(3) ■ **Theorem** (*No Free Lunch*): No single ML model that is optimal across every dataset; ~snowflake (unique) → no 1 model ∀ snowflakes.
■ **Curse of Dimensionality**: As features/covariates increase, data points become sparse in high-dimensional space, making it harder to generalize and requiring exponentially more data to cover the space adequately; many algorithms scale poorly with dimensionality (increase computational costs, etc); in high dimensions distances between points tend to converge (b/c sparsity); overfitting becomes an issue as high dimensional data often contains a lot of redundant features, which can lead the model to fit the noise.
■ **KNN Regression**: Predict the target value by averaging the values of the K nearest neighbors (training points). Choose K ($K = 1$ interpolation/complex, bias=0, var=↑↑; $K\uparrow$ = less complex, bias↑↑, var=↓). Poor results in high dimensions (curse of dimensionality).
■ **Regression**: Goal = estimate true model $E[Y|X] = f(x)$ with $\hat{f}(x_i)$. $y_i = f(x_i) + \varepsilon_i$, $\varepsilon_i \sim^{iid} N(0,\sigma_\varepsilon^2) \leftrightarrow P(Y|X) \sim N(f(X),\sigma_\varepsilon^2)$.
Derivation of MSE → MLE → $\max_f \frac{1}{(2\pi\sigma^2)^n} \exp\left\{-\frac{1}{2n\sigma^2}\sum_{i=1}^n (y_i - f(x_i))^2\right\}$ equivalently $\max_f -\frac{1}{2n\sigma^2}\sum_{i=1}^n (y_i - f(x_i))^2$ (drop some factors and take proportionality and it is the same as) $\min_f \frac{1}{2n}\sum_{i=1}^n (y_i - f(x_i))^2 \Leftrightarrow \min_f \frac{1}{2}||y - f(x_i)||_2^2$. ■ centering ys and xs is equivalent to fitting an intercept!
■ **Empirical Risk Minimization**: refers to minimizing any loss, risk, or error function (which could be MSE, cross-entropy, etc.) over a sample of data. Loss of MSE is equivalent to risk of Gaussian distribution $MSE = E\left[\left(f(x) - \hat{f}(x)\right)^2\right]$
■ **Population risk**: (not just the training – this is empirical risk) is taken wrt the population. ■ MSE decomp into bias; variance below
$E\left[\left(f(x) - \hat{f}(x)\right)^2\right] = E\left[\left((f - E[\hat{f}]) - (\hat{f} - E[\hat{f}])\right)^2\right] = E\left[(f - E[\hat{f}])^2\right] - 2E[(f - E[\hat{f}])(\hat{f} - E[\hat{f}])] + E\left[(\hat{f} - E[\hat{f}])^2\right]$
$= E\left[(f - E[\hat{f}])^2\right] + E\left[(\hat{f} - E[\hat{f}])^2\right] = \{Bias(\hat{f})\}^2 + Var(\hat{f})$. w/ $y = f + \varepsilon$, add σ_ε^2. Notes: $E[E[Y|X]] = E[Y]$; $E[f] = f$ (constant).
■ **Bias-Variance Tradeoff**: Overfitting is low bias, high variance. Underfitting is high bias, low variance. Bias-Var decomp is exact for MSE loss but is approximate, proportional decomp for most other losses used in ML. Formula: $MSE = Bias^2 + Variance$.
■ **OLS**: $f(x_i) = \beta_0 + x_i^T\beta$, $\beta \in \mathbb{R}^p$ (coef for features) - $\{y = X\beta + \varepsilon\}$. Solve $\min_\beta \frac{1}{2}||y - X\beta||_2^2$ (empirical MSE) for closed form solution.
■ **OLS Proof**: Use rule, $\nabla_x(Ax - b)^T(Ax - b) = 2A^T(Ax - b)$; $\min_\beta \frac{1}{2}(y - X\beta)^T(y - X\beta) \to \frac{\partial}{\partial\beta}\left\{\frac{1}{2}(X\beta - y)^T(X\beta - y)\right\} = 0 \to X^T(X\beta - y) = 0 \to$ $\hat{\beta}_{LS} = (X^TX)^{-1}X^Ty$ where $\hat{y} = X\hat{\beta}_{LS} = X(X^TX)^{-1}X^Ty$ and $X(X^TX)^{-1}X^T = H$ = Hat Matrix .
■ **Bias-Variance (OLS)**: $E[\hat{\beta}_{LS}] = E[(X^TX)^{-1}X^Ty] = (X^TX)^{-1}X^TE[y] = (X^TX)^{-1}X^TE[X\beta + \varepsilon] = (X^TX)^{-1}X^TX\beta = \beta$;unbiased if pop. model is linear, $y = X\beta + \varepsilon$; $Var(\hat{\beta}_{LS}) = Var((X^TX)^{-1}X^Ty) = (X^TX)^{-1}X^T\sigma_\varepsilon^2 X(X^TX)^{-1} = (X^TX)^{-1}\sigma_\varepsilon^2 X^TXX^{-1}(X^T)^{-1} = (X^TX)^{-1}\sigma_\varepsilon^2$.
■ **Gauss-Markov Theorem**: From all linear unbiased estimators of β, $\hat{\beta}_{LS}$ has the minimum variance best linear unbiased estimator, BLUE. As a result the LS has the best MSE. Limitations (even though bias, variance may be large! Want some bias for lower variance – regularize).
■ **Q**: When will $Var(\hat{\beta}_{LS})$ explode? (1) Features are super correlated; X^TX is not full rank (collinearity). (2) $p \gg n$ (high dim) – not unique, inf sol.; X^TX is not full rank; observe linearly dependent features. Property of high dimensional data is extreme correlation between features. (multicollinearity).
■ **C**: When $p > n \to$ training error $(\hat{\beta}_{LS}) = 0$ and $\hat{\beta}_{LS}$ is an interpolator. **Proof**: $\hat{y} = Hy$; $H = X(X^TX)^{-1}X^T$ $\{\text{if } p > n\} = I \to \hat{y} = y$! How to compute LS when $p > n$: gradient descent to get A solution from inf many. Approx. tangent line and go downhill iterative until min.
■ **Lab**: Fit LS model with 10 features. Get p-value low for 2 and other 8 high; can we drop the other features and take these two? No. These p-values are conditional! They are for the coefficients assuming that the other features are fixed, i.e., significant conditional on other features, unsure significance alone!

(4) ■ **Q**: in LS what happens to two correlated features for $x_i + x_j \to \beta_i, \beta_j$?■ **A**: Interpretation becomes difficult and coefficients become unstable = high variance, i.e., if β_i ↑↑, β_j ↓↓ to offset each other. ■ **Q**: How can we improve LS when $p > n$ or we have correlated features? ■ **A**: Regularize! Apply some conditions on β to stabilize the parameters (or filter or select features). We add a little bias in the hope of reducing the variance.
■ **Convex Set**: s.t. a line connecting any two points in the set lies wholly in the set. Any convex function has polynomial time algorithm for global solution.
■ **Convex**: Optimization where if any two points on function and connect them the line lies above the function U → has minimum (strictly=unique). When a set is not convex it is hard to go from primal to Lagrange form therefore a convex property is very desirable!
■ **Ridge Primal form** (strictly convex): $\min_\beta \frac{1}{2}||y - X\beta||_2^2$ subject to $||\beta||_2^2 \le \tau$; τ =hyperparameter. $\tau \to 0$, constrained soln, $\tau \to \infty$, LS soln.
■ **Lagrange form** (strictly convex): $\min_\beta \frac{1}{2}||y - X\beta||_2^2 + \frac{\lambda}{2}||\beta||_2^2$ (regularized empirical risk minimization); λ =hyperparameter. $\lambda \to 0$, LS soln, $\tau \to \infty$, constrained soln. Both forms are equivalent meaning there is a one-to-one mapping from τ to λ. ■ **Solution**: $\{\nabla_\beta||\beta||_2^2 = \beta\} \to \nabla_\beta = X^T(X\beta - y) + \lambda\beta = 0 \to X^TX\beta - X^Ty + \lambda\beta = 0 \to (X^TX + I\lambda)\beta = X^Ty \to \hat{\beta}_{ridge} = (X^TX + \lambda I)^{-1}X^Ty$; $\lambda > 0 \to (X^TX + \lambda I)^{-1}$ exists + always invertible! $\hat{\beta}_{ridge}$ is the unique global solution even when $p > n$ (where LS broke) b/c strictly convex. ■ **Notes**: need intercept and intercept should not be regularized (so it doesn't get forced to zero). ■ How do we fit and not regularize? We center ys and xs before ridge (cannot fit column of 1s because it regularizes the intercept!!).
■**Q:** What about the scale of X_j's (we care about with LS and ridge) because if the X_j's are on different scales then λ penalizes each X_j differently (features with high variance get penalized more)!! ■**A**: ALWAYS scale the features $(X_{ij} - \bar{X}_{ij})/[\widehat{Var}(X_{ij})]^{1/2}$ before applying regularization (default in software)!
■ **Ridge Decomp**: Is $\hat{\beta}_{ridge}$ MSE good? Bias-var decomp→ $E[\hat{\beta}_{ridge}] = E[(X^TX + \lambda I)^{-1}X^Ty] = (X^TX + \lambda I)^{-1}X^TE[y] = (X^TX + \lambda I)^{-1}X^TX\beta = (X^TX + \lambda I)^{-1}(X^TX + \lambda I - \lambda I)\beta = (X^TX + \lambda I)^{-1}(X^TX + \lambda I)\beta - \lambda(X^TX + \lambda I)^{-1}\beta = \beta - \lambda(X^TX + \lambda I)^{-1}\beta$; bias $= -\lambda(X^TX + \lambda I)^{-1}\beta$. $Var(\hat{\beta}_{ridge}) = Var((X^TX + \lambda I)^{-1}X^Ty) = (X^TX + \lambda I)^{-1}X^T\sigma_\varepsilon^2 X[(X^TX + \lambda I)^{-1}]^T \to$ ■ **Corollary**: $Var(\hat{\beta}_{ridge}) < Var(\hat{\beta}_{LS}), \forall\lambda > 0$.
■ **MSE Existence THM**: ∃ a $\lambda > 0$, s.t. $MSE(\hat{\beta}_{ridge}) < MSE(\hat{\beta}_{LS})$. Ridge is always better than LS in terms of MSE of prediction error!
■ **Properties Ridge**: (1) Ridge is great for prediction (2) (good w/ corr features) Ridge pushes correlated features together (groups), i.e., $\lambda\uparrow$; $Cor(\beta_j, \beta_k)\uparrow$, then $\beta_j \approx \beta_k$. Ridge does well in correlated settings, where LS pushes these features apart (same with LASSO). (3) Ridge shrinks large patterns in X less than small patterns (ridge is like doing LS on denoised data!), thus ridge does well in high noise settings. (ridge never shrinks coefficients to EXACTLY 0).

(5) ■ **Q**: Where does $||\beta||_2^2$ come from? **A**: Bayesian Interpretation→ $P(Y|X,\beta) \sim N(X\beta, \sigma_\varepsilon^2 I)$; $P(\beta) \sim N(0, \tau^2 I)$ (prior); find $P(\beta|X,y)$ (posterior). Using Bayes THM, $P(\beta|X,y) \propto P(Y|X,\beta)P(\beta) \propto \exp\left\{-\frac{1}{2\tau^2}\beta^T\beta\right\}\exp\left\{-\frac{1}{2\sigma^2}||y - X\beta||_2^2\right\}$. Maximum A Posteriori Estimator (MAP) (bayes version of MLE) - $\max_\beta Posterior$ equivalent to maximizing log posterior, $\max_\beta\left\{-\left(\frac{1}{2}||y - X\beta||_2^2 + \frac{\sigma^2}{\tau^2}||\beta||_2^2\right)\right\}$ where $\lambda = \frac{\sigma^2}{\tau^2}$ max - = min + → $\min_\beta \frac{1}{2}||y - X\beta||_2^2 + \frac{\lambda}{2}||\beta||_2^2$.
■ **Q**: How to improve LS when $p > n$? Use ridge, filter or select k features where $k \ll p$. If $p > n$ some features are noise/redundant ~ALWAYS holds.
■ **Feature Filtering vs Selection**: **Filtering** (univariate) removes features based on marginal associations (e.g. retain features $|Cor(X_j, y)| \ge \tau$). This is univariate method. This approach is good for independent-ish features but breaks down under multicollinearity. **Selection**: Considers multivariate models to find the best subset of features. This is better when features are not independent. **THM**: IF features are independent, THEN $filter = selection$.
■ **Best Subsets**: Find the best $k \ll p$ features for fitting a model, $\min_\beta \frac{1}{2}||y - X\beta||_2^2$ subject to$\sum_{j=1}^p \mathbb{1}\{\beta_j \ne 0\} \le k$; ($||\beta||_0 = l_0$ norm [not proper norm – triangle inequality does not hold]). Sum of non-zero βs. ***Q:*** How to solve? Not convex, need brute force fit over all $\binom{p}{k}$ subsets (NP-hard). $\sum_{k=0}^p \binom{p}{k} = 2^p$.
■ **Stepwise Selection Algorithm**: (Greedy algorithm; suboptimal) Iterative/stepwise algo that makes optimal decision regardless of past/future steps.
■**Forward Selection**: Start w/ β_0 + add single best feature, …, continues to add the single best feature given all other previous features in the model.
■**Backwards Selection**: Starts with $\hat{\beta}_{LS}$ soln + eliminates the single worst feature, (refits) and continue iteratively. Recursive Feature Elimination (RFE) same as backwards selection but if p is really large it just eliminates r features at a time in order to improve efficiency of algorithm. (stop according to AIC)
■ **Q**: How do we find the best or worst features in the model? P-values/MSE (not always the best), R^2, AIC/BIC (low is good; BIC is stricter than AIC).
■ ***LASSO* (L_1 norm)**: Want to use the best convex relaxation of the nonconvex L_0 penalty [i.e., solve a "relaxed" best subsets problem so we can optimize]. The l_1 norm is the best convex relaxation of l_0 norm.
■ **LASSO Primal/Lagrange**: $\min_\beta \frac{1}{2}||y - X\beta||_2^2$ sub to $||\beta||_1 \le \tau \leftrightarrow \min_\beta \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1$. Convex! Poly time algo to compute best global soln.
$\lambda \to 0$, $\hat{\beta}_{LASSO} = \hat{\beta}_{LS}$; λ ↑↑, $\hat{\beta}_{LASSO} \equiv 0$ (this is called λ_{max}, i.e., ∃ a value of λ s.t. $\hat{\beta}_{LASSO} \equiv 0$); hyperparameter tune $\forall\lambda \in [0, \lambda_{max}]$ on log-scale.

(6) ■ **LASSO Optimization**: Gradient→ l_1 norm is non-differentiable. Need iterative solver. LASSO signal approximator (no X, no features): $\min_\beta \frac{1}{2}||z - \beta||_2^2 + \lambda||\beta||_1$, $z \in \mathbb{R}^p$. Take gradient and set to 0 → $\frac{\partial}{\partial\beta} = -z + \beta + \lambda\Gamma(\beta) = 0$; sub gradient $= \Gamma(\beta) = \begin{cases} sign(\beta), \beta \ne 0 \\ [-1,1], \ \beta = 0 \end{cases}$. ■ **Projected Gradient Descent**: Go downhill on the differentiable part (MSE) + project onto the nondifferentiable part (l_1 norm ball ; soft-thresholding).
■ **LASSO Solution**: Show that for $\lambda \ge \lambda_{max}$, $\hat{\beta}_{LASSO} \equiv 0$. Optimize $\min_\beta \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1$ by $\nabla_\beta = -X^Ty + X^TX\beta + \lambda * Sgn(\beta)$. If we take a neg of the gradient we get a vector that points in the direction of steepest descent $-\nabla_\beta$, if λ is sufficiently large the Sgn term becomes large and $\hat{\beta}_{LASSO} \equiv 0$.
■ **Properties of LASSO**: (1) include intercept but do not regularize (do this by centering y and standardizing features before fitting). (2) Sparsity (Parsimony); sparse improves Interpretability; Model Compactness (genomics, don't want to measure 50k genes but only a few genes!). (3) LASSO better then OLS but worse than Ridge (unless true underlying data is sparse). (4) LASSO tends to select one feature of a correlated group. Selection consistency, i.e., under what conditions will LASSO select the correct features with high probability → only correct features under limited amounts of correlation. Interpret LASSO features with caution; they will be sparse but probably not the correct features! So bad under high correlation… (4) Regularization paths are piecewise linear; the paths only change direction when a feature enters or exits the model (b/c fitting in multivariate manner rather then univariate).
■ **Elastic Net**: ("rubberband" between ridge and LASSO). $P(\beta) = \alpha||\beta||_1 + (1 - \alpha)||\beta||_2^2$, $\alpha \in [0,1]$; $\alpha = 0$ (ridge), $\alpha = 1$ (LASSO). Improves LASSO with correlated features. ■ **Adaptive LASSO**: $P(\beta) = \sum_{j=1}^p w_j|\beta_j|$; we can find w_j's that induce less bias (estimate via Ridge or LS).
■ **Group LASSO**: Good for categorical features. ■ **Exclusive LASSO**: selects a single feature from a group (opposite of group LASSO).

(7) ■ **Non-Linear**: Fit non-linear using above methods? Basis expansion $\tilde{X}_{n\times q}$. Fit a linear model on $\tilde{X}_{n\times q}$. Downside = $\tilde{X}_{n\times q}$ dimensions huge $q \gg p$. Computationally burdensome. What basis to choose? ■ **Major Non-Linear Methods** (that can be fit using linear models): (1) Splines – univariate basis expansion on each feature. (2) Kernels – multivariate basis expansion.
■ **Kernel Regression**: Infinite dimensional basis expansion. $f(x) = \sum_{i=1}^\infty c_i\phi_i(x)$ where [$\phi_i(x)$ basis function; c_i coefficients].
■ **Optimization**: $\min_f \frac{1}{2}||y - f(x)||_2^2 + \lambda P(f(x))$ where $P(f(x))$ penalizes the "wiggiliness" of $f(x)$. $\lambda \to 0$, interpolator (as wiggly as possible); $\lambda\uparrow$ less curvature; λ ↑↑↑, intercept plane.
■ **Mercer's Representor THM**: Holds iff $f() \in \mathcal{H}$(Reproducing Kernel Hilbert Space). Says that even if $f()$ is infinite dimensional, we only need to evaluate $f()$ at the n training points!! Takes $\infty - dim$ to $n - dim$ for computation this is required! (intuition: regardless of the fit "how wiggly" we only need to compute the MSE at n training points to evaluate MSE; allows us to fit functions as wiggly as we want in polynomial time).
■ **Kernel Ridge**: $f(x) \in \mathcal{H}$, $f(x) = \eta(x)\beta$ where $\eta(x)$ is some ∞ dimensional basis expansion. $\min_\beta \frac{1}{2}||y - \eta(x)\beta||_2^2 + \lambda\beta^T\beta$ (ridge) . $\beta = \eta(x)^T\alpha$, $\alpha \ni \mathbb{R}^n$ (by Mercers THM) where $\eta(X)$ is a matrix of dimensions $n \times \infty$; → $\min_\beta \frac{1}{2}||y - \eta(X)\eta(X)^T\alpha||_2^2 + \lambda\alpha^T\eta(X)\eta(X)^T\alpha$ where $\mathrm{K}_{n\times n} = \eta(X)\eta(X)^T$ (Kernel Matrix); $k(x_i, x_i') \triangleq \eta(x_i)^T\eta(x_i')$; {inner product $n \times n$ dimensional}; $= cov(f(x_i), f(x_i'))$. The elements row i, col i' in $\mathrm{K}_{n\times n}$ is just equal to $k(x_i, x_i')$, i.e., $\mathrm{K}_{ii'} = k(x_i, x_i')$ {**Kernel Trick**} ; these are called kernels. Therefore, $\min_\beta \frac{1}{2}||y - \mathrm{K}\alpha||_2^2 + \lambda\alpha^T\mathrm{K}\alpha$. NOTE: λ will regularize the kernels.
■ **Optimize Kernel Ridge**: $\frac{\partial}{\partial\alpha} = -\mathrm{K}(\mathrm{y} - \mathrm{K}\alpha) + \lambda\mathrm{K}\alpha = 0 \to \hat{\alpha} = (K + \lambda I)^{-1}\mathrm{y}$; $\hat{y} = \mathrm{K}\hat{\alpha}$. Nice global closed form solution from infinite dimension basis! Prediction on a new point x^*; $\hat{f}(x^*) = \sum_{i=1}^n k(x_i, x^*)\hat{\alpha}_i$; kernel inner product of x^* with all training points; how we make a prediction for a new point. $\hat{\alpha}_i$ tells us which training points are most influential in kernel space.
■**Key takeaway**: Kernel trick allows us to solve for a nonlinear function that is potentially infinite dimensional with only n computational time.
■ **Example Kernels**: Example 1>> $k(x_i, x_i') = x_i^Tx_i$ (linear). Example 2 >> $k(x_i, x_i^T) = (a + x_i^Tx_{i'})^d$ (d^{th} order polynomial). What d? limited in flexibility for low d. **Advantage**: (1) computationally attractive way to do non-linear regression. (2) There is a lot of flexibility in options for kernels.

■ **Choose Kernels**: (a) symmetric, $k(x,z) = k(z,x)$. (b) Positive semi-definite $K \ge 0$, $\alpha^TK\alpha \ge 0$, $\forall\alpha$. (3) Any nonlinear functions of inner products, norms, or distances.
■ **Radial Basis Function** (Gaussian Kernel): $k(x,z) = \exp\{-\varphi||x - z||_2^2\}$; $-\varphi$ like inverse variance. φ ↑↑=interpolator; . φ ↓↓=intercept. FLEXIBLE!!
■ **Gaussian Processes**: Bayesian Version of Reproducing Kernel Hilbert Space. $p(y|X,f) \sim N(f(x), \sigma^2 I)$ and prior $p(f|X) \sim N(0, K(x,x))$. Fit posterior $p(f|X,y) = \frac{p(y|X,f)p(f|X)}{p(y|X)}$ (posterior=likelihood*prior/marginal)→ $p(f|X,y) \sim N(K(K + \sigma^2 I)^{-1}y, (K + \sigma^2 I)^{-1})$; the mean is just kernel ridge $\hat{\alpha} = (K + \lambda I)^{-1}\mathrm{y}$ with $\lambda = \sigma^2$; ■ **Key Takeaways**: The assumption on the prior of f is that the kernels give covariance and the Kernel Ridge Regression is just the MAP estimator of a Gaussian Process Regression.

(8) ■ **Kernel Trick**: To (not work in infinite dim space) work with covariance between infinite dimensional basis functions = $k(x,z)$ [the kernel!!!].
■ **ML Process/Pipeline**: Data Science Pipeline=Question – Data – Wrangling – Exploration/Pre-processing – Modeling (ML) – Visualize – Communicate.
■ **ML Modeling Process**: Select ML model family, evaluation metric, training, tune hyperparameters, report test prediction error.
■ **Notes:** Need new unseen data (test set); report how well our model generalizes. Goal of prediction vs interpretation changes ML modeling process.
■ **Validation**: For large n (big data), use validation set for hyperparameter tuning. For small n, use CV. Randomly split data into **training** (train all models on all hyperparameters), **validation** (evaluate all the models trained on all hyperparameters and choose optimal hyperparameter that minimizes MSE); after hyperparameter tuning on the validations set, we can retrain the model on the training and validation set (to not waste data)., **test** (test on the model with optimal hyperparameter; report prediction error). OPTIONAL – **Query** set which is used to choose between many ML model families (and find the best performing model – we can overuse the validation set so it becomes not unseen and thus query set is required); helps prevent overfitting.
■ **Example**: Consider selection between LASSO (λ) and Ridge (α), $\hat{\lambda}_{optimal} = \operatorname{argmin}_\lambda MSE(X^{Val}\hat{\beta}_{(\lambda)}^{LASSO}, y^{val})$ $\hat{\alpha}_{optimal} = \operatorname{argmin}_\alpha MSE(X^{Val}\hat{\beta}_{(\alpha)}^{ridge}, y^{val})$. To find this we train models on all hyperparameters and evaluate on validation set. Then we retrain final model on (train+val) and evaluate on test set. When training it is okay to use data we have seen before; this is not true for test (as we would be overfitting to the test set). What to do if limited amounts of data and cannot fully set aside a validation set? **Cross Validation**: Reuse random folds of data for training while keeping some unseen for validation.
■ **K-Fold CV**: Randomly split (idea everything is at random then the expectation will be the same; if there are rare classes need **stratified** random sampling, so each fold has rare class) training data into K folds (approx. equal size). Fit K different models to $(K - 1)/K$ training data and validate on $1/K$ data (every time we train did not see Kth chunk, so it is valid to use as validation). We report the average CV error = average prediction error / k-folds.
■ **Graph Curves**: Cross validation error tends to be an overestimate of the test error curve because it was trained and evaluated on less data, so it is always a bit of an overestimate; however, the shapes of CV error and test error are typically the same so it is good for selecting hyperparameters.
■ **Min-Rule**: $\lambda^* = \operatorname{argmin}_\lambda \overline{CV_{err(\lambda)}}$; select bottom of curve. Can be suboptimal because some variability in estimates of $CV_{err(\lambda)}$; Since we select optimal hyperparameter λ^* based on $\overline{CV_{err(\lambda)}}$ is k is small then there can be large variability in the estimates (i.e. mean with small number varies a lot). Furthermore, we can see a flat bottom of the curve so any λ in the range of the flat bottom is approximately the same! So intuitively leads to One-SE rule.
■ **1 SE Rule**: (plot CVerror curve and it is good for flat bottom curve) $\lambda^{*\,1SE} = \operatorname{argmin}_\lambda \overline{CV_{err(\lambda)}} < \overline{CV_{err(\lambda^*)}} + SE(\overline{CV_{err(\lambda^*)}})$. Choose the least complex model such that the average CV error is less than that of the minimum + 1 standard error of the mean. ■ **Example**: LASSO, want largest λ^* s.t. within 1SE to reduce complexity. ■ **Notes**: CV is stochastic (every time we perform get slightly different results). Which K to use in K-fold CV? $K = n \to$ LOOCV training on $n - 1$ points and testing on 1 left outpoint (computationally intensive; training sets are very correlated, doesn't shake up training sets enough – good for timeseries data). Typically, $k = 5{,}10$ (nice whole numbers and good rule of thumb).

(9) ■ **Grid search**: GOOD, CV on grid log scale for e.g. ~100 $\lambda \in [0, \lambda_{max}]$. ■ **Random search**: LEAST PRECISE, randomly try out some values of hyperparameter for $\lambda \in [0, \lambda_{max}]$. This if fastest; used with lots of hyperparameters to consider, i.e., 50-100 hyperparameters (a grid search is too computationally intensive).■**Binary search**: MOST PRECISE finetuning, take 2 values, middle? Another search outside?... widdle down until optimal.
■ **Validation for Interpretation**: Will the same strategies used for validating prediction work on interpretation? NO. Prediction error not the best for evaluating interpretation. **Q** How to validate interpretability? **Stability** – stable or reliable interpretations under many random perturbations of training data are likely true interpretations. If stable, it is likely to be a true feature. ■ (1) random perturbations can be done using bootstrapping (sample n obs w/ replace). (2) generalizability; interpretations should generalize well to new data; should be able to predict our interpretations (get interp from tr then pred ts)
■ **e.g.**, LASSO → select the best features > predictive ability. Choosing λ via CV almost always OVERSELECTS the true number of features.
■ **Generalizability e.g.**: select best features on TR and determine most stable features, then fit LS model w/ best features for 1,….,k stable features and report test MSE on each model and get curve to determine true # of features.
■ **Interpretability Statistical Inference**: Cannot use classical statistical inference because ML breaks all rules. Did not prespecify any model before looking at the data but looked at data to determine which regularization we want such as LASSO, etc. Can solve w/ data splitting (Tr,Val,Ts).

(10)■ **Classification**: Outcomes are labels not continuous. Binary $y_i \in \{0,1\}$, SVM $y_i \in \{-1,1\}$. Hard labels {0,1} and soft labels $P(y = 1) = \pi(y) \in [0,1]$. Multiclass $y_i \in \{1, \dots, K\}$. Why not fit MSE loss to $y_i \in \{0,1\} \to$ Brier Scoring… Not the best for interpretation reasons (and b/c multiclass case using MSE loss is very problematic if y is coded as classes). ■ **Discriminative Classifiers (conditional models)**: learn a decision boundary to separate the classes (learn some $f(x)$ not to predict y but to find some decision boundary).■ **Generative Classifiers (Bayesian)**: Assume a probability model for $X|y$, assume $p(X|y = k)$ and learn $p(y = k|X)$ *bayes thm* .
■ **Accuracy**: $misclassification = 1 - Accuracy$; $Accuracy = \frac{TP+TN}{TP+TN+FN+FP} = proportion\ correct$ (good for binary classification).
■ **Receiver Operating Characteristic (ROC curve)**: Plots True Positive Rate (sensitivity = $TPR = TP/(TP + FN)$) vs False Positive Rate (1 – specificity = $FPR = FP/(FP + TN)$).■ **Area Under Curve (AUC statistic)**: Integrate area under ROC curve. Good error metrics for soft labels
■ **Logistic Regression**: Idea = nonlinear transform to y in order to fit a linear model $f(x)$. Recall linear regression model, $E[Y|X] = x^T\beta \in \mathbb{R}$ but now we want to take this to lie within [0,1]. $E[\text{link}(y|X)] = x^T\beta \in \mathbb{R}$ s.t. link is a non-linear transform. Logistic: $E[\text{logit}(y|X)] = x^T\beta \in \mathbb{R}$ s.t. logit=log-odds . $\log\left[\frac{P(y=1|X)}{P(y=0|X)}\right] = x^T\beta$ (let $p(x) = P(y = 1|X)$) thus $\log\left[\frac{p(x)}{1-p(x)}\right] = x^T\beta \to p(x) = (1 - p(x))\exp\{x^T\beta\} \to p(x) = \frac{\exp\{x^T\beta\}}{1+\exp\{x^T\beta\}} = \frac{1}{1+\exp\{-x^T\beta\}}$.
As $x \to \infty$, $p(x) = \frac{1}{1+\exp\{-X^T\beta\}} \to 1$; As $x \to -\infty$, $p(x) \to 0$. ■ Do we need an intercept? Yes, the intercept changes the $p(x) = 0.5$ cutoff. This is done in logistic regression by including a column of 1s $\tilde{X} = [\vec{1} \quad X]$; cannot center because $y \in [0,1]$!! $p(x) = \frac{1}{1+\exp\{-\beta_0 - X^T\beta\}}$ w/ intercept but we just col 1s.
Predictions $= round(\hat{p}(x^*)) \in \{0,1\}$; soft → hard labels.
■ **Sigmoid Function**: $p(x) = \sigma(x) = \frac{1}{1+\exp\{-X^T\beta\}}$ is called the sigmoid function; this function squashes any real-values number into the range [0,1].
If $\beta > 0$, S-shaped slope. If $\beta < 0$, Ƨ-shaped slope. If $\beta = 0$, – flat intercept shape (on 0.5). Larger magnitude β the steeper the slope. Changing intercept! If $\beta_0 > 0$ the curve shifts left, if $\beta_0 < 0$ the curve shifts right.
■ **Decision Boundary**: $f(x) = x^T\beta + \beta_0$, e.g. $P(y = 1|X) = P(y = 0|X) = \frac{1}{2}$, $\log\left[\frac{P(y=1|X)}{P(y=0|X)}\right] = \log[1] = 0 \to x^T\beta + \beta_0 = 0$ (linear hyperplane).
■ **Logistic Regression Derivation**: Conditional model Bernoulli. $P(y = 1|X) \sim Bernoulli(p(x))$; $P(y|X) = p(x)^y(1 - p(x))^{1-y} = \exp\left\{\log p(x)^y(1 - p(x))^{1-y}\right\} = \exp\{y\log p(x) + (1 - y)\log(1 - p(x))\} = \exp\left\{y\log\left(\frac{p(x)}{1-p(x)}\right) + \log(1 - p(x))\right\}$ canonical form of exponential family.
This is a generalized linear model (GLM) so, $E[cannonical\ param(y|X)] = \beta_0 + x^T\beta$. Logistic Regression assumes a Bernoulli conditional model.
■ **Information Theory**: Logistic Regression loss function as Cross-Entropy (CE) = loss when using discrete probability distribution q() when real is p(). $CE(\hat{y}, y) = -y\log\hat{y} - (1 - y)\log(1 - \hat{y})$ where $\hat{y}$ is predicted hard label or soft label. For Bern (Logistic) Negative log likelihood = CE Loss .

---

(11)■ **Notes**: Hard label error metric = $1 - accuracy = misclassification$ is good. Soft label error metric = ROC curve and AUC summary statistic.
■ **Fit/Estimate Logistic Regression**: log likelihood is $l(\beta) = \sum_{i=1}^n[y_i\log p(x_i) + (1 - y_i)\log(1 - p(x_i))] = \sum_{i=1}^n[y_ix_i^T\beta - \log(1 + \exp\{x_i^T\beta\})] = yX\beta - \log(1 + \exp\{X\beta\})$. Do MLE → $\max_\beta l(\beta)$. $\nabla_\beta\{yX\beta - \log(1 + \exp\{X\beta\})\} = X^Ty - \frac{1}{1+\exp\{X\beta\}}X^T\exp\{X\beta\} = X^T\left(y - \frac{\exp\{X\beta\}}{1+\exp\{X\beta\}}\right) = X^T(y - p(X))$.
Now if we set the gradient to 0, there is no nice closed for solution as the sigmoid function makes things complicated [need iterative solver = gradient descent (very slow because sigmoid is flat at many places!) ; Newtons method = much faster iterative solver].
■ **Optimization**: $\max_\beta l(\beta) = \min_\beta -l(\beta)$ is convex and yields a global solution using iterative solver (newtons method).
■ **Multi-Class**: $y_i \in \{1, 2, \dots, K\}$ (K classes). Logistic Regression is binary $y_i \in \{0,1\}$. Therefore to still use Logistic Regression we can use **One Vs Rest (OVR)** which just fits each class k vs not class k, i.e., one class to the rest of the data points (so K separate logistic regressions); **One Vs One (OVO)** which just fits each features against another; $\binom{K}{2}$ models. MULTINOMIAL DECISION BOUNDARIES ARE LINEAR!!!
■ **Multinomial Model**: Conditional model $P(y = k|X) \sim Multinomial(p_1(x), \dots, p_K(x))$. ■ **Generalized Linear Model (Multi-Class)**: Multinomial derivation: $\log\left[\frac{P(y=k|X)}{P(y=K|X)}\right] = X\beta^k$, $k = 1, \dots, K - 1$, this is just the log odds of class k compared to last class, class K. Not a nice model! Bad interpretation!!
■ **Soft-Max (K-Formulation)**: $P(y = k|X) = \frac{\exp\{x^T\beta^{(k)}\}}{1+\sum_{m=1}^K \exp\{x^T\beta^{(m)}\}}$ defined for each class, $\beta^{(k)} \in \mathbb{R}^p$; $k = 1, \dots, K$. The sigmoid function here is = Soft-Max. Numerator = $\exp\{x^T\beta^{(k)}\}$ probability of class k ; Denominator = $1 + \sum_{m=1}^K \exp\{x^T\beta^{(m)}\}$ probability of all classes; so $P(y = k|X)$ is normalizing in [0,1]. The soft-max function is affiliated with soft labels, e.g. $K = 5$ and $\hat{p}_{k=1,\dots,5}(x) = [0.1, 0.1, 0.3, 0.1, 0.4]$ all sum to 1 and are probabilities of being in class k. Therefore, our predictions $\hat{p}_k(x^*)$, $k = 1, \dots, K$ as soft labels then convert to hard labels $\hat{y} = \operatorname{argmax}_k \hat{p}_k(x^*)$. ■ **Intuition**: give vector input (feature data to model) and hope to get classification of 1 of the K classes. The predictions are given as a probability for each class [0.1, 0.1, 0.3 ,0.1, 0.4] means class 3 and 5 are most likely to be identified from the input data!■ **Soft-Max Continued**: Loses the log odds interpretation; gains interpretability of $\beta's$ and $\hat{p}(x)$. We get a lot of parameters, $\vec{\beta}_k \in \mathbb{R}^p$ s.t. $B_{p\times K} = [\vec{\beta}_1 \ \cdots \ \vec{\beta}_K]$. $B_{j,k}$ is the weight of the j-th parameter for predicting the k-th class.
■ **Soft-Max Decision Boundary** (linear): Take all probabilities $P(y = k|X) = \frac{\exp\{x^T\beta^{(k)}\}}{1+\sum_{m=1}^K \exp\{x^T\beta^{(m)}\}} = 0.5$ and solve. Estimation via MLE (; Newton solver).
■ **Coef Interpretations**: e.g. take a prediction based on 3 age classes (0-20,20-40,40-60). We see that coefs for alcohol is (class1,class2,class3 coef respectively) $\beta^T = [0.13 \quad -1.48 \quad 1.35]^T$, therefore we can interpret this as alcohol is very good for separating class 2 and class 3!!
■ **Regularization on Logistic**: $\min_\beta -l(\beta) + \lambda P(\beta)$, $\lambda \ge 0$ and (l_1, l_2 as $\lambda \to 0$, Logistic; else lots of regularization). Why add regularization? Add bias to reduce variance with hopes of better generalization and prevent overfitting. Also address high dimensional settings $p > n$ as X^TwX is singular thus non-unique and difficult to compute. Regularization provides unique solutions with more stable computation (when features correlated, $p > n$, etc).
■ **Ridge/LASSO**: Ridge = [great for prediction, good w/ correlated features, numerical stability (sklearn)] ; LASSO = [features selection ; sparse $\hat{\beta}$].
■ **LASSO regularization path**: not piecewise linear for logistic regression but has curves (still performs selection and shrinks coef to 0).
■ **MOST IMPORTANT FEATURE ON REGULARIZATION PATH**: two methods (1) "enter path first", i.e., which coeff is nonzero first (2) $\max|\hat{\beta}_{CV}|$
■ **Multinomial Regularization (GROUP LASSO)**: LASSO feature selection is not useful on $B_{p\times K}$ but we want to shrink coefficients across classes .

■**KNN Classification**: K-Nearest Neighbors (KNN) is a non-parametric, instance-based learning algorithm that classifies a new data point based on the majority vote of its K nearest neighbors. To classify a point, the algorithm calculates the distance (e.g., Euclidean, Manhattan) to all existing data points, identifies the K closest neighbors, and assigns the most common class among them. A smaller K makes the model sensitive to noise (overfitting), while a larger K smooths the decision boundary (underfitting). KNN requires proper feature scaling and can be computationally expensive for large datasets but works well for small, well-structured data. For example, if K = 3 and the three nearest neighbors belong to classes [A, A, B], the new data point is classified as A (majority vote).
■ **KNN Regression**: In KNN regression, instead of assigning a class label, the algorithm predicts the target value by averaging (or weighting) the values of the K nearest neighbors. It follows the same process as KNN classification: finding the K closest points and computing the mean (or weighted mean) of their target values. KNN regression can model complex relationships without assuming a functional form, making it useful for non-linear problems. However, it is sensitive to outliers, requires careful choice of K, and performs poorly with high-dimensional data due to the curse of dimensionality. For example, if K = 3 and the target values of the three nearest neighbors are [10, 12, 14], the predicted value would be (10 + 12 + 14) / 3 = 12.

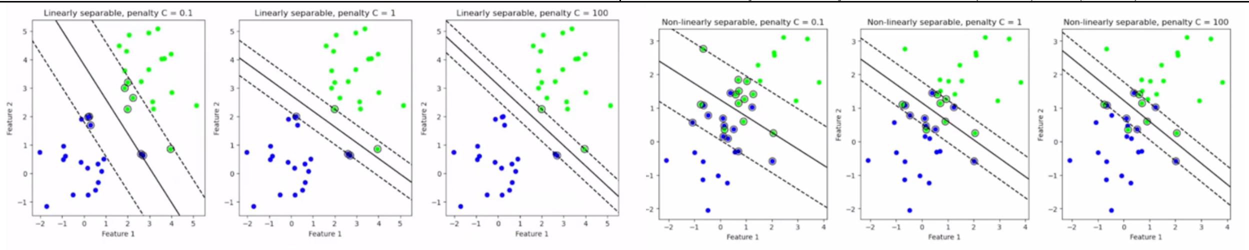
(12) ■ **Generative Classifiers**: Assume $P(X|y=k) \sim distribution$ a generating distribution for our data being in class k . Learn $P(y=k|X)$ using bayes theorem, $P(y=k|X) = \frac{P(X|y=k)P(y=k)}{P(X)}$. Classify to the most probable class, $\hat{y} = \underset{k}{\operatorname{argmax}} P(y=k|x^*)$.

■ **Naïve Bayes Classifier (Gaussian)**: $P(X|y=k) \sim N(\mu_k, \sigma^2 I)$, $\mu_k \in \mathbb{R}^p$ (centroids) and common variance across classes ; $P(y=k) = \pi_k$. **ASSUME ALL FEATURES ARE INDPENDENT**. Applying bayes→ $P(y=k|X) = \frac{\pi_k \exp\left\{-\frac{1}{2\sigma^2}\|x-\mu_k\|_2^2\right\}}{\sum_{m=1}^K \pi_m \exp\left\{-\frac{1}{2\sigma^2}\|x-\mu_m\|_2^2\right\}}$ (similar to softmax). Params to estimate π_k, σ^2, μ_k .

$\hat{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{n_k} x_i$, for all classes $k \in \{1, \dots, K\}$. $\hat{\sigma}^2 = \frac{1}{n-k}\sum_{i=1}^{n}\sum_{i=1}^{n_k}(x_i - \bar{x}_k)$. $\hat{\pi}_k = \frac{n_k}{n}$. To predict just $P(y=k|X) = \frac{\pi_k \exp\left\{-\frac{1}{2\sigma^2}\|x-\mu_k\|_2^2\right\}}{\sum_{m=1}^K \pi_m \exp\left\{-\frac{1}{2\sigma^2}\|x-\mu_m\|_2^2\right\}}$ but sub est.
We get linear decision boundaries (maximize log $P(y=k|X)$ w/ respect to k ; can drop all non k related terms).
■ **Nearest Centroid Classifier**: estimate centroids + classify observations to the nearest centroid.
■**NBC vs NCC**: NBC is equivalent to NCC under the following assumptions: feature independence, gaussian NBC w/ equal variance, Euclidean distance on NCC, equal priors for π_k for NBC (if unequal we see a shift in the decision boundary). This can be used to prove that the decision boundary is linear for NBC. **Estimation of params is done by using MLE.**
■**NBC Cons**: Weakness = assumptions of feature independence, equal variance across classes, data follows gaussian→ these imply that the data is spherical
■ **NBC unequal var**: if we relax the equal var assumption, $P(X|y=k) \sim N(\mu_k, \sigma_k^2 I)$ gives a QUADRATIC DECISION BOUNDARY!!!
■ **NBC w/ Correlation**: if there is correlation the NBC breaks and we need to use Linear Discriminant Analysis (LDA).
■ **Linear Discriminant Analysis (LDA)**: Assume (multivariate normal) $P(X|y=k) \sim N(\mu_k, \Sigma)$ where $\Sigma_{\mathrm{p\times p}}$ with in class covariance. Fit using MLE. $\hat{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{n_k} x_i$, $\hat{\pi}_k = \frac{n_k}{n}$, $\hat{\Sigma}_w = \frac{1}{n-k}\sum_{k=1}^{K}\sum_{i \in C_i(k)}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$.
■ **Decision Boundary**: $P(y=k|X) = P(y=k'|X) = \frac{1}{2}$, take log of both → $\log P(y=k|X) - \log P(y=k'|X) = 0$. Thus, $\log \hat{\pi}_k - \log \hat{\pi}_{k'} - \frac{1}{2}(x-\hat{\mu}_k)^T \hat{\Sigma}_w^{-1}(x-\hat{\mu}_k) + \frac{1}{2}(x-\hat{\mu}_{k'})^T \hat{\Sigma}_w^{-1}(x-\hat{\mu}_{k'}) = 0$. Then, simplify and we get a decision boundary that is linear in x.
■ **Whiten (or Sphere)**: $\hat{\Sigma}_w^{-1}$ whitens the difference between centroids. If we whiten or sphere the data then LDA on x is NBC on $\tilde{x} = X\hat{\Sigma}_w^{-1}$!!!!!!!!
■ **LDA Interpretation**: LDA is like whitening the data into sphere then just applying Naïve Bayes, i.e., remove correlation and then use NBC.
■ **LDA vs NBC**: $p > n$, $\hat{\Sigma}_w$ is singular so we cannot compute the LDA solution! NBC is fine. If we knew that there was high correlation and wanted to still apply LDA we could regularize $\tilde{\Sigma}_w(\alpha) = \hat{\Sigma}_w + \alpha I$. When p is large, NBC is very fast, but LDA can be intensive to compute inverse of $\hat{\Sigma}_w$.
■ **LDA vs Logistic**: (1) LDA assumes Gaussian x's. Logistic makes no assumptions on x's. (2) LDA is fit via full log likelihood; logistic fit via conditional log likelihood. (3) If the data is gaussian, LDA far outperforms Logistic, o/w if the data is not gaussian logistic outperforms LDA. E.g., say we have categorical variables we one-hot encoded (this is not gaussian) and LDA would not perform well and we would prefer logistic.
■ **Fisher's Discriminant Analsysis**: this performs the same function as whitening the data so we can apply NBC on whitened data is equivalent to LDA.

(13) ■ **Geometric Based Classifiers (Discriminative)**: Maximum margin classifiers. Binary classification SVM $y_i \in \{-1,1\}$, $i = 1, \dots, n$.
■ **Optimal Separating Hyperplanes** – assumes linearly separable classes: We want to place as much space as possible (margin) between two classes that are linearly separable. Goal: find a separating hyperplane that maximizes the margin between each class and the decision boundary. Assume that the margin is symmetric about the decision boundary. Minus plane: $x^-: \beta_0 + (x^-)^T\beta = -1$; Plus plane: $x^+: \beta_0 + (x^+)^T\beta = 1$. Take x^- and x^+ to be two points on $+/-$ plane that are closest in distance, i.e., distance$(x^+, x^-) = 2M$ where $M = ||margin||$. Want to write M in terms of $\beta + \beta_0$; note direction $= -\beta/\|\beta\|_2$, thus $x^+ = x^- + \frac{M\beta}{\|\beta\|_2}$. Solving for $M = \frac{1}{\|\beta\|_2}$.

■ **Optimization**: $\max_M M$ subject to data in each class lies outside the correct $+/-$ plane; mathematically, $\max_{\beta_0,\beta} \frac{1}{\|\beta\|_2}$ subject to $\begin{cases} if\ y_i = 1, \beta_0 + x_i^T\beta \geq 1 \\ if\ y_i = -1, \beta_0 + x_i^T\beta \leq -1 \end{cases}$ (pushes the data into correct plane outside of margin). Now we can simplify the constraint to, OPTIMAL SEPARATING HYPERPLANE (OSH) is $\min_{\beta_0,\beta}\|\beta\|_2$ subject to $y_i(\beta_0 + x_i^T\beta) \geq 1$, $i = 1, \dots, n$. This is a CONVEX problem so it will have a unique global solution in polynomial time (need an iterative solver).
■ **Properties + Interpretations OSH**: (1) There area always training points lying directly on the $+/-$ plane. (at least 2 points, usually 2-3). These points that directly lie on the +/- plane are called **Support Vectors** $\{x_i: |x_i^T\hat{\beta} + \hat{\beta}_0| = 1\}$ because they are holding up the hyperplanes.
■ **Is it a good property that 2-3 (points/support vectors) determine the slop of your decision boundary?** No. The decision boundary is subject to extreme shifts (outliers can change results, not robust and **VERY sensitive**!); this **requires separable classes**.
■ **Linear SVMS (extension of OSH)** – assumes not linearly separable classes: Idea, introduce slack to allow some points to be outside the correct +/- plane while pushing points to where they should be. Slack variable $\xi_i \geq 0, i = 1, \dots, n$ is the distance of x_i to the correct +/- plane. (1) Stabilizes the OSH so they are less sensitive to training data (2) allows for non-separable case.
■ **Optimization Linear SVMs**: $\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|_2^2$ subject to $y_i(\beta_0 + x_i^T\beta) \geq 1 - \xi_i$; $\xi_i \geq 0$. Now we want some constraint on ξ_i so we need to regularize with L_1 norm in order for them to be sparse! LASSO penalty. Encourages sparsity in the ξ_i which is the distance of points from correct plane (we want to be 0 for most points!). ■ **LaGrange form**: $\min_{\beta_0,\beta} \frac{1}{2}\|\beta\|_2^2 + C\|\xi\|_1$; (this can also be written as $\sum_{i=1}^n \xi_i$ since $\xi_i \geq 0$ so we don't need the absolute value). The hyperparameter is $C \geq 0$.
■ **Hyperparameter (C) results:** (1) $C\uparrow$, ξ_i are very sparse (when many ξ_i are small this means the margin size is very small as we push all points into correct +/- planes). (2) $C\downarrow$, there will be many nonzero ξ_i so the margins can be large.
■ **Interpretations Suppor Vectors**: **(type 1)** = $\{x_i: |x_i^T\hat{\beta} + \hat{\beta}_0| = 1\}$ which are on the +/- plane, $\xi_i = 0$. **(type2)** $\{x_i: 0 < \xi_i \leq 1\}$ outside of the +/- plane, so within the margins, but ALSO must be on the correct side of the decision boundary, i.e., correctly classified! **(type3)** $\{x_i: \xi_i > 1\}$ points that are incorrectly classified. ■ **Insights**: Can get insights on which points are significant in shaping the decision boundary! Also, insight into observations and features. (very large ξ_i helpful for finding outliers) ■ **Prediction**: $\hat{y} = sgn\left((x^*)^T\hat{\beta} + \hat{\beta}_0\right)$ and the magnitude of $(x^*)^T\hat{\beta} + \hat{\beta}_0$ tells us how confident we are in classifying that point.
■ **Notes**: SVMs are only defined for binary classification – solution is (one vs rest) OVR or (one vs one) OVO.



■ **KEY POINTS**: As we allow more points in the margin (less sparse ξ) we stabilize the slope of the decision boundary.

Matrix Calculus ■Matrix Formula (important): $Var(AX) = AVar(X)A^T$;

$\nabla_x x = \nabla_x x^T = I \in \mathbb{R}^{k\times k}$ $\nabla_x \mathbf{1}^T x = \nabla_x x^T \mathbf{1} = \mathbf{1} \in \mathbb{R}^k$ $\nabla_x (Ax - b) = A^T$ $\nabla_x (x^T A - b^T) = A$	$\nabla_x a^T x^T x b = 2xa^T b$ $\nabla_x a^T x x^T b = (ab^T + ba^T)x$ $\nabla_x a^T x^T x a = 2xa^T a$ $\nabla_x a^T x x^T a = 2aa^T x$ $\nabla_x a^T x x^T a = 2aa^T x$	$\nabla_x a^T y x^T b = ba^T y$ $\nabla_x a^T y^T x b = yb^T a$ $\nabla_x a^T x y^T b = ab^T y$ $\nabla_x a^T x^T y b = ya^T b$	$\nabla_X a^T X b = \nabla_X b^T X^T a = ab^T$ $\nabla_X a^T X^T X b = X(ab^T + ba^T)$ $\nabla_X a^T X X^T b = (ab^T + ba^T)X$ $\nabla_X a^T X^T X a = 2Xaa^T$ $\nabla_X a^T X X^T a = 2aa^T X$	$\frac{\partial}{\partial v}(v^T A v) = 2Av$; $\frac{\partial}{\partial v}(a^T v) = \frac{\partial}{\partial v}(v^T a) = a$; $\frac{\partial}{\partial x}(b^T A x) = A^T b$; $\frac{\partial}{\partial x}(x^T A x) = (A + A^T)x$. $\nabla_x (Ax - b)^T(Ax - b) = 2A^T(Ax - b)$; $\nabla_x \mathbf{1}^T e^{Ax} = A^T e^{Ax}$; $\nabla_x \log(\mathbf{1}^T e^x) = \frac{1}{\mathbf{1}^T e^x} e^x$