1}^{m} log\,(1 + exp(-y^{(i)}.\,g(\mathbf{x}^{(i)}))) + \frac{1}{2}||\mathbf{w}||_2^2$$

  - Where the function $l_l(y, g(\mathbf{x})) = log(1 + exp(-y^{(i)}.\,g(\mathbf{x}^{(i)})))$ is called the *logistic loss function*
- Zero-One Loss
  - Both the logistic loss and the hinge loss are convex upper bounds on the zero-one loss:

$$l_{01}(y, g(\mathbf{x})) = \mathbb{I}[y \neq sign(g(\mathbf{x}))]$$

  - Average of $01 - loss$ is exactly the classification error rate
  - This is the loss function we'd like to minimize, but this generally isn't computationally feasible, thus the need for surrogate loss functions
- Comparison (x-axis is the decision function)



Lagrange Duality

- Lagrange Duality: In mathematical optimization theory, **duality** or the **duality principle** is the principle that optimization problems may be viewed from either of two perspectives, the **primal problem** or the **dual problem**. The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem. However in general the optimal values of the primal and dual problems need not be equal. Their difference is called the duality gap. For convex optimization problems, the duality gap is zero under a constraint qualification condition.

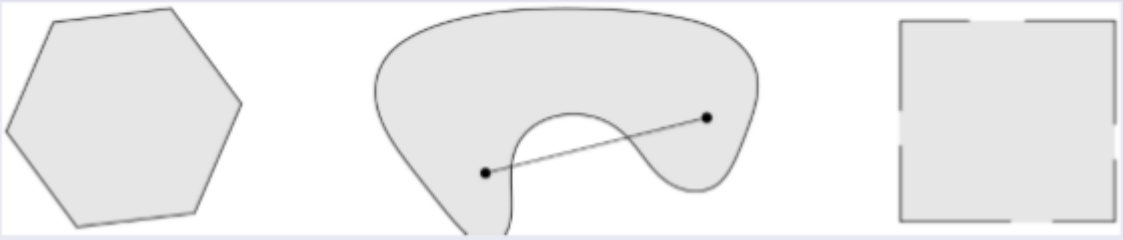
- Convex set

Definition: Convex Set

A convex set contains line segment between any two points in the set

$$\mathbf{w}, \tilde{\mathbf{w}} \in W,\ 0 \leq \theta \leq 1 \implies \theta\mathbf{w} + (1-\theta)\tilde{\mathbf{w}} \in W$$

Example



- Convex function

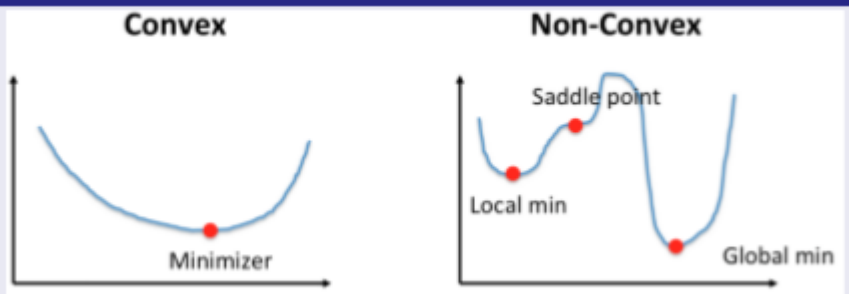
Definition: Convex Function

$f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex if the domain of f, $\text{dom}(f)$, is a convex set and

$$f(\theta\mathbf{w} + (1-\theta)\tilde{\mathbf{w}}) \leq \theta f(\mathbf{w}) + (1-\theta) f(\tilde{\mathbf{w}})$$

for all $\mathbf{w}, \tilde{\mathbf{w}} \in \text{dom}(f)$, $0 \leq \theta \leq 1$.

Example



- Standard form of optimization (minimize for instance) problems

$$\begin{aligned} & min_{\mathbf{w}} f_0(\mathbf{w}) \\ subject\ to\ & f_i(\mathbf{w}) \leq 0, i = 1, \ldots, r \\ & h_i(\mathbf{w}) = 0, i = 1, \ldots, s \end{aligned}$$

  - $\mathbf{w} \in \mathbb{R}^n$ is the optimization variable
  - $f_0 : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective or cost function
  - $f_i : \mathbb{R}^n \mapsto \mathbb{R}, i = 1, \ldots, r$ are the inequality constraint function
  - $h_i : \mathbb{R}^n \mapsto \mathbb{R}, i = 1, \ldots, s$ is the equality constraint function

    - If f_0 and a set of f_i are convex functions, and h_i's are affine functions, it becomes a **convex optimization problem**
        - Affine function: $f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}, A \in \mathbb{R}^{s \times n}, \mathbf{x} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^s$
- Lagrangian: $L : \mathbb{R}^n \times \mathbb{R}^r \times \mathbb{R}^s \mapsto \mathbb{R}$, with $dom(L) : W \times \mathbb{R}^r \times \mathbb{R}^s$ (fixed weight matrix)

$$L(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{w}) + \sum_{i=1}^{r} \lambda_i f_i(\mathbf{w}) + \sum_{i=1}^{s} \nu_i h_i(\mathbf{w})$$

    - It's a weighted sum of objective and constraint function
    - λ_i is Lagrange multiplier associated with $f_i(x) \leq 0$
    - ν_i is Lagrange multiplier associated with $h_i(x) = 0$
- Lagrange dual function: $g : \mathbb{R}^r \times \mathbb{R}^s \mapsto \mathbb{R}$, (inf means min)

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}) = inf_{\mathbf{w} \in \mathbf{W}}(L(\mathbf{w}, \boldsymbol{\lambda}, \boldsymbol{\nu})) = inf_{\mathbf{w} \in \mathbf{W}}(f_0(\mathbf{w}) + \sum_{i=1}^{r} \lambda_i f_i(\mathbf{w}) + \sum_{i=1}^{s} \nu_i h_i(\mathbf{w}))$$

    - g is **concave** and can be $-\infty$ for some $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$
    - According to the constraints: $f_i(x) \leq 0,\ h_i(x) = 0, \lambda_i > 0$, we have:

$$f_0(\mathbf{w}^{\tilde{}}) \geq L(\mathbf{w}^{\tilde{}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \geq inf_{\mathbf{w}}(L(\mathbf{w}^{\tilde{}}, \boldsymbol{\lambda}, \boldsymbol{\nu})) = g(\boldsymbol{\lambda}, \boldsymbol{\nu})$$

    - Definition of p^* (the value we want):

$$p^* = inf\{f_0(\mathbf{w}) | f_i(\mathbf{w}) \leq 0, i = 1, \dots r, h_i(\mathbf{w}) = 0, i = 1, \dots, s\}$$

    - Then we have the following equation

$$d^* = max_{\lambda \geq 0, \nu}\ g(\lambda, \nu) \leq min\ f_0(\mathbf{w}^{\tilde{}}) = p^*$$

- Lagrange dual problem (transformed by the Lagrange dual function)

$$max_{\lambda, \nu}\ g(\lambda, \nu)$$
$$subject\ to\ \lambda \geq 0$$

    - It is a convex optimization problem, with optimal value d^*
    - d^* is the best lower bound on p^*
    - **Weak duality**: $d^* \leq p^*$, which always holds for convex and nonconvex problems
    - **Strong duality**: $d^* = p^*$ does not hold in general, but (**usually**) holds for convex problems, e.g., SVM

Optimization of SVM: From Primal to Dual

- Primal Form:
    - Notice that for constraints f_i, there is a set of constraints

$$min_{\mathbf{w}, b}\ \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m} \xi_i$$
$$subject\ to\ y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) \geq 1 - \xi_i, i = 1, \dots, m$$
$$\xi_i \geq 0, i = 1, \dots, m$$

- Dual Form (Weak Duality):

$$L(\mathbf{w}, \mathbf{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i[y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + b) - 1 + \xi_i] - \sum_{i=1}^{m} \beta_i \xi_i$$
$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)} - \sum_{i=1}^{m} \alpha_i y^{(i)} b + \sum_{i=1}^{m} (C - \alpha_i - \beta_i)\xi_i + \sum_{i=1}^{m} \alpha_i$$

- Notice that α and β are two vectors contains the Lagrange Multiplier λ , there is no affine constraint
- We minimize the **Lagrangian** (definition of g) $w.r.t$ the primal variables $\mathbf{w}, \mathbf{b}, \boldsymbol{\xi}$

$$\nabla_{\mathbf{w}} L = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)} = 0 \ (1)$$
$$\nabla_{\mathbf{b}} L = -\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \ (2)$$
$$\nabla_{\boldsymbol{\xi}_i} L = C - \alpha_i - \beta_i = 0 \ (3)$$
$$i = 1, \ldots, m$$

- Plugin the gradient equality to the Lagrangian to get the **Lagrange dual function**:

$$g(\boldsymbol{\alpha}, \boldsymbol{\beta}) = inf_{\mathbf{w},\mathbf{b},\boldsymbol{\xi}}(L(\mathbf{w}, \mathbf{b}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta})) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)}$$
$$- \sum_{i=1}^{m} \alpha_i y^{(i)} b (= 0 \ according \ to \ (2))$$
$$+ \sum_{i=1}^{m} (C - \alpha_i - \beta_i)\xi_i (= 0 \ according \ to \ (3)) + \sum_{i=1}^{m} \alpha_i$$
$$= \sum_{i=1}^{m} \alpha_i + \frac{1}{2}(\sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)})^T (\sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)}) - \sum_{i=1}^{m} \alpha_i y^{(i)} (\sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)})^T \mathbf{x}^{(i)}$$
$$(according \ to \ (1))$$
$$= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$

  - The property: $(a+b)^T(c+d) = (a^T + b^T)(c+d) = a^Tc + a^Td + b^Tc + b^Td$ is used in the last step
- Finally, we get the **dual form** of the primary form of SVM optimization:

$$max_{\alpha} \ \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (\mathbf{x}^{(i)})^T \mathbf{x}^{(j)}$$
$$subject \ to \ \nabla_{\mathbf{b}} L = -\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \ (2)$$
$$0 \leq \alpha_i \leq C (\alpha_i \geq 0, \beta_i \geq 0, \beta_i = C - \alpha_i \ (3))$$

- Strong Duality Form:
  - Complementary slackness: Assume the strong duality holds, and optimal variables are: $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \mathbf{w}^*, \mathbf{b}^*, \boldsymbol{\xi}^*$, we have:

$$p^* = f_0(\mathbf{w}^*) = f_0(\mathbf{w}^*) + \sum_{i=1}^{r} \lambda_i^* f_i(\mathbf{w}^*) + (\sum_{i=1}^{s} \nu_i^* h_i(\mathbf{w}^*) = 0)$$
$$\leq inf_{\mathbf{w}}(f_0(\mathbf{w}) + \sum_{i=1}^{r} \lambda_i^* f_i(\mathbf{w}) + (\sum_{i=1}^{s} \nu_i^* h_i(\mathbf{w}) = 0))$$

    - w^* minimize both $f_0(\mathbf{w})$ and $L(\mathbf{w}, \lambda^*, \nu^*)$ with Lagrange multiplier λ^* and ν^*
    - $\sum_{i=1}^{r} \lambda_i^* f_i(\mathbf{w}^*) = 0$, notice that $\lambda_i^* f_i(\mathbf{w}^*) \leq 0$ for each $i = 1, \ldots, m$(know as **complementary slackness**), which implies that $\lambda_i^* f_i(\mathbf{w}^*) = 0$.Therefore, there are only 2 situations:
      - $\lambda_i > 0,\ f_i(\textbf{w}^*)=0; i=1,...,m$
      - $\lambda_i = 0,\ f_i(\mathbf{w}^*) < 0; i = 1, \ldots, m$

- In the context of SVM with soft margin, we have the following complementary slackness (assume $C>0$):

$$\alpha_i^\star \left[y^{(i)} \left((x^{(i)})^T w^\star + b^\star\right) - 1 + \xi_i^\star\right] = 0,\ i = 1, \ldots, m,$$
$$\beta_i^\star \xi_i^\star = 0,\ i = 1, \ldots, m.$$

Combining with

- $y^{(i)}((w^\star)^T x^{(i)} + b^\star) \geq 1 - \xi_i^\star$,
- $\xi_i^\star \geq 0$,
- $C - \alpha_i^\star - \beta_i^\star = 0$,
- $\alpha_i^\star \geq 0$,
- $\beta_i^\star \geq 0$,

we have

- if $\alpha^\star = 0$

$$\begin{cases} \alpha_i^\star = 0 \\ C - \alpha_i^\star - \beta_i^\star = 0 \end{cases} \implies \beta_i^\star = C, \tag{88}$$

$$\begin{cases} \beta_i^\star = C \\ \beta_i^\star \xi_i^\star = 0 \end{cases} \implies \xi_i^\star = 0, \tag{89}$$

$$\begin{cases} \xi_i^\star = 0 \\ y^{(i)}((w^\star)^T x^{(i)} + b^\star) \geq 1 - \xi_i^\star \end{cases} \implies y^{(i)}((w^\star)^T x^{(i)} + b^\star) \geq 1; \tag{90}$$

- if $\alpha^\star = C$

$$\begin{cases} \alpha_i^\star = C \\ \alpha_i^\star \left[y^{(i)} \left((x^{(i)})^T w^\star + b^\star\right) - 1 + \xi_i^\star\right] = 0 \end{cases} \implies y^{(i)} \left((x^{(i)})^T w^\star + b^\star\right) - 1 + \xi_i^\star = 0, \tag{91}$$

$$\begin{cases} y^{(i)} \left((x^{(i)})^T w^\star + b^\star\right) - 1 + \xi_i^\star = 0 \\ \xi_i^\star \geq 0 \end{cases} \implies y^{(i)} \left((x^{(i)})^T w^\star + b^\star\right) \leq 1; \tag{92}$$

- if $0 < \alpha^\star < C$

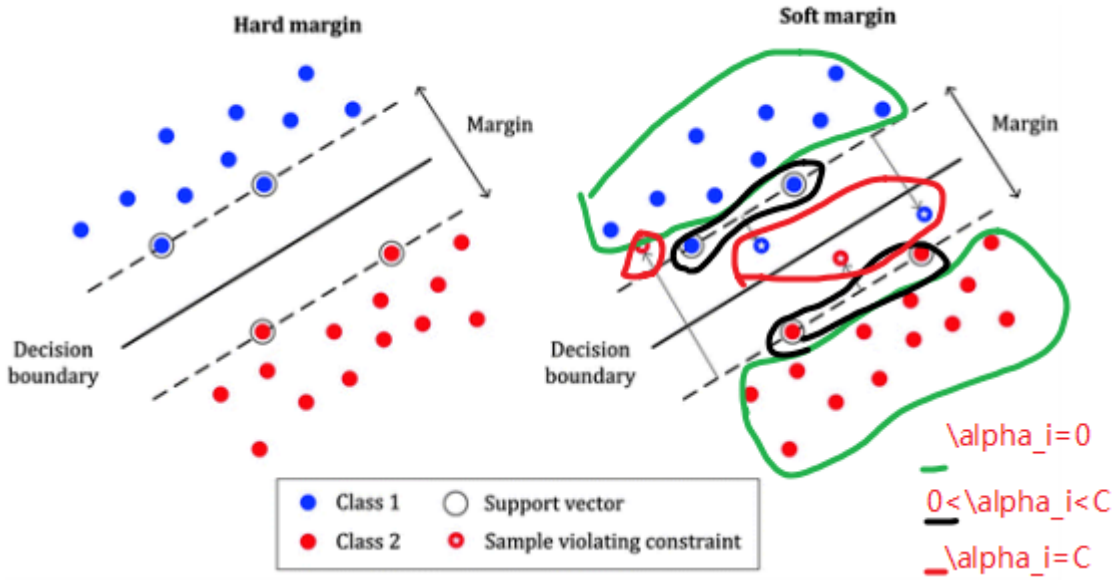
$$\begin{cases} 0 < \alpha^\star < C \\ C - \alpha_i^\star - \beta_i^\star = 0 \end{cases} \implies \beta_i^\star > 0, \tag{93}$$

$$\begin{cases} 0 < \alpha^\star < C \\ \alpha_i^\star \left[y^{(i)} \left((x^{(i)})^T w^\star + b^\star \right) - 1 + \xi_i^\star \right] = 0 \end{cases} \implies y^{(i)} \left((x^{(i)})^T w^\star + b^\star \right) - 1 + \xi_i^\star = 0, \tag{94}$$

$$\begin{cases} \beta_i^\star > 0 \\ \beta_i^\star \xi_i^\star = 0 \end{cases} \implies \xi_i^\star = 0, \tag{95}$$

$$\begin{cases} \xi_i^\star = 0 \\ y^{(i)}((w^\star)^T x^{(i)} + b^\star) - 1 + \xi_i^\star = 0 \end{cases} \implies y^{(i)}((w^\star)^T x^{(i)} + b^\star) = 1. \tag{96}$$

- Support Vectors:



- Why Bother SVM Dual
  - How would one classify a new point x with primal SVM solved
    - We need to explicitly compute the scalar product $\mathbf{w}^T x + b$
  - How would one classify a new point x with dual SVM solved
    - Very efficient as we only compute a few "support vectors" whose α_i's are nonzero

$$\mathbf{w}^T x + b = (\sum_{i=1}^{m} \alpha_i y^{(i)} \mathbf{x}^{(i)})^T \mathbf{x} + b$$

$$= \sum_{i=1}^{m} \alpha_i y^{(i)} (\mathbf{x}^{(i)})^T \mathbf{x} + b$$

- More importantly, the dual form lends itself easily to the *kernel trick*